INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2025IPPAXXXX

Thèse de doctorat

Supervised Learning Methods for Offline Reinforcement Learning

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

Thèse de doctorat de l'Université Internationale de Rabat
École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat: Réseaux, Informations et Communications

Thèse à présenter et soutenir à Rabat, le 20/10/2025, par

Abdelghani GHANEM

Composition du Jury :

Houda BENBRAHIM Professeure, Université Mohamed V de Rabat	Présidente
Mohammed EL HASSOUNI Professeur, Université Mohamed V de Rabat	Rapporteur
Mustapha OUDANI Professeur associé, Université Internationale de Rabat	Rapporteur
Matthieu JONCKHEERE Directeur de recherche, LAAS-CNRS	Rapporteur
Philippe MARY Professeur, INSA Rennes	Examinateur
Philippe CIBLAT Professeur, Télécom Paris	Directeur de thèse
Mounir GHOGHO Professeur, Université Mohammed VI Polytechnique	Co-directeur de thèse
Mohammed BOULMALF Professeur, Université Internationale de Rabat	Co-directeur de thèse

Contents

List of Figures vii

List of Tables ix

Résumé xi

Abstract xiii

List of Abbreviations xv

Introduction 1

Notation Guide and Conventions 7

1 Multi-Objective Decision Transformers for Offline Reinforcement Learning 13

1.1 Introduction . . . 13
1.2 Related Work . . . 16
1.3 Preliminaries . . . 18
1.3.1 Setup and Notation . . . 19
1.3.2 Decision Transformer . . . 19
1.4 Multi-Objective Decision Transformers . . . 20
1.4.1 Multi-Task Learning for Decision Transformers . . . 20
1.4.2 Trust Region Decision Transformer . . . 21
1.4.3 Inference Procedure . . . 23
1.5 Results and Analysis . . . 23
1.5.1 Hyperparameters of MO-DT and TRDT . . . 25
1.5.2 Investigating the learned attention patterns . . . 28
1.6 Theoretical Insights . . . 30
1.6.1 Self-Attention Formulation for Analysis . . . 30
1.6.2 Token-Priority Graphs for Transformer Models . . . 32
1.6.3 Graph Structures and Convergence in Single-Task vs. Multi-Task Learning . . . 33
1.6.4 Implications for Decision Transformers . . . 35
1.6.5 Numerical Validation . . . 36
1.7 Conclusion . . . 38

2 State Prediction for Offline Reinforcement Learning via Sequence-to-Sequence Modeling **39**
2.1 Introduction . . . 39
2.2 Preliminaries . . . 41
2.2.1 Setup and Notation . . . 41
2.2.2 Sequence Modeling for Offline RL . . . 42
2.3 Offline RL as a Sequence-to-Sequence Modeling Problem . . . 43
2.3.1 Model Architecture . . . 43
2.3.2 Training the Sequence-to-Sequence Decision Translator . . . 44
2.3.3 Policy Extraction and Inference . . . 45
2.4 Experiments . . . 46
2.5 Conclusion . . . 48

3 When Tasks Collide: A Gradient-Flow Analysis of Alternating and Joint Training in Transformer Models **49**
3.1 Introduction . . . 49
3.2 Related Work . . . 51
3.3 Preliminaries . . . 52
3.3.1 Notation and Multi-Task Learning Setup . . . 53
3.3.2 The Modified Gradient Flow Framework . . . 53
3.3.3 Problem Statement . . . 54
3.4 Model agnostic Implicit regularization of Sequential vs Multi-Objective Training . . . 54
3.4.1 Modified Gradient Flow Analysis for Sequential Multi-Task Updates 57
3.5 Implicit Regularisation in One-Layer Self-Attention . . . 59
3.5.1 Mathematical Setup of the Self-Attention Model . . . 59
3.5.2 Preparatory Lemmas . . . 63
3.5.3 Notation for First- and Second-Order Quantities . . . 66
3.5.4 Implicit Regularization Effect . . . 67
3.6 Geometric Analysis of Task Competition . . . 68
3.6.1 Task Competition . . . 68
3.6.2 Competition Regions as Functions of Gradient Angle . . . 69
3.6.3 Special Cases and Implications . . . 70
3.7 Numerical Validation . . . 70
3.7.1 Optimization Trajectories . . . 71
3.7.2 Implicit Regularization Effect on Gradient Norms . . . 72
3.7.3 Implicit Regularization Effect on Flatness of Local Minima . . . 74
3.8 Discussion and Practical Implications . . . 75
3.8.1 Design Principles for Multi-Task Systems . . . 76
3.8.2 Implications for Transformer-based RL . . . 76
3.9 Conclusion . . . 77

4 When Can Sequence Modeling Approaches Recover the Target Policy in Offline Reinforcement Learning? A Statistical Analysis **79**
4.1 Introduction . . . 79
4.2 Related Work . . . 80
4.3 System Model . . . 81

4.4 Statistical Trajectory Model 82
4.5 Problem Statement 83
4.6 Main Results 83
4.7 Numerical Results 86
4.8 Conclusion and Future Directions 88

Conclusion and Future Perspectives **91**

Bibliography **97**

List of Figures

1.1 Attention patterns per block for DT during action prediction on the walker2d environment. **Top row:** Attention heatmaps where rows represent state tokens (as we focus only on next action prediction) and columns represent the full 60 tokens (20 tokens per modality context). The lower triangular structure results from masking future tokens, with color intensity indicating attention strength between the corresponding tokens. **Bottom row:** Mean attention weights from states to each modality (Returns, States, Actions) with standard deviation shown as error bars. The bar plots reveal that DT exhibits relatively uniform attention across modalities with minimal variation between layers, suggesting limited attention specialization. . 15

1.2 Block-wise attention patterns for MO-DT during action prediction on walker2d. **Top row:** Unlike DT, MO-DT demonstrates diversified focus on different tokens per block, indicative of more efficient utilization of the transformer's attention mechanism. **Bottom row:** Mean attention weights reveal progressive specialization across layers, with later layers showing increased attention to action tokens while maintaining balanced attention to other modalities. The error bars indicate greater variance in attention patterns compared to vanilla DT, suggesting that multi-objective training encourages attention head specialization. 28

1.3 Block-wise attention patterns for TRDT during action prediction on walker2d. **Top row:** The inclusion of action regions promotes more localized attention distribution, potentially indicating enhanced capacity to distinguish between tokens of the same type. Notably, attention at later blocks is allocated between both actions and action regions. **Bottom row:** Mean attention weights show that TRDT distributes attention more evenly between action regions and actions in later layers, reducing over-reliance on specific action tokens. The substantial error bars indicate highly specialized attention heads, with different heads focusing on different modalities—a key advantage over vanilla DT's homogeneous attention patterns. 29

1.4 Ablation studies showing the effect of each design choice in our proposed methods across w-m-r (walker2d-medium-replay-v2), ht-m-r (halfcheetah-medium-replay), and ho-m-r (hopper-medium-replay) datasets. 30

1.5 Comparison of Token-Priority Graphs for single-task and multi-task Decision Transformers. The different graph structures lead to different optimization problems that shape the attention mechanism in fundamentally different ways. 35

1.6 Empirical validation of our theoretical analysis. The left column shows results for the vanilla Decision Transformer (action prediction only), while the right column shows results for the Multi-Task Decision Transformer. The two different Graph-SVM problems lead to fundamentally different attention patterns. In the attention maps, even indices represent state tokens and odd indices represent action tokens. 37

2.1 RGDT framework overview. Our sequence-to-sequence architecture maps future returns to past states via an encoder-decoder structure, with actions inferred through an inverse dynamics model. This design disentangles modality processing, separately handling vector states/actions and scalar returns. 39

2.2 (a) Pearson correlation between each state and its subsequent states in Walker2d-Medium, decreasing with timestep offset. (b) Attention weights for GPT model with full context token prediction. (c) Attention weights with final token prediction only. Full context prediction (b) biases attention toward linear state correlations observed in (a), while final token prediction (c) shows more balanced attention distribution. 45

2.3 Comparison of RGDT training strategies on Gym D4RL medium tasks. . . 47

3.1 **Two paradigms of multi-task optimisation.** *(a)* Simultaneous GD aggregates both gradients before stepping. *(b)* Sequential GD applies task x first, producing $\theta^{(1)}$ that feeds task y; this shift is the source of the Hessian-driven correction analysed in §3.4. Colours match task identities. 49

3.2 Optimization trajectories for sequential versus multi-objective gradient descent under varying learning rates. 71

3.3 Optimization dynamics under similar task settings. The top row displays the evolution of (a) cosine similarity between the task gradients, (b) the norm of the summed gradients, and (c) the training loss for experiments conducted with a learning rate of 0.001. The bottom row presents the corresponding results for a learning rate of 0.01. 73

3.4 Optimization dynamics for dissimilar task settings. The top row shows the evolution of (a) cosine similarity between task gradients, (b) the norm of the summed gradients, and (c) the training loss for experiments conducted with a learning rate of 0.001. The bottom row presents the corresponding metrics for a learning rate of 0.01. 74

3.5 Loss surface analysis via weight perturbation. **Top row:** Results for models trained on similar tasks. **Bottom row:** Results for models trained on dissimilar tasks. For each setting, the left subfigure corresponds to a learning rate of 0.001 while the right subfigure corresponds to a learning rate of 0.01. The plots depict the change in the loss surface slope—computed as the variation in test error under multiplicative Gaussian perturbations applied to all network weights—with lower slopes indicating flatter minima and, consequently, enhanced generalization. 75

4.1 Empirical approximation error versus theoretical thresholds for varying ϵ values. Error bars represent standard deviation across 10 random seeds. . . 87

4.2 Approximation error as a function of high-return samples for $\epsilon = 0.2$. Vertical line indicates theoretical minimum sample size. 88
4.3 Sample complexity analysis for $\epsilon = 0.1$, showing increased sample requirements for lower error tolerance. 89

List of Tables

2 Core reinforcement learning notation used throughout all chapters 9
3 Notation specific to Chapter 1 10
4 Notation specific to Chapter 2 10
5 Notation specific to Chapter 3 11
6 Notation specific to Chapter 4 11

1.1 Summary of Hyperparameters used to train MO-DT and TRDT 25
1.2 Initial Return value for each environment 26
1.3 Averaged normalized scores on gym D4RL over 10 random seeds. Both of our methods outperform our primary baseline DT on almost all tasks, and matches or outperforms the best prior methods. 27

2.1 Normalized scores on Gym D4RL tasks (10 seeds). Our method **RGDT** achieves the best performance. 47

Résumé

L'apprentissage par renforcement hors ligne (RL) promet de dériver des politiques efficaces à partir d'ensembles de données statiques sans interaction coûteuse avec l'environnement, mais les approches existantes peinent face au décalage de distribution et à l'expressivité limitée. Cette thèse fait progresser les approches de modélisation de séquences pour le RL hors ligne à travers de nouvelles architectures, des perspectives théoriques et une validation empirique, démontrant comment les méthodes basées sur les transformers peuvent résoudre les limitations clés des algorithmes traditionnels de RL hors ligne.

Nous apportons quatre contributions principales. Premièrement, nous introduisons les Multi-Objective Decision Transformers (MO-DT), qui optimisent conjointement les tâches de prédiction d'états, d'actions et de retours pour induire des motifs d'attention diversifiés à travers les têtes du transformer—résolvant l'homogénéité d'attention observée dans les Decision Transformers standards. En s'appuyant sur cela, les Trust Region Decision Transformers (TRDT) augmentent les trajectoires avec des régions d'actions discrétisées, fournissant des représentations d'actions plus grossières qui réduisent le surapprentissage à des motifs comportementaux spécifiques. Deuxièmement, nous développons le Return-Guided Decision Translator (RGDT), une architecture séquence-à-séquence qui reformule le RL hors ligne comme la traduction de séquences de retours futurs en séquences d'états passés, avec des actions inférées par dynamique inverse. Cette approche encodeur-décodeur démêle naturellement le traitement des modalités.

Notre analyse théorique fournit des perspectives sur ces succès empiriques. À travers l'analyse du flot de gradient modifié sur des modèles d'auto-attention simplifiés, nous prouvons que la descente de gradient multi-tâches présente une régularisation implicite : elle encourage le désaccord entre tâches en minimisant les produits scalaires des gradients. Nous démontrons en outre que l'entraînement conjoint et séquentiel induisent des dynamiques d'optimisation fondamentalement différentes, les mises à jour séquentielles introduisant des corrections hessiennes du second ordre qui peuvent déstabiliser l'apprentissage. En utilisant le cadre des graphes de priorité de tokens, nous expliquons formellement pourquoi l'entrainement multi-objectif crée une attention équilibrée entre modalités tandis que l'entraînement mono-tâche induit des hiérarchies strictes de tokens. De plus, nous établissons des bornes de complexité d'échantillonnage pour la modélisation de séquences en RL hors ligne, révélant une transition critique entre les régimes dominés par la variance et ceux dominés par le biais qui fournit des orientations théoriques pour les stratégies de collecte de données.

Empiriquement, nos méthodes atteignent des performances compétitives sur les benchmarks de locomotion D4RL, avec TRDT améliorant le Decision Transformer standard jusqu'à 31% sur les tâches difficiles medium-replay tout en égalant ou dépassant les méthodes de pointe sur d'autres. Les visualisations d'attention confirment nos prédictions théoriques : l'entraînement multi-objectif produit des têtes d'attention spécialisées avec des motifs

inter-modaux distincts, tandis que les modèles mono-tâche présentent une attention redondante centrée sur les actions. Notre cadre théorique prédit avec précision les trajectoires d'optimisation et fournit des principes actionnables : des taux d'apprentissage plus petits réduisent la compétition entre tâches dans l'entraînement séquentiel, tandis que des poids multi-objectifs équilibrés favorisent une convergence stable.

Cette thèse démontre que des approches d'apprentissage supervisé fondées sur des principes, guidées par une compréhension théorique, peuvent efficacement relever les défis du RL hors ligne dans le cadre des tâches de contrôle continu. En unifiant les innovations architecturales avec une analyse rigoureuse, nous établissons la modélisation de séquences comme un paradigme viable pour apprendre à partir de données de trajectoires statiques, fournissant des fondations pour des travaux futurs dans des domaines plus complexes.

Abstract

Offline Reinforcement Learning (RL) promises to derive effective policies from static datasets without costly environment interaction, yet existing approaches struggle with distribution shift and limited expressivity. This thesis advances sequence modeling approaches to offline RL through novel architectures, theoretical insights, and empirical validation, demonstrating how transformer-based methods can address key limitations of traditional offline RL algorithms.

We make four primary contributions. First, we introduce Multi-Objective Decision Transformers (MO-DT), which jointly optimize state, action, and return prediction tasks to induce diverse attention patterns across transformer heads—addressing the attention homogeneity observed in vanilla Decision Transformers. Building on this, Trust Region Decision Transformers (TRDT) augment trajectories with discretized action regions, providing coarser action representations that reduce overfitting to specific behavioral patterns. Second, we develop Return-Guided Decision Translator (RGDT), a sequence-to-sequence architecture that reformulates offline RL as translating future return sequences to past state sequences, with actions inferred through inverse dynamics. This encoder-decoder approach naturally disentangles modality processing.

Our theoretical analysis provides insights into these empirical successes. Through modified gradient flow analysis on simplified self-attention models, we prove that multi-task gradient descent exhibits implicit regularization: it encourages task disagreement by minimizing gradient inner products. We further demonstrate that joint and sequential training induce fundamentally different optimization dynamics, with sequential updates introducing second-order Hessian corrections that can destabilize learning. Using the Token-Priority Graph framework, we formally explain why multi-objective training creates balanced cross-modal attention while single-task training induces strict token hierarchies. Additionally, we establish sample complexity bounds for sequence modeling in offline RL, revealing a critical transition between variance-dominated and bias-dominated regimes that provides theoretical guidance for data collection strategies.

Empirically, our methods achieve competitive performance on D4RL locomotion benchmarks, with TRDT improving upon vanilla Decision Transformer by up to 31% on challenging medium-replay tasks while matching or exceeding state-of-the-art methods on others. Attention visualizations confirm our theoretical predictions: multi-objective training produces specialized attention heads with distinct cross-modal patterns, while single-task models exhibit redundant, action-focused attention. Our theoretical framework accurately predicts optimization trajectories and provides actionable principles: smaller learning rates reduce task competition in sequential training, while balanced multi-objective weights promote stable convergence.

This thesis demonstrates that principled supervised learning approaches, guided by the-

oretical understanding, can effectively address the challenges of offline RL within the scope of continuous control tasks. By unifying architectural innovations with rigorous analysis, we establish sequence modeling as a viable paradigm for learning from static trajectory data, providing foundations for future work in more complex domains.

List of Abbreviations

Abbreviation	Full Form
Adam	Adaptive Moment Estimation
AdamW	Adam with Weight decay
AWAC	Advantage Weighted Actor Critic
BC	Behavioral Cloning
BCQ	Batch Constrained Q-learning
BEAR	Bootstrapping Error Accumulation Reduction
BERT	Bidirectional Encoder Representations from Transformers
BRAC	Behavior Regularized Actor Critic
CNN	Convolutional Neural Network
CQL	Conservative Q-Learning
D4RL	Datasets for Deep Data-Driven Reinforcement Learning
DT	Decision Transformer
EEG	Electroencephalography
GD	Gradient Descent
GF	Gradient Flow
GPT	Generative Pre-trained Transformer
HC-M	HalfCheetah-Medium
HC-ME	HalfCheetah-Medium-Expert
HP-M	Hopper-Medium
HP-ME	Hopper-Medium-Expert
IQL	Implicit Q-Learning
LAMB	Layer-wise Adaptive Moments optimizer for Batch training
LMCodec	Language Model Codec
MDP	Markov Decision Process
MGF	Modified Gradient Flow
MLP	Multi-Layer Perceptron
MO-DT	Multi-Objective Decision Transformer
MTL	Multi-Task Learning
NLL	Negative Log-Likelihood
NLP	Natural Language Processing
ODE	Ordinary Differential Equation
ODT-O	Offline Decision Transformer
ReLU	Rectified Linear Unit
RGDT	Return-Guided Decision Transformer

Continued on next page

Abbreviation	Full Form
RL	Reinforcement Learning
SCC	Strongly Connected Component
SM	Sequence Modeling
TPG	Token-Priority Graph
TRDT	Trust Region Decision Transformer
TT	Trajectory Transformer
TV	Total Variation
WK-M	Walker2d-Medium
WK-ME	Walker2d-Medium-Expert

Introduction

The Promise of Learning from Experience

In the grand narrative of artificial intelligence, few capabilities are as fundamental as learning from experience. Reinforcement learning (RL) has emerged as a powerful paradigm for this challenge, enabling agents to discover optimal behaviors through trial and error. From mastering complex games [1] to controlling plasma in fusion reactors [2], RL has demonstrated remarkable success when agents can freely explore their environments. Yet this very strength—the need for exploration—becomes a critical limitation in many real-world applications.

Consider an autonomous vehicle learning to navigate busy intersections, a robotic surgeon refining its technique, or an AI system optimizing drug dosing protocols. In these scenarios, the cost of exploration is measured not in computational cycles but in human safety and wellbeing. We cannot afford to let these systems learn through potentially catastrophic trial and error. This fundamental tension—between the need for experiential learning and the constraints of safety—motivates the field of offline reinforcement learning.

Offline RL promises a compelling resolution: what if we could learn optimal policies solely from historical data, without any risky exploration? Healthcare systems generate extensive collections of treatment records, autonomous vehicles collect millions of miles of driving data, and industrial robots accumulate years of operational logs. These datasets represent valuable repositories of experience that can be transformed into intelligent decision-making policies. The challenge lies in developing methods that can effectively extract this knowledge while avoiding the pitfalls that arise when learning from fixed, potentially suboptimal data.

The Fundamental Challenges of Learning from Static Data

The transition from online to offline learning fundamentally changes the nature of the reinforcement learning problem. In online RL, agents can test hypotheses, explore uncertain regions of the state space, and iteratively refine their understanding through interaction. Offline RL removes this feedback loop, creating several interconnected challenges that have proven remarkably difficult to overcome.

The most prominent challenge is *distribution shift*—the mismatch between the state-action distribution in the dataset and the distribution induced by the learned policy [3]. When an agent learns a policy that deviates from the behavior in the dataset, it must evaluate actions in states that may be poorly represented or entirely absent from the training data. Traditional value-based methods, which bootstrap value estimates from future states, can catastrophically overestimate the values of these out-of-distribution actions, leading to poor performance when deployed.

This challenge is compounded by the *coverage problem*. Unlike supervised learning, where we typically assume independent and identically distributed data, offline RL datasets consist of trajectories generated by unknown behavior policies. These trajectories may provide excellent coverage of some regions of the state-action space while leaving others completely unexplored. The sequential nature of the data means that certain state-action pairs may never appear together, not because they are suboptimal, but simply because the behavior policy never tried them.

Early approaches to offline RL attempted to address these challenges by adapting successful online algorithms with various forms of conservatism. Methods like Conservative Q-Learning (CQL) [4] penalize value estimates for out-of-distribution actions, while others like BEAR [5] constrain the learned policy to remain close to the behavior policy. While these approaches have shown success, they introduce a delicate balance: too much conservatism prevents the discovery of better policies than those in the dataset, while too little leads to overoptimistic extrapolation.

These challenges have led researchers to question whether the dynamic programming paradigm itself is well-suited to the offline setting. The iterative nature of value estimation, the need for explicit uncertainty quantification, and the sensitivity to hyperparameters all suggest that alternative approaches might be more natural for learning from static datasets.

A Paradigm Shift: From Dynamic Programming to Supervised Learning

The frustrations with traditional offline RL methods have catalyzed a fundamental rethinking of the problem. What if, instead of viewing offline RL through the lens of dynamic programming and value iteration, we approached it as a supervised learning problem? This shift in perspective, while radical, is grounded in a simple observation: trajectory data in offline RL has more in common with sequences in natural language processing than with the incremental feedback loops of online RL.

This reconceptualization transforms trajectories from Markov decision processes into sequences of tokens—states, actions, and rewards become the vocabulary of a new language. Just as language models learn to predict the next word given context, we can train models to predict the next action given a history of states and desired outcomes. This insight has profound implications: it allows us to leverage the remarkable advances in sequence modeling, particularly the transformer architecture [6], for reinforcement learning.

The Decision Transformer [7] implemented this approach, demonstrating that a transformer trained with simple supervised learning could match or exceed the performance of sophisticated offline RL algorithms. By conditioning on desired returns and predicting actions, it sidesteps the challenges of value estimation entirely. The Trajectory Transformer [8] extended this concept further, modeling entire trajectories autoregressively and enabling planning through beam search.

This paradigm shift offers several compelling advantages. First, it eliminates the need for bootstrapping and value iteration, avoiding the extrapolation errors that plague traditional methods. Second, it provides a natural way to condition on desired outcomes, enabling more flexible policy specification. Third, it leverages the substantial progress in sequence modeling architectures and training techniques developed by the natural language processing community.

Yet this new paradigm also raises fundamental questions. How should we represent trajectories for optimal learning? What implicit biases do these models have, and how do they differ from traditional RL methods? Can we provide theoretical guarantees about their performance? These questions motivate the research presented in this thesis.

Thesis Statement and Contributions

This thesis advances the following central argument: **Supervised learning methods, when properly designed with appropriate architectures and training objectives, and understood through rigorous theoretical analysis, can effectively address the fundamental challenges of offline reinforcement learning while providing competitive empirical performance and greater interpretability than traditional approaches.**

The practical implications of this work extend across numerous domains where safe learning from historical data is paramount. Our methods could enable more reliable deployment of AI systems in healthcare, robotics, and autonomous systems, where the supervised learning paradigm offers both performance improvements and greater interpretability for human operators.

We support this thesis through four interconnected contributions that span architectural innovations, training methodologies, and theoretical foundations:

First, we address a critical limitation in how Decision Transformers process trajectory data. Our analysis reveals that vanilla Decision Transformers develop homogeneous attention patterns, failing to fully utilize the expressive power of the transformer architecture. We introduce Multi-Objective Decision Transformers (MO-DT), which jointly optimize state, action, and return prediction tasks. This multi-task approach encourages the model to develop specialized attention patterns, with different heads focusing on different aspects of the trajectory. Building on this insight, we develop Trust Region Decision Transformers (TRDT), which augment trajectories with discretized action regions. These coarser action representations serve as an inductive bias that reduces overfitting to specific behav-

ioral patterns while maintaining the model's ability to generate precise actions. Through the Token-Priority Graph framework, we formally characterize how multi-objective training creates balanced attention across modalities.

Second, we explore alternative architectural paradigms beyond the decoder-only transformers used in prior work. We develop the Return-Guided Decision Translator (RGDT), which reconceptualizes offline RL as a sequence-to-sequence translation problem. By using an encoder-decoder architecture to translate future return sequences into past state sequences, with actions inferred through inverse dynamics, we achieve natural disentanglement of modality processing. This architectural choice enables more effective credit assignment and allows for asymmetric processing of different trajectory components.

Third, we provide theoretical foundations for understanding these empirical successes. Through modified gradient flow analysis, we uncover a fundamental property of multi-task learning: gradient descent implicitly encourages task disagreement by minimizing the inner products between task gradients. We prove that joint and sequential training induce fundamentally different optimization dynamics, with sequential updates introducing potentially destabilizing second-order corrections. These theoretical insights provide a rigorous explanation for why multi-objective training approaches outperform single-task alternatives in offline RL settings.

Fourth, we establish fundamental limits and guidelines for offline RL with sequence modeling. We derive sample complexity bounds that reveal a critical transition between variance-dominated and bias-dominated regimes, providing theoretical guidance for data collection strategies. Our analysis shows that success depends not just on dataset size but on the intricate balance between context coverage breadth and sampling depth within each context.

Throughout this thesis, we validate our approaches on standard benchmarks including D4RL locomotion tasks, demonstrating consistent improvements over both traditional offline RL methods and vanilla sequence modeling approaches.

Thesis Organization

This thesis is organized to provide both empirical innovations and theoretical understanding, with each chapter building upon previous insights while maintaining sufficient independence for selective reading.

Chapter 1 introduces Multi-Objective Decision Transformers, establishing the foundation for our architectural contributions. We begin by analyzing the limitations of vanilla Decision Transformers through attention visualization, then develop our multi-objective framework and trust region augmentation. We employ the Token-Priority Graph framework to provide theoretical justification for the observed attention patterns. This chapter demonstrates how architectural modifications can substantially improve performance on standard benchmarks while providing more interpretable attention patterns.

Chapter 2 explores the sequence-to-sequence paradigm through our Return-Guided

Decision Translator. We show how reconceptualizing offline RL as a translation problem enables more effective processing of trajectory data. This chapter highlights the benefits of encoder-decoder architectures and provides insights into credit assignment in sequential decision-making.

Chapter 3 provides the theoretical foundation for understanding multi-task training in transformers. We develop a comprehensive framework based on modified gradient flow analysis, revealing the implicit regularization effects of different training objectives. This chapter bridges the gap between our empirical observations and fundamental optimization principles, explaining why multi-objective approaches succeed in offline RL.

Chapter 4 examines the statistical foundations of offline RL with sequence modeling. We establish sample complexity bounds and identify critical factors that determine when sequence modeling approaches can successfully recover optimal policies. This chapter provides theoretical guidelines for data collection and helps explain when and why these methods succeed or fail.

The thesis concludes with a synthesis of our contributions and a discussion of future directions. We reflect on the broader implications of treating offline RL as supervised learning and identify promising avenues for extending this paradigm.

A Note on Reading This Thesis

This thesis is designed to accommodate different reading approaches depending on your background and interests:

For practitioners interested in implementation: Chapters 1 and 2 provide concrete architectural innovations with empirical validation. These chapters include implementation details and can be read independently.

For theorists interested in foundations: Chapters 3 and 4 offer rigorous theoretical analysis. While they reference the empirical work, the theoretical results stand on their own.

For researchers new to offline RL: Reading the chapters in order provides a natural progression from empirical observations to theoretical understanding.

For experts familiar with Decision Transformers: You may wish to start with Chapter 3 for the theoretical insights, then examine how they manifest in the architectural designs of Chapters 1 and 2.

Each chapter begins with its own introduction that provides the necessary context, allowing for flexible navigation through the material while maintaining coherence across the thesis.

List of Publications

The research presented in this thesis has resulted in the following publications:

1. **Multi-Objective Decision Transformers for Offline Reinforcement Learning**
   Under review, IEEE Transactions on Neural Networks and Learning Systems (TNNLS)

2. **State Prediction for Offline Reinforcement Learning via Sequence-to-Sequence Modeling**
   Accepted for publication, IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2025)

3. **When Tasks Collide: A Gradient-Flow Analysis of Alternating and Joint Training in Transformer Models**
   To be submitted

4. **When Can Sequence Modeling Approaches Recover the Target Policy in Offline Reinforcement Learning? A Statistical Analysis**
   Accepted for publication, European Signal Processing Conference (EUSIPCO 2025)

Notation Guide and Conventions

This chapter provides a comprehensive reference for the mathematical notation and conventions used throughout this thesis. We adopt standard conventions from the machine learning literature while maintaining consistency across all chapters.

General Notation Conventions

Typography and Mathematical Objects

Throughout this thesis, we employ the following typographical conventions to distinguish between different mathematical objects:

Scalars: Regular lowercase letters denote scalar values (e.g., x, a, α, η). Scalar functions and indices also use regular notation (e.g., t for time, i for sample index).

Vectors: Bold lowercase letters represent column vectors (e.g., $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{\theta}$, $\boldsymbol{h}_i$). The i-th element of vector $\boldsymbol{x}$ is denoted x_i (non-bold with subscript).

Matrices: Bold uppercase letters denote matrices (e.g., $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, $\boldsymbol{H}$). The element in row i and column j of matrix $\boldsymbol{X}$ is denoted $x_{i,j}$ or X_{ij}. Row vectors extracted from matrices are denoted as $\boldsymbol{x}_i$ (the i-th row of $\boldsymbol{X}$).

Sets and Spaces: Calligraphic uppercase letters represent sets and spaces (e.g., $\mathcal{S}$ for state space, $\mathcal{A}$ for action space, $\mathcal{T}$ for trajectory dataset, $\mathcal{V}$ for vocabulary).

Sequences: Sequences of vectors are treated as matrices and use bold uppercase notation. For sequences of scalars, we use bold lowercase to emphasize their vector nature (e.g., $\boldsymbol{g} = (g_1, \ldots, g_T) \in \mathbb{R}^T$).

Indexing Conventions

Subscripts serve multiple purposes:

- **Element indexing**: x_i denotes the i-th element of vector $\boldsymbol{x}$
- **Time indexing**: $\boldsymbol{s}_t$ denotes the state at time t

- **Sequence indexing**: $\boldsymbol{h}_k$ denotes the k-th element in a sequence
- **Task/component specification**: $\boldsymbol{\theta}_{DT}$ denotes parameters for the Decision Transformer component

Superscripts are reserved for:

- **Sample indexing**: $\boldsymbol{x}^{[i]}$ denotes the i-th data sample, with brackets to distinguish from exponentiation
- **Iteration indexing**: $\boldsymbol{\theta}^{(t)}$ denotes parameters at iteration t, with parentheses for clarity
- **Type specification**: $\mathcal{X}^h$ denotes high-return contexts, α_k^h denotes high-return proportion

Bracket Conventions:

- Square brackets $[\cdot]$ for sample indices: $\boldsymbol{H}^{[i]}$, $x^{[i]}$
- Parentheses $(\cdot)$ for iteration indices: $\boldsymbol{\theta}^{(1)}$, $\boldsymbol{W}^{(2)}$
- Curly braces $\{\cdot\}$ for sets: $\{1, 2, \ldots, N\}$
- Angle brackets $\langle\cdot\rangle$ for inner products: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$

Sequence and Range Notation

Subsequence notation: $\boldsymbol{S}_{t_1}^{t_2} = (\boldsymbol{s}_{t_1}, \ldots, \boldsymbol{s}_{t_2})$ denotes elements from index t_1 to t_2 (inclusive).

Relative indexing:

- $\boldsymbol{S}_{-K_s}$ denotes the past K_s states (shorthand for $\boldsymbol{S}_{t-K_s}^{t}$)
- $\boldsymbol{g}_{+K_r}$ denotes the future K_r returns (shorthand for $\boldsymbol{g}_{t+1}^{t+K_r+1}$)

Integer sets: $[N] := \{1, 2, \ldots, N\}$ denotes the set of integers from 1 to N.

Special Symbols

Operators and Functions:

- $\|\cdot\|_2$: Euclidean norm
- $\|\cdot\|_F$: Frobenius norm

- $\|\cdot\|_1$: 1-norm
- $\sigma(\cdot)$: Softmax function
- $\mathcal{N}(\mu, \Sigma)$: Gaussian distribution with mean μ and covariance Σ
- $\mathcal{B}(p)$: Bernoulli distribution with parameter p
- Δ^{k-1}: $(k-1)$-dimensional probability simplex

Constants:

- η: Learning rate
- γ: Discount factor
- ϵ: Small positive constant (context-dependent)
- ξ: Numerical stability constant

Chapter-Specific Notation

Symbol	Type	Description
$\mathcal{M}$	Set	Markov Decision Process
$\mathcal{S}$	Set	State space
$\mathcal{A}$	Set	Action space
$P(\cdot\|\cdot,\cdot)$	Function	Transition probability function
$R(\cdot,\cdot)$	Function	Reward function
γ	Scalar	Discount factor
$\boldsymbol{s}_t$	Vector	State at time t
$\boldsymbol{a}_t$	Vector	Action at time t
r_t	Scalar	Reward at time t
g_t	Scalar	Return-to-go at time t
G_t	Scalar	Discounted return at time t
π	Function	Policy (behavioral or general)
π^*	Function	Optimal/target policy
$\boldsymbol{\tau}$	Sequence	Trajectory
$\mathcal{T}$	Set	Dataset of trajectories

Table 2: Core reinforcement learning notation used throughout all chapters

Symbol	Type	Description
$\boldsymbol{S}$	Matrix	Sequence of states in trajectory
$\boldsymbol{A}$	Matrix	Sequence of actions in trajectory
$\boldsymbol{g}$	Vector	Sequence of returns in trajectory
$\bar{\boldsymbol{A}}$	Matrix	Sequence of action regions
Γ_t	Set	Context at time t
Γ_t^+	Set	Extended context with action regions
K	Scalar	Context length
b	Scalar	Action space discretization granularity
m_s	Scalar	State space dimensionality
m_a	Scalar	Action space dimensionality
$\boldsymbol{\theta}_{DT}$	Vector	Decision Transformer parameters
$\boldsymbol{\theta}_s$	Vector	State prediction parameters
$\boldsymbol{\theta}_g$	Vector	Return prediction parameters
$\boldsymbol{\lambda}$	Vector	Task weights on unit simplex
$\mathcal{L}_{DT}$	Function	Action prediction loss
$\mathcal{L}_s$	Function	State prediction loss
$\mathcal{L}_g$	Function	Return prediction loss

Table 3: Notation specific to Chapter 1

Symbol	Type	Description
K_r	Scalar	Encoder context length (returns)
K_s	Scalar	Decoder context length (states)
$\boldsymbol{S}_{-K_s}$	Matrix	Past K_s states
$\boldsymbol{g}_{+K_r}$	Vector	Future K_r returns
$\boldsymbol{\theta}_{RGDT}$	Vector	RGDT model parameters
$\boldsymbol{\phi}_{RGDT}$	Vector	Inverse dynamics parameters
$\rho_{\boldsymbol{\theta}}$	Function	State distribution function
$\pi_{\boldsymbol{\phi}}$	Function	Inverse dynamics policy

Table 4: Notation specific to Chapter 2

Symbol	Type	Description
$\boldsymbol{\theta}$	Vector	Shared model parameters
$\mathcal{L}_x$	Function	Task x loss function
$\mathcal{L}_y$	Function	Task y loss function
$\boldsymbol{g}_x$	Vector	Gradient of task x
$\boldsymbol{g}_y$	Vector	Gradient of task y
Φ	Scalar	Angle between task gradients
$\boldsymbol{W}$	Matrix	Attention weight matrix
$\boldsymbol{H}^{[i]}$	Matrix	Token embedding sequence
$\bar{\boldsymbol{h}}^{[i]}$	Vector	Query token
$\boldsymbol{c}^{[i]}$	Vector	Context vector
$\boldsymbol{\zeta}^{[i]}$	Vector	Attention weights
$\mathcal{V}$	Set	Token vocabulary
$\mathcal{V}_x$	Set	Task x vocabulary
$\mathcal{V}_y$	Set	Task y vocabulary
d_{in}	Scalar	Embedding dimension
$\boldsymbol{W}^x$	Matrix	Task x prediction matrix
$\boldsymbol{W}^y$	Matrix	Task y prediction matrix

Table 5: Notation specific to Chapter 3

Chapter 4 - Sample Complexity Analysis

Symbol	Type	Description
$\mathcal{D}$	Set	Supervised learning dataset
$c^{[i]}$	Element	Discrete context (from $\mathcal{X}$)
$\ell^{[i]}$	Element	Next token label
$\mathcal{X}$	Set	Context space
$\mathcal{Y}$	Set	Token/action space
C	Scalar	Number of contexts ($\lvert\mathcal{X}\rvert$)
V	Scalar	Vocabulary size ($\lvert\mathcal{Y}\rvert$)
$\mathcal{X}^h$	Set	High-return contexts
$\mathcal{X}^l$	Set	Low-return contexts
C^h	Scalar	Number of high-return contexts
α_k	Scalar	Expected proportion from policy k
α_k^h	Scalar	High-return proportion from policy k
β_c^h	Scalar	Estimated high-return proportion for context c
N_c	Scalar	Number of samples with context c
ν_{min}^h	Scalar	Minimum high-return samples
$\boldsymbol{p}$	Matrix	Learned conditional distribution
$\boldsymbol{\pi}^*$	Matrix	Target policy matrix

Table 6: Notation specific to Chapter 4

Function and Distribution Notation

Probability Distributions

- **Discrete distributions**: $\pi(a|s)$ denotes the probability of action a given state s
- **Continuous distributions**: $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for Gaussian, $\mathcal{B}(p)$ for Bernoulli
- **Empirical distributions**: $p(v|c)$ denotes learned conditional probabilities

Loss Functions and Optimization

- **Loss functions**: Always denoted with calligraphic $\mathcal{L}$ with appropriate subscripts
- **Gradients**: $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ or abbreviated as $\boldsymbol{g}$ when the context is clear
- **Hessians**: $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}$ for second-order derivatives

Norms and Inner Products

- **Vector norms**: $\|\boldsymbol{x}\|_2$ (Euclidean), $\|\boldsymbol{x}\|_1$ (Manhattan)
- **Matrix norms**: $\|\boldsymbol{X}\|_F$ (Frobenius), $\|\boldsymbol{X}\|_2$ (spectral)
- **Inner products**: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ or $\boldsymbol{x}^T\boldsymbol{y}$ (equivalent for real vectors)

Usage Guidelines

1. **Consistency**: When the same mathematical object appears across chapters, it maintains the same notation unless explicitly noted in a remark.
2. **Context clarity**: Subscripts and superscripts are chosen to maximize clarity. Time indices use subscripts, sample indices use bracketed superscripts.
3. **Dimension matching**: Vector and matrix dimensions are implicitly defined by context but explicitly stated when first introduced or when clarity demands.
4. **Overloading**: Some symbols are overloaded (e.g., π for both behavioral and specific policies) but the context makes the usage clear.
5. **Remarks**: When notation differs from established conventions or when connecting concepts across chapters, explicit remarks are provided to guide the reader.

This notation system follows established conventions in the machine learning and reinforcement learning literature while maintaining internal consistency throughout the thesis.

Chapter 1

Multi-Objective Decision Transformers for Offline Reinforcement Learning

1.1 Introduction

In the realm of offline Reinforcement Learning (RL), traditionally referred to as batch RL [3, 9], the objective is for an agent to acquire effective policies solely from static, finite datasets. These datasets are often harvested by an arbitrary, possibly unknown process, devoid of online interaction. This paradigm is particularly appealing across a range of real-world applications where data is readily available, but exploratory actions using untrained policies are prohibitive due to high costs or potential risks. Examples of such domains include robotics [10], recommender systems [11] , education [12], autonomous driving [13], and healthcare[14].

Early work in offline RL extended traditional off-policy algorithms by introducing various policy or value-regularization schemes to mitigate distributional shift when bootstrapping from fixed data. Seminal methods include BCQ [3], BEAR [5], CQL [4], AWAC [15], and BRAC [16]. While these methods demonstrated stable offline policy learning without environment interaction, they still suffer from extrapolation and overestimation biases inherent to off-policy value updates [3, 4], a conservatism–vs.–over-restriction trade-off [16], and strong hyperparameter sensitivity in the absence of online validation [17, 18]. Moreover, off-policy updates can amplify errors over long horizons, making the choice of discount factor γ a delicate balance between bias and variance [19, 20].

Alternatively, the challenges of traditional off-policy methods have motivated researchers to pursue simpler supervised learning approaches for offline RL. In particular, the sequential nature of trajectory data has inspired methods that recast offline RL as a sequence modeling task [7, 21], leveraging the algorithmic elegance and scalability of techniques from natural language processing. The key insight behind these approaches is that trajectories contain valuable behavioral information even when suboptimal for a specific reward function—behavior deemed ineffective for one task might be optimal for another. By conditioning on desired outcomes (e.g., target returns) or employing planning heuristics (e.g., beam search),

these sequence modeling methods can extract effective policies through direct *filtered* imitation without value bootstrapping or complex regularization schemes [22].

Two seminal architectures have emerged in the sequence modeling approach to offline RL: the Trajectory Transformer (TT) [21] and the Decision Transformer (DT) [7]. While both leverage the transformer architecture, they differ fundamentally in their representation of trajectories and learning objectives.

TT adopts a *component-level tokenization* scheme where each dimension of states, actions, and returns is independently discretized and treated as a distinct token. For m_s-dimensional states and m_a-dimensional actions over $\boldsymbol{\tau}$ timesteps, this results in sequences of length $\boldsymbol{\tau}(m_s + m_a + 1)$. TT is trained to model the joint distribution over all trajectory components autoregressively, enabling flexible policy extraction through beam search planning. The authors demonstrate that this fine-grained tokenization leads to distinctive attention patterns, where heads learn either Markovian strategies or dimension-specific "action smoothing" behaviors [21]. This architectural design maintains theoretical consistency with the original transformer paradigm, but the discretization process introduces computational challenges during both training and inference.

In contrast, DT employs *modality-level tokenization* where states, actions, and returns are embedded through shared linear layers, focusing primarily on action prediction. While this architectural choice offers computational efficiency and simplicity, empirical evidence suggests limitations in how the model utilizes its attention mechanism. Recent work [23] systematically compared tokenization strategies and found that finer component-level tokenization consistently outperforms modality-level approaches in representation quality and downstream performance, suggesting that DT's coarse tokenization may create an information bottleneck.

Analysis of attention patterns confirms this limitation. As shown in Figure 1.1, DT exhibits attention patterns that are remarkably homogeneous across different transformer blocks, with minimal specialization or diversity. This contrasts sharply with patterns observed in transformers across other domains. In NLP, [24] demonstrated that BERT's [25] attention heads develop specialized functions, with different heads attending to specific linguistic structures. Similarly, in computer vision, [26] found that Vision Transformer heads learn distinct spatial attention patterns, from local neighborhood focus to global image-wide attention.

The patterns observed in TT (Figure 4 in [21]) more closely resemble this expected diversity, suggesting that the issue stems from DT's architectural design rather than being inherent to offline RL data. [27] further reinforced this hypothesis by showing that simply disentangling action dimensions in DT architectures enhances attention diversity and improves performance, as different heads can then specialize in different action components.

These observations suggest a critical question: *without altering DT's computationally efficient tokenization approach, how can we encourage more diverse and effective attention patterns?* We propose that multi-task learning provides a natural solution. By training the model to predict not only actions but also states and returns under the framework of multi-objective optimization, we can force the attention mechanism to differentiate between tokens of the same modality type and develop more specialized attention heads. This

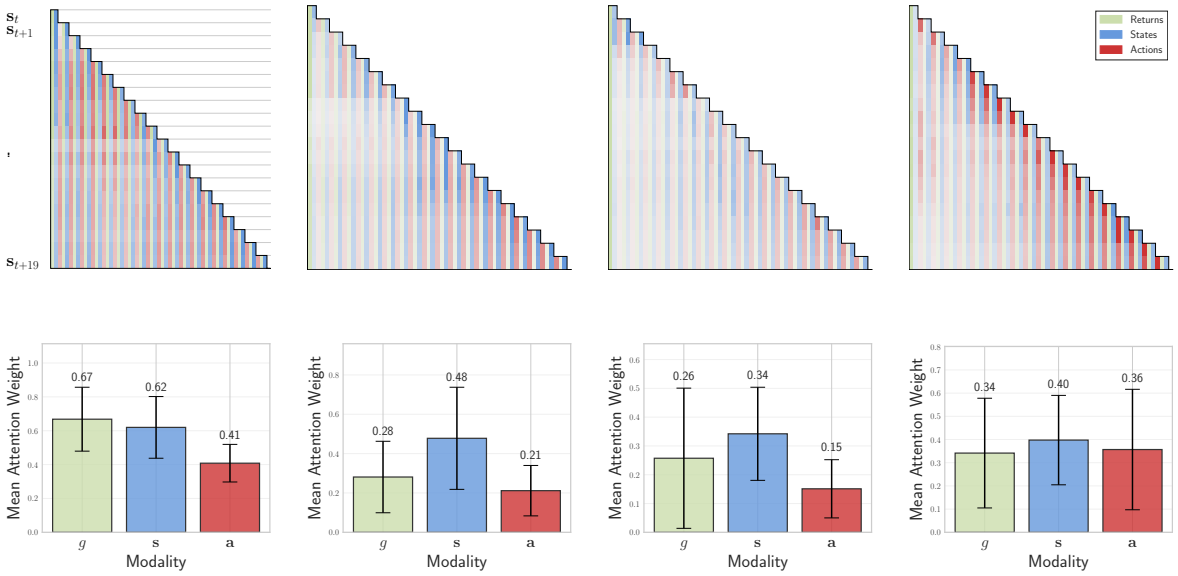


Figure 1.1: Attention patterns per block for DT during action prediction on the walker2d environment. **Top row:** Attention heatmaps where rows represent state tokens (as we focus only on next action prediction) and columns represent the full 60 tokens (20 tokens per modality context). The lower triangular structure results from masking future tokens, with color intensity indicating attention strength between the corresponding tokens. **Bottom row:** Mean attention weights from states to each modality (Returns, States, Actions) with standard deviation shown as error bars. The bar plots reveal that DT exhibits relatively uniform attention across modalities with minimal variation between layers, suggesting limited attention specialization.

approach maintains DT's computational advantages while addressing its limitations in attention utilization. We name our proposed approach *Multi-Objective Decision Transformer (MO-DT)*.

Following our analysis of Figure 1.1, we observe another concerning pattern: the model appears to be overly reliant on previous action tokens, attributing considerable attention weight to them. This is problematic as it biases the model toward pure behavioral cloning rather than learning generalizable state-action relationships, potentially leading to causal confusion [28] – a phenomenon where models leverage spurious correlations in training data rather than true causal factors. [28] demonstrated that providing a policy with the expert's past actions can paradoxically worsen performance by encouraging reliance on these misleading signals.

We hypothesize that this phenomenon in DT stems from fundamental differences in how states and actions evolve in dynamical systems [29]. States typically change more smoothly due to physical constraints and system dynamics, while actions (as direct control inputs) can change more abruptly between timesteps. This inherent difference creates more distinctive patterns in action token sequences compared to state (and return) tokens, making them easier targets for attention mechanisms to latch onto. In control systems,

this distinction is well-understood: state trajectories exhibit smoothness due to inertia and dynamics integration, while control signals can contain higher-frequency components [30].

To address this limitation, we propose to introduce action space regions — a higher-level abstraction that groups similar actions— as additional tokens in the trajectory representation. This approach, which we call *Trust Region Decision Transformer (TRDT)*, is designed specifically to modify how the transformer's attention mechanism processes trajectories. By providing a coarser representation of the action space alongside the original actions, we encourage the model to develop attention patterns that are less fixated on exact action reproduction and more focused on meaningful state-action relationships. This is conceptually similar to addressing causal confusion as studied by [31], who found that preventing models from overfitting to spurious correlations improves generalization. In our case, action regions serve as an inductive bias that helps the transformer avoid over-relying on specific action tokens, leading to more balanced attention distribution across modalities.

Our experiments on locomotion benchmarks from the D4RL dataset [32] demonstrate that both MO-DT and TRDT significantly outperform our primary baseline (vanilla DT) while achieving performance that matches or exceeds state-of-the-art off-policy methods. Moreover, analysis of the learned attention patterns confirms our hypothesis about increased diversity and more balanced cross-modal attention. To theoretically understand these empirical improvements, we present a formal analysis in Section 1.6 that leverages recent advances in analyzing the implicit bias of transformer attention mechanisms. Specifically, we employ the Token-Priority Graph framework [33] to prove that single-objective training induces a strict hierarchy in the attention structure that prioritizes action tokens while suppressing state tokens. In contrast, multi-objective training creates strongly connected components spanning different modalities, enabling more nuanced attention distributions. This theoretical insight not only explains the empirical success of our proposed methods but also provides a principled foundation for transformer-based approaches to offline RL.

The remainder of this paper is organized as follows. Section 1.2 reviews related work on transformers for reinforcement learning and attention diversity. Section 1.3 establishes notation and provides background on Decision Transformers. Section 1.4 introduces our multi-objective approach and trust region augmentation method, detailing their formulations and inference procedures. Section 1.5 presents experimental results on D4RL benchmarks, ablation studies, and attention pattern analysis. Section 1.6 provides theoretical insights using the Token-Priority Graph framework to explain why multi-objective training induces more diverse attention patterns. We conclude with a discussion of contributions and future directions.

1.2 Related Work

In this section, we provide an overview of the related work relevant to our study, focusing on two main fields: (1) transformers for reinforcement learning, and (2) explicit regularization techniques in transformers.

Transformers for Reinforcement Learning Recent advancements in casting reinforcement learning as a supervised learning problem have been noteworthy [34, 35]. As a further development of this paradigm, numerous studies have suggested treating offline RL as a context-conditioned sequence modeling problem [7, 21, 36], with [37] providing theoretical analysis of when such return-conditioned supervised learning succeeds or fails. This approach places greater emphasis on the predictive modeling of action sequences based on task specifications (e.g., returns or goals) rather than explicitly learning Q-functions or policy gradients, as commonly done in conventional model-free RL methods. [7] implemented a transformer [6, 38] as a model-free, context-conditioned deterministic policy, while [36] introduced a more general probabilistic formulation that employs a stochastic policy, which proved to be more effective in offline settings and can be fine-tuned efficiently in online scenarios. [21] employed a transformer as both a model and a policy, revealing that the integration of beam search can improve performance beyond several purely off-policy RL approaches. Despite their successes, these approaches face limitations in certain scenarios, with [39] demonstrating that Decision Transformers can struggle in stochastic environments due to overfitting to lucky trajectories.

Later research has explored various architectural innovations and training methodologies to address these limitations. Architectural extensions include Multi-Game Decision Transformers for transfer learning across multiple tasks [40], Context-Former for improved trajectory stitching via latent conditioning [41], DeFog for handling missing observations [42], and Graph Decision Transformer for processing graph-structured state spaces [43]. Additionally, the efficacy of these sequence modeling techniques can be enhanced through advanced training methodologies borrowed from NLP, including transfer learning [44], self-training [45], or contrastive learning [46].

In our work, we interrogate the foundational design choices of DT, particularly focusing on attention mechanisms and optimization objectives, as opposed to leveraging complex training methodologies or architectural extensions. By examining how multi-objective training affects transformer attention patterns, we provide insights that complement existing theoretical analyses [37] while maintaining the computational efficiency that makes DT appealing in practice. Naturally, the incorporation of sophisticated techniques explored in recent literature could further enhance our proposed models in future research.

Attention Mechanisms and Diversity in Transformers. Attention mechanisms form the core of transformer architectures, allowing models to selectively focus on different input elements when generating outputs. A growing body of research has sought to understand how these mechanisms function and develop specialized roles. Studies across both language and vision domains have revealed that attention heads can develop distinct functional specializations when properly trained. In NLP, [47] demonstrated that different heads focus on specific linguistic patterns, with some tracking long-range dependencies while others handle local details. Similarly, in computer vision, [26] found that vision transformer heads exhibit distinct spatial attention patterns ranging from local neighborhood focus to global image-wide attention. This functional specialization appears to be particularly pronounced in multi-task settings, where [48] showed heads can evolve to attend to different aspects of the input data when simultaneously processing multiple objectives.

Despite the potential for specialization, researchers have observed that transformers often develop redundant attention patterns. [49] demonstrated that many heads in transformers are functionally redundant, with pruning a large fraction of heads having minimal impact on performance. This redundancy can limit model expressivity and computational efficiency, as noted by [50], who found that attention matrices often lie in a low-dimensional space with many heads attending in similar ways. The problem is particularly pronounced in single-task settings, where the model lacks diverse learning signals that might encourage attention diversity. This redundancy is not merely a computational inefficiency but can also limit the transformer's ability to capture complex patterns in the data.

To address this limitation, various regularization approaches have been proposed to promote diversity among attention heads. These techniques aim to ensure that different heads attend to distinct aspects of the input, effectively expanding the model's representational capacity. Several methods have been explored, including introducing additional loss terms to explicitly encourage diversity among attention head outputs [51], leveraging dropout mechanisms [52] to randomly mask attention heads [53, 54] or entire encoder/decoder block layers [54] during training, and employing relaxed attention methods [55, 56] that modify how attention weights are computed. These approaches share the common goal of preventing heads from learning redundant patterns, though they differ in their implementation and theoretical underpinnings.

While explicit regularization has shown promise, an alternative approach is to provide the model with multiple learning objectives that naturally encourage attention diversity. [48] found that multi-task learning can enhance functional specialization among attention heads by reducing negative transfer between tasks. Similarly, [57] observed the emergence of specialized "induction heads" in language models trained on diverse corpora, which learn to replicate token sequences and enable higher-level capabilities. These findings suggest that multi-objective training may serve as an implicit form of attention regularization, encouraging heads to specialize in different aspects of the input data. In our work, we leverage this insight by incorporating state, action region, and return prediction tasks alongside the primary action prediction task, achieving increased attention diversity without explicit regularization terms.

1.3 Preliminaries

We adopt the Markov decision process (MDP) framework to model our environment, denoted by $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces, respectively. The probability distribution over transitions is denoted by $P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$, while the reward function is defined as $R(\boldsymbol{s}_t, \boldsymbol{a}_t)$. The discount factor, γ, is used to weigh the importance of future rewards. The agent starts in an initial state $\boldsymbol{s}_1$ sampled from the fixed distribution $p(\boldsymbol{s}_1)$ and chooses an action $\boldsymbol{a}_t \in \mathcal{A}$ from the state $\boldsymbol{s}_t \in \mathcal{S}$ at each timestep t. The agent then transitions to a new state $\boldsymbol{s}_{t+1}$ according to the probability distribution $P(\cdot \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$. After each action, the agent receives a deterministic reward $r_t = R(\boldsymbol{s}_t, \boldsymbol{a}_t)$.

1.3.1 Setup and Notation

Throughout this paper, we use bold capital letters (e.g., $\boldsymbol{X}$, $\boldsymbol{W}$) to denote matrices, bold lowercase letters (e.g., $\boldsymbol{x}$, $\boldsymbol{v}$) for vectors, and regular lowercase letters (e.g., x, a) for scalars. Subscripts on vectors denote rows from matrices (e.g., $\boldsymbol{x}_i$ is the i-th row vector of $\boldsymbol{X}$) and double subscript indices denote scalar entries of matrices (e.g., $x_{i,j}$ is the element in i-th row and j-th column of $\boldsymbol{X}$). We use superscripts (e.g., $\boldsymbol{X}^{[i]}$) to denote the i-th data sample.

Our goal is to model the offline RL problem as a sequence modeling problem, with the agent having access to a fixed dataset of trajectories $\mathcal{T}$ collected using an unknown behavioral policy. We use $\boldsymbol{\tau}$ to represent a trajectory and $|\boldsymbol{\tau}|$ to denote its length. The return-to-go of a trajectory $\boldsymbol{\tau}$ at timestep t is defined as the sum of future rewards starting from that timestep, i.e., $g_t = \sum_{t'=t}^{|\boldsymbol{\tau}|} r_{t'}$. Note that throughout this paper, we use 'return' as shorthand for 'return-to-go' to avoid confusion with traditional returns in RL. The sequence of states, actions, action regions, and returns of trajectory $\boldsymbol{\tau}$ are represented by $\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{|\boldsymbol{\tau}|})$, $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{|\boldsymbol{\tau}|})$, $\bar{\boldsymbol{A}} = (\bar{\boldsymbol{a}}_1, \ldots, \bar{\boldsymbol{a}}_{|\boldsymbol{\tau}|})$, and $\boldsymbol{g} = (g_1, \ldots, g_{|\boldsymbol{\tau}|})$, respectively. We denote the quantization granularity of the action space as b, the dimensionality of the action space as m_a, and the dimensionality of the state space as m_s.

Remark 1 (Notation for Sequences). Throughout this thesis, we treat sequences of scalars as vectors and denote them using bold notation. For instance, while individual returns g_t are scalars, the sequence of returns $\boldsymbol{g} = (g_1, \ldots, g_{|\boldsymbol{\tau}|})$ is denoted in bold as it can be viewed as a vector in $\mathbb{R}^{|\boldsymbol{\tau}|}$. This convention maintains consistency with our general principle that bold symbols represent multi-dimensional objects.

Remark 2 (Return Notations). Throughout this thesis, we primarily use g_t to denote the return-to-go (sum of future rewards). In subsequent chapters, we may introduce additional notations for different types of returns when needed. For instance, G_t may be used to denote discounted returns when the distinction from return-to-go is important for the analysis.

1.3.2 Decision Transformer

The DT model is designed to process a trajectory $\boldsymbol{\tau}$ as a sequence of three types of input tokens: returns, states, and actions. Formally, the trajectory $\boldsymbol{\tau}$ is represented as follows:

$$\boldsymbol{\tau} = (g_1, \boldsymbol{s}_1, \boldsymbol{a}_1, g_2, \boldsymbol{s}_2, \boldsymbol{a}_2, \ldots, g_{|\boldsymbol{\tau}|}, \boldsymbol{s}_{|\boldsymbol{\tau}|}, \boldsymbol{a}_{|\boldsymbol{\tau}|}) \tag{1.1}$$

The initial return g_1 is equivalent to the return of the trajectory. At each timestep t, DT utilizes the latest K tokens to generate an action $\boldsymbol{a}_t$. The value of K is a hyperparameter referred to as the context length for the transformer, which may be shorter during evaluation than the one used during training.

For clarity and compactness, we define the context up to time t as

$$\Gamma_t = \{g_{t-K}^t, \boldsymbol{S}_{t-K}^t, \boldsymbol{A}_{t-K}^{t-1}\} \tag{1.2}$$

where $\boldsymbol{S}_{t-K}^t$ represents the sequence of K past states from $\max(1, t-K+1)$ to t, and similarly for g_{t-K}^t and $\boldsymbol{A}_{t-K}^{t-1}$. Note that by convention, Γ_t does not include the action $\boldsymbol{a}_t$ at

the current timestep, as this is precisely what the model aims to predict. For models that also utilize action regions, we extend our context notation to $\Gamma_t^+ = \{\Gamma_t, \bar{\boldsymbol{A}}_{t-K}^{t-1}\}$.

DT can learn either a deterministic or a stochastic policy $\pi_{\boldsymbol{\theta}_{DT}}(\boldsymbol{a}_t \mid \Gamma_t)$. This creates an autoregressive model of order K. DT parameterizes the policy using a GPT architecture [38], which applies a causal mask to enforce the autoregressive structure in the predicted action sequence.

1.4 Multi-Objective Decision Transformers

In a standard probabilistic framework, the goal of DT is to learn a stochastic policy that maximizes the likelihood of the training data [36].

For continuous action spaces, we model the policy using a multivariate Gaussian distribution with a diagonal covariance matrix [58]. Formally, the probability density assigned to action vector $\boldsymbol{a}_t$ at time step t, given the context of past tokens, is expressed as:

$$\pi_{\boldsymbol{\theta}_{DT}}(\boldsymbol{a}_t \mid \Gamma_t) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}_{DT}}(\Gamma_t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}_{DT}}(\Gamma_t)), \quad \forall t \tag{1.3}$$

The mean vector $\boldsymbol{\mu}_{\boldsymbol{\theta}_{DT}}$ and covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}_{DT}}$ are produced by the transformer model with parameters $\boldsymbol{\theta}_{DT}$.

The primary task loss is the negative log-likelihood:

$$\mathcal{L}_{DT}(\boldsymbol{\theta}_{DT}) = -\frac{1}{K}\mathbb{E}_{\boldsymbol{\tau}\sim\mathcal{T}}\left[\sum_{k=1}^{K} \log \pi_{\boldsymbol{\theta}_{DT}}(\boldsymbol{a}_k \mid \Gamma_k)\right] \tag{1.4}$$

When the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}_{DT}}$ is diagonal with uniform variances across dimensions, this problem reduces to learning a deterministic policy with standard ℓ_2 loss, as implemented in the original DT [7]. This type of regression problem is referred to as heteroscedastic regression [59].

1.4.1 Multi-Task Learning for Decision Transformers

In addition to the primary task of action prediction, we introduce two auxiliary tasks, namely, state prediction and return prediction. These are parameterized by $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_g$ respectively, which together with $\boldsymbol{\theta}_{DT}$ form the complete parameter set $\boldsymbol{\theta}$.

For state prediction, we model the next state distribution as a multivariate Gaussian:

$$\rho_{\boldsymbol{\theta}_s}^{(1)}(\boldsymbol{s}_t \mid \Gamma_{t-1}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}_s}(\Gamma_{t-1}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}_s}(\Gamma_{t-1})), \quad \forall t > 1 \tag{1.5}$$

For return prediction, we model the next return distribution as a univariate Gaussian:

$$\rho_{\boldsymbol{\theta}_g}^{(2)}(g_t \mid \Gamma_{t-1}) = \mathcal{N}(\mu_{\boldsymbol{\theta}_g}(\Gamma_{t-1}), \sigma_{\boldsymbol{\theta}_g}^2(\Gamma_{t-1})), \quad \forall t > 1 \tag{1.6}$$

Note that unlike action prediction which uses the current context Γ_t, state and return predictions use the previous context Γ_{t-1} since we are predicting the next state and return given past information.

The auxiliary loss functions correspond to negative log-likelihoods of these distributions:

$$\mathcal{L}_s(\boldsymbol{\theta}_s) = -\frac{1}{K}\mathbb{E}_{\boldsymbol{\tau}\sim\mathcal{T}}\left[\sum_{k=1}^{K}\log \rho^{(1)}_{\boldsymbol{\theta}_s}(\boldsymbol{s}_k \mid \Gamma_{k-1})\right] \tag{1.7}$$

$$\mathcal{L}_g(\boldsymbol{\theta}_g) = -\frac{1}{K}\mathbb{E}_{\boldsymbol{\tau}\sim\mathcal{T}}\left[\sum_{k=1}^{K}\log \rho^{(2)}_{\boldsymbol{\theta}_g}(g_k \mid \Gamma_{k-1})\right] \tag{1.8}$$

The Multi-Objective Decision Transformer (MO-DT) jointly optimizes these auxiliary losses together with the primary action prediction loss using linear scalarization:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i\in\{DT,s,g\}} \lambda_i \mathcal{L}_i(\boldsymbol{\theta}_i) \tag{1.9}$$

where $\boldsymbol{\theta} = \bigcup_{i\in\{DT,s,g\}} \boldsymbol{\theta}_i$ represents the complete set of model parameters, and $\boldsymbol{\lambda} = [\lambda_{DT}, \lambda_s, \lambda_g]$ are task-specific weights on the unit simplex

$$\Delta^2 = \left\{ \boldsymbol{\lambda} \in \mathbb{R}^3 : \lambda_i \geq 0 \text{ for all } i \text{ and } \sum_{i\in\{DT,s,g\}} \lambda_i = 1 \right\} \tag{1.10}$$

This linear scalarization approach provides computational efficiency, though it may not guarantee Pareto optimality in non-convex optimization landscapes. More advanced multi-objective optimization techniques [60, 61] could potentially improve performance but introduce additional computational complexity and hyperparameters, which we leave for future work.

1.4.2 Trust Region Decision Transformer

In the offline RL setting, trajectories are typically generated by behavior policies that may be sub-optimal and highly stochastic, leading to diverse action distributions within each trajectory. This creates a potential limitation for MO-DT, which could become overly focused on these specific actions during prediction. To address this challenge, we introduce TRDT, which augments trajectories with action regions that group semantically similar actions.

Action Region Augmented Trajectories. We modify the trajectory representation in Eq. 1.1 by incorporating action regions:

$$\boldsymbol{\tau} = (g_1, \boldsymbol{s}_1, \bar{\boldsymbol{a}}_1, \boldsymbol{a}_1, g_2, \boldsymbol{s}_2, \bar{\boldsymbol{a}}_2, \boldsymbol{a}_2, \ldots, g_{|\boldsymbol{\tau}|}, \boldsymbol{s}_{|\boldsymbol{\tau}|}, \bar{\boldsymbol{a}}_{|\boldsymbol{\tau}|}, \boldsymbol{a}_{|\boldsymbol{\tau}|}) \tag{1.11}$$

Here, $\bar{\boldsymbol{a}}_t$ represents the region vector for action $\boldsymbol{a}_t$, obtained through coarse uniform discretization of the continuous action space. This approach preserves Euclidean distance

relationships from the original action space, potentially capturing the underlying problem structure better than the empirical training distribution.

Efficient Action Region Representation. To manage high-dimensional action spaces efficiently, we represent action regions using ordinal encodings [62] instead of one-hot encodings. For an action space of dimensionality m_a with discretization granularity b, each action dimension is represented by an ordinal encoding vector of length b. The complete action region encoding is formed by concatenating these vectors, resulting in a binary vector of dimensionality $b \times m_a$.

This representation offers two key advantages: (1) it scales linearly with action dimensionality rather than exponentially, and (2) it maintains ordinal relationships between discretized actions. Formally, we model the distribution of action regions using a multivariate Bernoulli distribution:

$$\rho^{(3)}_{\boldsymbol{\theta}_{\bar{a}}}(\bar{\boldsymbol{a}}_t \mid \Gamma^+_t) = \prod_{j=1}^{b \times m_a} \mathcal{B}(p_{\boldsymbol{\theta}_{\bar{a}},j}(\Gamma^+_t)) \tag{1.12}$$

where $p_{\boldsymbol{\theta}_{\bar{a}},j}$ represents the probability of the j-th element in the binary action region vector, and $\mathcal{B}(p)$ represents the Bernoulli probability mass function with parameter p.

The corresponding negative log-likelihood loss is:

$$\mathcal{L}_{\bar{a}}(\boldsymbol{\theta}_{\bar{a}}) = -\frac{1}{K}\mathbb{E}_{\boldsymbol{\tau}\sim\mathcal{T}}\left[\sum_{k=1}^{K} \log \rho^{(3)}_{\boldsymbol{\theta}_{\bar{a}}}(\bar{\boldsymbol{a}}_k \mid \Gamma^+_k)\right] \tag{1.13}$$

During inference, we use a simple threshold-based approach, classifying values above 0.5 as 1 and below as 0. While more sophisticated techniques exist for ordinal encoding prediction, such as sequential prediction [62] or factorized architectures [63], our approach balances effectiveness with computational efficiency.

Multi-Objective Optimization with Action Regions. When incorporating action regions, we adapt the probabilistic functions $\rho^{(1)}_{\boldsymbol{\theta}_s}$, $\rho^{(2)}_{\boldsymbol{\theta}_g}$, and $\pi_{\boldsymbol{\theta}_{DT}}$ to condition on previous action regions in addition to actions, states, and returns. Importantly, we do not condition the prediction of $\boldsymbol{a}_t$ on the corresponding action region $\bar{\boldsymbol{a}}_t$ at the same timestep for two reasons: (1) it eliminates the need for action region prediction during inference, and (2) it prevents error propagation if the model predicts sub-optimal action regions.

The complete multi-objective optimization problem for TRDT becomes:

$$\boldsymbol{\theta}^*_{TR} = \arg\min_{\boldsymbol{\theta}_{TR}} \sum_{i\in\{DT,s,g,\bar{a}\}} \lambda_i \mathcal{L}_i(\boldsymbol{\theta}_{TR}) \tag{1.14}$$

where $\boldsymbol{\theta}_{TR} = \bigcup_{i\in\{DT,s,g,\bar{a}\}} \boldsymbol{\theta}_i$ represents the complete set of model parameters, and $\boldsymbol{\lambda} = [\lambda_{DT}, \lambda_s, \lambda_g, \lambda_{\bar{a}}]$ are task-specific weights on the unit simplex Δ^3.

1.4.3 Inference Procedure

During evaluation, both MO-DT and TRDT utilize a closed-loop inference process that leverages their multi-task capabilities.

For MO-DT, we begin with an initial state vector $\boldsymbol{s}_1$ and a desired return scalar g_1. Since there is no prior history at the first timestep, the model generates the initial action using a restricted version of the action prediction head:

$$\boldsymbol{a}_1 = \boldsymbol{\mu}_{\boldsymbol{\theta}_{DT}}(\Gamma_1) \tag{1.15}$$

where $\Gamma_1 = \{g_1, \boldsymbol{s}_1\}$ contains only the initial state and target return. After executing this action, the model predicts the next return using the return prediction head:

$$g_2 = \mu_{\boldsymbol{\theta}_g}(\Gamma_1^+) \tag{1.16}$$

where $\Gamma_1^+ = \{g_1, \boldsymbol{s}_1, \boldsymbol{a}_1\}$ represents the initial history augmented with the first action. This return prediction capability represents a key departure from standard DT, which typically derives the next return by subtracting the received reward. Our empirical results demonstrate that model-based return prediction leads to more consistent performance. The agent then observes the next state $\boldsymbol{s}_2$ from the environment transition function $P(\cdot \mid \boldsymbol{s}_1, \boldsymbol{a}_1)$ and generates the next action using the complete history:

$$\boldsymbol{a}_2 = \boldsymbol{\mu}_{\boldsymbol{\theta}_{DT}}(\Gamma_2) \tag{1.17}$$

where $\Gamma_2 = \{g_1, g_2, \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{a}_1\}$. This cycle continues until episode termination, with each step leveraging the full context of previously observed states, actions, and predicted returns.

For TRDT, the inference procedure follows the same overall structure with one addition: at each timestep, we also append the corresponding action region $\bar{\boldsymbol{a}}_t$ to the trajectory after generating action $\boldsymbol{a}_t$. Following our training methodology, the action region is not used to condition the action prediction at the same timestep, but becomes part of the context for future predictions. Both methods are summarized in Algorithm 1.

1.5 Results and Analysis

Our experiments are designed to comparatively evaluate our two proposed multi-objective decision transformers against prior offline RL methods. Specifically, we aim to understand how the incorporation of multi-objective optimization affects the attention mechanism in the Decision Transformer. Furthermore, we vary different components of our approach to demonstrate their significance.

Benchmark and Compared Baselines. We evaluate our algorithms, MO-DT and TRDT, on the D4RL offline dataset for continuous control tasks [32]. Our experiments are conducted in the Gym domain [64], encompassing three environments (halfcheetah, hopper,

Algorithm 1 Multi-Objective Decision-Transformer Training & Inference

Require: fixed dataset of trajectories $\mathcal{T}$; context length K; learning-rate η; weight vector $\boldsymbol{\lambda}$; discretisation granularity b

initialise parameters $\boldsymbol{\theta}_{DT}, \boldsymbol{\theta}_s, \boldsymbol{\theta}_g, \boldsymbol{\theta}_{\bar{a}}$

Training phase

for each training iteration **do**

  sample mini-batch $\{\boldsymbol{\tau}_i\} \sim \mathcal{T}$

  if model = MO-DT **then**

    compute losses $\mathcal{L}_{DT}, \mathcal{L}_s, \mathcal{L}_g$

    $\mathcal{L} \leftarrow \sum_{i \in \{DT,s,g\}} \lambda_i \mathcal{L}_i$

  else ▷ TRDT branch

    augment each trajectory with action-regions $\bar{\boldsymbol{a}}_t$

    compute losses $\mathcal{L}_{DT}, \mathcal{L}_s, \mathcal{L}_g, \mathcal{L}_{\bar{a}}$

    $\mathcal{L} \leftarrow \sum_{i \in \{DT,s,g,\bar{a}\}} \lambda_i \mathcal{L}_i$

  end if

  LAMB-update: $\boldsymbol{\theta} \leftarrow \text{LAMB}(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} \mathcal{L}, \eta)$

end for

Inference phase

Require: initial state $\boldsymbol{s}_1$, target return g_1

$\Gamma_1 \leftarrow \{g_1, \boldsymbol{s}_1\}$

for $t \leftarrow 1$ **to** T_{episode} **do**

  if model = MO-DT **then**

    $\boldsymbol{a}_t \leftarrow \boldsymbol{\mu}_{\boldsymbol{\theta}_{DT}}(\Gamma_t)$

  else ▷ TRDT branch

    $\boldsymbol{a}_t \leftarrow \boldsymbol{\mu}_{\boldsymbol{\theta}_{DT}}(\Gamma_t^+)$

    compute $\bar{\boldsymbol{a}}_t$ from $\boldsymbol{a}_t$

  end if

  execute $\boldsymbol{a}_t$; observe $r_t, \boldsymbol{s}_{t+1}$

  predict g_{t+1} with $\boldsymbol{\theta}_g$

  update history Γ_{t+1} (and Γ_{t+1}^+ for TRDT)

  trim history to the last K tokens

end for

walker2d), each with three levels (medium, medium-replay, medium-expert). All experiments utilize the 'v2' version of D4RL. For comparison, we consider our primary baseline, DT [7], and the offline variant of the online decision transformer, ODT-O, proposed in [36]. This variant substitutes the deterministic policy of DT with a Gaussian policy. Results from the original papers of both methods are reported. For the experiments of 'medium-expert' for ODT-O are based on our reproduction following the hyperparameters and code of the authors. We also compare against the two strongest dynamic programming-based state-of-the-art methods CQL[4] and IQL[4]. As the original paper for CQL reports performance on the 'V0' version of D4RL, which generally performs worse than 'V2', we refer to results

reported in [65]. Additionally, we compare with pure Behavioral Cloning (BC) and its variant, 10% BC, which mimics the behavior of the top 10%. For BC, we report results from [65], and for 10% BC, we refer to [7].

1.5.1 Hyperparameters of MO-DT and TRDT

In this section, we detail the optimal architectural and hyperparameter settings for MO-DT and TRDT. Our chosen model is half the size of the ODT-O model presented in [36]. Specifically, our model employs a Transformer structure comprising four layers, each equipped with four attention heads. The embedding size is set to 256, deviating from the 512 dimension embedding used in ODT-O. Our empirical studies revealed no significant performance gain by increasing the embedding size to 512.

For both models, we used the LAMB optimizer [66] to optimize the model parameters. In terms of scalarization coefficients, uniform values have consistently yielded superior results across all conducted experiments with both models. Finally, consistent with the findings from [36], our models do not incorporate positional embeddings. We found this approach to yield superior results compared to configurations that include them. For TRDT, we used an action region granularity $b = 3$ for all experiments. The full list of hyperparameters is summarized in Table 1.1.

Hyperparameter	**Value**
Number of bins (TRDT)	$b = 3$
Linear scalarization coefficients (MO-DT)	Uniform
Linear scalarization coefficients (TRDT)	Uniform
Number of layers	4
Number of attention heads	4
Embedding dimension	256
Context Length K	20
Dropout	0.1
Nonlinearity function	ReLU
Batch size	256
Learning rate	0.0001
Weight decay	0.001
Gradient norm clip	0.25
positional embedding	No
Learning rate warmup	linear warmup for 10^4 training steps
Total number of updates	10^5

Table 1.1: Summary of Hyperparameters used to train MO-DT and TRDT

In terms of the initial return prompt for both models, empirical evidence indicated that initializing with a return value twice that of the expert's return yielded superior results across all tested datasets, with the exception of the HalfCheetah dataset. For this particular

dataset, the use of the expert return proved most effective. The exact return values are summarized in Table 1.2

Domain	**Initial Return Value**
Halfcheetah	12000
Walker2d	10000
Hopper	7200

Table 1.2: Initial Return value for each environment

Table 1.3: Averaged normalized scores on gym D4RL over 10 random seeds. Both of our methods outperform our primary baseline DT on almost all tasks, and matches or outperforms the best prior methods.

Dataset	BC	10% BC	CQL	IQL	DT	ODT-O	MO-DT(Ours)	TRDT(Ours)
halfcheetah-medium-v2	42.6	42.5	44.0	**47.4**	42.6	42.72	43.34± 3.34	43.20 ± 3.0
hopper-medium-v2	52.9	56.9	58.5	**66.3**	**67.6**	**66.95**	**67.45** ± 11.21	**65.89** ± 8.38
walker2d-medium-v2	75.3	75.0	72.5	**78.3**	74.0	72.19	**78.56** ± 6.30	75.89 ± 7.26
halfcheetah-medium-replay-v2	36.6	40.6	**45.5**	**44.2**	36.6	39.99	40.93 ± 2.13	42.61 ± 2.0
hopper-medium-replay-v2	18.1	75.9	**95.0**	**94.7**	82.7	86.64	79.43 ± 5.2	**97.82** ± 2.3
walker2d-medium-replay-v2	26.0	62.5	77.2	73.9	66.6	68.92	81.3 ± 9.72	**86.94** ± 6.23
halfcheetah-medium-expert-v2	55.2	92.9	91.6	86.7	86.8	88.76	**94.57** ± 2.78	**94.37** ± 3.19
hopper-medium-expert-v2	52.5	**110.9**	105.4	91.5	107.6	107.24	**111.71** ± 1.52	**111.53** ± 1.31
walker2d-medium-expert-v2	**107.5**	**109.0**	**108.8**	**109.6**	**108.1**	**108.65**	**108.08** ± 0.54	**108.52** ± 0.45
Total	466.7	666.2	698.5	692.4	672.6	674.06	705.37	**726.77**

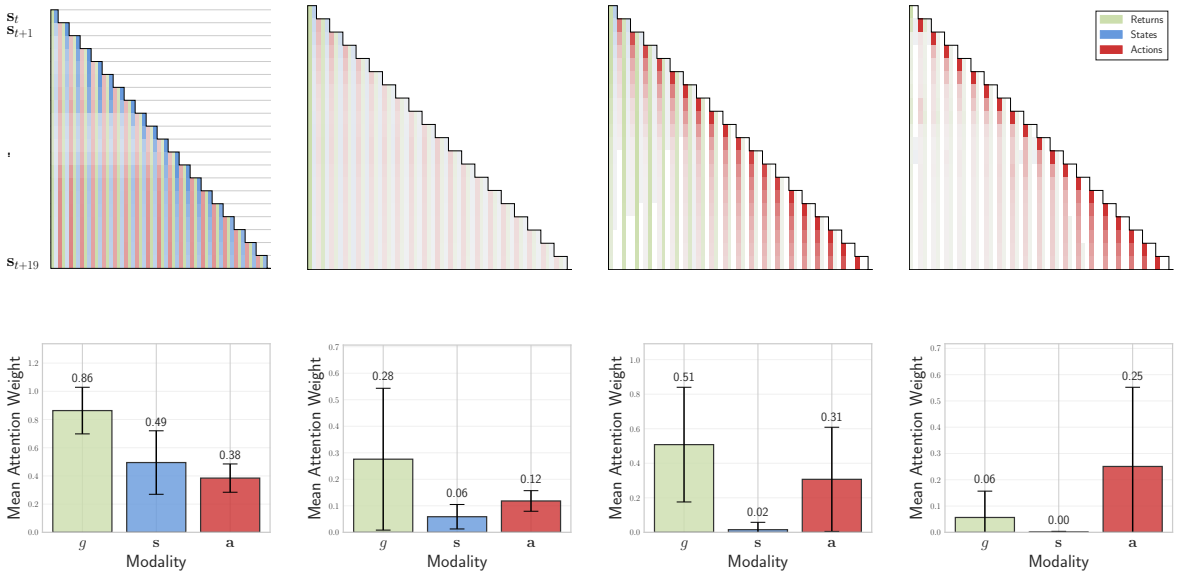


Figure 1.2: Block-wise attention patterns for MO-DT during action prediction on walker2d. **Top row:** Unlike DT, MO-DT demonstrates diversified focus on different tokens per block, indicative of more efficient utilization of the transformer's attention mechanism. **Bottom row:** Mean attention weights reveal progressive specialization across layers, with later layers showing increased attention to action tokens while maintaining balanced attention to other modalities. The error bars indicate greater variance in attention patterns compared to vanilla DT, suggesting that multi-objective training encourages attention head specialization.

1.5.2 Investigating the learned attention patterns

In Figures 1.2 and 1.3, we present attention maps for MO-DT and TRDT models, respectively. Firstly, we discern that both models prompt the decision transformer to focus selectively on distinct tokens per block, with the attention appearing sparser for the TRDT case, contrasting the uniform attention patterns displayed by DT as shown in Figure 1.1. This implies that multi-objective optimization in our configuration potentially mirrors the effects of contemporary transformer regularization techniques. Secondly, we note that later blocks of the transformer model predominantly engage with high-level representations of action tokens over return or state tokens. This observation aligns with the findings by [21] for the Trajectory Transformer. Lastly, for TRDT, attention in subsequent blocks is divided between action regions and action tokens, hinting at a decreased dependency on the behavioral policy. Despite the derivation of action regions from the behavior policy, their coarse nature (we use $b = 3$ across all experiments) leads to their presence in a wider array of trajectories and contexts, possibly enhancing the DT model's capacity to effectively integrate different trajectory segments.

Gaussian vs Deterministic Attention Heads To substantiate our choice of Gaussian prediction heads for states and returns over linear prediction heads (trained using mean

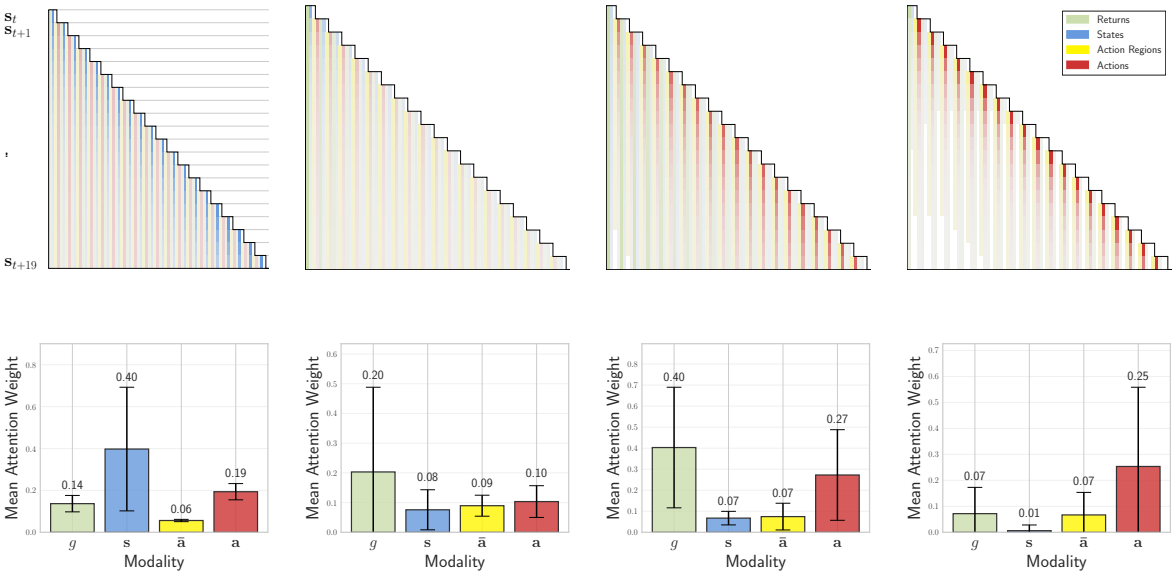
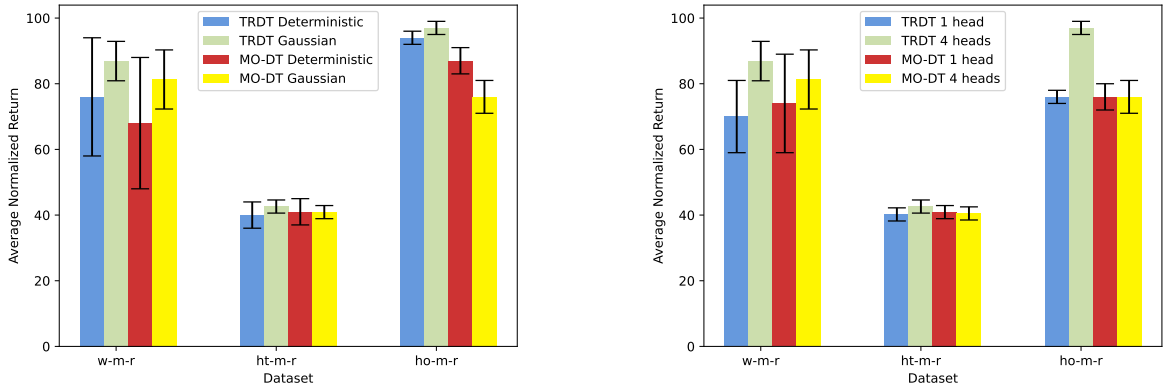


Figure 1.3: Block-wise attention patterns for TRDT during action prediction on walker2d. **Top row:** The inclusion of action regions promotes more localized attention distribution, potentially indicating enhanced capacity to distinguish between tokens of the same type. Notably, attention at later blocks is allocated between both actions and action regions. **Bottom row:** Mean attention weights show that TRDT distributes attention more evenly between action regions and actions in later layers, reducing over-reliance on specific action tokens. The substantial error bars indicate highly specialized attention heads, with different heads focusing on different modalities—a key advantage over vanilla DT's homogeneous attention patterns.

squared error minimization), we compared MO-DT and TRDT both with and without Gaussian prediction heads in Figure 1.4a. For simpler environment dynamics, as in the Hopper environment (action dimensionality 3, state dimensionality 6), deterministic prediction heads perform on par with their Gaussian counterparts. However, in the face of complex dynamics, deterministic heads yield high variance within model predictions, an established observation in RL literature [58].

Importance of Multi-Head Attention In Figure 1.4b, our aim is to evaluate the extent to which our model leverages the inherent capabilities of the transformer architecture. We observe that employing a singular attention head detrimentally affects the performance of both models. The common purpose of multiple attention heads is to facilitate simultaneous attention to diverse representation subspaces at distinct positions. Given this, the performance degradation with a single attention head implies that our model is effectively utilizing multiple representation subspaces in parallel, thus capitalizing on the transformer architecture's potential.



(a) The effect of Gaussian prediction heads.

(b) The effect of multi-head attention.

Figure 1.4: Ablation studies showing the effect of each design choice in our proposed methods across w-m-r (walker2d-medium-replay-v2), ht-m-r (halfcheetah-medium-replay), and ho-m-r (hopper-medium-replay) datasets.

1.6 Theoretical Insights

In order to understand why incorporating additional tasks leads to improved attention diversity, we adopt recent advances from the literature of implicit regularization, which studies the asymptotic convergence behavior of models trained using gradient descent. In particular, we employ the Token-Priority Graph framework introduced by [33], which establishes that self-attention mechanisms trained with gradient descent converge to solutions that can be characterized as maximum margin classifiers in a graph-structured space. This theoretical lens allows us to analyze the attention patterns observed in Figure 1.1 and explain why vanilla DT develop such similar attention patterns across different blocks and heads. Furthermore, we demonstrate that incorporating auxiliary prediction tasks fundamentally alters the structure of these graphs, leading to the more diverse and effective attention patterns we observe empirically in our multi-objective models.

1.6.1 Self-Attention Formulation for Analysis

To apply the Token-Priority Graph framework to Decision Transformers, we need to introduce theoretical simplifications that make our model amenable to analysis. For clarity and simplicity, we focus on action and state prediction tasks, though our analysis extends naturally to any number of modalities or tasks.

In the standard Decision Transformer, different modalities (states, actions, returns) are embedded through separate projection matrices before being processed by the attention mechanism. For theoretical tractability, we simplify this representation by assuming all tokens exist in a common embedding space of dimension d_{in}. This allows us to convert the trajectory dataset $\mathcal{T}$ into a supervised learning dataset $\mathcal{D}_{multi} = \{(\boldsymbol{H}^{[i]}, a^{[i]}, s^{[i]})\}_{i=1}^{N}$, where $\boldsymbol{H}^{[i]} \in \mathbb{R}^{K \times d_{\text{in}}}$ represents a sequence of token embeddings, and $a^{[i]} \in \mathcal{A}$, $s^{[i]} \in \mathcal{S}$ represent target action and state token indices respectively.

Let $\bar{\boldsymbol{h}}^{[i]} = \boldsymbol{h}_K^{[i]}$ denote the final token in the sequence $\boldsymbol{H}^{[i]}$. For our theoretical analysis, we employ a simplified self-attention formulation. While standard attention mechanisms use separate query, key, and value projection matrices, we follow [33] in using a single learnable matrix $\boldsymbol{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ that combines query-key interactions, and we assume identity value projections. The self-attention operation then embeds the final token in a d_{in}-dimensional space as follows:

$$\boldsymbol{c}^{[i]} = \boldsymbol{H}^{[i]\top} \sigma(\boldsymbol{H}^{[i]} \boldsymbol{W} \bar{\boldsymbol{h}}^{[i]}), \tag{1.18}$$

where $\sigma(\cdot)$ denotes the softmax operation. This formulation represents a weighted linear combination of input tokens, where the weights are determined by the similarity between each token and the final token, as measured by $\boldsymbol{W}$. The vector $\boldsymbol{c}^{[i]} \in \mathbb{R}^{d_{\text{in}}}$ is referred to as the context vector of the final token.

For our theoretical analysis to be tractable, we employ the following technical conditions:

1. The embedding matrix $\boldsymbol{H}$ has full row rank, ensuring linear independence among token embeddings.
2. The token prediction matrices satisfy orthogonality with their respective token embeddings, as defined below.
3. The dataset satisfies the realizability condition: for any sequence-target pair $(\boldsymbol{H}^{[i]}, a^{[i]}, s^{[i]})$, both the target action token and the target state token are contained in the input sequence $\boldsymbol{H}^{[i]}$.

Given this simplified attention mechanism, we introduce prediction matrices for our two tasks:

- $\boldsymbol{W}^a \in \mathbb{R}^{|\mathcal{A}| \times d_{\text{in}}}$ for action prediction, where $|\mathcal{A}|$ is the number of action tokens[1]
- $\boldsymbol{W}^s \in \mathbb{R}^{|\mathcal{S}| \times d_{\text{in}}}$ for state prediction, where $|\mathcal{S}|$ is the number of state tokens

Let us denote by $\boldsymbol{w}_i^a$ the row vector in $\boldsymbol{W}^a$ corresponding to action token $i \in \mathcal{A}$, and similarly $\boldsymbol{w}_j^s$ for the row in $\boldsymbol{W}^s$ corresponding to state token $j \in \mathcal{S}$. These prediction matrices satisfy the orthogonality condition with respect to the token embeddings:

$$\boldsymbol{w}_i^a \cdot \boldsymbol{h}_i = 1 \quad \text{and} \quad \boldsymbol{w}_i^a \cdot \boldsymbol{h}_{i'} = 0 \quad \forall i' \neq i, i' \in \mathcal{A} \tag{1.19}$$

$$\boldsymbol{w}_j^s \cdot \boldsymbol{h}_j = 1 \quad \text{and} \quad \boldsymbol{w}_j^s \cdot \boldsymbol{h}_{j'} = 0 \quad \forall j' \neq j, j' \in \mathcal{S} \tag{1.20}$$

[1]Throughout this thesis, we primarily work with continuous state and action spaces as defined in our MDP framework. However, in certain theoretical analyses (e.g., Sections 1.6.1 and 1.6), we employ discretized or tokenized representations of these spaces for analytical tractability. In such contexts, we use $|\mathcal{A}|$ and $|\mathcal{S}|$ with a slight abuse of notation to denote the cardinality of the discretized token sets, rather than the original continuous spaces. This discretization is particularly relevant when analyzing transformer architectures that operate on finite vocabularies of tokens.

where $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ represent the token embeddings for action token i and state token j respectively. This condition ensures that each prediction vector can precisely identify its corresponding token when that token is present in the context vector.

The prediction process produces token scores as follows:

$$\hat{v}_a^{[i]} = \boldsymbol{w}_{a^{[i]}}^a \cdot \boldsymbol{c}^{[i]} = \boldsymbol{w}_{a^{[i]}}^a \cdot \boldsymbol{H}^{[i]\top} \sigma(\boldsymbol{H}^{[i]} \boldsymbol{W} \bar{\boldsymbol{h}}^{[i]}) \tag{1.21}$$

$$\hat{v}_s^{[i]} = \boldsymbol{w}_{s^{[i]}}^s \cdot \boldsymbol{c}^{[i]} = \boldsymbol{w}_{s^{[i]}}^s \cdot \boldsymbol{H}^{[i]\top} \sigma(\boldsymbol{H}^{[i]} \boldsymbol{W} \bar{\boldsymbol{h}}^{[i]}) \tag{1.22}$$

These scores measure how strongly the context vector aligns with each possible token. Under the orthogonality conditions and realizability assumption, if the attention mechanism places all weight on the target token, the score would be exactly 1 for the correct token and 0 for all others.

Given an input sequence $\boldsymbol{H}^{[i]}$ with target tokens $a^{[i]}$ and $s^{[i]}$, the empirical risk for both tasks is formulated using the negative log-likelihood loss:

$$\mathcal{L}_a(\boldsymbol{W}) = \frac{1}{N} \sum_{i=1}^{N} -\log(\hat{v}_a^{[i]}) \tag{1.23}$$

$$\mathcal{L}_s(\boldsymbol{W}) = \frac{1}{N} \sum_{i=1}^{N} -\log(\hat{v}_s^{[i]}) \tag{1.24}$$

The negative log-likelihood is appropriate here because it strongly penalizes low scores for correct tokens, effectively encouraging the attention mechanism to prioritize target tokens in the context vector.

For multi-objective training, we optimize the combined loss:

$$\mathcal{L}_{\text{multi}}(\boldsymbol{W}) = \mathcal{L}_a(\boldsymbol{W}) + \mathcal{L}_s(\boldsymbol{W}) \tag{1.25}$$

A critical aspect of our formulation is that while we have multiple prediction tasks (action and state prediction), they share the same attention mechanism parameterized by $\boldsymbol{W}$. This shared parameterization is key to understanding how multi-task learning affects attention patterns - both tasks influence the same attention weights, potentially leading to different Token-Priority Graph structures than would emerge from single-task learning. In the following section, we formalize how these Token-Priority Graphs characterize the implicit bias of the attention mechanism under both single-task and multi-task training.

1.6.2 Token-Priority Graphs for Transformer Models

Building on the self-attention model described in the previous section and following the theoretical framework described in [33], we introduce the concept of Token-Priority Graphs (TPGs) that capture the relationship between different tokens in the transformer vocabulary. A TPG is a directed graph where each node represents a token in the transformer's

vocabulary. A directed edge from token i to token j indicates that token i has higher priority than token j in the attention mechanism. Tokens that are mutually reachable from each other form a strongly connected component (SCC), indicating equal priority.

We use $(i \asymp j) \in \mathcal{G}$ to indicate that nodes i and j are in the same strongly connected component of graph $\mathcal{G}$, and $(i \Rightarrow j) \in \mathcal{G}$ to denote that a directed path exists from node i to node j in graph $\mathcal{G}$, but no path from j to i. We also denote by $\boldsymbol{h}_i$ the embedding vector corresponding to token i in the vocabulary, and by $\|\boldsymbol{W}\|_F$ the Frobenius norm of matrix $\boldsymbol{W}$.

The structure of the TPG depends critically on which prediction tasks are used during training. When training only for action prediction with loss $\mathcal{L}_a(\boldsymbol{W})$, the resulting TPG will differ from that obtained when training with the multi-objective loss $\mathcal{L}_{\text{multi}}(\boldsymbol{W})$. This difference in graph structure is key to understanding why multi-task learning leads to more diverse attention patterns.

We define the gradient descent optimization process used to train the attention mechanism as follows: given a starting point $\boldsymbol{W}(0)$ and step size $\eta > 0$, the gradient descent update rule for iteration $t \geq 0$ is $\boldsymbol{W}(t+1) = \boldsymbol{W}(t) - \eta \nabla \mathcal{L}(\boldsymbol{W}(t))$, where $\boldsymbol{W}(t)$ denotes the attention weight matrix at iteration t and $\mathcal{L}$ represents either the single-task loss $\mathcal{L}_a$ or the multi-task loss $\mathcal{L}_{\text{multi}}$.

1.6.3 Graph Structures and Convergence in Single-Task vs. Multi-Task Learning

Having established the concept of Token-Priority Graphs, we now analyze how different training objectives lead to distinct TPG structures in Decision Transformers. When analyzing the attention mechanism, we observe fundamentally different patterns in the TPGs depending on whether single-task or multi-task learning is employed.

An Action-Only TPG, denoted $\mathcal{G}_a$, is constructed when the model is trained only with the action prediction loss $\mathcal{L}_a(\boldsymbol{W})$. This graph has several key properties: first, for all $i \in \mathcal{A}$ and $j \in \mathcal{S}$, we have $(i \Rightarrow j) \in \mathcal{G}_a$, indicating that action tokens have higher priority than state tokens. Second, for all $j \in \mathcal{S}$ and $i \in \mathcal{A}$, we have $(j \Rightarrow i) \notin \mathcal{G}_a$, meaning there are no directed paths from state tokens to action tokens. Finally, strongly connected components can only be formed by tokens within $\mathcal{A}$, not across modalities, reflecting the fact that only action tokens can be prediction targets.

In contrast, a Multi-Modality TPG, denoted $\mathcal{G}_m$, arises when the model is trained with the multi-objective loss $\mathcal{L}_{\text{multi}}(\boldsymbol{W})$. In this graph, for some $i \in \mathcal{A}$ and $j \in \mathcal{S}$, both $(i \Rightarrow j) \in \mathcal{G}_m$ and $(j \Rightarrow i) \in \mathcal{G}_m$ can exist, creating strongly connected components that can span across modalities. This reflects the fact that both action and state tokens can be prediction targets.

These different graph structures lead to fundamentally different optimization problems when training the attention mechanism. For the Action-Only TPG $\mathcal{G}_a$, the corresponding

Graph-SVM problem [33] is:

$$\boldsymbol{W}_a = \arg\min_{\boldsymbol{W}} \|\boldsymbol{W}\|_F \tag{1.26}$$

$$\text{s.t.} \quad (\boldsymbol{h}_i - \boldsymbol{h}_j)^\top \boldsymbol{W}\bar{\boldsymbol{h}} = 0 \quad \forall (i \asymp j) \in \mathcal{G}_a \tag{1.27}$$

$$(\boldsymbol{h}_i - \boldsymbol{h}_j)^\top \boldsymbol{W}\bar{\boldsymbol{h}} \geq 1 \quad \forall (i \Rightarrow j) \in \mathcal{G}_a \tag{1.28}$$

Similarly, for the Multi-Modality TPG $\mathcal{G}_m$, the corresponding Graph-SVM problem is:

$$\boldsymbol{W}_m = \arg\min_{\boldsymbol{W}} \|\boldsymbol{W}\|_F \tag{1.29}$$

$$\text{s.t.} \quad (\boldsymbol{h}_i - \boldsymbol{h}_j)^\top \boldsymbol{W}\bar{\boldsymbol{h}} = 0 \quad \forall (i \asymp j) \in \mathcal{G}_m \tag{1.30}$$

$$(\boldsymbol{h}_i - \boldsymbol{h}_j)^\top \boldsymbol{W}\bar{\boldsymbol{h}} \geq 1 \quad \forall (i \Rightarrow j) \in \mathcal{G}_m \tag{1.31}$$

The critical mathematical distinction between these two problems lies in their constraint structure. In the Action-Only Graph-SVM, there are many inequality constraints between action and state tokens (enforcing $\boldsymbol{h}_i^\top \boldsymbol{W}\bar{\boldsymbol{h}} \geq \boldsymbol{h}_j^\top \boldsymbol{W}\bar{\boldsymbol{h}} + 1$ for $i \in \mathcal{A}, j \in \mathcal{S}$). These constraints force the attention mechanism to prioritize action tokens over state tokens, effectively suppressing attention to state tokens. Equality constraints exist only between action tokens that form strongly connected components.

In contrast, in the Multi-Modality Graph-SVM, many of these inequalities are replaced by equality constraints (when tokens from different modalities form an SCC) or are even reversed, when some states get higher priority compared to actions. This allows for more diverse attention allocation across modalities, enabling state tokens to receive meaningful attention weights. This difference is illustrated in Figure 1.5.

The following result proves that Decision Transformers trained with different objectives converge to the corresponding Graph-SVM solutions, explaining why they exhibit different attention patterns:

Theorem 1 (Modality-Dependent Attention Bias)**.** *Consider the self-attention model defined in equation 1.18. Under the technical conditions established earlier and with reference to Theorem 2 in [33]:*

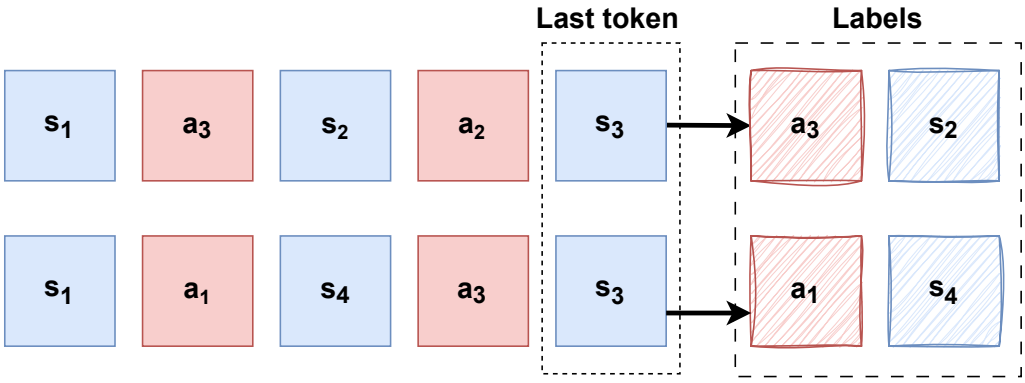
(a) When trained only on action prediction ($\mathcal{L}(\boldsymbol{W}) = \mathcal{L}_a(\boldsymbol{W})$), the attention weights converge in direction to the solution of the Action-Only Graph-SVM problem:

$$\lim_{t\to\infty} \frac{\boldsymbol{W}(t)}{\|\boldsymbol{W}(t)\|_F} = \frac{\boldsymbol{W}_a}{\|\boldsymbol{W}_a\|_F} \tag{1.32}$$

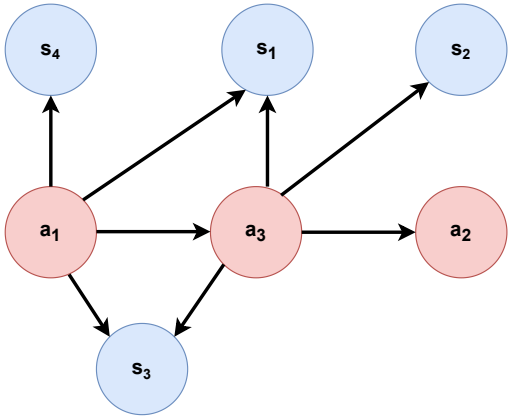
(b) When trained on both action and state prediction ($\mathcal{L}(\boldsymbol{W}) = \mathcal{L}_a(\boldsymbol{W}) + \mathcal{L}_s(\boldsymbol{W})$), the attention weights converge in direction to the solution of the Multi-Modality Graph-SVM problem:

$$\lim_{t\to\infty} \frac{\boldsymbol{W}(t)}{\|\boldsymbol{W}(t)\|_F} = \frac{\boldsymbol{W}_m}{\|\boldsymbol{W}_m\|_F} \tag{1.33}$$

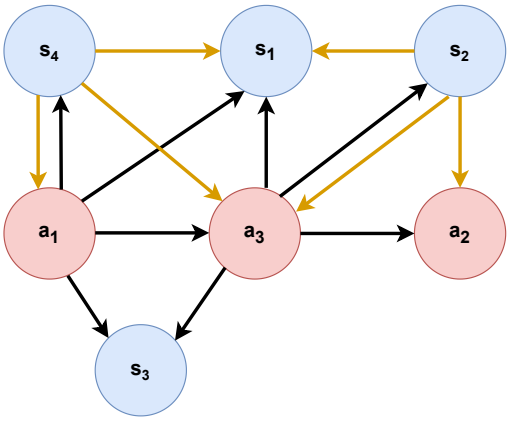
Proof. The proof follows by constructing the appropriate TPGs for each scenario and applying the convergence result from Theorem 2 in [33]. □



(a) Example of two sequences in offline reinforcement learning, showing the relationship between states and actions.



(b) Action-Only TPG ($\mathcal{G}_a$) for vanilla Decision Transformer, showing the strict hierarchy where action tokens dominate state tokens.



(c) Multi-Modality TPG ($\mathcal{G}_m$) for Multi-Objective Decision Transformer, where bi-directional paths between action and state tokens create cross-modal strongly connected components.

Figure 1.5: Comparison of Token-Priority Graphs for single-task and multi-task Decision Transformers. The different graph structures lead to different optimization problems that shape the attention mechanism in fundamentally different ways.

1.6.4 Implications for Decision Transformers

While our theoretical analysis is based on a simplified attention mechanism, it provides valuable insights into why multi-objective training might lead to more diverse attention patterns in practice. Our results suggest that even in a simplified scenario, the structure of the optimization problem fundamentally changes when moving from single-task to multi-task learning.

In the Action-Only setting, the Graph-SVM formulation creates a strict hierarchical relationship between token types, which may translate in practice to the homogeneous attention patterns we observe across different heads and layers in vanilla Decision Transformers.

The constraints force the optimization to prioritize one modality over others.

Conversely, in the Multi-Modality setting, the emergence of cross-modal strongly connected components allows for more balanced attention distribution. This mathematical insight aligns with our empirical observations that multi-objective Decision Transformers exhibit more diverse attention patterns across both heads and layers, with different components of the network potentially specializing in different aspects of the task.

It's important to note that our theoretical formulation omits many components present in practical DT implementations, such as layer normalization, MLP layers, separate query-key matrices [67, 68], dropout, and residual connections. Moreover, practical DTs employ multiple attention heads and layers with complex interactions. Therefore, we cannot claim that our analysis provides a complete explanation for the empirical diversity of attention patterns.

Nevertheless, the fundamental principle we've identified—that multi-task learning induces more complex attention relationships through different TPG structures—offers a plausible mechanism for how auxiliary objectives might encourage specialization among attention heads. The key insight is that different prediction tasks create different optimization geometries that shape the attention mechanism in fundamentally different ways. By intentionally incorporating auxiliary tasks, we can guide the inductive bias of the transformer toward more balanced, nuanced, and ultimately more effective attention patterns.

1.6.5 Numerical Validation

To validate our theoretical analysis, we conducted controlled experiments comparing attention patterns and convergence behavior of single-task (action prediction only) and multi-task Decision Transformers. These experiments used a simplified setup with vocabulary size $|\mathcal{V}| = |\mathcal{A}| + |\mathcal{S}| = 6$, embedding dimension $d_{\text{in}} = 8$, number of training samples $N = 6$, and sequence length $K = 4$.

For both models, we trained for 4000 iterations with step size $\eta = 0.01$. To accelerate convergence, we used normalized gradient descent:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) - \eta \frac{\nabla \mathcal{L}(\boldsymbol{W}(t))}{\|\nabla \mathcal{L}(\boldsymbol{W}(t))\|_F} \tag{1.34}$$

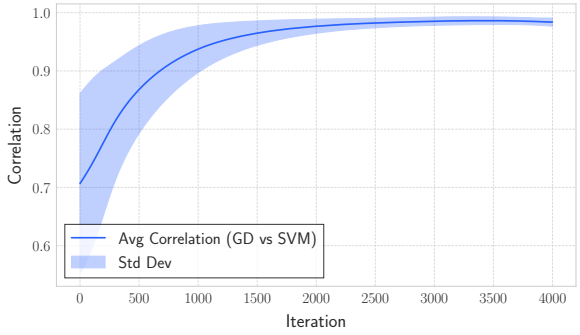
where $\mathcal{L}$ represents either $\mathcal{L}_a$ for the single-task model or $\mathcal{L}_{\text{multi}}$ for the multi-task model.

At each iteration, we measured the correlation between the current weights and the theoretical Graph-SVM solution:

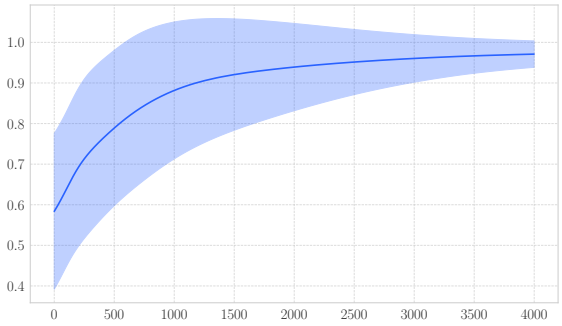
$$\text{Correlation}(t) = \frac{\langle \boldsymbol{W}(t), \boldsymbol{W}_a \rangle}{\|\boldsymbol{W}(t)\|_F \|\boldsymbol{W}_a\|_F} \tag{1.35}$$

for the single-task model, and analogously using $\boldsymbol{W}_m$ for the multi-task model. Results were averaged over 10 random initializations.

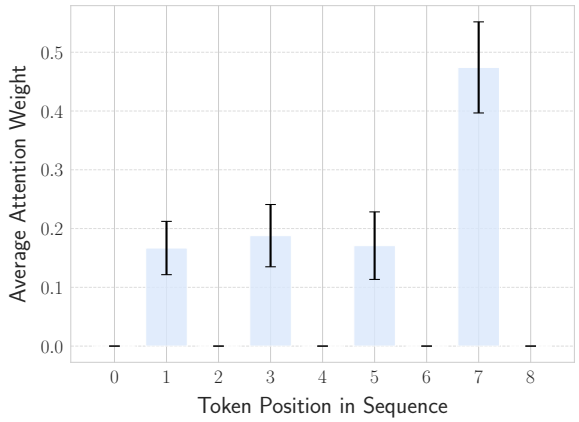
The single-task model's weights strongly correlated with $\boldsymbol{W}_a$ (reaching approximately 0.98), confirming our theoretical prediction of directional convergence. Its attention heatmap



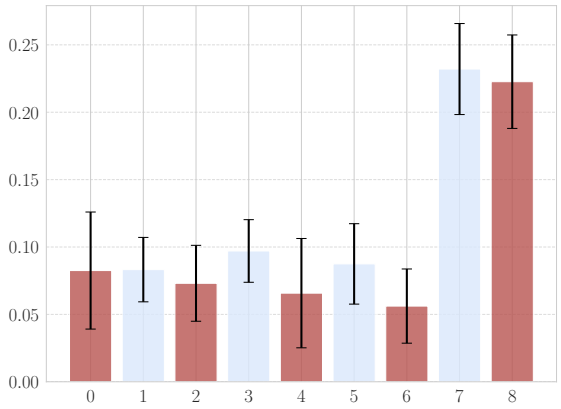
(a) Convergence of attention weights in Action-Only DT to the solution of the Action-Only Graph-SVM problem.



(b) Convergence of attention weights in Multi-Modality DT to the solution of the Multi-Modality Graph-SVM problem.



(c) Attention pattern for Action-Only DT, showing strong bias toward action tokens with negligible attention to state tokens.



(d) Attention pattern for Multi-Modality DT, showing more balanced attention across token types and more selective attention within each modality.

Figure 1.6: Empirical validation of our theoretical analysis. The left column shows results for the vanilla Decision Transformer (action prediction only), while the right column shows results for the Multi-Task Decision Transformer. The two different Graph-SVM problems lead to fundamentally different attention patterns. In the attention maps, even indices represent state tokens and odd indices represent action tokens.

(Figure 1.6c) shows the predicted priority pattern: action tokens receive substantial attention while state tokens receive negligible attention.

Similarly, the multi-task model's weights correlated with $\boldsymbol{W}_m$, and its attention heatmap (Figure 1.6d) reveals a more balanced distribution of attention across modalities. This confirms that multi-task learning fundamentally alters the Token-Priority Graph structure, allowing state tokens to receive meaningful attention weights.

These results demonstrate that the choice of prediction tasks shapes the optimization geometry, which in turn determines how attention is allocated across different token types in Decision Transformers.

1.7 Conclusion

We introduced Multi-Objective Decision Transformer (MO-DT), which enhances transformer effectiveness in offline RL by jointly optimizing state, action, and return prediction. This approach encourages diverse attention patterns across transformer heads, addressing a key limitation in vanilla Decision Transformers. We further extended this to Trust Region Decision Transformer (TRDT), incorporating action region prediction to reduce dependency on behavioral patterns in training data and improve generalization.

Our theoretical analysis through the Token-Priority Graph framework explains why single-task transformers develop homogeneous attention patterns, while multi-objective training enables balanced cross-modal attention. This analysis establishes that transformer implicit bias is significantly influenced by prediction tasks, with multi-task learning creating strongly connected components spanning different modalities in the token-priority structure.

Empirically, both MO-DT and TRDT outperform vanilla Decision Transformer across most D4RL locomotion benchmarks, with TRDT achieving state-of-the-art total performance. Notable improvements appear in challenging scenarios like medium-replay datasets, where our approaches demonstrate robust performance despite heterogeneous training data. Ablation studies validate our design choices, confirming benefits of Gaussian prediction heads for complex environments and multi-head attention for capturing diverse representation subspaces.

Future work could explore more sophisticated multi-objective optimization techniques beyond linear scalarization, extend our approach to discrete action domains and partially observable environments, and investigate pre-training on diverse datasets. The principles of multi-objective optimization and attention diversification introduced here could extend to other sequence modeling applications where transformers show limited attention diversity.

This work advances offline reinforcement learning while deepening our understanding of transformer properties in sequential decision-making tasks. By bridging theoretical insights with algorithmic improvements, we provide a foundation for future research leveraging transformer capabilities in reinforcement learning.

The content of this chapter is based on the paper "Multi-Objective Decision Transformers for Offline Reinforcement Learning", currently under review at IEEE Transactions on Neural Networks and Learning Systems (TNNLS).

Chapter 2

State Prediction for Offline Reinforcement Learning via Sequence-to-Sequence Modeling

2.1 Introduction

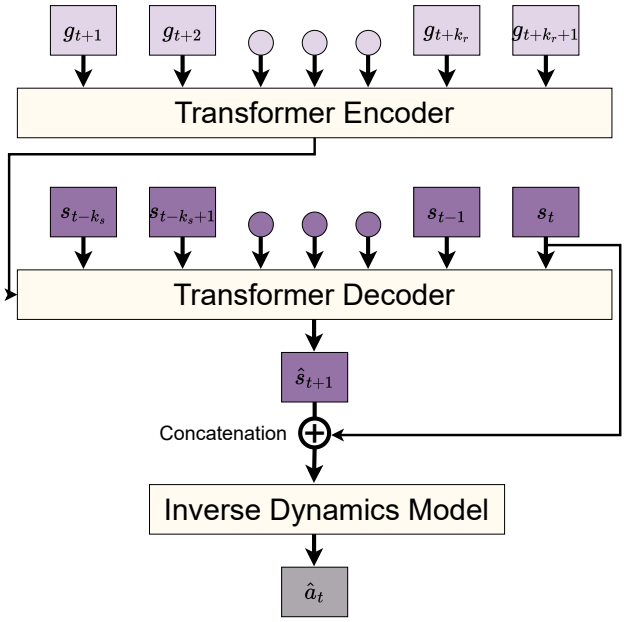


Figure 2.1: RGDT framework overview. Our sequence-to-sequence architecture maps future returns to past states via an encoder-decoder structure, with actions inferred through an inverse dynamics model. This design disentangles modality processing, separately handling vector states/actions and scalar returns.

Offline RL has emerged as a promising approach for deriving effective policies from static datasets, circumventing the necessity for online interactions with the environment. This paradigm is particularly advantageous in domains where real-world exploration is expensive, hazardous, or impractical, such as robotics, healthcare, and finance [17].

Offline RL datasets typically comprise sequences of three distinct modalities: encountered states, actions executed by a single or a mixture of policies, which may not necessarily be optimal, and scalar reward information, which can be in the form of Q-values or the sum of future rewards per timestep. This sequential nature of offline RL data makes it an ideal candidate for the application of sequence modeling techniques [7]. Recent work in the signal processing community has embraced this perspective, with [69] proposing an offline RL method based on next state supervision, [70] introducing uncertainty estimation with generative adversarial networks, and Wu et al. introducing pre-trained policy guidance in offline learning. These methods harness the capabilities of transformer models, particularly the decoder component, to directly learn policies from trajectories [38].

Transformers, initially designed for natural language processing to handle discrete input tokens, have been adapted for offline RL either by discretization-based tokenization of input features from each modality [8], or by a higher-level tokenization that considers only three tokens: states, actions, and returns [7]. This approach has parallels in signal processing, where Ohta et al. designed a sequential Audio Spectrogram Transformer with a memory token for real-time sound event detection, and Seraphim et al. developed structure-preserving Transformers for sequences of symmetric positive definite matrices in EEG classification.

The primary advantage of discretization is the flexibility it affords the model to learn distinct statistical properties of each feature, as each discretized value is associated with its own parameters [38]. However, this approach faces significant scalability challenges with even relatively small datasets and dimensions [8]. The higher-level modality-based tokenization offers greater scalability and simplicity, but with all transformer layers beyond the embedding layer shared across modalities, it may not adequately capture the unique characteristics of each data type. For instance, in continuous control tasks, action sequences often represent joint torques with non-smooth trajectories [71], while state sequences are governed by the environment's dynamics and exhibit smoother patterns [72].

This disparity explains why discretization, despite being modality-agnostic, often yields superior performance. The advantage of disentangling transformer components per modality has been explored extensively in signal processing applications. [73] introduced RESTAD, a Transformer-based model for time-series anomaly detection that effectively separates the processing of normal and anomalous signal patterns, while [74] employed contrastive representation learning on wireless channel-state sequences for human-orientation detection. [75] proposed "Speed," a scalable preprocessing pipeline for EEG data enabling self-supervised sequence learning, demonstrating how proper signal-domain preprocessing aids downstream representation learning. Specifically, in offline RL, there is work that explores the effect of disentangling transformer components per modality [76]. This approach hard-codes the importance of each modality into the design of a multimodal architecture after attention analysis of the decision transformer, using multiple small transformers. However, instead of training a model to discover the importance and leverage it with a mixture

of models, an automated approach within a single model would be more desirable.

In this work, we develop a modality-aware sequence-based approach for offline RL that exhibits both the flexibility of disentangled parameters across different modalities and the scalability of high-level modality-specific tokenization within a single model. We adopt a sequence-to-sequence modeling framework where the model translates from one modality to another—specifically, from sequences of future returns to sequences of past states. This approach aligns with successful encoder-decoder architectures in signal processing: [77] designed CNN-Transformer architectures for audio captioning that map audio features to text descriptions, [78] presented YourMT3+, which transcribes polyphonic audio into multiple instrument tracks, and [79] proposed parameter-efficient transfer learning for Audio Spectrogram Transformers.

Our sequence-to-sequence modeling leverages the vanilla transformer architecture with an encoder-decoder structure [6], ensuring each modality has its own transformer components (see Figure 2.1). This approach is conceptually similar to [80]'s LMCodec, which uses a causal Transformer for neural speech coding with coarse/fine token hierarchies. With our formulation, each modality has disentangled weights while still using the high-level modality tokenization employed by decision transformers. The idea of translating from one modality to another using an encoder-decoder architecture is widely common in the literature and forms the cornerstone for fields like speech-to-text, text-to-speech, and image-to-text translation. Finally, actions are predicted using an inverse dynamics model, a design choice we explain in subsequent sections. Our experimental results demonstrate that this approach substantially outperforms decoder-based counterparts and matches or surpasses state-of-the-art off-policy methods across a range of continuous control tasks.

2.2 Preliminaries

Following the notation established in Section 1.3, we adopt the MDP framework to model our environment, denoted by $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ (as defined in Section 1.3), where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces, respectively. The probability distribution over transitions is given by $P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$, while the reward function is defined as $R(\boldsymbol{s}_t, \boldsymbol{a}_t)$. The discount factor, γ, is used to weigh the importance of future rewards. The agent starts in an initial state $\boldsymbol{s}_1$ sampled from the fixed distribution $p(\boldsymbol{s}_1)$ and chooses an action $\boldsymbol{a}_t \in \mathcal{A}$ from the state $\boldsymbol{s}_t \in \mathcal{S}$ at each timestep t. The agent then transitions to a new state $\boldsymbol{s}_{t+1}$ according to the probability distribution $P(\cdot \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$. After each action, the agent receives a deterministic reward $r_t = R(\boldsymbol{s}_t, \boldsymbol{a}_t)$.

2.2.1 Setup and Notation

Throughout this paper, we maintain the notation conventions established in our notation chapter: bold uppercase letters (e.g., $\boldsymbol{X}$, $\boldsymbol{W}$) denote matrices, bold lowercase letters (e.g., $\boldsymbol{s}$, $\boldsymbol{a}$) denote vectors, and regular letters (e.g., r, g) represent scalars. This distinction is particularly important as we deal with various modalities of different dimensionalities in

our sequence-to-sequence formulation.

As in Section 1.3, our goal is to model the offline RL problem as a sequence modeling problem, with the agent having access to a fixed-size static training dataset $\mathcal{T}$. We use $\boldsymbol{\tau}$ to represent a trajectory and $|\boldsymbol{\tau}|$ to denote its length. The return-to-go (which we continue to refer to as 'return' for brevity, consistent with Section 1.3) of a trajectory $\boldsymbol{\tau}$ at timestep t is defined as the sum of future rewards starting from that timestep, i.e., $g_t = \sum_{t'=t}^{|\boldsymbol{\tau}|} r_{t'}$. We use G_t to represent the discounted returns which are computed as $G_t = \sum_{t'=t}^{|\boldsymbol{\tau}|-1} \gamma^{t'-t} r_{t'}$. We use m_s and m_a to denote the dimensionality of the states and actions respectively.

Remark 3 (Sequence Notation). For consistency with Section 1.3, we denote sequences of states, actions, and returns for trajectory $\boldsymbol{\tau}$ as $\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{|\boldsymbol{\tau}|})$, $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{|\boldsymbol{\tau}|})$, and $\boldsymbol{g} = (g_1, \ldots, g_{|\boldsymbol{\tau}|})$, respectively. Note that we use bold uppercase for sequences of vectors and lowercase for the sequence of scalars (returns).

To represent segments of trajectories, we define $\boldsymbol{\tau}_{t_1}^{t_2}$ as a segment of the trajectory $\boldsymbol{\tau}$ from timestep t_1 to t_2. The sequences of states and returns in the segment $\boldsymbol{\tau}_{t_1}^{t_2}$ are denoted by $\boldsymbol{S}_{t_1}^{t_2} = (\boldsymbol{s}_{t_1}, \ldots, \boldsymbol{s}_{t_2})$ and $\boldsymbol{g}_{t_1}^{t_2} = (g_{t_1}, \ldots, g_{t_2})$, respectively.

2.2.2 Sequence Modeling for Offline RL

Recall from Section 1.3 that sequence modeling approaches in offline RL treat trajectory data as sequences suitable for transformer architectures. We previously saw two trajectory representations in Equations 1.1 and their component-level variants. Here, we present these representations again for completeness:

$$\boldsymbol{\tau}_{DT} := (g_1, \boldsymbol{s}_1, \boldsymbol{a}_1, g_2, \boldsymbol{s}_2, \boldsymbol{a}_2, \ldots, g_{|\boldsymbol{\tau}|}, \boldsymbol{s}_{|\boldsymbol{\tau}|}, \boldsymbol{a}_{|\boldsymbol{\tau}|}), \tag{2.1}$$

$$\boldsymbol{\tau}_{TT} := \{\boldsymbol{s}_t^1, \boldsymbol{s}_t^2, \ldots, \boldsymbol{s}_t^{m_s}, \boldsymbol{a}_t^1, \boldsymbol{a}_t^2, \ldots, \boldsymbol{a}_t^{m_a}, r_t, G_t\}_{t=1}^{|\boldsymbol{\tau}|} \tag{2.2}$$

In this context, the subscripts on all tokens denote the timestep, while the superscripts on states and actions signify dimensions. The core objective is to develop a model capable of predicting action distributions. This can either be conditioned on desired returns and preceding states, a strategy employed in DT [7] (as discussed in Section 1.4), which typically aligns with the trajectory representation in (2.1); or it can entail crafting a distribution of actions based on previous states, accompanied by the utilization of a discounted return guided beam search for selecting actions, a methodology followed by TT[8] and commonly corresponding to the trajectory configuration outlined in (2.2). Despite the distinct technical variations between these approaches (e.g., the implementation of quantization in [8] as opposed to [7]), both avenues employ a decoder-only architecture and fundamentally aim to employ the transformer architecture as a tool for policy modeling.

2.3 Offline RL as a Sequence-to-Sequence Modeling Problem

Remark 4 (Key Distinction from Previous Approaches). While the methods discussed in Section 1.4 employ decoder-only architectures (similar to GPT), we introduce a sequence-to-sequence approach using an encoder-decoder architecture. This key distinction allows us to disentangle the processing of different modalities.

In this work, our goal is to disentangle the processing of different modalities in transformer-based approaches for offline RL by formulating the problem as a sequence-to-sequence modeling task. Given a dataset $\mathcal{T}$ of trajectories, where each trajectory $\boldsymbol{\tau}$ is composed of sequences of states, actions, and returns-to-go tuples $\{\boldsymbol{S}, \boldsymbol{A}, g\}$, we model the relationship:

$$\text{Model: } \boldsymbol{g}_{t+1}^{t+K_r+1} \mapsto \boldsymbol{S}_{t-K_s}^{t}, \quad \forall t > 0 \tag{2.3}$$

This formulation captures our sequence-to-sequence approach, where we learn to map sequences of future returns $\boldsymbol{g}_{t+1}^{t+K_r+1}$ to sequences of historical states $\boldsymbol{S}_{t-K_s}^{t}$. Here, $\boldsymbol{S}_{t-K_s}^{t}$ denotes states from timestep $t-K_s$ to t, and $\boldsymbol{g}_{t+1}^{t+K_r+1}$ denotes returns from timestep $t+1$ to $t+K_r+1$. For simplicity, we introduce the following shorthand notation that will be used throughout this chapter:

$$\boldsymbol{S}_{-K_s} := \boldsymbol{S}_{t-K_s}^{t} \quad \text{(past } K_s \text{ states up to time } t) \tag{2.4}$$

$$\boldsymbol{g}_{+K_r} := \boldsymbol{g}_{t+1}^{t+K_r+1} \quad \text{(future } K_r \text{ returns from time } t+1) \tag{2.5}$$

where the subscript t is implicit and determined by context.

This sequence-to-sequence formulation offers several advantages. Firstly, it enables mapping segments from different timesteps, allowing direct conditioning on a sequence of upcoming returns rather than a single scalar sum. Our experiments indicate that this direct conditioning of state generation enhances performance. We attribute this improvement to the richer, more detailed representation of future behavior provided by the sequence, which facilitates more effective credit assignment.

Additionally, this approach permits the use of different context lengths for returns and states. Considering a larger context length for returns, K_r, helps avoid short-sightedness in policy decisions, similar to the role of the discount factor in value-based RL algorithms. This extended context allows the model to capture long-term dependencies more effectively. In contrast, the context length for states, K_s, can be smaller due to the Markov property of the environment, which implies that fewer past states are needed for next state prediction.

2.3.1 Model Architecture

An encoder-decoder architecture [6] provides a robust framework for addressing sequence-to-sequence modeling problems. In our setting, the source segments of returns, $\boldsymbol{g}_{+K_r}$, are processed by an encoder model, which consists of a stack of identical layers. Each layer has

two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. The target sequence, $\boldsymbol{S}_{-K_s}$, is processed by a decoder model, which also comprises a stack of layers. Notably, the number of layers in the decoder does not necessarily have to match the number of layers in the encoder.

Each decoder layer includes its own masked multi-head self-attention mechanism and position-wise fully connected feed-forward network. Additionally, the decoder introduces a third sub-layer that performs multi-head attention over the encoder's output. This architecture allows for separate processing of each token modality with distinct sets of weights and the use of networks with varying depths. In contrast, the GPT architecture [38], used in prior sequence modeling approaches to offline RL [7, 8] (as discussed in Section 1.4), which only employs a variant of the decoder part of the transformer model, lacks this level of flexibility.

2.3.2 Training the Sequence-to-Sequence Decision Translator

In a sequence-to-sequence framework, the mapping from the source space to the target space is learned by performing next token prediction in the target space. In our case, the mapping from $\boldsymbol{g}_{+K_r}$ to $\boldsymbol{S}_{-K_s}$ is learned by performing next state prediction. To this end, we introduce a probabilistic function $\rho_{\boldsymbol{\theta}_{RGDT}}$ which models the distribution of states. This function is parameterized by the transformer model with both its components—the encoder and decoder—which we denote with weights $\boldsymbol{\theta}_{RGDT}$, and it assigns probability density at the t-th time step, conditioned on the future K_r instances of returns and K_s historical state instances.

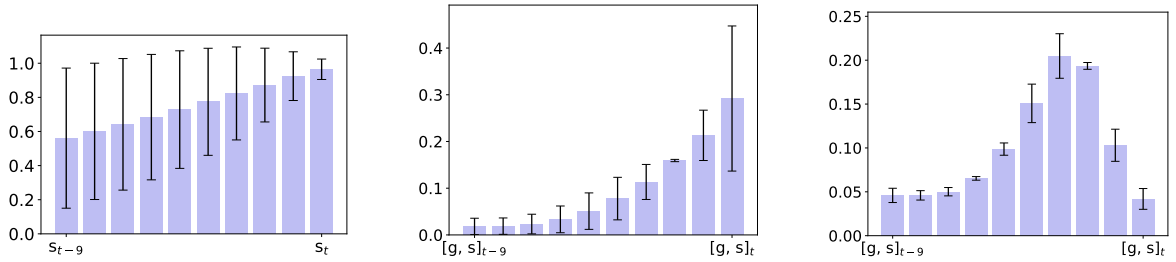
More precisely, the state distribution function $\rho_{\boldsymbol{\theta}_{RGDT}}$ is modeled using a multivariate Gaussian distribution parameterized by the transformer model which outputs its mean $\boldsymbol{\mu}_{\boldsymbol{\theta}_{RGDT}}$ and a diagonal covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}_{RGDT}}$, i.e.,

$$\rho_{\boldsymbol{\theta}_{RGDT}}(\boldsymbol{s}_{t+1} \mid \boldsymbol{S}_{-K_s}, \boldsymbol{g}_{+K_r}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}_{RGDT}}(\boldsymbol{S}_{-K_s}, \boldsymbol{g}_{+K_r}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}_{RGDT}}(\boldsymbol{S}_{-K_s}, \boldsymbol{g}_{+K_r})), \quad \forall t > 1 \tag{2.6}$$

Subsequently, $\rho_{\boldsymbol{\theta}_{RGDT}}$ can be learned by minimizing the negative log-likelihood (NLL) loss, which can be expressed as follows:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}_{RGDT}) = -\mathbb{E}_{\boldsymbol{S}^{t+1}_{t-K_s}, \boldsymbol{g}_{+K_r} \sim \mathcal{T}} \Big[& \log \rho_{\boldsymbol{\theta}_{RGDT}}(\boldsymbol{s}_{t+1} \mid \boldsymbol{S}_{-K_s}, \boldsymbol{g}_{+K_r}) + \\ & \sum_{i=t-K_s}^{t-1} \mathbf{1}_{\{i+1 \leq K_s\}} \log \rho_{\boldsymbol{\theta}_{RGDT}}(\boldsymbol{s}_{i+1} \mid \boldsymbol{S}^{i}_{i-K_s}, \boldsymbol{g}_{+K_r}) \Big] \end{aligned} \tag{2.7}$$

By setting $\mathbf{1}_{\{i+1 \leq K_s\}}$ to 1, we recover the training regime commonly employed in sequence-to-sequence approaches, where the model is trained to predict other tokens in the context, effectively varying the context length from 1 to K_s. This strategy of training with varying context lengths, even when the full context is available, is widely used in sequence prediction tasks to expose the model to diverse scenarios, potentially promoting robustness, adaptability, and generalization to unseen sequences.



(a) Linear correlation in the dataset

(b) Full context prediction

(c) Final token prediction

Figure 2.2: (a) Pearson correlation between each state and its subsequent states in Walker2d-Medium, decreasing with timestep offset. (b) Attention weights for GPT model with full context token prediction. (c) Attention weights with final token prediction only. Full context prediction (b) biases attention toward linear state correlations observed in (a), while final token prediction (c) shows more balanced attention distribution.

However, in our case, since subsequent states are highly correlated, as can be seen in Figure 2.2, this approach may lead the model to overfit easily to these correlations by assigning high attention to nearby states, potentially neglecting more distant tokens that could be valuable for the ultimate objective of return maximization. To address this issue, we propose a simple adaptation: setting $\mathbf{1}_{\{i+1\leq K_s\}}$ to 0. As Figure 2.2 shows, this final token prediction approach makes the model less biased toward the linear correlations between subsequent states in the dataset.

2.3.3 Policy Extraction and Inference

Predicting states using the transformer model is insufficient for defining a controller. Nevertheless, a policy can be inferred by estimating the action $\boldsymbol{a}_t$ that caused the transition from state $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$ at any timestep t in $\boldsymbol{\tau}$. Given two consecutive states, we generate an action according to the inverse dynamics model [81, 82], which we model as a conditional multivariate Gaussian distribution, as follows:

$$\pi_{\phi_{RGDT}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+1}) = \mathcal{N}(\boldsymbol{\mu}_{\phi_{RGDT}}(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}), \boldsymbol{\Sigma}_{\phi_{RGDT}}(\boldsymbol{s}_t, \boldsymbol{s}_{t+1})) \tag{2.8}$$

Note that the same offline data utilized to train the state predictor model $\rho_{\boldsymbol{\theta}_{RGDT}}$ can also be employed to learn $\pi_{\phi_{RGDT}}$. Furthermore, we employ the same framework as used for the transformer model, utilizing maximum likelihood estimation to learn the parameters $\boldsymbol{\phi}_{RGDT}$.

During inference, a desired sequence of returns $\boldsymbol{g}_2^{K_r+1}$ and an initial state $\boldsymbol{s}_1$ are specified. In practice, to filter for good behavior during inference, the whole sequence of returns g is constructed by taking the best return per-timestep from the dataset of trajectories $\mathcal{T}$. The RGDT model then generates the first state $\hat{\boldsymbol{s}}_2 = \boldsymbol{\mu}_{\boldsymbol{\theta}_{RGDT}}(\boldsymbol{s}_1, \boldsymbol{g}_2^{K_r+2})$. Then, both $\boldsymbol{s}_1$ and $\hat{\boldsymbol{s}}_2$ are fed to the inverse dynamics model $\pi_{\phi_{RGDT}}$ which generates a deterministic action according to $\boldsymbol{a}_1 = \boldsymbol{\mu}_{\phi_{RGDT}}(\boldsymbol{s}_1, \hat{\boldsymbol{s}}_2)$. After executing the action $\boldsymbol{a}_1$, the agent observes

Algorithm 2 Inference with RGDT

Input: $\boldsymbol{g}_{+K_r}, \boldsymbol{S}_{-K_s}$
Output: $\hat{\boldsymbol{a}}_t$
procedure INFERENCE($\boldsymbol{g}_{+K_r}, \boldsymbol{S}_{-K_s}$)
    Initialize $t \leftarrow 1$
    while not done **do**
        $\hat{\boldsymbol{s}}_{t+1} \leftarrow \boldsymbol{\mu}_{\boldsymbol{\theta}_{RGDT}}(\boldsymbol{S}_{-K_s}, \boldsymbol{g}_{+K_r})$
        $\hat{\boldsymbol{a}}_t \leftarrow \boldsymbol{\mu}_{\phi_{RGDT}}(\boldsymbol{s}_t, \hat{\boldsymbol{s}}_{t+1})$
        Execute $\hat{\boldsymbol{a}}_t$
        Observe $\boldsymbol{s}_{t+1} \sim P(\cdot \mid \boldsymbol{s}_t, \hat{\boldsymbol{a}}_t)$
        $t \leftarrow t + 1$
    end while
end procedure

the next state $\boldsymbol{s}_2$ from the transition probability distribution $P(\cdot \mid \boldsymbol{s}_1, \boldsymbol{a}_1)$. Subsequently, the RGDT generates the next goal state $\hat{\boldsymbol{s}}_3$ based on $\boldsymbol{g}_3^{K_r+3}, \boldsymbol{s}_1, \boldsymbol{s}_2$ as inputs. This process continues until the episode terminates.

2.4 Experiments

We evaluate RGDT against leading offline RL approaches on the D4RL benchmark [32], comparing performance with both sequence modeling methods and off-policy algorithms. Our experiments focus on three aspects: (1) overall performance across continuous control tasks, (2) benefits of final token prediction over full context prediction, and (3) advantages of our sequence-to-sequence formulation.

Implementation Details and Hyperparameters. Our RGDT implementation utilizes a transformer with hidden dimension 64 and 2 attention heads. The encoder and decoder comprise 2 and 3 layers, respectively, reflecting the asymmetric nature of our sequence-to-sequence formulation. We employed context lengths of $K_r = 20$ for returns and $K_s = 5$ for states, enabling the model to consider extended future horizons while maintaining computational efficiency. For optimization, we used the AdamW optimizer with a learning rate of 5×10^{-4}, linear warmup over 10,000 iterations, weight decay of 10^{-3}, and dropout probability of 0.1. Training continued for 150,000 gradient steps with a batch size of 256. The inverse dynamics model consists of a neural network with 2 hidden layers of 512 units each, trained using the Adam optimizer with learning rate 10^{-4}, weight decay of 10^{-4}, and dropout of 0.1. This model was trained for 200,000 gradient steps with a batch size of 1024.

Benchmark and Compared Baselines. We evaluate RGDT on the D4RL benchmark [32] using the Gym environment [64], focusing on six continuous control tasks: halfcheetah-medium (HC-M), hopper-medium (HP-M), walker2d-medium (WK-M) and their medium-expert (HC-ME, HP-ME, WK-ME) variants. We compare against four established

Table 2.1: Normalized scores on Gym D4RL tasks (10 seeds). Our method **RGDT** achieves the best performance.

Dataset	10% BC	CQL	IQL	DT	RGDT
HC-M	42.5	44.0	**47.4**	42.6	44.4
HP-M	56.9	58.5	66.3	67.6	**93.2**
WK-M	75.0	72.5	78.3	74.0	**83.3**
HC-ME	92.9	91.6	86.7	86.8	**93.4**
HP-ME	**110.9**	105.4	91.5	107.6	**112.4**
WK-ME	109.0	108.8	109.6	108.1	**113.3**
Total	487.2	480.8	479.8	486.7	**539.0**

baselines: 10% BC [7], which replicates the top decile of behaviors in the dataset; and two leading dynamic programming methods - Conservative Q-Learning (CQL) [4] and Implicit Q-Learning (IQL) [65], as well as DT [7] (discussed in Section 1.4), the primary sequence modeling approach. All baseline results are sourced from their respective papers with CQL results adapted from [65] for the v2 datasets.

Full Context Token Prediction vs. Final Token Prediction. An important aspect of our method is the training regime of final token prediction as implied by equation 2.7. As demonstrated in Figure 2.3, this approach leads to significantly better performance compared to full context token prediction across the D4RL medium tasks.

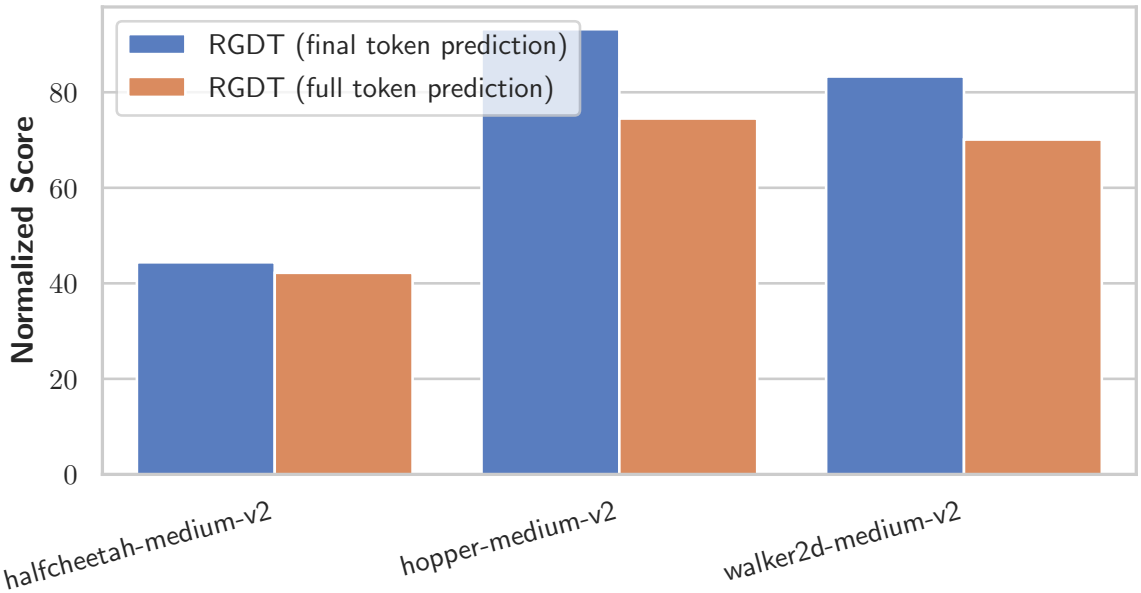


Figure 2.3: Comparison of RGDT training strategies on Gym D4RL medium tasks.

2.5 Conclusion

We introduced RGDT, a sequence-to-sequence approach for offline RL that disentangles modality processing through an encoder-decoder architecture. By translating future returns to state sequences and using an inverse dynamics model for action inference, our method effectively addresses the limitations of decoder-only architectures (discussed in Section 1.4) and achieves superior performance on the D4RL benchmark.

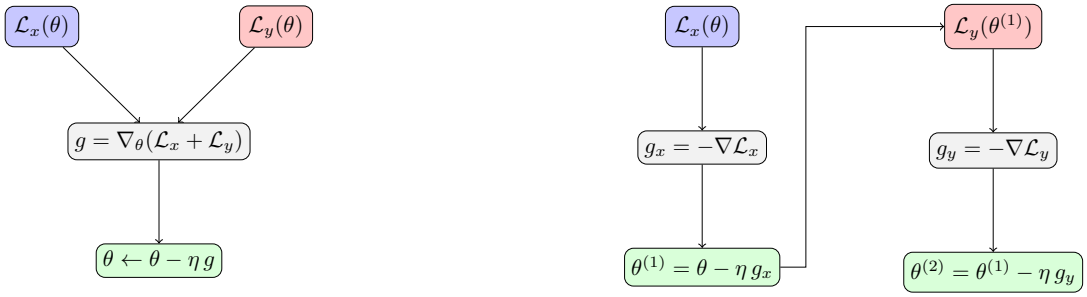
Future work could explore: (1) adapting RGDT to visual observations and complex state spaces, (2) investigating transfer learning across tasks with similar dynamics but different return structures. The sequence-to-sequence formulation presented here offers a promising foundation for addressing the multimodal nature of sequential decision-making while maintaining computational efficiency.

The content of this chapter is based on the paper "State Prediction for Offline Reinforcement Learning via Sequence-to-Sequence Modeling", accepted for publication at IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2025).

Chapter 3

When Tasks Collide: A Gradient-Flow Analysis of Alternating and Joint Training in Transformer Models

3.1 Introduction



(a) Simultaneous step

(b) Alternating step

Figure 3.1: **Two paradigms of multi-task optimisation.** *(a)* Simultaneous GD aggregates both gradients before stepping. *(b)* Sequential GD applies task x first, producing $\theta^{(1)}$ that feeds task y; this shift is the source of the Hessian-driven correction analysed in §3.4. Colours match task identities.

Multi-Task Learning (MTL) jointly trains a single model on several related objectives, encouraging the emergence of shared representations that transfer information across tasks [83]. This paradigm leverages the observation that many practical problems—from computer vision [84, 85, 86, 87, 88] and language understanding [89, 90] to bioinformatics [91, 92] and speech recognition [93, 94]—possess latent commonalities that can be exploited to improve sample efficiency, reduce model footprint, and boost generalisation performance compared to training one network per task [83, 95, 96]. As such, MTL has become a cornerstone of modern deep learning and an indispensable component of large-scale foundation models.

Despite its strong empirical record, we still lack a principled understanding of *why* MTL often outperforms its single-task counterpart. It is particularly unclear how the implicit regularisation induced by gradient-based optimisers in the multi-task setting differs from that in the classical single-task regime. Does sharing parameters fundamentally alter the bias that gradient descent (GD) imposes on the learned solution, and can this difference explain the superior generalisation observed in practice?

To address this gap, we develop a rigorous theoretical framework using the *Modified Gradient Flow* (MGF) [97], which reveals the implicit biases of discrete optimisation algorithms. Through this lens, we uncover a surprising phenomenon in the standard MTL setting: **joint gradient descent implicitly encourages task disagreement by minimizing the inner product between task gradients.** This finding extends the implicit regularization literature [98, 99, 97, 100, 101] by explicitly characterizing how multiple loss functions interact to shape the optimisation landscape.

This initial insight, however, pertains to the most common MTL optimisation scheme: minimising a weighted sum of task losses in a single step. Yet, a rich spectrum of alternative *training schedules* exists, including sequential, alternating, or bandit-style task sampling [102]. This raises a crucial follow-up question: how does the choice of schedule—conceptually illustrated in Figure 3.1—modify the optimiser's implicit bias? Understanding this is key to explaining why joint training has become the de facto standard and identifying when alternatives might be preferable. We extend our analysis to alternating updates—a common alternative—and show that they introduce a second-order Hessian correction absent from joint training, a finding that theoretically explains the empirical instability often observed with this method.

The practical implications of this schedule-dependent bias are particularly salient in offline RL, where the choice between joint and alternating updates is a central design decision. For instance, the TT [21] uses an alternating schedule to predict returns, states, and actions, while the MO-DT [103] opts for joint, multi-objective updates. Our framework provides the ideal tools to dissect this choice. By specialising our general theory to a one-layer self-attention model, we derive closed-form expressions for the implicit regulariser, interpreting its geometric action on the attention weights and providing theoretical insights that may help explain performance differences between these architectures.

Finally, we conduct extensive numerical experiments that confirm our theoretical predictions across these different settings. The results provide a theoretical perspective on the different optimization dynamics of joint versus alternating training, which may contribute to the empirical performance differences observed between MO-DT and TT. Taken together, our work provides the first rigorous account of how the optimisation schedule shapes the inductive bias of MTL, offering both fundamental insights and practical guidance for designing the next generation of scalable multi-task systems.

In summary, this work makes the following key contributions:

1. **Implicit Regulariser for Joint MTL:** We derive the exact modified objective for joint multi-task training, revealing a novel regulariser that encourages gradient anti-alignment between tasks, thus providing a theoretical basis for MTL's generalisation

benefits (Section 3.4).

2. **Analysis of Alternating Schedules:** We extend the MGF framework to alternating/sequential training, uncovering a second-order Hessian correction term that explains the method's potential instability and highlights its fundamental difference from joint training (Section 3.4).

3. **Theory for Multi-Task Transformers:** We provide the first theoretical analysis of implicit bias in multi-task self-attention models, deriving closed-form expressions that characterize how tasks compete via attention weight redistribution (Section 3.5).

4. **Practical Design Principles & Empirical Validation:** We establish precise geometric criteria linking learning rates and loss curvature to task competition, offering actionable guidance for MTL design. We validate all theoretical predictions with controlled experiments that accurately forecast optimisation dynamics (Sections 3.6 and 3.7).

The remainder of this paper is organized as follows. Section 3.2 reviews related work on multi-task learning, implicit regularization, and transformer architectures. Section Section 3.3 sets up the required preliminaries. Section 3.4 develops our model-agnostic theoretical framework, deriving the modified gradient flow for both multi-objective and sequential training regimes. Section 3.5 specializes these results to self-attention mechanisms, providing closed-form expressions for gradient dynamics in simplified transformer models. Section 3.6 presents our geometric analysis of task competition, establishing precise conditions for cooperation versus competition. Section 3.7 validates our theoretical predictions through controlled experiments. Finally, Section 3.9 concludes with a discussion of implications and future directions.

3.2 Related Work

To situate our work within the existing literature, this section reviews the three key research domains that form the backdrop for our contributions. We begin with the foundations of Multi-Task Learning and the gradient-based methods developed to address its challenges. We then survey the literature on implicit regularization, the primary theoretical tool we employ. Finally, we discuss the recent progress in understanding the training dynamics of transformer architectures, the specific context in which we apply and validate our general theory.

Multi-Task Learning and Gradient-Based Optimization. MTL is founded on the principle that joint training on related tasks can improve generalization by promoting shared representations [83, 104]. However, this benefit is not guaranteed; a persistent challenge is *negative transfer*, where multi-task performance degrades below single-task baselines [96]. The root cause is often attributed to destructive interference between task objectives, manifesting as *conflicting gradients* during optimization [102, 105]. Consequently, a significant

body of research has focused on developing algorithms to manage these conflicts. Prominent methods include projecting conflicting gradients (PCGrad [106]), dynamically balancing gradient magnitudes (GradNorm [107]), finding conflict-averse update directions (CAGrad [108]), computing Pareto-optimal updates (MGDA [60]), and framing MTL as a bargaining game (Nash-MTL [109]). While empirically successful, these methods are largely algorithmic interventions that treat the symptoms of gradient conflict. They do not provide a first-principles explanation for how and why standard gradient descent induces task competition in the first place—a fundamental gap our modified gradient flow analysis directly addresses.

Implicit Regularization and Modified Gradient Flow. Our theoretical approach is grounded in the literature on *implicit regularization*, which studies the biases that optimization algorithms impose on the learned solution. For classical deep networks, it is well-established that gradient descent on regression tasks implicitly minimizes parameter norms [110, 111, 112, 113, 114, 115, 116, 117, 118, 119], while for classification with exponential-tailed losses, it promotes margin maximization [120, 121, 122]. These insights are often derived using tools like the MGF, which shows that a discrete-time optimizer like GD can be seen as optimizing a modified continuous-time objective that includes additional regularization terms [97]. Our work extends this powerful framework in two key directions: first, by characterizing the regularizer that arises from the interaction of *multiple* loss functions, and second, by analyzing non-homogeneous models like transformers, for which many prior analyses on homogeneous networks do not directly apply [123, 124].

Training Dynamics of Transformer Architectures. The remarkable success of transformers has spurred a new wave of theoretical work aimed at understanding their training dynamics [125, 126, 127, 128, 129, 130, 131, 132, 133]. This research has shown how transformers learn data structure [125, 127] and has begun to characterize their generalization properties [126]. The line of work most relevant to ours uses implicit bias analysis to connect the attention mechanism to max-margin separation, akin to a Support Vector Machine (SVM) operating on tokens [134, 135, 68]. Recent studies have also investigated the dynamics of next-token prediction [136, 33, 137]. However, this entire body of work has predominantly focused on single-task objectives. Our research provides a critical extension by being the first to apply the MGF framework to a *multi-task transformer*. We thereby bridge the gap between the MTL gradient-conflict literature and the implicit bias analysis of self-attention, providing a precise mechanism for how tasks compete through the redistribution of attention weights.

3.3 Preliminaries

In this section, we establish the notation, formally define the multi-task learning problem, introduce the Modified Gradient Flow framework that serves as our primary analytical tool, and state the central questions this work addresses.

3.3.1 Notation and Multi-Task Learning Setup

We consider a model parameterized by a vector $\boldsymbol{\theta} \in \mathbb{R}^d$. In a hard-parameter-sharing Multi-Task Learning (MTL) setting, the model is trained to perform two tasks, denoted by x and y. Each task has an associated loss function, $\mathcal{L}_x(\boldsymbol{\theta})$ and $\mathcal{L}_y(\boldsymbol{\theta})$, which are both functions of the shared parameters $\boldsymbol{\theta}$. We assume the losses are twice continuously differentiable. We analyze two primary training regimes for minimizing these losses.

1. Joint (Multi-Objective) Training. In this regime, the parameters are updated by performing gradient descent on the summed loss, $\mathcal{L}_{\text{multi}}(\boldsymbol{\theta}) = \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{L}_y(\boldsymbol{\theta})$. A single gradient descent step with learning rate $\eta > 0$ is given by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{multi}}(\boldsymbol{\theta}_t) = \boldsymbol{\theta}_t - \eta \left(\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}_t) + \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}_t) \right). \tag{3.1}$$

2. Sequential (Alternating) Training. In this regime, the tasks are optimized in a fixed sequence within each update step. An update first performs a step for task x, then uses the resulting parameters to perform a step for task y:

$$\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}_t), \tag{3.2}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}^{(1)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}^{(1)}). \tag{3.3}$$

Remark 5 (Connection to Chapter 1). The general multi-task framework presented here encompasses the specific setting of Chapter 1's Multi-Objective Decision Transformer. In that context, the abstract tasks x and y correspond to concrete prediction objectives in offline RL: task x might represent state prediction (with loss $\mathcal{L}_s$), task y might represent action prediction (with loss $\mathcal{L}_{DT}$), and additional tasks could include return prediction (with loss $\mathcal{L}_g$). The general parameter vector $\boldsymbol{\theta}$ here unifies the task-specific parameters $\{\boldsymbol{\theta}_{DT}, \boldsymbol{\theta}_s, \boldsymbol{\theta}_g\}$ from Chapter 1.

3.3.2 The Modified Gradient Flow Framework

Our analysis relies on the MGF framework [97], which uses backward error analysis to find a continuous-time differential equation whose flow better approximates the discrete steps of an optimization algorithm like gradient descent. This allows us to characterize the *implicit regularization* induced by the choice of optimizer and its step size.

For a single-task setting with loss $\mathcal{L}(\boldsymbol{\theta})$, the standard gradient flow is $\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$. The MGF corrects this to account for the discrete nature of GD with a finite learning rate η.

Definition 1 (Modified Gradient Flow [97])**.** The modified gradient flow for gradient descent with learning rate η on a loss $\mathcal{L}(\boldsymbol{\theta})$ is the solution to the ordinary differential equation (ODE):

$$\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) - \frac{\eta}{2} \nabla^2_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) + \mathcal{O}(\eta^2). \tag{3.4}$$

Crucially, this modified flow can be expressed as the gradient flow of a *modified loss function.*

Lemma 1 (Modified Loss [97])**.** *The optimization trajectory under the modified gradient flow from Eq.* (3.4) *is equivalent to performing gradient descent on a modified loss* $\tilde{\mathcal{L}}(\boldsymbol{\theta})$*:*

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \frac{\eta}{4}|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})|_2^2 + \mathcal{O}(\eta^2). \tag{3.5}$$

This modified loss was first derived in [97], with which they showed that the additional term that arises is nothing but the slope of the original loss function, and hence gradient descent attempts to find local minima flat local minima. This same mathematical framework, was later used by [138] to show how GD with dropout further introduces additional terms. The additional term, $\frac{\eta}{4}\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\|_2^2$, is the implicit regularizer. It encourages the optimizer to find solutions not only with a low loss but also in "flat" regions of the landscape where the gradient norm is small. This term is the foundation of our analysis.

3.3.3 Problem Statement

The existence of these two distinct training regimes (joint vs. sequential) and the powerful lens of MGF raise fundamental questions about the optimization dynamics of MTL:

1. How does the implicit regularization of joint training differ from that of sequential training?
2. Can we derive the explicit form of the modified loss (for joint training) and the effective update rule (for sequential training) to reveal how task gradients and curvatures interact?
3. What are the practical implications of these differences, especially in complex models like transformers where the choice of schedule can significantly impact performance?

This paper develops a rigorous theoretical framework to answer these questions, first in a model-agnostic setting and then specialized to a one-layer self-attention model.

3.4 Model agnostic Implicit regularization of Sequential vs Multi-Objective Training

Building on the MGF framework, we now derive the specific forms of implicit regularization induced by joint and sequential training schedules.

For the joint training regime, we can directly apply the modified loss from Lemma 1 to the aggregate loss $\mathcal{L}_{\text{multi}} = \mathcal{L}_x + \mathcal{L}_y$. This immediately yields the implicit regularizer for the multi-task setting.

Remark 6 (Generalization to m Tasks). For clarity of exposition, we focus our analysis on the two-task case. However, our framework extends straightforwardly to an arbitrary number of tasks ≥ 2. In the m-task setting, the aggregate loss becomes $\mathcal{L}_{\text{multi}} = \sum_{i=1}^{m} \mathcal{L}_i$, and the implicit regularizer becomes proportional to $\| \sum_{i=1}^{m} \nabla \mathcal{L}_i \|^2$. This term decomposes into individual gradient norm regularizers and a sum of all pairwise interaction terms $\langle \nabla \mathcal{L}_i, \nabla \mathcal{L}_j \rangle$, preserving the core finding that joint training regularizes through gradient inner products. Indeed, the TRDT variant in Chapter 1 exemplifies this m-task setting with $m = 4$, incorporating action region prediction as an additional task alongside state, action, and return prediction.

Remark 7 (On Task Weighting Coefficients). We also note that practical MTL often involves weighting coefficients, i.e., minimizing $\sum_{i=1}^{m} \lambda_i \mathcal{L}_i$ (as we did in chapter 1). For notational simplicity, we set all weights to one. This is done without loss of generality, as any constant weight $\lambda_i > 0$ can be absorbed into the definition of the loss function itself (i.e., by considering a rescaled loss $\mathcal{L}'_i = \lambda_i \mathcal{L}_i$). This scaling does not alter the fundamental structure of the resulting implicit regularizer, which would still be governed by the inner products of the (now-weighted) task gradients.

Theorem 2 (Multi-Task Modified Loss)**.** *For tasks with losses $\mathcal{L}_x$ and $\mathcal{L}_y$, the modified loss optimized by GD is given by:*

$$\begin{aligned} \widetilde{\mathcal{L}}_{multi}(\boldsymbol{\theta}) &= \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{L}_y(\boldsymbol{\theta}) \\ &+ \frac{\eta}{4} \|\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta})\|_2^2 + \mathcal{O}(\eta^2). \end{aligned} \tag{3.6}$$

Proof. The total loss is $\mathcal{L}_{\text{multi}}(\boldsymbol{\theta}) = \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{L}_y(\boldsymbol{\theta})$. Applying the modified loss formula from Lemma 1 directly:

$$\begin{aligned} \widetilde{\mathcal{L}}_{\text{multi}}(\boldsymbol{\theta}) &= \mathcal{L}_{\text{multi}}(\boldsymbol{\theta}) + \frac{\eta}{4} \|\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{multi}}(\boldsymbol{\theta})\|^2 + \mathcal{O}(\eta^2) \\ &= \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{L}_y(\boldsymbol{\theta}) \\ &\quad + \frac{\eta}{4} \|\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta})\|_2^2 + \mathcal{O}(\eta^2) \end{aligned} \tag{3.7}$$

□

The most remarkable property of this additional loss is that it is minimized when $\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) = -\nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta})$. This behavior of GD under MTL is surprisingly different from its behavior in single task settings. In single-task optimization, the implicit regularization term is minimized only at critical points where the gradient vanishes. In the multi-task setting, however, the regularizer can also be minimized through gradient anti-alignment, creating a tension between minimizing the original losses and encouraging gradient disagreement.

Which leads us to the following proposition:

Proposition 1 (Gradient Anti-Alignment Bias)**.** *Multi-objective gradient descent implicitly biases the optimization toward regions where task gradients are anti-aligned, though this effect competes with the primary objective of minimizing the individual losses.*

The disagreement between the gradients is equivalent to making the angle Φ between the two tasks equal to π. Which becomes clear from the following corollary:

Corollary 1. *The modified multi-task loss decomposes into task-specific terms and an interaction term:*

$$\widetilde{\mathcal{L}}_{multi}(\boldsymbol{\theta}) = \underbrace{\mathcal{L}_x(\boldsymbol{\theta}_x) + \mathcal{L}_y(\boldsymbol{\theta}_y)}_{\text{task-specific losses}} + \tag{3.8}$$

$$\frac{\eta}{4} \underbrace{\left(\|\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}_x)\|_2^2 + \|\nabla_{\theta} \mathcal{L}_y(\boldsymbol{\theta}_y)\|_2^2\right)}_{\text{task-specific gradient regularization}} + \tag{3.9}$$

$$\frac{\eta}{2} \underbrace{\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}_x), \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}_y) \rangle}_{\text{multi-task interaction regularization}} + \mathcal{O}(\eta^2) \tag{3.10}$$

where $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean inner product in $\mathbb{R}^p$.

Note how the multi-task interaction term can be either positive or negative based on the angle Φ between the gradients of the two tasks. The role of this term is to capture how both tasks interact over shared parameters, which is demonstrated by the following corollary:

Corollary 2 (Task Interaction through Shared Parameters)**.** *Let $\Theta = \Theta_x \cup \Theta_y \cup \Theta_s$ where Θ_s represents shared parameters, and let $\boldsymbol{\theta} = [\boldsymbol{\theta}_x, \boldsymbol{\theta}_y, \boldsymbol{\theta}_s]$ be the corresponding parameter vector. If $\Theta_s = \emptyset$ (no parameter sharing), then the cross-term vanishes:*

$$\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \rangle = 0 \tag{3.11}$$

With shared parameters ($\Theta_s \neq \emptyset$), this term is zero if and only if the gradients are orthogonal in the parameter space or both null vectors.

Proof. We partition Θ into shared parameters Θ_s and task-specific parameters Θ_x, Θ_y. In the fully disjoint case where there are no shared parameters, $\Theta = \Theta_x \cup \Theta_y$ where $\Theta_x \cap \Theta_y = \emptyset$.

Let $\boldsymbol{\theta} = [\theta_1, ..., \theta_p]$ be the concatenated vector of all parameters. The inner product can be written as:

$$\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \rangle = \sum_{i=1}^{p} \frac{\partial \mathcal{L}_x}{\partial \theta_i} \frac{\partial \mathcal{L}_y}{\partial \theta_i} \tag{3.12}$$

For parameters $\theta_i \in \Theta_x$:

$$\frac{\partial \mathcal{L}_y}{\partial \theta_i} = 0 \tag{3.13}$$

For parameters $\theta_i \in \Theta_y$:

$$\frac{\partial \mathcal{L}_x}{\partial \theta_i} = 0 \tag{3.14}$$

Since each parameter belongs to either Θ_x or Θ_y (but not both), for each term in the sum, at least one of the partial derivatives is zero. Therefore, each term in the sum is zero, making the entire inner product zero. □

Another direct result is given by the following proposition:

Proposition 2. *As long as the angle between the two task gradients satisfies:*

$$\Phi < \arccos\left(-\frac{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_y\|}{2\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_x\|}\right) \tag{3.15}$$

GD induces stronger regularization compared to single-task settings.

Proof. The difference between multi-task and single-task regularization is:

$$\begin{aligned}&\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_x + \nabla_{\boldsymbol{\theta}}\mathcal{L}_y\|_2^2 - \|\nabla_{\boldsymbol{\theta}}\mathcal{L}_x\|_2^2 \\ &= \|\nabla_{\boldsymbol{\theta}}\mathcal{L}_y\|_2^2 + 2\langle\nabla_{\boldsymbol{\theta}}\mathcal{L}_x, \nabla_{\boldsymbol{\theta}}\mathcal{L}_y\rangle\end{aligned} \tag{3.16}$$

This is positive when:

$$2\langle\nabla_{\boldsymbol{\theta}}\mathcal{L}_x, \nabla_{\boldsymbol{\theta}}\mathcal{L}_y\rangle > -\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_y\|_2^2 \tag{3.17}$$

$$2\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_x\|\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_y\|\cos(\Phi) > -\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_y\|^2 \tag{3.18}$$

$$\cos(\Phi) > -\frac{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_y\|}{2\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_x\|} \tag{3.19}$$

which gives the stated condition. □

The insights from our modified gradient flow analysis perfectly align with the well-known fact from multi-objective optimization, that optimizing two tasks using linear scalarization – the same technique we are considering here, with the simple distinction that for it, we also add ponderation coefficients for both tasks – leads to solutions that are completely biased for one task over the other [61].

3.4.1 Modified Gradient Flow Analysis for Sequential Multi-Task Updates

In contrast to the multi-objective update

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\left[\nabla_{\boldsymbol{\theta}}\mathcal{L}_x(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\mathcal{L}_y(\boldsymbol{\theta})\right],$$

we study a sequential update scheme. Starting from $\boldsymbol{\theta}$, we first perform an update for task x:

$$\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta} - \eta\,\nabla_{\boldsymbol{\theta}}\mathcal{L}_x(\boldsymbol{\theta}), \tag{3.20}$$

and then update for task y:

$$\boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(1)} - \eta\,\nabla_{\boldsymbol{\theta}}\mathcal{L}_y\left(\boldsymbol{\theta}^{(1)}\right). \tag{3.21}$$

We assume that both losses are twice continuously differentiable.

Theorem 3 (Effective Update for Sequential Multi-Task Gradient Descent)**.** *Under the sequential updates* (3.20) *and* (3.21)*, the overall parameter update is given by*

$$\begin{aligned}\boldsymbol{\theta}^{(2)} = \boldsymbol{\theta} - \eta \left[\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta})\right] \\ + \eta^2 \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{O}(\eta^3).\end{aligned} \tag{3.22}$$

Moreover, the gradient for task y computed at the intermediate point $\boldsymbol{\theta}^{(1)}$ satisfies

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_y\left(\boldsymbol{\theta}^{(1)}\right) = \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) - \eta \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{O}(\eta^2). \tag{3.23}$$

Proof. The first update (3.20) yields

$$\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}).$$

Expanding $\nabla_{\boldsymbol{\theta}} \mathcal{L}_y$ at $\boldsymbol{\theta}^{(1)}$ by a first-order Taylor expansion about $\boldsymbol{\theta}$ gives

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_y\left(\boldsymbol{\theta}^{(1)}\right) = \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) + \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \left(\boldsymbol{\theta}^{(1)} - \boldsymbol{\theta}\right) + O\big(\|\boldsymbol{\theta}^{(1)} - \boldsymbol{\theta}\|^2\big).$$

Since $\boldsymbol{\theta}^{(1)} - \boldsymbol{\theta} = -\eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})$, we obtain (3.23). The second update (3.21) then gives

$$\boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(1)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_y\left(\boldsymbol{\theta}^{(1)}\right) \tag{3.24}$$

$$= \left[\boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})\right] - \eta \left[\nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) - \eta \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})\right] + \mathcal{O}(\eta^3) \tag{3.25}$$

$$= \boldsymbol{\theta} - \eta \left[\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta})\right] + \eta^2 \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{O}(\eta^3). \tag{3.26}$$

This completes the proof. □

In the multi-objective update scheme, one minimizes the aggregate loss

$$\mathcal{L}_{\text{multi}}(\boldsymbol{\theta}) = \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{L}_y(\boldsymbol{\theta}),$$

and the modified gradient flow analysis (via backward error analysis) shows that an extra regularization term of order η arises proportional to

$$\frac{\eta}{2} \langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}),\ \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \rangle.$$

This term directly reflects the alignment (or disagreement) between the two task gradients.

In contrast, the sequential update analyzed in Theorem 3 yields an effective update (3.22) with an *additional* Hessian-induced correction term,

$$\eta^2 \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}),$$

which does not appear in the multi-objective update. Equation (3.23) reveals that the gradient for task y is evaluated at a shifted parameter value $\boldsymbol{\theta}^{(1)}$ and is corrected by the term $\eta\,\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_y(\boldsymbol{\theta})\,\nabla_{\boldsymbol{\theta}}\mathcal{L}_x(\boldsymbol{\theta})$.

The implication is that—even if the two tasks are similar and their gradients are aligned at $\boldsymbol{\theta}$—the re-evaluation of the gradient for task y after the update on task x can lead to a significant directional change when the curvature (as measured by $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_y(\boldsymbol{\theta})$) is large. In extreme cases, this curvature effect may cause the effective gradient for task y (computed at $\boldsymbol{\theta}^{(1)}$) to differ substantially from its original direction. In particular, while the multi-objective update encourages direct interaction via the inner product $\langle\nabla_{\boldsymbol{\theta}}\mathcal{L}_x(\boldsymbol{\theta}),\,\nabla_{\boldsymbol{\theta}}\mathcal{L}_y(\boldsymbol{\theta})\rangle$, the sequential update may induce an effective anti-alignment in regions of high curvature—though the precise outcome depends on both the magnitude and the structure of the Hessian.

Remark 8 (Implications for Training Regimes). Our analysis reveals that:

- Multi-objective training directly exposes gradient alignment through the cross-term $\langle\nabla_{\boldsymbol{\theta}}\mathcal{L}_x(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}}\mathcal{L}_y(\boldsymbol{\theta})\rangle$
- Sequential training introduces additional curvature-dependent effects that can artificially induce task competition
- The distinction becomes negligible as $\eta \to 0$, explaining convergent behavior at small learning rates

3.5 Implicit Regularisation in One-Layer Self-Attention

The preceding model-agnostic analysis established the fundamental mechanisms of implicit regularization in multi-task settings, revealing a first-order interaction term for joint training and a second-order Hessian correction for sequential updates. To understand the practical implications of these findings, we now specialize our analysis to a concrete and highly relevant architecture: the transformer. By examining a simplified one-layer self-attention model, we can derive closed-form expressions for these regularization effects and interpret their geometric action on the attention mechanism itself. This allows us to bridge the gap from abstract theory to the specific dynamics governing modern sequence models.

3.5.1 Mathematical Setup of the Self-Attention Model

Following a large body of previous work [134, 135, 68, 136, 33, 137], we consider a simplified self-attention mechanism designed to make theoretical analysis tractable while preserving the essential characteristics of attention-based learning.

Definition 2 (Token Vocabulary)**.** et $\mathcal{V} = \mathcal{V}_x \cup \mathcal{V}_y$ be the vocabulary of all possible tokens, where $\mathcal{V}_x$ and $\mathcal{V}_y$ are the token sets for tasks X and Y respectively. We assume $\mathcal{V}_x \cap \mathcal{V}_y = \emptyset$.

If overlap is desired, distinct embeddings per task must be supplied. Each token $j \in \mathcal{V}$ has a fixed embedding $\boldsymbol{e}_j \in \mathbb{R}^{d_{\text{in}}}$.

Remark 9 (Mapping to Decision Transformer Vocabulary). In the context of Chapter 1's Decision Transformer for offline RL, the generic vocabularies $\mathcal{V}_x$ and $\mathcal{V}_y$ map to specific modalities: $\mathcal{V}_x \equiv \mathcal{S}$ (state tokens) and $\mathcal{V}_y \equiv \mathcal{A}$ (action tokens). The assumption $\mathcal{V}_x \cap \mathcal{V}_y = \emptyset$ naturally holds in this setting since states and actions represent distinct data types. Similarly, the prediction matrices $\boldsymbol{W}^x$ and $\boldsymbol{W}^y$ in our general framework correspond to $\boldsymbol{W}^s$ and $\boldsymbol{W}^a$ respectively in the RL setting.

Definition 3 (Input Representation)**.** Given a trajectory dataset $\mathcal{T}$, we construct a supervised learning dataset $\mathcal{D}_{multi} = \{(\boldsymbol{H}^{[i]}, x^{[i]}, y^{[i]})\}_{i=1}^{N}$ where $\boldsymbol{H}^{[i]} = [\boldsymbol{h}_1^{[i]}, \ldots, \boldsymbol{h}_K^{[i]}]^\top \in \mathbb{R}^{K \times d_{\text{in}}}$ represents a sequence of K token embeddings[1]. Each embedding $\boldsymbol{h}_k^{[i]}$ belongs to the set $\{\boldsymbol{e}_j : j \in \mathcal{V}\}$, ensuring that the embedding map is global and fixed across all sequences. The targets $x^{[i]} \in \mathcal{V}_x$ and $y^{[i]} \in \mathcal{V}_y$ denote the token indices for tasks X and Y respectively. We denote the final token $\bar{\boldsymbol{h}}^{[i]} = \boldsymbol{h}_K^{[i]} \in \mathbb{R}^{d_{\text{in}}}$, which serves as the query in our attention mechanism.

This abstract supervised learning formulation is directly motivated by the modern paradigm of treating sequential decision-making problems as sequence modeling, as explored in Chapter 1. A prominent example is offline reinforcement learning (RL), where trajectories are sequences composed of multiple data types or *modalities* (e.g., returns-to-go, states, and actions). In this context, our setup can be interpreted as a multi-task prediction problem: given a history of trajectory tokens $\boldsymbol{H}^{[i]}$, the model might be trained to concurrently predict the next state token (task X, with target $x^{[i]} \in \mathcal{S}$) and the next action token (task Y, with target $y^{[i]} \in \mathcal{A}$). This perspective was popularized by architectures like the Decision Transformer [7] and the Trajectory Transformer [21], which frame RL as autoregressive prediction over tokenized trajectories. Thus, our simplified model is not merely a toy problem; it captures the essential structure of multi-modal prediction tasks that are central to these influential models, including the MO-DT and TRDT variants analyzed in Chapter 1.

Remark 10 (Query Token). We fix the query token to position K for simplicity. The derivations for arbitrary query position $q^{[i]}$ are analogous with $\bar{\boldsymbol{h}}^{[i]} = \boldsymbol{h}_{q^{[i]}}^{[i]}$.

Definition 4 (Simplified Self-Attention)**.** The self-attention operation is parameterized by a single learnable matrix $\boldsymbol{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$. Given an input sequence $\boldsymbol{H}^{[i]}$, the attention mechanism computes:

1. **Attention logits**: $\boldsymbol{z}^{[i]} = \boldsymbol{H}^{[i]} \boldsymbol{W} \bar{\boldsymbol{h}}^{[i]} \in \mathbb{R}^K$
2. **Attention weights**: $\boldsymbol{\zeta}^{[i]} = \sigma(\boldsymbol{z}^{[i]}) \in \Delta^{K-1}$, where the softmax function is defined component-wise as:

$$\sigma(\boldsymbol{z})_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.27}$$

[1] K is the sequence length, possibly with token repetitions, hence $K \geq |\mathcal{V}|$ is allowed.

3. **Context vector**:

$$\boldsymbol{c}^{[i]} = \boldsymbol{H}^{[i]\top}\boldsymbol{\zeta}^{[i]} = \sum_{k=1}^{K} \zeta_k^{[i]} \boldsymbol{h}_k^{[i]} \in \mathbb{R}^{d_{\text{in}}} \tag{3.28}$$

For simplicity, we set the value projection matrix to the identity, i.e., $\boldsymbol{V} = \boldsymbol{I}_{d_{\text{in}}}$.

Remark 11 (Connection to Token-Priority Analysis). The simplified self-attention model presented here is identical to the one analyzed in Chapter 1, Section 1.6.1. While Chapter 1 employs this model to study asymptotic convergence through the Token-Priority Graph framework, revealing how multi-objective training affects the final attention patterns, here we analyze the transient optimization dynamics through modified gradient flow. These analyses are complementary: Chapter 1 characterizes where the optimization converges, while this chapter reveals how the implicit regularization shapes the path to convergence.

This connection between two distinct mathematical frameworks—asymptotic Token-Priority analysis and modified gradient flow—opens an intriguing research direction. Specifically, one could investigate whether the strongly connected components and priority structures identified in the Token-Priority Graphs of Chapter 1 are primarily determined by the implicit regularization terms $\frac{\eta}{2}\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x, \nabla_{\boldsymbol{\theta}} \mathcal{L}_y \rangle$ derived in this chapter. Understanding this relationship would provide a complete picture of multi-task optimization: not only where we converge and how we get there, but whether the transient dynamics fundamentally determine the asymptotic structure. This would constitute a valuable bridge between local optimization dynamics and global convergence properties in multi-task transformers.

Remark 12 (Attention Weight Dependencies). The attention weights $\boldsymbol{\zeta}^{[i]}$ depend on the parameter matrix $\boldsymbol{W}$ through the attention logits: $\boldsymbol{\zeta}^{[i]} = \boldsymbol{\zeta}^{[i]}(\boldsymbol{W}) = \sigma(\boldsymbol{H}^{[i]}\boldsymbol{W}\bar{\boldsymbol{h}}^{[i]})$. This dependency is crucial for understanding the gradient flow through the attention mechanism.

Remark 13 (Mathematical Conventions). We use Δ^{k-1} to denote the $(k-1)$-dimensional probability simplex, δ_{km} for the Kronecker delta ($\delta_{km} = 1$ if $k = m$, 0 otherwise).

Remark 14 (Architectural Simplifications). Our simplified attention mechanism differs from the standard transformer architecture in three fundamental ways. First, we employ a single matrix $\boldsymbol{W}$ instead of separate query, key, and value projection matrices ($\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V$). Second, we set the value projection to the identity ($\boldsymbol{V} = \boldsymbol{I}$) rather than learning task-specific value transformations. Third, both tasks share the same attention matrix $\boldsymbol{W}$, creating a natural setting for studying task interactions through shared parameters. These simplifications enable exact gradient analysis while preserving the core attention mechanism, allowing us to derive precise conditions for task cooperation versus competition in multi-task settings.

Definition 5 (Task-Specific Prediction Heads)**.** Each task employs a dedicated prediction matrix: $\boldsymbol{W}^x \in \mathbb{R}^{|\mathcal{V}_x| \times d_{\text{in}}}$ for task X and $\boldsymbol{W}^y \in \mathbb{R}^{|\mathcal{V}_y| \times d_{\text{in}}}$ for task Y. We denote by $\boldsymbol{w}_j^x$ the j-th row of $\boldsymbol{W}^x$ corresponding to token $j \in \mathcal{V}_x$, with an analogous notation for task Y.

Assumption 1 (Orthogonality Condition)**.** *The prediction matrices satisfy:*

$$\boldsymbol{w}_j^x \cdot \boldsymbol{e}_k = \begin{cases} 1 & \text{if } j = k \text{ and } j, k \in \mathcal{V}_x \\ 0 & \text{otherwise} \end{cases} \tag{3.29}$$

$$\boldsymbol{w}_j^y \cdot \boldsymbol{e}_k = \begin{cases} 1 & \text{if } j = k \text{ and } j, k \in \mathcal{V}_y \\ 0 & \text{otherwise} \end{cases} \tag{3.30}$$

In particular, $\boldsymbol{w}_j^x \cdot \boldsymbol{e}_k = 0$ *for all* $j \in \mathcal{V}_x, k \in \mathcal{V}_y$ *and vice versa. This requires* $d_{in} \geq \max\{|\mathcal{V}_x|, |\mathcal{V}_y|\}$.

Assumption 2 (Realizability)**.** *For each training example* $(\boldsymbol{H}^{[i]}, x^{[i]}, y^{[i]})$*, both target tokens appear in the sequence:*

$$\exists k_x^{[i]}, k_y^{[i]} \in \{1, \ldots, K\} : \boldsymbol{h}_{k_x^{[i]}}^{[i]} = \boldsymbol{e}_{x^{[i]}} \text{ and } \boldsymbol{h}_{k_y^{[i]}}^{[i]} = \boldsymbol{e}_{y^{[i]}} \tag{3.31}$$

Definition 6 (Token Prediction Scores)**.** Given the context vector $\boldsymbol{c}^{[i]}$ and the orthogonality assumption, the prediction scores for each task are computed as:

$$\hat{v}_x^{[i]} = \boldsymbol{w}_{x^{[i]}}^x \cdot \boldsymbol{c}^{[i]} = \sum_{k=1}^{K} \zeta_k^{[i]} (\boldsymbol{w}_{x^{[i]}}^x \cdot \boldsymbol{h}_k^{[i]}) \tag{3.32}$$

$$\hat{v}_y^{[i]} = \boldsymbol{w}_{y^{[i]}}^y \cdot \boldsymbol{c}^{[i]} = \sum_{k=1}^{K} \zeta_k^{[i]} (\boldsymbol{w}_{y^{[i]}}^y \cdot \boldsymbol{h}_k^{[i]}) \tag{3.33}$$

Lemma 2 (Score Simplification under Orthogonality)**.** *Under Assumption 1, the prediction scores simplify to:*

$$\hat{v}_x^{[i]} = \zeta_{k_x^{[i]}}^{[i]} \tag{3.34}$$

$$\hat{v}_y^{[i]} = \zeta_{k_y^{[i]}}^{[i]} \tag{3.35}$$

where $k_x^{[i]}$ *and* $k_y^{[i]}$ *are the positions of the target tokens in the sequence.*

Proof. By the orthogonality assumption, $\boldsymbol{w}_{x^{[i]}}^x \cdot \boldsymbol{h}_k^{[i]} = 1$ if $k = k_x^{[i]}$ and 0 otherwise. Therefore:

$$\hat{v}_x^{[i]} = \sum_{k=1}^{K} \zeta_k^{[i]} (\boldsymbol{w}_{x^{[i]}}^x \cdot \boldsymbol{h}_k^{[i]}) = \zeta_{k_x^{[i]}}^{[i]} \tag{3.36}$$

The same reasoning applies for task Y. □

Definition 7 (Loss Functions)**.** The empirical risk for each task employs the negative log-likelihood with smooth clipping to ensure differentiability. For task X, we have $\mathcal{L}_x(\boldsymbol{W}) = \frac{1}{N} \sum_{i=1}^{N} -\log(\hat{v}_x^{[i]} + \xi)$, while task Y uses $\mathcal{L}_y(\boldsymbol{W}) = \frac{1}{N} \sum_{i=1}^{N} -\log(\hat{v}_y^{[i]} + \xi)$. The multi-task objective combines both losses as $\mathcal{L}_{\text{multi}}(\boldsymbol{W}) = \mathcal{L}_x(\boldsymbol{W}) + \mathcal{L}_y(\boldsymbol{W})$, where $\xi \in (0, 1)$ is a small constant ensuring numerical stability.

Assumption 3 (Initialization)**.** *We assume* $\boldsymbol{W}$ *is initialized such that* $\zeta_k^{[i]}(\boldsymbol{W}_0) > 0$ *for all* $k \in \{1, \ldots, K\}$ *and all training examples* $i \in \{1, \ldots, N\}$*, where* $\boldsymbol{W}_0$ *denotes the initial parameter values. This ensures the losses are well-defined at initialization.*

3.5.2 Preparatory Lemmas

We now derive the gradients of the task losses with respect to the shared attention matrix $\boldsymbol{W}$. Note that we compute gradients of the negative log-likelihood of the clipped score $\log(\hat{v}_x + \xi)$.

Lemma 3 (Softmax Derivative)**.** *The derivative of the softmax function is:*

$$\frac{\partial \sigma(\boldsymbol{z})_k}{\partial z_m} = \sigma(\boldsymbol{z})_k[\delta_{km} - \sigma(\boldsymbol{z})_m] = \zeta_k[\delta_{km} - \zeta_m] \tag{3.37}$$

where δ_{km} is the Kronecker delta and we use the shorthand $\zeta_k = \sigma(\boldsymbol{z})_k$.

Definition 8 (Attention-Weighted Target Embeddings)**.** For each example i and task, define:

$$\boldsymbol{u}_x^{[i]} := \zeta^{[i]}_{k_x^{[i]}} \boldsymbol{h}^{[i]}_{k_x^{[i]}} = \hat{v}_x^{[i]} \boldsymbol{h}^{[i]}_{k_x^{[i]}} \tag{3.38}$$

$$\boldsymbol{u}_y^{[i]} := \zeta^{[i]}_{k_y^{[i]}} \boldsymbol{h}^{[i]}_{k_y^{[i]}} = \hat{v}_y^{[i]} \boldsymbol{h}^{[i]}_{k_y^{[i]}} \tag{3.39}$$

Lemma 4 (Gradient of Token Score)**.** *Since $\hat{v}_x^{[i]} = \zeta^{[i]}_{k_x^{[i]}}$, we have:*

$$\nabla_{\boldsymbol{W}} \hat{v}_x^{[i]} = \left(\boldsymbol{u}_x^{[i]} - \hat{v}_x^{[i]} \boldsymbol{c}^{[i]}\right) \bar{\boldsymbol{h}}^{[i]\top} \tag{3.40}$$

where $\boldsymbol{c}^{[i]} = \sum_{k=1}^{K} \zeta_k^{[i]} \boldsymbol{h}_k^{[i]}$ is the context vector.

Proof. Computing the derivative:

$$\frac{\partial \hat{v}_x^{[i]}}{\partial W_{pq}} = \frac{\partial \zeta^{[i]}_{k_x^{[i]}}}{\partial W_{pq}} \tag{3.41}$$

$$= \sum_{m=1}^{K} \frac{\partial \zeta^{[i]}_{k_x^{[i]}}}{\partial z_m^{[i]}} \frac{\partial z_m^{[i]}}{\partial W_{pq}} \tag{3.42}$$

$$= \sum_{m=1}^{K} \zeta^{[i]}_{k_x^{[i]}} \left[\delta_{k_x^{[i]},m} - \zeta_m^{[i]}\right] h_{mp}^{[i]} \bar{h}_q^{[i]} \tag{3.43}$$

$$= \sum_{m=1}^{K} \hat{v}_x^{[i]} \left[\delta_{k_x^{[i]},m} - \zeta_m^{[i]}\right] h_{mp}^{[i]} \bar{h}_q^{[i]} \tag{3.44}$$

$$= \hat{v}_x^{[i]} h^{[i]}_{k_x^{[i]},p} \bar{h}_q^{[i]} - \hat{v}_x^{[i]} \sum_{m=1}^{K} \zeta_m^{[i]} h_{mp}^{[i]} \bar{h}_q^{[i]} \tag{3.45}$$

$$= [\boldsymbol{u}_x^{[i]} - \hat{v}_x^{[i]} \boldsymbol{c}^{[i]}]_p \bar{h}_q^{[i]} \tag{3.46}$$

□

Remark 15 (Geometric Interpretation). The gradient $(\boldsymbol{u}_x^{[i]} - \hat{v}_x^{[i]} \boldsymbol{c}^{[i]}) \bar{\boldsymbol{h}}^{[i]\top}$ is a rank-1 update that aligns $\boldsymbol{W} \bar{\boldsymbol{h}}^{[i]}$ with the target embedding while suppressing alignment with the current context. Note that the clipping affects only the $\frac{1}{\hat{v}_x + \xi}$ prefactor in the loss gradient, but not the formula for $\nabla_{\boldsymbol{W}} \hat{v}_x$.

Lemma 5 (Complete Gradient Expression). *The gradients of the task losses are:*

$$\nabla_{\boldsymbol{W}} \mathcal{L}_x(\boldsymbol{W}) = -\frac{1}{N} \sum_{i=1}^{N} \frac{1}{\hat{v}_x^{[i]} + \xi} \left[\boldsymbol{u}_x^{[i]} - \hat{v}_x^{[i]} \boldsymbol{c}^{[i]} \right] \bar{\boldsymbol{h}}^{[i]\top} \tag{3.47}$$

$$\nabla_{\boldsymbol{W}} \mathcal{L}_y(\boldsymbol{W}) = -\frac{1}{N} \sum_{i=1}^{N} \frac{1}{\hat{v}_y^{[i]} + \xi} \left[\boldsymbol{u}_y^{[i]} - \hat{v}_y^{[i]} \boldsymbol{c}^{[i]} \right] \bar{\boldsymbol{h}}^{[i]\top} \tag{3.48}$$

For the sequential training analysis, we require the Hessian of each task loss.

Lemma 6 (Second-Order Softmax Derivative). *The second derivative of the softmax function is:*

$$\frac{\partial^2 \sigma(\boldsymbol{z})_k}{\partial z_u \partial z_v} = \sigma(\boldsymbol{z})_k \Big[(\delta_{ku} - \sigma(\boldsymbol{z})_u)(\delta_{kv} - \sigma(\boldsymbol{z})_v) \tag{3.49}$$

$$- (\delta_{uv} - \sigma(\boldsymbol{z})_v)\sigma(\boldsymbol{z})_u \Big] \tag{3.50}$$

Lemma 7 (Hessian of Task Loss). *The Hessian of task X loss has the form:*

$$[\nabla^2_{\boldsymbol{W}} \mathcal{L}_x(\boldsymbol{W})]_{pq,rs} = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{G_{pq}^{[i]} G_{rs}^{[i]}}{(\hat{v}_x^{[i]} + \xi)^2} - \frac{\mathcal{H}_{pq,rs}^{[i]}}{\hat{v}_x^{[i]} + \xi} \right] \tag{3.51}$$

where:

$$G_{pq}^{[i]} = [\boldsymbol{u}_x^{[i]} - \hat{v}_x^{[i]} \boldsymbol{c}^{[i]}]_p \bar{h}_q^{[i]} \tag{3.52}$$

$$\mathcal{H}_{pq,rs}^{[i]} = \sum_{u,v=1}^{K} \frac{\partial^2 \zeta_{k_x^{[i]}}^{[i]}}{\partial z_u^{[i]} \partial z_v^{[i]}} h_{up}^{[i]} \bar{h}_q^{[i]} h_{vr}^{[i]} \bar{h}_s^{[i]} \tag{3.53}$$

Proof of Lemma 7. We derive the Hessian by taking the second derivative of the task loss $\mathcal{L}_x(\boldsymbol{W})$ with respect to the attention matrix entries. Our approach builds systematically on the first-order gradient computation established in Theorem 5.

Step 1: Starting from the first-order gradient. From Theorem 5, we have:

$$\frac{\partial \mathcal{L}_x}{\partial W_{pq}} = -\frac{1}{N} \sum_{i=1}^{N} \frac{1}{\hat{v}_x^{[i]} + \xi} \frac{\partial \hat{v}_x^{[i]}}{\partial W_{pq}} \tag{3.54}$$

where $\frac{\partial \hat{v}_x^{[i]}}{\partial W_{pq}} = G_{pq}^{[i]}$, giving us:

$$\frac{\partial \mathcal{L}_x}{\partial W_{pq}} = -\frac{1}{N} \sum_{i=1}^{N} \frac{G_{pq}^{[i]}}{\hat{v}_x^{[i]} + \xi} \tag{3.55}$$

Step 2: Computing the second derivative. To obtain the Hessian entry $[\nabla^2_{\boldsymbol{W}}\mathcal{L}_x(\boldsymbol{W})]_{pq,rs}$, we differentiate the first-order gradient with respect to W_{rs}:

$$\frac{\partial^2 \mathcal{L}_x}{\partial W_{rs}\partial W_{pq}} = -\frac{1}{N}\sum_{i=1}^{N} \frac{\partial}{\partial W_{rs}}\left[\frac{G_{pq}^{[i]}}{\hat{v}_x^{[i]}+\xi}\right] \tag{3.56}$$

Step 3: Applying the quotient rule. Using the quotient rule for differentiation:

$$\frac{\partial}{\partial W_{rs}}\left[\frac{G_{pq}^{[i]}}{\hat{v}_x^{[i]}+\xi}\right] = \frac{\frac{\partial G_{pq}^{[i]}}{\partial W_{rs}}\cdot(\hat{v}_x^{[i]}+\xi) - G_{pq}^{[i]}\cdot\frac{\partial(\hat{v}_x^{[i]}+\xi)}{\partial W_{rs}}}{(\hat{v}_x^{[i]}+\xi)^2} \tag{3.57}$$

$$= \frac{\frac{\partial G_{pq}^{[i]}}{\partial W_{rs}}\cdot(\hat{v}_x^{[i]}+\xi) - G_{pq}^{[i]}\cdot\frac{\partial \hat{v}_x^{[i]}}{\partial W_{rs}}}{(\hat{v}_x^{[i]}+\xi)^2} \tag{3.58}$$

Since $\frac{\partial \hat{v}_x^{[i]}}{\partial W_{rs}} = G_{rs}^{[i]}$, this becomes:

$$\frac{\partial}{\partial W_{rs}}\left[\frac{G_{pq}^{[i]}}{\hat{v}_x^{[i]}+\xi}\right] = \frac{\frac{\partial G_{pq}^{[i]}}{\partial W_{rs}}\cdot(\hat{v}_x^{[i]}+\xi) - G_{pq}^{[i]}\cdot G_{rs}^{[i]}}{(\hat{v}_x^{[i]}+\xi)^2} \tag{3.59}$$

Separating the terms:

$$= \frac{\frac{\partial G_{pq}^{[i]}}{\partial W_{rs}}}{\hat{v}_x^{[i]}+\xi} - \frac{G_{pq}^{[i]}G_{rs}^{[i]}}{(\hat{v}_x^{[i]}+\xi)^2} \tag{3.60}$$

Step 4: Computing $\frac{\partial G_{pq}^{[i]}}{\partial W_{rs}}$. Recall that $G_{pq}^{[i]} = \frac{\partial \hat{v}_x^{[i]}}{\partial W_{pq}}$ where $\hat{v}_x^{[i]} = \zeta^{[i]}_{k_x^{[i]}}$. Therefore:

$$\frac{\partial G_{pq}^{[i]}}{\partial W_{rs}} = \frac{\partial^2 \hat{v}_x^{[i]}}{\partial W_{rs}\partial W_{pq}} = \frac{\partial^2 \zeta^{[i]}_{k_x^{[i]}}}{\partial W_{rs}\partial W_{pq}} \tag{3.61}$$

Step 5: Applying the chain rule for second derivatives. Since $\zeta^{[i]}_{k_x^{[i]}}$ depends on $\boldsymbol{W}$ through the attention logits $z_u^{[i]} = \sum_{j=1}^{d_{\text{in}}}\sum_{k=1}^{d_{\text{in}}} h_{uj}^{[i]} W_{jk} \bar{h}_k^{[i]}$, we apply the chain rule:

$$\frac{\partial^2 \zeta^{[i]}_{k_x^{[i]}}}{\partial W_{rs}\partial W_{pq}} = \sum_{u,v=1}^{K} \frac{\partial^2 \zeta^{[i]}_{k_x^{[i]}}}{\partial z_u^{[i]}\partial z_v^{[i]}} \frac{\partial z_u^{[i]}}{\partial W_{pq}} \frac{\partial z_v^{[i]}}{\partial W_{rs}} \tag{3.62}$$

Step 6: Evaluating the logit derivatives. From the definition of the attention logits, we have:

$$\frac{\partial z_u^{[i]}}{\partial W_{pq}} = h_{up}^{[i]} \bar{h}_q^{[i]} \tag{3.63}$$

$$\frac{\partial z_v^{[i]}}{\partial W_{rs}} = h_{vr}^{[i]} \bar{h}_s^{[i]} \tag{3.64}$$

Substituting these into our expression:

$$\frac{\partial^2 \zeta^{[i]}_{k_x^{[i]}}}{\partial W_{rs} \partial W_{pq}} = \sum_{u,v=1}^{K} \frac{\partial^2 \zeta^{[i]}_{k_x^{[i]}}}{\partial z_u^{[i]} \partial z_v^{[i]}} h_{up}^{[i]} \bar{h}_q^{[i]} h_{vr}^{[i]} \bar{h}_s^{[i]} \tag{3.65}$$

This is precisely the definition of $\mathcal{H}^{[i]}_{pq,rs}$ given in the theorem statement.

Step 7: Assembling the final result. Substituting back into our expression from Step 3:

$$\frac{\partial}{\partial W_{rs}} \left[\frac{G^{[i]}_{pq}}{\hat{v}_x^{[i]} + \xi} \right] = \frac{\mathcal{H}^{[i]}_{pq,rs}}{\hat{v}_x^{[i]} + \xi} - \frac{G^{[i]}_{pq} G^{[i]}_{rs}}{(\hat{v}_x^{[i]} + \xi)^2} \tag{3.66}$$

Therefore, the Hessian entry is:

$$[\nabla^2_{\boldsymbol{W}} \mathcal{L}_x(\boldsymbol{W})]_{pq,rs} = -\frac{1}{N} \sum_{i=1}^{N} \left[\frac{\mathcal{H}^{[i]}_{pq,rs}}{\hat{v}_x^{[i]} + \xi} - \frac{G^{[i]}_{pq} G^{[i]}_{rs}}{(\hat{v}_x^{[i]} + \xi)^2} \right] \tag{3.67}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left[\frac{G^{[i]}_{pq} G^{[i]}_{rs}}{(\hat{v}_x^{[i]} + \xi)^2} - \frac{\mathcal{H}^{[i]}_{pq,rs}}{\hat{v}_x^{[i]} + \xi} \right] \tag{3.68}$$

This completes the proof of the theorem. □

This completes the setup and gradient derivations needed for analyzing the multi-task training dynamics.

3.5.3 Notation for First- and Second-Order Quantities

For each training example $i \in [N]$[2], recall from definition 6 the token-level quantities:

$$\hat{v}_x^{[i]} := \zeta^{[i]}_{k_x^{[i]}}, \quad \hat{v}_y^{[i]} := \zeta^{[i]}_{k_y^{[i]}}, \quad \boldsymbol{c}^{[i]} := \sum_{k=1}^{K} \zeta_k^{[i]} \boldsymbol{h}_k^{[i]}, \quad \bar{\boldsymbol{h}}^{[i]} := \boldsymbol{h}_K^{[i]}. \tag{3.69}$$

We define scalar coefficients and rank-one matrices:

$$\begin{aligned} \omega_x^{[i]} &:= -\frac{1}{\hat{v}_x^{[i]} + \xi}, \quad \omega_y^{[i]} := -\frac{1}{\hat{v}_y^{[i]} + \xi}, \\ \boldsymbol{Q}_x^{[i]} &:= \left(\boldsymbol{u}_x^{[i]} - \hat{v}_x^{[i]} \boldsymbol{c}^{[i]} \right) \bar{\boldsymbol{h}}^{[i]\top}, \quad \boldsymbol{Q}_y^{[i]} := \left(\boldsymbol{u}_y^{[i]} - \hat{v}_y^{[i]} \boldsymbol{c}^{[i]} \right) \bar{\boldsymbol{h}}^{[i]\top}. \end{aligned} \tag{3.70}$$

By lemma 5, the task gradients satisfy:

$$\boldsymbol{g}_x(\boldsymbol{W}) = \frac{1}{N} \sum_{i=1}^{N} \omega_x^{[i]} \boldsymbol{Q}_x^{[i]}, \qquad \boldsymbol{g}_y(\boldsymbol{W}) = \frac{1}{N} \sum_{i=1}^{N} \omega_y^{[i]} \boldsymbol{Q}_y^{[i]} \tag{3.71}$$

Because every $\boldsymbol{Q}_*^{[i]}$ is an outer product, their pairwise Frobenius inner product factorizes:

[2] We use the standard notation $[N] := \{1, 2, \ldots, N\}$ to denote the set of integers from 1 to N.

Lemma 8 (Outer-product factorisation)**.** *For any* $i, j \in [N]$,

$$\langle \boldsymbol{Q}_x^{[i]}, \boldsymbol{Q}_y^{[j]} \rangle_F = \left(\bar{\boldsymbol{h}}^{[i]} \cdot \bar{\boldsymbol{h}}^{[j]} \right) \left(\boldsymbol{u}_x^{[i]} - \hat{v}_x^{[i]} \boldsymbol{c}^{[i]} \right)^\top \left(\boldsymbol{u}_y^{[j]} - \hat{v}_y^{[j]} \boldsymbol{c}^{[j]} \right). \tag{3.72}$$

Proof. Let $\boldsymbol{q}_x^{[i]} := \boldsymbol{u}_x^{[i]} - \hat{v}_x^{[i]} \boldsymbol{c}^{[i]}$. Then $\boldsymbol{Q}_x^{[i]} = \boldsymbol{q}_x^{[i]} \bar{\boldsymbol{h}}^{[i]\top}$, so:

$$\langle \boldsymbol{Q}_x^{[i]}, \boldsymbol{Q}_y^{[j]} \rangle_F = \left(\bar{\boldsymbol{h}}^{[i]} \cdot \bar{\boldsymbol{h}}^{[j]} \right) \boldsymbol{q}_x^{[i]\top} \boldsymbol{q}_y^{[j]}$$

which yields (3.72). □

3.5.4 Implicit Regularization Effect

We refine the analysis sketched in section 3.4 by specialising every step of the modified-gradient-flow (MGF) framework to the *simplified self-attention* architecture of definition 4. All derivations are carried out for a single shared attention matrix $\boldsymbol{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$; both task-specific heads $(\boldsymbol{W}^x, \boldsymbol{W}^y)$ are held fixed (assumption 1). We provide full first- and second-order computations and highlight the geometric interpretation of every term.

Joint (Simultaneous) Training When both losses are optimised jointly, $\mathcal{L}_{\text{multi}} = \mathcal{L}_x + \mathcal{L}_y$, the MGF argument from section 3.4 shows gradient descent is equivalent to flow on the modified loss:

$$\widetilde{\mathcal{L}}_{\text{multi}}(\boldsymbol{W}) = \mathcal{L}_x(\boldsymbol{W}) + \mathcal{L}_y(\boldsymbol{W}) + \frac{\eta}{4} \left\| \boldsymbol{g}_x(\boldsymbol{W}) + \boldsymbol{g}_y(\boldsymbol{W}) \right\|_F^2 + \mathcal{O}(\eta^2). \tag{3.73}$$

Theorem 4 (Explicit implicit term under joint updates)**.**

$$\begin{aligned} \frac{\eta}{4} \left\| \boldsymbol{g}_x + \boldsymbol{g}_y \right\|_F^2 = \frac{\eta}{4N^2} \sum_{i,j} \omega_x^{[i]} \omega_x^{[j]} \langle \boldsymbol{Q}_x^{[i]}, \boldsymbol{Q}_x^{[j]} \rangle_F \\ + \frac{\eta}{4N^2} \sum_{i,j} \omega_y^{[i]} \omega_y^{[j]} \langle \boldsymbol{Q}_y^{[i]}, \boldsymbol{Q}_y^{[j]} \rangle_F \\ + \frac{\eta}{2N^2} \sum_{i,j} \omega_x^{[i]} \omega_y^{[j]} \underbrace{\langle \boldsymbol{Q}_x^{[i]}, \boldsymbol{Q}_y^{[j]} \rangle_F}_{\textit{cross-task interaction}}. \end{aligned} \tag{3.74}$$

Proof. Use $\|\boldsymbol{g}_x + \boldsymbol{g}_y\|_F^2 = \|\boldsymbol{g}_x\|_F^2 + \|\boldsymbol{g}_y\|_F^2 + 2\langle \boldsymbol{g}_x, \boldsymbol{g}_y \rangle_F$ and insert (3.71). □

The cross-term in the implicit regularization depends on two key geometric factors. First, the query similarity $\bar{\boldsymbol{h}}^{[i]} \cdot \bar{\boldsymbol{h}}^{[j]}$ captures how aligned the query tokens are across different training examples. Second, the residual-context similarity $\left(\boldsymbol{u}_x^{[i]} - \hat{v}_x^{[i]} \boldsymbol{c}^{[i]} \right)^\top \left(\boldsymbol{u}_y^{[j]} - \hat{v}_y^{[j]} \boldsymbol{c}^{[j]} \right)$ measures the alignment between the attention-weighted target embeddings and their corresponding context vectors. When these factors produce negative alignment, the resulting negative regularizer term encourages maximal task disagreement.

Sequential (Alternating) Training Sequential update: first task X, then task Y:

$$\boldsymbol{W}^{(1)} = \boldsymbol{W} - \eta \boldsymbol{g}_x(\boldsymbol{W}), \quad \boldsymbol{W}^{(2)} = \boldsymbol{W}^{(1)} - \eta \boldsymbol{g}_y(\boldsymbol{W}^{(1)}). \tag{3.75}$$

Theorem 5 (Second-order correction under sequential updates)**.** *The overall update is:*

$$\boldsymbol{W}^{(2)} = \boldsymbol{W} - \eta \left[\boldsymbol{g}_x(\boldsymbol{W}) + \boldsymbol{g}_y(\boldsymbol{W})\right] + \frac{\eta^2}{n^2} \sum_{i,j} \omega_y^{[i]} \omega_x^{[j]} \boldsymbol{\Xi}^{[i,j]} + \mathcal{O}(\eta^3) \tag{3.76}$$

with:

$$\begin{aligned} \boldsymbol{\Xi}^{[i,j]} &:= \left[\left(\boldsymbol{u}_y^{[i]} - \hat{v}_y^{[i]} \boldsymbol{c}^{[i]}\right) \odot \boldsymbol{O}^{[i,j]}\right] \bar{\boldsymbol{h}}^{[i]\top}, \\ O_k^{[i,j]} &:= \left(\boldsymbol{u}_x^{[j]} - \hat{v}_x^{[j]} \boldsymbol{c}^{[j]}\right)^\top \boldsymbol{h}_k^{[i]}. \end{aligned} \tag{3.77}$$

Proof. Insert the Hessian structure from lemma 7 into the general sequential update from theorem 3. □

The second-order term is $\mathcal{O}(\eta^2)$ and becomes negligible as $\eta \to 0$, explaining why both regimes agree in the infinitesimal limit (see fig. 3.2).

Our analysis reveals fundamental differences in how joint and sequential training shape multi-task optimization dynamics. Joint training introduces the first-order implicit term (3.74), which directly encourages anti-alignment between task gradients through the inner product regularization. Sequential training, on the other hand, adds a second-order curvature correction (3.76) that can potentially amplify disagreement through Hessian-mediated interactions. Notably, as the learning rate η approaches zero, both implicit regularization terms vanish, causing the training trajectories to converge. These theoretical predictions will be empirically validated in Section 3.7.

3.6 Geometric Analysis of Task Competition

Our analysis has so far revealed the precise mathematical forms of the implicit regularizers for both training regimes. To build a more intuitive and actionable understanding, this section develops a geometric perspective on the problem. We formalize the notion of "task competition" and derive precise conditions based on the angle between task gradients and the curvature of the loss landscape.

3.6.1 Task Competition

Definition 9 (Task Competition - Multi-Objective Training)**.** In multi-objective training, tasks x and y are said to be **competing** at parameter $\boldsymbol{\theta}$ if:

$$\mathcal{C}_{\text{multi}}(\boldsymbol{\theta}) := \langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \rangle < 0 \tag{3.78}$$

Definition 10 (Task Competition - Sequential Training)**.** In sequential training where task x is optimized before task y, tasks are **competing** at parameter $\boldsymbol{\theta}$ if:

$$\mathcal{C}_{\text{seq}}(\boldsymbol{\theta}) := \langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}^{(1)}) \rangle < 0 \tag{3.79}$$

where $\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}^{(1)})$ is the effective gradient from Equation 3.23.

Lemma 9 (Relationship Between Competition Definitions)**.** *The sequential competition metric can be expressed as:*

$$\mathcal{C}_{seq}(\boldsymbol{\theta}) = \mathcal{C}_{multi}(\boldsymbol{\theta}) - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})^T \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{O}(\eta^2) \tag{3.80}$$

As $\eta \to 0$, both definitions converge: $\mathcal{C}_{seq} \to \mathcal{C}_{multi}$.

Proof. Substituting the expression for $\nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}^{(1)})$ from Equation 3.23:

$$\begin{aligned} \mathcal{C}_{\text{seq}} &= \langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) - \eta \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{O}(\eta^2) \rangle & (3.81) \\ &= \mathcal{C}_{\text{multi}} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})^T \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) + \mathcal{O}(\eta^2) & (3.82) \end{aligned}$$

□

3.6.2 Competition Regions as Functions of Gradient Angle

Let $\Phi \in [0, \pi]$ denote the angle between task gradients:

$$\cos(\Phi) = \frac{\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \rangle}{\|\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})\| \|\nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta})\|} \tag{3.83}$$

Proposition 3 (Competition Region - Multi-Objective Training)**.** *Tasks compete in multi-objective training if and only if $\Phi > \pi/2$.*

Proposition 4 (Competition Region – Sequential Training)**.** *Define the normalized curvature factor:*

$$\kappa := \frac{\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})^T \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})\|^2 \|\nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta})\|} \tag{3.84}$$

Tasks compete in sequential training when:

$$\cos(\Phi) < \frac{\eta \kappa \|\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})\|}{\|\nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta})\|} \tag{3.85}$$

Proof. From Lemma 9, $\mathcal{C}_{\text{seq}} < 0$ when:

$$\begin{aligned} \|\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})\| \|\nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta})\| \cos(\Phi) &< \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})^T \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}) & (3.86) \\ \cos(\Phi) &< \frac{\eta \kappa \|\nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})\|}{\|\nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta})\|} & (3.87) \end{aligned}$$

□

3.6.3 Special Cases and Implications

Corollary 3 (Competition Enhancement in Sequential Training)**.** *When task y has a convex loss ($\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_y(\boldsymbol{\theta}) \succeq 0$ and $\kappa > 0$), sequential training can induce competition even between naturally aligned tasks ($\Phi < \pi/2$). Competition emerges when:*

$$\eta > \frac{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_y(\boldsymbol{\theta})\| \cos(\Phi)}{\kappa\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_x(\boldsymbol{\theta})\|} \tag{3.88}$$

The critical angle where competition begins is $\Phi_c = \arccos\left(\frac{\eta\kappa\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_x(\boldsymbol{\theta})\|}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_y(\boldsymbol{\theta})\|}\right)$.

Example 1 (Isotropic Curvature)**.** *If $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_y(\boldsymbol{\theta}) = \iota\boldsymbol{I}$ with $\iota > 0$, then $\kappa = \iota$ and tasks compete when:*

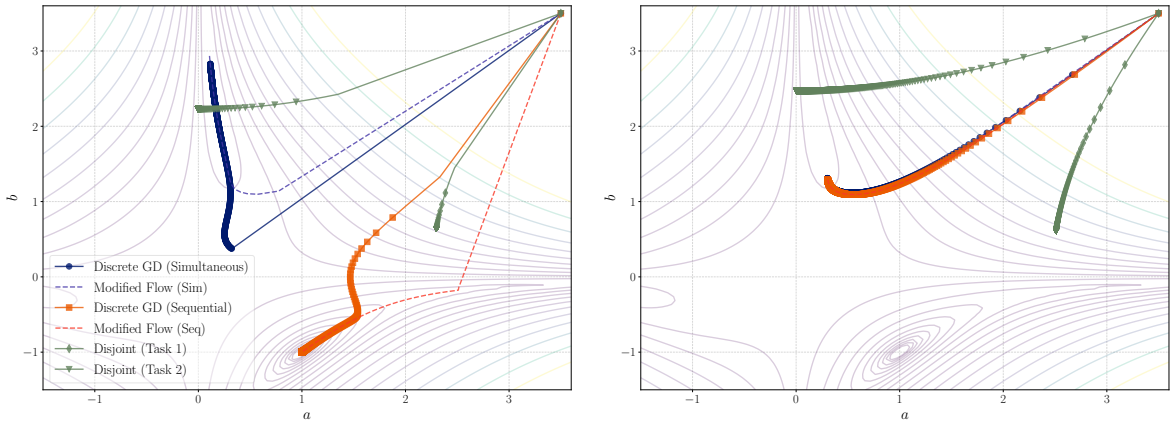
$$\Phi > \arccos\left(\frac{\eta\iota\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_x(\boldsymbol{\theta})\|}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}_y(\boldsymbol{\theta})\|}\right) \tag{3.89}$$

Remark 16 (Practical Implications). Our geometric framework reveals that:

- Multi-objective training maintains a fixed competition boundary at $\Phi = \pi/2$, preserving natural task alignment when $\Phi < \pi/2$
- Sequential training's competition boundary depends on η, gradient magnitudes, and loss curvature, potentially inducing artificial competition
- Smaller learning rates reduce the gap between training regimes, providing more predictable optimization dynamics
- For transformer architectures with shared attention mechanisms, these insights guide the choice between joint and sequential training strategies

3.7 Numerical Validation

The preceding sections have developed a comprehensive theoretical framework, predicting that joint and sequential training induce fundamentally different implicit biases. Specifically, our analysis points to a first-order gradient alignment term in joint training versus a more complex, second-order Hessian correction in sequential training, with the divergence between them amplified by the learning rate and loss curvature. To validate these predictions and demonstrate their tangible impact on optimization, we now turn to a series of numerical experiments. We design controlled settings to first visualize the optimization paths on a simple landscape, then measure the evolution of gradient alignment and solution flatness in a multi-task neural network.



(a) Learning rate $\eta = 0.01$

(b) Learning rate $\eta = 0.001$

Figure 3.2: Optimization trajectories for sequential versus multi-objective gradient descent under varying learning rates.

3.7.1 Optimization Trajectories

To validate our theoretical analysis of gradient flow dynamics in multi-task learning settings, we designed a simplified two-dimensional optimization problem that captures essential aspects of task interaction while remaining analytically tractable. Specifically, we examined the optimization trajectories of two distinct loss functions sharing a common parameter space:

$$\mathcal{L}_1(x, y) = \frac{1}{2}(xy^2 - 1)^2 \tag{3.90}$$

$$\mathcal{L}_2(x, y) = \frac{1}{2}(x^2 y + 1)^2 \tag{3.91}$$

These functions were deliberately constructed to possess different curvature properties, allowing us to observe how gradient interactions shape the optimization trajectory. The multi-objective loss function is the direct sum $\mathcal{L}_{\text{multi}}(x, y) = \mathcal{L}_1(x, y) + \mathcal{L}_2(x, y)$.

We examined four optimization approaches: (i) multi-objective gradient descent, updating parameters using the combined gradient $\nabla(\mathcal{L}_1 + \mathcal{L}_2)$; (ii) the modified gradient flow approximation of multi-objective updates; (iii) sequential gradient descent, which updates parameters first with respect to $\mathcal{L}_1$, then with respect to $\mathcal{L}_2$; and (iv) the corresponding sequential modified flow with Hessian correction. For comparison, we also tracked the trajectories of disjoint optimization, where each task is optimized independently.

Figure 3.2 presents optimization trajectories under two learning rate regimes: $\eta = 0.01$ (left) and $\eta = 0.001$ (right). The contour lines represent the combined loss landscape $\mathcal{L}_{\text{multi}}$. With the larger learning rate of $\eta = 0.01$, we observe a clear divergence between multi-objective and sequential optimization paths. The multi-objective approach follows one trajectory toward a particular local minimum, while the sequential approach takes a markedly different path converging to a distinct local optimum. Notably, the discrete gradient descent trajectories closely follow their corresponding modified flows, providing strong empirical confirmation of our theoretical analysis.

In contrast, with the smaller learning rate of $\eta = 0.001$, both multi-objective and sequential optimization trajectories become nearly indistinguishable, converging to the same local minimum. This observation precisely aligns with our theoretical prediction that the Hessian correction term in the sequential update scales quadratically with the learning rate ($\eta^2 \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})$). As this term becomes negligible at smaller learning rates, both methods essentially follow the standard gradient flow trajectory.

These results provide compelling evidence for our theoretical framework. The distinct optimization paths observed at larger learning rates demonstrate that the training regime—multi-objective versus sequential—can significantly impact the solution found through gradient-based optimization. Furthermore, the close correspondence between discrete updates and their continuous flow approximations validates our derivation of the modified gradient flow, including the Hessian correction term that emerges in the sequential setting.

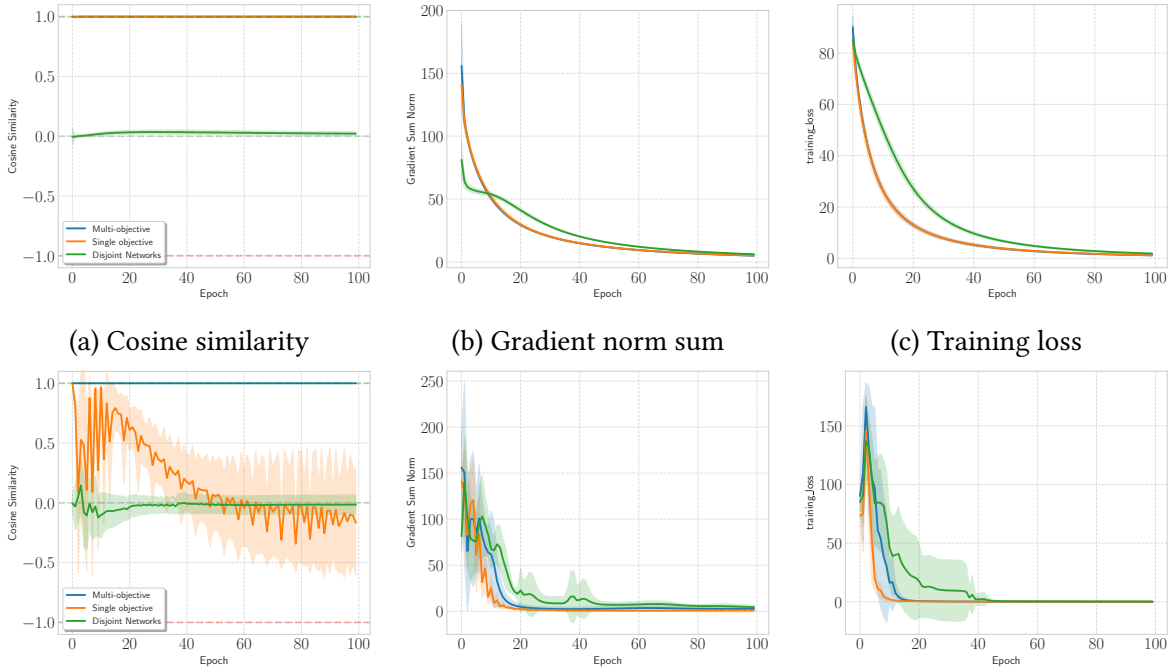
3.7.2 Implicit Regularization Effect on Gradient Norms

We conducted a comprehensive evaluation of our theoretical findings through carefully controlled experiments on synthetic datasets. This experimental design allowed us to precisely isolate the effects of our proposed training regimes and architectural choices. Our experimental protocol maintained consistent hyperparameter settings across all conditions, with networks trained for 100 epochs using a hidden dimension of 256 and 2 layers. To thoroughly assess the sensitivity of implicit regularization effects to step size, we examined two learning rates: $\eta = 0.001$ and $\eta = 0.01$.

Our investigation compared two fundamental architectural approaches. The first employed disjoint networks, where each task was modeled independently without parameter sharing. The second utilized a shared architecture with task-specific heads, where a common backbone was shared across tasks except for the final output layer. For the shared architecture, we explored two distinct training regimes: a multi-objective approach that simultaneously updated both task-specific heads using an aggregated loss, and a sequential regime that implemented a two-stage process, updating parameters first with respect to one task and then the second.

To analyze the implicit regularization effects in greater depth, we tracked three key metrics throughout training: cosine similarity between task gradients, norm of the summed gradients, and the training loss curve. These metrics allow us to directly examine how different training regimes influence the alignment of task gradients and validate our theoretical predictions regarding gradient disagreement.

Figure 3.3 presents the results for similar tasks, where both prediction heads were initialized with identical weights and optimized for the same objective. Under the larger learning rate regime ($\eta = 0.01$, bottom row), we observe a striking divergence in behavior between training approaches. The sequential training regime progressively drives tasks toward disagreement, as evidenced by the cosine similarity trending toward -1. This indicates that even with initially aligned tasks, the sequential update introduces gradient conflicts through the Hessian correction term. In contrast, the multi-objective approach maintains perfect gradient alignment (cosine similarity ≈ 1) throughout training. Notably, both train-



(a) Cosine similarity (b) Gradient norm sum (c) Training loss

(d) Cosine similarity (e) Gradient norm sum (f) Training loss

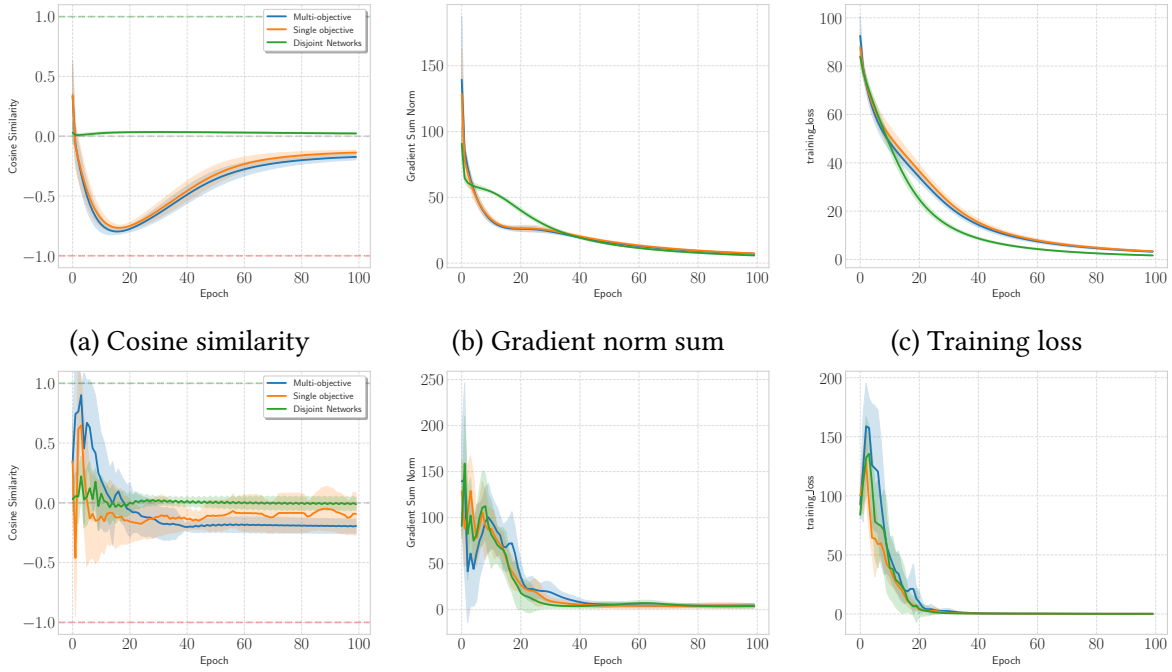
Figure 3.3: Optimization dynamics under similar task settings. The top row displays the evolution of (a) cosine similarity between the task gradients, (b) the norm of the summed gradients, and (c) the training loss for experiments conducted with a learning rate of 0.001. The bottom row presents the corresponding results for a learning rate of 0.01.

ing regimes maintain stable training loss and gradient norm profiles, confirming that the observed gradient alignment effects are not artifacts of divergent optimization dynamics.

Under the smaller learning rate ($\eta = 0.001$, top row), both training approaches exhibit nearly identical behavior, maintaining maximum task agreement throughout the optimization process. This pattern reinforces our theoretical prediction that the influence of the implicit regularization terms diminishes at smaller learning rates.

Figure 3.4 presents the corresponding analysis for dissimilar tasks, where each prediction head was assigned a different objective. In this scenario, both multi-objective and sequential training regimes encourage task disagreement regardless of learning rate. With the smaller learning rate ($\eta = 0.001$, top row), both approaches follow nearly identical paths of gradually increasing gradient disagreement. With the larger learning rate ($\eta = 0.01$, bottom row), the task disagreement emerges more rapidly, though the multi-objective training regime demonstrates more consistent behavior across experimental seeds, as indicated by the tighter confidence intervals.

When tasks have inherently competing objectives, the multi-objective training regime renders this competition explicit through its implicit regularization term $\frac{\eta}{2}\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \rangle$, which is minimized when gradients are anti-aligned. When tasks are naturally compatible, however, this same term preserves their alignment, maintaining higher training efficiency. The sequential training approach, by contrast, introduces an additional source of disagree-



(a) Cosine similarity (b) Gradient norm sum (c) Training loss

(d) Cosine similarity (e) Gradient norm sum (f) Training loss

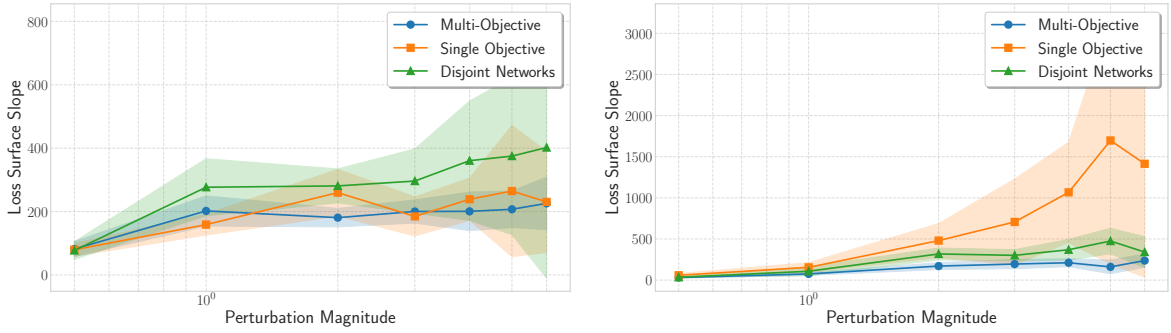
Figure 3.4: Optimization dynamics for dissimilar task settings. The top row shows the evolution of (a) cosine similarity between task gradients, (b) the norm of the summed gradients, and (c) the training loss for experiments conducted with a learning rate of 0.001. The bottom row presents the corresponding metrics for a learning rate of 0.01.

ment through the Hessian correction term $\eta^2 \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_y(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_x(\boldsymbol{\theta})$, which can induce artificial competition even between compatible tasks.

These findings reveal that the choice of training regime profoundly influences how multi-task models negotiate competing objectives, with the multi-objective approach providing a more direct and consistent mechanism for balancing task interactions. This has significant implications for attention-based models like Decision Transformers, where gradient disagreement may contribute to the formation of specialized attention patterns when trained on multiple prediction tasks.

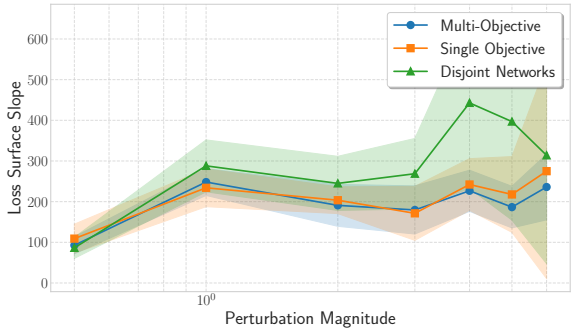
3.7.3 Implicit Regularization Effect on Flatness of Local Minima

The robustness of the obtained local minima was evaluated by systematically perturbing the weights of the trained models using multiplicative Gaussian noise. Specifically, for each model we applied noise with varying magnitudes (ranging from 0.0 to 0.4) and subsequently measured the test error, defined as the sum of mean squared errors over the tasks. By plotting the test error on a log-log scale as a function of the perturbation magnitude, we derived a quantitative estimate of the loss surface slope. This experimental design allows us to assess the flatness of the minima: flatter regions (i.e., lower slopes) are generally associated with better generalization. The analysis was carried out for models trained under

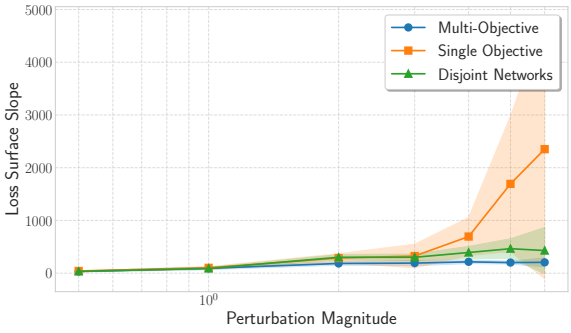


(a) Similar tasks, $\eta = 0.001$

(b) Similar tasks, $\eta = 0.01$



(c) Dissimilar tasks, $\eta = 0.001$



(d) Dissimilar tasks, $\eta = 0.01$

Figure 3.5: Loss surface analysis via weight perturbation. **Top row:** Results for models trained on similar tasks. **Bottom row:** Results for models trained on dissimilar tasks. For each setting, the left subfigure corresponds to a learning rate of 0.001 while the right subfigure corresponds to a learning rate of 0.01. The plots depict the change in the loss surface slope—computed as the variation in test error under multiplicative Gaussian perturbations applied to all network weights—with lower slopes indicating flatter minima and, consequently, enhanced generalization.

both similar and dissimilar task conditions, and for two learning rates (0.001 and 0.01), thereby providing a comprehensive evaluation of the implicit regularization effects inherent to the different training regimes.

3.8 Discussion and Practical Implications

Our theoretical analysis and empirical validation provide several actionable insights for practitioners designing multi-task learning systems. The fundamental finding that gradient descent implicitly encourages task disagreement, combined with the distinct mechanisms through which joint and sequential training manifest this bias, leads to concrete design principles.

3.8.1 Design Principles for Multi-Task Systems

The choice of learning rate emerges as a critical factor in multi-task optimization. Our analysis on learning rates of $\eta \in \{0.001, 0.01\}$ suggests that smaller learning rates reduce the gap between training regimes, potentially providing more predictable optimization dynamics. This is particularly important when the desired task interactions are known a priori, as larger learning rates can introduce unintended competition through the Hessian correction term in sequential training.

Task compatibility assessment becomes essential for selecting appropriate training strategies. Practitioners should monitor gradient alignment $\cos(\Phi)$ during early training epochs to identify potential task conflicts. When tasks exhibit natural alignment ($\Phi < \pi/2$), joint training preserves this beneficial cooperation, while sequential training may inadvertently induce artificial competition. Conversely, for inherently competing tasks ($\Phi > \pi/2$), both methods encourage disagreement, but sequential training amplifies this effect through curvature-dependent corrections.

The architectural choice between shared and task-specific parameters fundamentally alters the optimization landscape. As shown in Corollary 2, task interactions only arise through shared parameters. This suggests that careful architecture design—selectively sharing parameters based on expected task relationships—can modulate the strength of implicit regularization effects.

3.8.2 Implications for Transformer-based RL

Our theoretical framework provides insights into the different optimization dynamics of MO-DT and TT that may contribute to their performance differences in offline reinforcement learning settings. Recall that in the RL setting, our generic tasks x and y correspond to state and action prediction respectively, with the multi-objective loss $\mathcal{L}_{\text{multi}} = \mathcal{L}_s + \mathcal{L}_{DT} + \mathcal{L}_g$ in the MO-DT formulation. The joint training employed by MO-DT avoids the Hessian-induced competition that affects TT's sequential approach. This distinction becomes particularly crucial when predicting correlated modalities like states and actions, where artificial competition introduced by sequential training may hinder the model's ability to capture natural dependencies.

Our framework suggests that the choice between joint and sequential training should depend not only on computational constraints but also on the fundamental relationships between tasks. For transformer architectures processing multi-modal sequences, joint training provides a more principled approach when modalities exhibit natural correlations, while sequential training may be preferred when explicit task specialization is desired.

3.9 Conclusion

In this work, we developed a theoretical framework for understanding the implicit biases of gradient descent in multi-task learning. Through modified gradient flow analysis, we revealed that multi-task optimization fundamentally encourages task disagreement by implicitly minimizing the inner product between task gradients. This finding reveals a nuanced picture: while parameter sharing enables task interaction, the optimization dynamics through gradient descent can implicitly encourage task disagreement.

Our analysis uncovered critical differences between training regimes: while multi-objective optimization introduces a first-order regularization term that directly reflects gradient alignment, sequential training adds a second-order Hessian correction that can create artificial competition even between compatible tasks. By specializing to self-attention mechanisms, we demonstrated how these effects manifest in transformer architectures through attention weight redistribution, providing the first theoretical characterization of multi-task dynamics in attention-based models.

The geometric framework we developed offers practical insights for MTL design. Our conditions linking gradient angles, learning rates, and loss curvature enable practitioners to predict when tasks will cooperate or compete, guiding architectural and algorithmic choices. The empirical validation on controlled experiments confirms our theoretical predictions about optimization trajectories. These insights provide a potential explanation for performance differences between multi-objective and sequential training approaches, though the connection to complex models like transformers requires further investigation.

While our analysis provides valuable insights, several limitations merit future investigation. First, our theoretical results rely on simplified models—extending to full transformer architectures with layer normalization, residual connections, and multi-head attention remains an open challenge. Second, we focused on gradient descent dynamics, but modern optimizers like Adam may exhibit different implicit biases worthy of study.

Future research directions include: (i) developing optimization algorithms that explicitly control task interactions based on our geometric insights, (ii) extending the analysis to continual learning settings where tasks arrive sequentially, (iii) investigating how our findings translate to other multi-task architectures beyond transformers, and (iv) exploring connections between task disagreement and generalization performance. Our work provides a foundation for understanding multi-task optimization dynamics, opening new avenues for principled MTL algorithm design.

The content of this chapter forms the basis of the paper "When Tasks Collide: A Gradient-Flow Analysis of Alternating and Joint Training in Transformer Models", to be submitted.

Chapter 4

When Can Sequence Modeling Approaches Recover the Target Policy in Offline Reinforcement Learning? A Statistical Analysis

4.1 Introduction

Remark 17 (Connection to Previous Work). This chapter provides a theoretical analysis of sequence modeling approaches to offline RL, complementing the empirical methods developed in Chapters 1 and 2. While previous chapters focused on architectural innovations (MO-DT, TRDT, RGDT), here we establish fundamental sample complexity bounds for when these approaches can successfully recover optimal policies.

Offline RL addresses the challenge of learning effective policies from fixed datasets without online interaction with the environment [3, 9]. This paradigm is particularly relevant in domains such as robotics, logistics, and operations research, where exploration with untrained policies is impractical or unsafe. The offline RL setting has prompted the development of various approaches, initially focusing on adapting classical RL algorithms. These algorithms, such as off-policy methods [58, 139], were primarily designed for the online paradigm—a fundamentally different setting where the agent can interact with and learn from the environment in real-time. These adaptations typically incorporate mechanisms to mitigate action distribution shift while pursuing policy improvement [5, 3, 20, 65, 4]. The goal is to learn an optimal policy that maximizes expected return, leveraging the information contained in the offline dataset, which often contains data from multiple policies or training stages. However, off-policy methods in offline RL settings are known for their sensitivity to hyperparameters and lack a theoretical basis for selecting among different distribution shift mitigation strategies [20, 65].

The limitations of classical RL methods in offline settings have motivated a shift towards

framing offline RL as a supervised learning problem [140, 141]. Leveraging the inherently sequential nature of offline RL datasets, sequence modeling (SM) approaches have emerged as a promising direction [7, 8]. These methods offer several advantages over their off-policy counterparts, including algorithmic simplicity, reduced sensitivity to hyperparameters, and inherent resilience to action distribution shift [7]. Unlike off-policy techniques that aim to directly learn an optimal policy, SM approaches model the entire conditional distribution of policies present in the dataset, typically using transformer architectures [6, 38]. This comprehensive modeling approach, while powerful, introduces unique challenges. By capturing the full spectrum of policies, including suboptimal ones, these methods may be more susceptible to the influence of poor-quality data. During inference, the best policy is extracted by conditioning on the most high return contexts, but the success of this process heavily depends on the composition of the training dataset. The presence of suboptimal policies in the data could potentially hinder the extraction of truly optimal behavior, a challenge that is less pronounced in off-policy methods that explicitly target the best possible policy. This characteristic of SM approaches underscores the critical need for a thorough understanding of how dataset composition affects the quality of the extracted policy.

In this work, we address this challenge by providing a theoretical framework for analyzing the sample complexity of learning the target policy in offline RL using SM approaches. The analysis yields a novel bound on the required number of high-return samples, expressed in terms of the minimum number of samples and the expected minimum proportion of high-return data across contexts. This formulation allows for the characterization of the relationship between sample complexity and dataset composition, revealing distinct small-data and large-data regimes. A critical transition point between these regimes is identified and analyzed, providing insights into the diminishing returns of increasing dataset size beyond this point. The theoretical results suggest a fundamental trade-off between the breadth of context coverage and the depth of sampling within each context. This analysis may inform data collection strategies and algorithm design in offline RL, particularly in scenarios with imbalanced or limited data across different contexts.

4.2 Related Work

The study of sample complexity in reinforcement learning has a rich history, with seminal works establishing bounds for various settings [142, 143]. In the context of offline RL, recent research has focused on the challenges of distribution shift and policy constraint [144, 5]. SM approaches to RL, while relatively new, have shown promising empirical results [7, 21]. These methods draw inspiration from advances in natural language processing, particularly the use of transformer architectures [6]. Our work bridges the gap between these empirical advances and theoretical foundations, building on techniques from statistical learning theory [145].

4.3 System Model

We cast offline RL as a sequence modeling problem within a MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, P, R, and γ denote the state space, action space, transition probability function, reward function, and discount factor, respectively. We assume discrete data, noting that continuous spaces can be addressed through discretization techniques [21].

The core of our analysis revolves around a fixed-size static training dataset $\mathcal{T}$, comprising trajectories generated by T distinct *unknown* policies $\{\pi_k\}_{k=1}^{T}$. We transform $\mathcal{T}$ into a sequence modeling dataset $\mathcal{D} = \{(c^{[i]}, \ell^{[i]})\}_{i=1}^{N}$, where $c^{[i]} \in \mathcal{X}$ represents the context (e.g., previous states, actions, returns) and $\ell^{[i]} \in \mathcal{Y}$ represents the next token (typically an action). The vocabulary size $V = |\mathcal{Y}|$ corresponds to the number of possible actions, while $C = |\mathcal{X}|$ denotes the number of possible contexts. To enhance interpretability, our approach prioritizes actions as tokens, though the methodology generalizes to vocabularies that incorporate state and return tokens. Additionally, we adopt returns—defined as the cumulative sum of future rewards—as our primary reward metric. This choice is made without loss of generality, as alternative metrics, such as Monte Carlo value estimates [21], are also applicable.

Remark 18 (Relationship to Multi-Task Framework). This single-objective formulation specializes the multi-task framework of Chapter 3. While Chapter 3 uses continuous history representations $\boldsymbol{H}^{[i]}$ with separate token predictions $x^{[i]}, y^{[i]}$ for multiple tasks, here we work with discrete contexts $c^{[i]} \in \mathcal{X}$ that summarize the relevant history. We focus solely on predicting the next action $\ell^{[i]} \in \mathcal{Y}$, corresponding to restricting the multi-task framework to a single action prediction objective.

Remark 19 (Context Notation). In this chapter, $c^{[i]}$ denotes a discrete context from the finite set $\mathcal{X}$, contrasting with Chapter 3's $\boldsymbol{c}^{[i]}$ which represents a continuous vector in $\mathbb{R}^{d_{\text{in}}}$ computed via attention mechanisms.

To characterize the dataset composition, we define α_k as the *expected* proportion of samples in $\mathcal{D}$ generated by policy π_k, ensuring $\sum_{k=1}^{T} \alpha_k = 1$. We distinguish between high-return and low-return contexts, denoting $\mathcal{X}^h \subset \mathcal{X}$ as the set of high-return contexts and $\mathcal{X}^l = \mathcal{X} \setminus \mathcal{X}^h$ as the set of low-return contexts, with $C^h = |\mathcal{X}^h|$. For each policy π_k, we decompose $\alpha_k = \alpha_k^h + \alpha_k^l$, where α_k^h and α_k^l represent the proportions of high-return and low-return data, respectively. The overall expected proportions of high-return and low-return data are denoted as $\alpha^h := \sum_{k=1}^{T} \alpha_k^h$ and $\alpha^l := \sum_{k=1}^{T} \alpha_k^l$.

For a context $c \in \mathcal{X}^h$, let $N_c = N_c^h + N_c^l$ denote the number of samples in $\mathcal{D}$ containing c, where N_c^h and N_c^l represent the numbers of high-return and low-return samples, respectively. For $c \in \mathcal{X}^l$, we have that $N_c = N_c^l$. This decomposition is crucial, as even if a context c is in $\mathcal{X}^h$, not all of its occurrences in the dataset are necessarily optimal. This is particularly evident in episodic environments where trajectories near the end timesteps may have the same return, but the chosen actions might be sub-optimal depending on the policy. The illustrative example below provides further clarification on this point.

Illustrative Example Consider a 5×5 grid world where an agent must navigate from a start position to a goal. We define high-return trajectories as those reaching the goal in 10 steps or fewer. Let context A represent the agent's position one step away from the goal, and context B represent the starting position. Context A is classified as a high-return context ($A \in \mathcal{X}^h$) due to its proximity to the goal.

Consider a dataset $\mathcal{T}$ which contains the following two trajectories:

- $\boldsymbol{\tau}_1$: $B \to \cdots \to A \to$ Goal (10 steps, high-return)
- $\boldsymbol{\tau}_2$: $B \to \cdots \to A \to$ [suboptimal actions] $\to$ Goal (20 steps, low-return)

In this example, $N_A = 2$ (total occurrences of context A), while $N_A^h = 1$ (occurrences of A in high-return trajectories). Thus, $N_A \neq N_A^h$ despite $A \in \mathcal{X}^h$. This discrepancy arises because context A's classification as high-return is based on its potential for high returns, but the actual returns depend on subsequent actions in the trajectory. □

Finally, we use $\beta_c^h = \frac{N_c^h}{N}$ to denote the estimated proportion of high-return samples in which context c appeared. By design, we have that $\mathbb{E}\left[\sum_{c \in \mathcal{X}^h} \beta_c^h\right] = \alpha^h$.

4.4 Statistical Trajectory Model

We define each policy π_k as a conditional probability distribution over actions given contexts. Specifically, $\pi_k(v|c)$ represents the probability of taking action v given context c under policy k. For each context $c \in \mathcal{X}$, $\pi_k(\cdot|c)$ forms a probability distribution over the action vocabulary $[V]$, satisfying $\sum_{v=1}^{V} \pi_k(v|c) = 1$. Subsequently, we define the behavior policy π as a mixture of π_k and the target policy π^* for offline RL as follows:

$$\pi = \sum_{k=1}^{T} \alpha_k \pi_k \tag{4.1}$$

$$\pi^* = \frac{1}{\alpha^h} \sum_{k=1}^{T} \alpha_k^h \pi_k \tag{4.2}$$

Note that π^* uses α_k^h as coefficients, distinguishing it from the behavior policy π. An effective offline RL algorithm should aim to approximate π^*. Off-policy methods attempt this by directly modeling the policy with the maximum Q-value, effectively targeting π^*. In contrast, SM approaches model π, but attempt to recover π^* during inference by conditioning on contexts $c \in \mathcal{X}^h$ at each timestep. This approach implicitly assumes that high-return contexts are predominantly generated by π^*, theoretically allowing its recovery.

We define our learned model p as an estimate of the behavior policy, where $p(v|c) = \frac{1}{N_c} \sum_{l=1}^{N_c} X_l^{c,v}$, and $X_l^{c,v}$ are indicators of the occurrence of the pair (c, v) given c for the l-th sample in $\mathcal{D}$. We assume that p is an unbiased estimator of the true underlying distribution π, i.e., $\mathbb{E}[p] = \pi$. Consequently, $X_l^{c,v}$ can be interpreted as a Bernoulli random variable,

with the probability of $X_l^{c,v} = 1$ given by the conditional probability $\pi(v|c)$. Additionally, to simplify our theoretical analysis, we assume that the $X_l^{c,v}$'s are independent across different samples. It can be shown that the empirical conditional distribution p is obtained by minimizing the known categorical cross-entropy loss [146]. For theoretical analysis, we assume a simplified model where the *canonical* vectors of context-action pairs are directly used. In practice, these would be derived from a transformer model [6].

4.5 Problem Statement

Given a dataset with a minimum number of samples generated by the behavior policy for any high-return context, our goal is to find a lower bound on the minimum number of high-return samples needed so that our learned model approximates the target policy in offline RL.

Formally, let $\nu_{min}^h := \beta_{min}^h N_{min}$ be the minimum number of samples generated by π for any $c \in \mathcal{X}^h$ in $\mathcal{D}$, where $\beta_{min}^h = \min_{c\in\mathcal{X}^h} \mathbb{E}[\beta_c^h]$, and $N_{min} = \min_{c\in\mathcal{X}^h} N_c$. Our goal is to find a lower bound on ν_{min}^h such that our learned conditional model p approximates π^* on all $c \in \mathcal{X}^h$. To achieve this, we consider the matrix representations of p and π^*:

$$\boldsymbol{p} = (p(v|c))_{c\in\mathcal{X},v\in[V]} \in [0,1]^{C\times V} \tag{4.3}$$

$$\boldsymbol{\pi}^* = (\pi^*(v|c))_{c\in\mathcal{X},v\in[V]} \in [0,1]^{C\times V} \tag{4.4}$$

In order to measure the approximation error, we use the 1-norm:

$$\|\boldsymbol{p} - \boldsymbol{\pi}^*\|_1 = \sum_{c\in\mathcal{X}^h} \sum_{v\in\mathcal{Y}} |p(v|c) - \pi^*(v|c)| \tag{4.5}$$

Throughout the rest of the paper, we use $\sum_{c,v}$ to denote this double summation for brevity. It is worth noting that the 1-norm is twice the total variation distance $\|\boldsymbol{p} - \boldsymbol{\pi}^*\|_{\text{TV}}$, another commonly used metric for probability distributions [146, 147]. Additionally, we compare π^* and p only on contexts from $\mathcal{X}^h$, which delineates a key flexibility of offline RL compared to scenarios of *offline imitation learning* (i.e., BC) [148] where we would require the model to approximate π^* on all $c \in \mathcal{X}$.

4.6 Main Results

This section presents our main theoretical results, establishing sample complexity bounds for learning near-optimal policies in offline RL using SM approaches, and analyzes the implications of these bounds across different data regimes.

Theorem 6 (Sample Complexity Bound). *For any $\epsilon > 0$, if ν_{min}^h satisfies:*

$$\nu_{min}^h \geq \max\left\{\beta_{min}^h \left(\frac{C^h V}{\epsilon}\right)^2, N_{min}\left(1 - \frac{\epsilon}{4C^h}\right)\right\}, \tag{4.6}$$

then, we have that:

$$\mathbb{E}[\|\boldsymbol{p} - \boldsymbol{\pi}^*\|_1] < \epsilon \tag{4.7}$$

Proof. We begin by decomposing the error into variance and bias terms using the triangle inequality:

$$\mathbb{E}[\|\boldsymbol{p} - \boldsymbol{\pi}^*\|_1] \leq \mathbb{E}[\|\boldsymbol{p} - \mathbb{E}[\boldsymbol{p}]\|_1] + \|\mathbb{E}[\boldsymbol{p}] - \boldsymbol{\pi}^*\|_1 \tag{4.8}$$

Step 1: Bounding the variance term $\mathbb{E}[\|\boldsymbol{p} - \mathbb{E}[\boldsymbol{p}]\|_1]$.

We begin by expressing the 1-norm and applying linearity of expectation:

$$\mathbb{E}[\|\boldsymbol{p} - \mathbb{E}[\boldsymbol{p}]\|_1] = \sum_{c,v} \mathbb{E}[|p(v|c) - \mathbb{E}[p(v|c)]|] \tag{4.9}$$

Now, we apply Jensen's inequality to each term:

$$\begin{aligned}\sum_{c,v} \mathbb{E}[|p(v|c) - \mathbb{E}[p(v|c)]|] &\leq \sum_{c,v} \sqrt{\mathbb{E}[(p(v|c) - \mathbb{E}[p(v|c)])^2]} \\ &= \sum_{c,v} \sqrt{\text{Var}(p(v|c))}\end{aligned} \tag{4.10}$$

For each $p(v|c)$, we have that:

$$\text{Var}(p(v|c)) = \frac{\pi(v|c)(1 - \pi(v|c))}{N_c} \leq \frac{1}{4N_c} \leq \frac{1}{4N_{min}} \tag{4.11}$$

Therefore,

$$\mathbb{E}[\|\boldsymbol{p} - \mathbb{E}[\boldsymbol{p}]\|_1] \leq \frac{C^h V}{2\sqrt{N_{min}}} \tag{4.12}$$

Now, to ensure that $\mathbb{E}[\|\boldsymbol{p} - \mathbb{E}[\boldsymbol{p}]\|_1] \leq \epsilon/2$, we need:

$$\nu_{min}^h \geq \beta_{min}^h \left(\frac{C^h V}{\epsilon}\right)^2 \tag{4.13}$$

Step 2: Bounding the bias term $\|\mathbb{E}[\boldsymbol{p}] - \boldsymbol{\pi}^*\|_1$.

Since $\mathbb{E}[\boldsymbol{p}] = \boldsymbol{\pi} = \sum_{i=1}^{T} \alpha_i \boldsymbol{\pi}_i$, where $\boldsymbol{\pi}$ is the matrix representation of π, we have that:

$$\|\mathbb{E}[\boldsymbol{p}] - \boldsymbol{\pi}^*\|_1 = \left\| \sum_{i=1}^{T} \alpha_i \boldsymbol{\pi}_i - \frac{1}{\alpha^h} \sum_{i=1}^{T} \alpha_i^h \boldsymbol{\pi}_i \right\|_1 \tag{4.14}$$

$$= \left\| \sum_{i=1}^{T} \left(\alpha_i^l + \alpha_i^h - \frac{\alpha_i^h}{\alpha^h} \right) \boldsymbol{\pi}_i \right\|_1 \tag{4.15}$$

$$\leq C^h \left(\sum_{i=1}^{T} \alpha_i^l + \left| 1 - \frac{1}{\alpha^h} \right| \sum_{i=1}^{T} \alpha_i^h \right) \tag{4.16}$$

$$= 2C^h(1 - \alpha^h) = 2C^h \left(1 - \mathbb{E}\left[\sum_{c \in \mathcal{X}^h} \beta_c^h \right] \right) \tag{4.17}$$

$$\leq 2C^h(1 - \beta_{min}^h) \tag{4.18}$$

The key step is recognizing that $\|\boldsymbol{\pi}_i\|_1 = C^h$ for all i. To ensure that this is at most equal to $\epsilon/2$, the following inequality must be satisfied:

$$\nu_{min}^h \geq N_{min} \left(1 - \frac{\epsilon}{4C^h} \right) \tag{4.19}$$

Combining the two bounds completes the proof. □

Proposition 5 (Sample Complexity Regimes)**.** *The sample complexity ν_{min}^h exhibits distinct regimes as a function of the minimum number of samples N_{min}:*

1. *For $N_{min} \ll N_{min}^*$: $\nu_{min}^h \approx \beta_{min}^h \left(\frac{C^h V}{\epsilon} \right)^2$ (small-data regime)*
2. *For $N_{min} \gg N_{min}^*$: $\nu_{min}^h \approx N_{min} \left(1 - \frac{\epsilon}{4C^h} \right)$ (large-data regime)*

where the transition point N_{min}^* is given by:

$$N_{min}^* = \frac{4\beta_{min}^h (C^h)^3 V^2}{\epsilon^2 (4C^h - \epsilon)} \tag{4.20}$$

The analysis highlights key factors influencing sample complexity in offline RL, particularly emphasizing the role of worst-case context coverage. In data-scarce scenarios, reducing action space complexity and carefully selecting high-return contexts become critical. As data increases, the focus shifts to improving the minimum sample count for any high-return context, with diminishing returns from increasing overall dataset size. This suggests a fundamental trade-off between ensuring a good proportion of high-return samples across contexts (breadth) and sufficient samples per context (depth). These findings have significant implications for data collection strategies in offline RL, suggesting adaptive sampling methods that balance exploration of diverse contexts with exploitation of known high-return areas, particularly focusing on underrepresented high-return contexts in larger datasets.

Proposition 6 (Approximated Critical Minimum Sample Size). *Let N^*_{min} be the critical minimum sample size at which the two terms in the bound of Theorem 6 are equal. For $\epsilon \ll C^h$, N^*_{min} can be approximated as:*

$$N^*_{min} \approx \frac{\beta^h_{min}(C^h)^2V^2}{\epsilon^2} + \frac{\beta^h_{min}C^hV^2}{4\epsilon} \tag{4.21}$$

Proof. To express N^*_{min} in a more analytically tractable form, we consider the typical case where $\epsilon \ll C^h$. Under this assumption, we use a Taylor series expansion:

$$\frac{1}{4C^h - \epsilon} = \frac{1}{4C^h} \cdot \frac{1}{1 - \frac{\epsilon}{4C^h}} \tag{4.22}$$

$$\approx \frac{1}{4C^h}\left(1 + \frac{\epsilon}{4C^h} + \left(\frac{\epsilon}{4C^h}\right)^2 + \cdots\right) \tag{4.23}$$

$$\approx \frac{1}{4C^h}\left(1 + \frac{\epsilon}{4C^h}\right) \quad \text{(keeping only first-order terms)} \tag{4.24}$$

Substituting this back into our expression for N^*_{min} from (4.20) yields the result in (4.21). □

The formula for the critical minimum sample size N^*_{min} highlights key factors driving the transition between small-data and large-data regimes in offline RL. The dominant term, $\frac{\beta^h_{min}(C^h)^2V^2}{\epsilon^2}$, shows that N^*_{min} scales quadratically with the number of high-return contexts (C^h) and the action space size (V), and inversely with the accuracy (ϵ). This indicates that larger datasets are required for problems with complex action spaces or many high-return contexts to transition into the large-data regime.

The secondary term, $\frac{\beta^h_{min}C^hV^2}{4\epsilon}$, becomes more significant as ϵ increases, suggesting that lower accuracy thresholds require more data to balance growth. The linear dependence on β^h_{min} across the formula emphasizes that datasets with more evenly distributed high-return samples across contexts will require proportionally larger sizes to reach the transition point.

4.7 Numerical Results

We empirically validate our theoretical findings through controlled experiments.

Experimental Setup We designed a synthetic, controlled environment with $|\mathcal{S}| = 10$ states and $|\mathcal{A}| = 5$ actions, using an optimal policy (favoring first action with 0.7 probability), a uniform random policy, and a suboptimal policy. The behavior policy weights follow $\alpha^h_k \leq \alpha_k$ with controlled β^h_{min} and N_{min} to ensure consistent coverage across high-return contexts.

Results and Analysis For values of $\epsilon \in \{0.2, 0.3, 0.4, 0.5\}$, we calculated the theoretical minimum sample size ν^h_{min} and tested whether the resulting empirical error remained below the theoretical threshold.

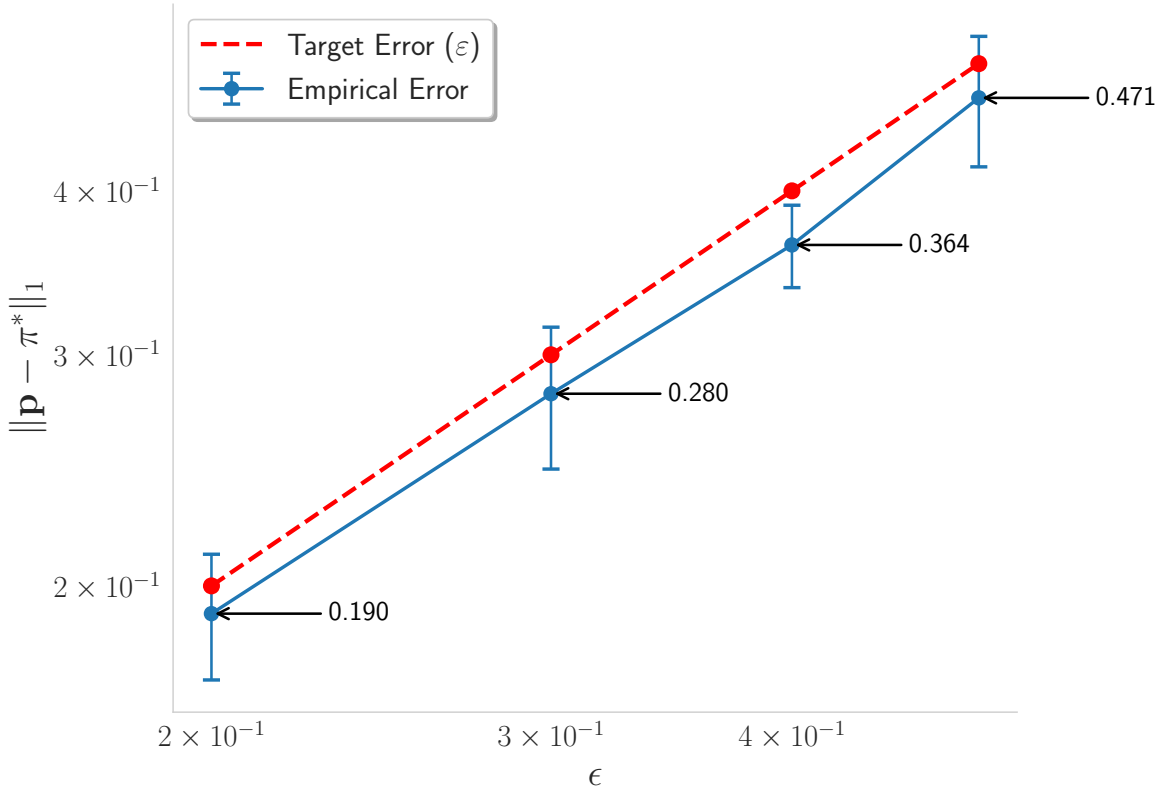


Figure 4.1: Empirical approximation error versus theoretical thresholds for varying ϵ values. Error bars represent standard deviation across 10 random seeds.

As shown in Figure 4.1, the empirical error $\|\boldsymbol{p} - \boldsymbol{\pi}^*\|_1$ consistently remains below the theoretical threshold ϵ across all tested values.

We then varied high-return samples as a fraction $\{0.1, 0.75, 1.0, 1.5, 2.0\}$ of the theoretical minimum to test bound tightness.

For $\epsilon = 0.2$ (Figure 4.2), using $0.75\times$ the theoretical sample size yields errors above threshold, while the full theoretical size maintains error below threshold, confirming our bound's tightness.

With the more stringent $\epsilon = 0.1$ (Figure 4.3), the required samples increase significantly, and using fewer samples than predicted consistently fails to achieve the target error.

Discussion Our experiments confirm that using the minimum number of high-return samples prescribed by our theory ensures the empirical error remains below the target threshold, while using fewer samples results in higher errors. This simultaneously validates both the effectiveness and tightness of our bounds, providing theoretically grounded guidance for minimum data requirements in offline RL.

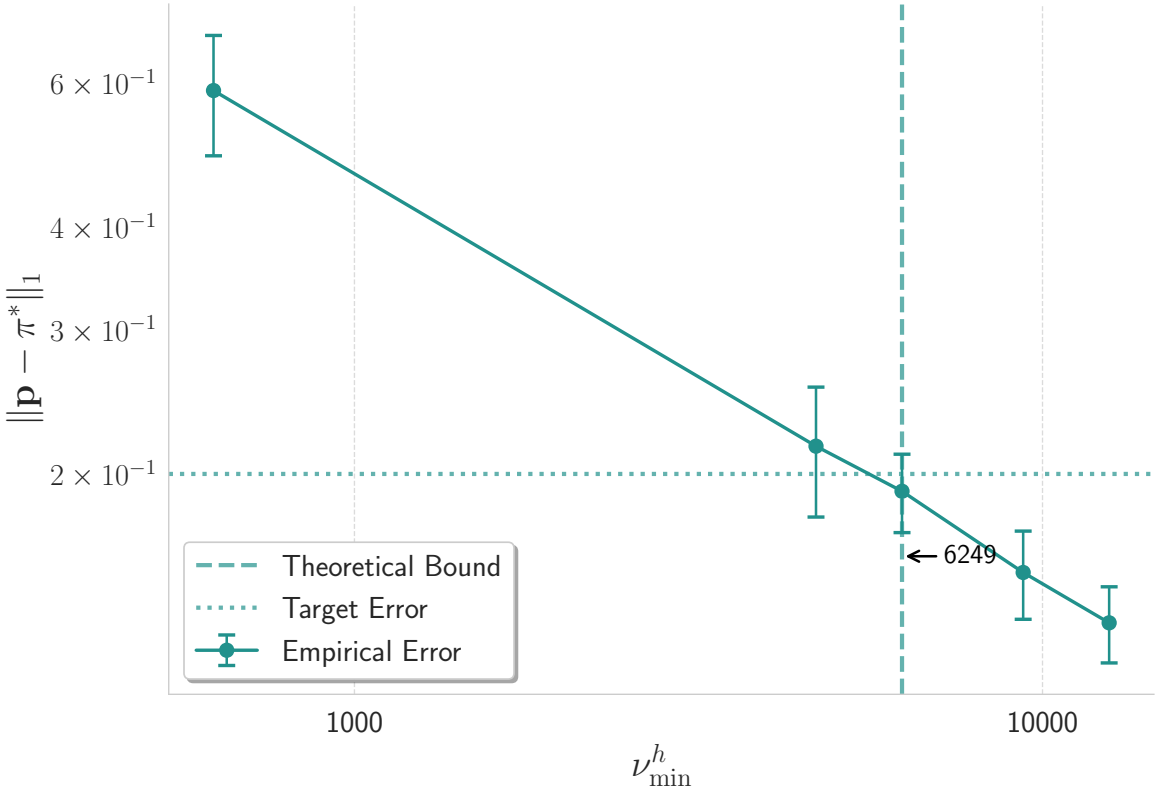


Figure 4.2: Approximation error as a function of high-return samples for $\epsilon = 0.2$. Vertical line indicates theoretical minimum sample size.

4.8 Conclusion and Future Directions

This theoretical analysis of sample complexity in offline RL using SM approaches reveals critical insights into the relationship between dataset composition and learning effectiveness. By identifying distinct small-data and large-data regimes separated by a critical transition point, the study challenges the notion that simply increasing dataset size is sufficient for improved performance. Instead, it emphasizes the importance of balanced data collection strategies that ensure adequate coverage of high-return contexts. The revealed trade-off between context coverage breadth and sampling depth suggests that adaptive sampling methods may be more effective than uniform strategies. These findings provide a foundation for developing more efficient offline RL algorithms and data collection strategies, potentially leading to improved performance in real-world applications where data collection is costly or constrained. Future work should focus on validating these theoretical results using realistic transformer architectures on standard offline RL benchmarks, while exploring their practical implications for algorithm design, particularly in developing efficient data selection and weighting strategies that bridge the gap between theoretical insights and real-world applications.

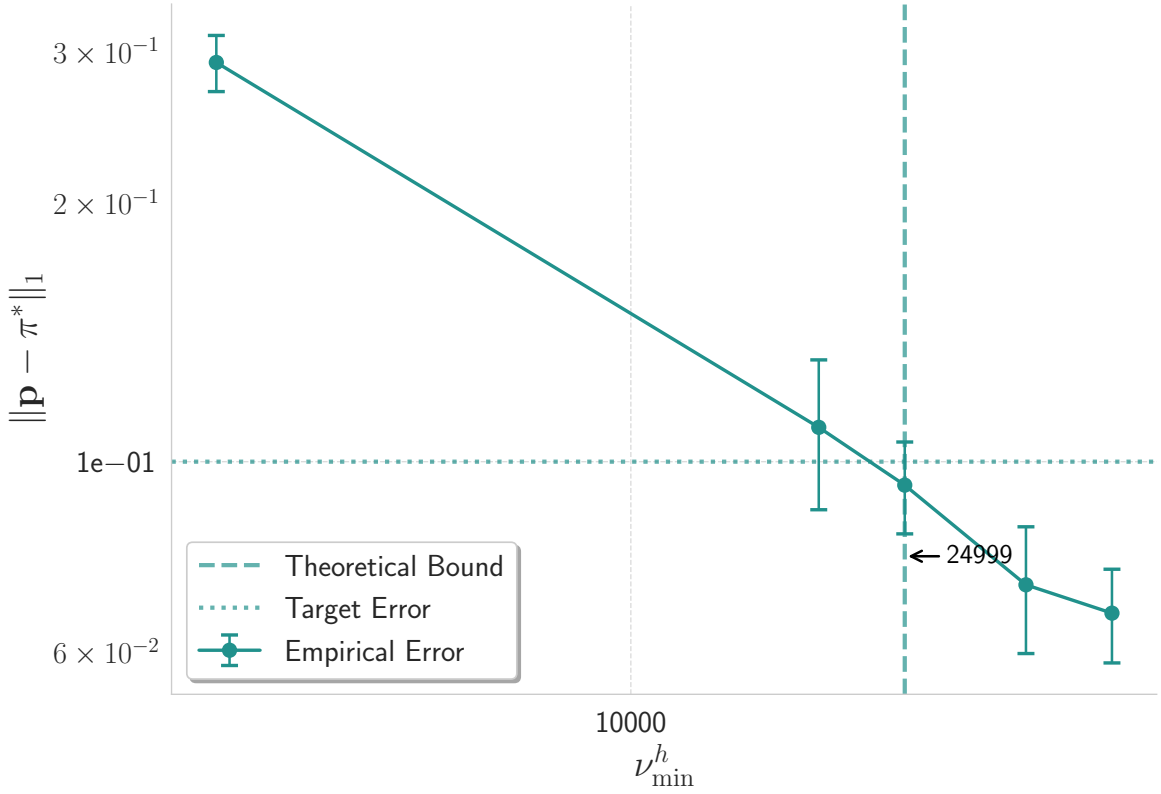


Figure 4.3: Sample complexity analysis for $\epsilon = 0.1$, showing increased sample requirements for lower error tolerance.

The content of this chapter is based on the paper "When Can Sequence Modeling Approaches Recover the Target Policy in Offline Reinforcement Learning? A Statistical Analysis", accepted for publication at European Signal Processing Conference (EUSIPCO 2025).

Conclusion and Future Perspectives

Synthesis of Contributions

This thesis has advanced the supervised learning paradigm for offline reinforcement learning through architectural innovations and theoretical analysis. Our investigation began with the empirical observation that Decision Transformers [7], despite their strong performance, exhibited homogeneous attention patterns that suggested suboptimal use of the transformer architecture. This observation motivated a series of contributions that enhance sequence modeling approaches to offline RL.

Our first contribution, the Multi-Objective Decision Transformer (MO-DT), demonstrated that single-task training was responsible for the homogeneous attention patterns. By jointly optimizing state, action, and return prediction objectives, we induced specialized attention patterns across different heads. The Trust Region Decision Transformer (TRDT) extended this multi-objective framework by augmenting trajectories with discretized action regions, reducing overfitting to specific behavioral patterns while maintaining action precision. Both methods achieved state-of-the-art performance on D4RL benchmarks.

The Return-Guided Decision Translator (RGDT) explored an alternative architectural paradigm, separate from the multi-objective framework. By reconceptualizing offline RL as a sequence-to-sequence translation problem using an encoder-decoder architecture, RGDT achieved effective disentanglement of modality processing. This approach demonstrated that decoder-only architectures are not the only viable option for trajectory modeling in offline RL.

Our theoretical contributions provided rigorous foundations for these empirical observations. The modified gradient flow analysis revealed that multi-task learning induces an implicit regularization term proportional to the inner product of task gradients, explaining why gradient descent encourages task specialization. This analysis was conducted on simplified self-attention models to maintain mathematical tractability. The Token-Priority Graph framework, applied specifically to our multi-objective setting, characterized how different training objectives create distinct attention hierarchies.

Our statistical analysis established sample complexity bounds for sequence modeling in offline RL, identifying the critical factors of context coverage and sampling depth. These bounds reveal a phase transition between variance-dominated and bias-dominated regimes, providing theoretical guidance for data collection strategies.

Technical Implications

Our findings have several technical implications for offline RL and transformer-based sequence modeling:

Multi-Task Training Dynamics: The modified gradient flow analysis shows that multi-task training fundamentally alters optimization dynamics through the cross-task gradient term $\frac{\eta}{2}\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_x, \nabla_{\boldsymbol{\theta}} \mathcal{L}_y \rangle$. This term is minimized when gradients are anti-aligned, providing a mechanistic explanation for the emergence of specialized representations.

Architectural Design Principles: The success of both decoder-only (MO-DT, TRDT) and encoder-decoder (RGDT) architectures suggests that architectural choices should be guided by the specific structure of the offline RL problem. The encoder-decoder architecture proved particularly effective for explicit return conditioning.

Sample Complexity Considerations: Our bounds show that the required number of high-return samples scales as $\mathcal{O}((C^h V/\epsilon)^2)$ in the small-data regime and linearly with N_{min} in the large-data regime, where C^h is the number of high-return contexts and V is the action vocabulary size.

Limitations and Open Problems

Several limitations of our work point to important open problems:

Experimental Scope: All experiments were conducted on D4RL locomotion tasks with relatively low-dimensional state and action spaces. Scaling to high-dimensional observations (e.g., images) and more complex action spaces remains unexplored.

Computational Analysis: We did not analyze the computational costs of our methods compared to baselines. While sequence modeling approaches avoid value iteration, the transformer architectures may have different scaling properties that warrant investigation.

Theoretical Gaps: Our theoretical analysis relied on simplified models. Extending the modified gradient flow analysis to practical transformers with layer normalization, residual connections, and multiple layers remains challenging. The gap between our simplified self-attention model and practical architectures is substantial.

Generalization Beyond D4RL: The effectiveness of our methods on other offline RL benchmarks and real-world applications requires further validation. The D4RL benchmark, while standard, may not capture all challenges in practical offline RL settings.

Partial Observability: Our methods assume fully observable states. Extending to partially observable settings would require architectural modifications to handle belief state representation.

Future Research Directions

Based on our contributions and identified limitations, we outline several concrete research directions:

1. Scaling to Complex Observations

Extending our methods to image-based observations requires addressing the computational and memory constraints of processing high-dimensional inputs. Potential approaches include:

- Hierarchical tokenization schemes that preserve spatial structure
- Efficient attention mechanisms designed for long sequences
- Integration with learned state abstractions

2. Foundation Models for Sequential Decision-Making

The supervised learning paradigm naturally enables pre-training on diverse trajectory datasets. This suggests the possibility of building foundation models that:

- Pre-train on trajectories from multiple environments and tasks
- Fine-tune efficiently on new tasks with limited data
- Leverage the multi-objective framework to learn general trajectory representations

The key challenge is designing architectures and training objectives that enable positive transfer across diverse sequential decision-making problems.

3. Theoretical Extensions

Several theoretical directions merit investigation:

- Extending modified gradient flow analysis to stochastic optimization algorithms (e.g., Adam)
- Analyzing the effect of architectural components (layer normalization, positional encodings) on implicit bias
- Establishing finite-sample convergence guarantees for multi-objective training
- Connecting our Token-Priority Graph analysis to the broader literature on attention mechanism interpretability

4. Bridging Offline and Online Learning

Developing principled methods to transition from offline pre-training to online fine-tuning while maintaining the simplicity of the supervised learning framework. This includes:

- Uncertainty-aware sequence models that can identify when to explore
- Safe exploration strategies that leverage the offline policy as a prior
- Theoretical guarantees on the sample complexity of online adaptation

5. Alternative Training Objectives

Our multi-objective framework used simple summation of losses. Future work could explore:

- Adaptive weighting schemes based on task uncertainty
- Hierarchical objectives operating at different temporal scales
- Contrastive objectives that explicitly encourage diversity

Technical Requirements for Deployment

The deployment of sequence modeling approaches to offline RL in safety-critical applications requires addressing several technical challenges:

Verification and Validation: Unlike value-based methods where Q-values provide interpretable value estimates, sequence models operate as black-box predictors. Developing methods to verify policy behavior and provide safety guarantees is essential.

Computational Efficiency: Real-time deployment requires efficient inference. While transformers have well-optimized implementations, the context length requirements for trajectory modeling may necessitate specialized architectures or approximations.

Distribution Shift Detection: Methods to detect when the deployment distribution differs significantly from the training distribution are critical for safe operation. This could leverage the attention patterns as indicators of out-of-distribution inputs.

Interpretability: Developing tools to interpret the decision-making process of sequence models, potentially leveraging our understanding of attention patterns and task specialization.

Concluding Remarks

This thesis has demonstrated that the supervised learning paradigm for offline reinforcement learning, introduced by the Decision Transformer, can be significantly enhanced through careful architectural design and multi-objective training. Our theoretical analysis provides a rigorous foundation for understanding why these enhancements succeed, revealing the implicit biases that shape learning in multi-task settings.

The key insight that multi-objective training induces specialized attention patterns through gradient disagreement has implications beyond offline RL. As transformers continue to be applied to diverse sequential decision-making problems, understanding how training objectives shape learned representations becomes increasingly important.

Our work contributes to the growing convergence between reinforcement learning and supervised learning communities. By demonstrating that trajectory modeling can be enhanced through architectural innovations and multi-task learning, we provide evidence that the boundaries between these fields are more fluid than traditionally conceived.

The supervised learning approach to offline RL offers practical advantages: stable training, reduced hyperparameter sensitivity, and the ability to leverage advances in sequence modeling. Our contributions show that these advantages can be amplified through principled design choices guided by theoretical understanding.

As the field progresses toward more complex applications, the principles established in this thesis—multi-objective learning for representation diversity, architectural flexibility for different problem structures, and rigorous theoretical analysis—will continue to guide the development of effective offline RL methods. The future of offline reinforcement learning lies not in choosing between dynamic programming and supervised learning paradigms, but in understanding how to combine their strengths to build systems that can learn safely and effectively from historical data.

Bibliography

[1] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[2] Jonas Degrave et al. "Magnetic control of tokamak plasmas through deep reinforcement learning". In: *Nature* 602.7897 (2022), pp. 414–419.

[3] Scott Fujimoto, David Meger, and Doina Precup. *Off-Policy Deep Reinforcement Learning without Exploration*. 2019. arXiv: 1812.02900 [cs.LG].

[4] Aviral Kumar et al. *Conservative Q-Learning for Offline Reinforcement Learning*. 2020. arXiv: 2006.04779 [cs.LG].

[5] Aviral Kumar et al. *Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction*. 2019. arXiv: 1906.00949 [cs.LG].

[6] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[7] Lili Chen et al. "Decision transformer: Reinforcement learning via sequence modeling". In: *Advances in neural information processing systems* 34 (2021), pp. 15084–15097.

[8] Michael Janner, Qiyang Li, and Sergey Levine. "Offline reinforcement learning as one big sequence modeling problem". In: *Advances in neural information processing systems* 34 (2021), pp. 1273–1286.

[9] Sascha Lange, Thomas Gabel, and Martin Riedmiller. "Batch reinforcement learning". In: *Reinforcement learning: State-of-the-art*. Springer, pp. 45–73.

[10] Chengzhong Ma et al. "Improving Offline Reinforcement Learning With In-Sample Advantage Regularization for Robot Manipulation". In: *IEEE Transactions on Neural Networks and Learning Systems* (2024), pp. 1–13. DOI: 10.1109/TNNLS.2024.3443102.

[11] Xin Wen et al. "CDCM: ChatGPT-Aided Diversity-Aware Causal Model for Interactive Recommendation". In: *IEEE Transactions on Multimedia* 26 (2024), pp. 6488–6500. DOI: 10.1109/TMM.2024.3352397.

[12] Yue Yun et al. “Doubly constrained offline reinforcement learning for learning path recommendation”. In: *Knowledge-Based Systems* 284 (2024), p. 111242. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2023.111242. URL: https://www.sciencedirect.com/science/article/pii/S0950705123009917.

[13] Haohong Lin et al. “Safety-Aware Causal Representation for Trustworthy Offline Reinforcement Learning in Autonomous Driving”. In: *IEEE Robotics and Automation Letters* 9.5 (2024), pp. 4639–4646. DOI: 10.1109/LRA.2024.3379805.

[14] Chamani Shiranthika et al. “Supervised Optimal Chemotherapy Regimen Based on Offline Reinforcement Learning”. In: *IEEE Journal of Biomedical and Health Informatics* 26.9 (2022), pp. 4763–4772. DOI: 10.1109/JBHI.2022.3183854.

[15] Ashvin Nair et al. *AWAC: Accelerating Online Reinforcement Learning with Offline Datasets*. 2021. arXiv: 2006.09359 [cs.LG].

[16] Yifan Wu, George Tucker, and Ofir Nachum. *Behavior Regularized Offline Reinforcement Learning*. 2019. arXiv: 1911.11361 [cs.LG].

[17] Sergey Levine et al. “Offline reinforcement learning: Tutorial, review, and perspectives on open problems”. In: *arXiv preprint arXiv:2005.01643* (2020).

[18] Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. “A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems”. In: *IEEE Transactions on Neural Networks and Learning Systems* 35.8 (2024), pp. 10237–10257. DOI: 10.1109/TNNLS.2023.3250269.

[19] Ilya Kostrikov et al. *Offline Reinforcement Learning with Fisher Divergence Critic Regularization*. 2021. arXiv: 2103.08050 [cs.LG].

[20] Scott Fujimoto and Shixiang Shane Gu. “A minimalist approach to offline reinforcement learning”. In: *Advances in neural information processing systems* 34 (2021), pp. 20132–20145.

[21] Michael Janner, Qiyang Li, and Sergey Levine. “Offline reinforcement learning as one big sequence modeling problem”. In: *Advances in neural information processing systems* 34 (2021), pp. 1273–1286.

[22] Scott Emmons et al. “RvS: What is Essential for Offline RL via Supervised Learning?” In: *arXiv preprint arXiv:2112.10751* (2021).

[23] Raphael Boige et al. *PASTA: Pretrained Action-State Transformer Agents*. 2023. arXiv: 2307.10936 [cs.AI]. URL: https://arxiv.org/abs/2307.10936.

[24] Kevin Clark et al. *What Does BERT Look At? An Analysis of BERT’s Attention*. 2019. arXiv: 1906.04341 [cs.CL]. URL: https://arxiv.org/abs/1906.04341.

[25] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[26] Yiran Li et al. *How Does Attention Work in Vision Transformers? A Visual Analytics Attempt*. 2023. arXiv: 2303.13731 [cs.LG]. URL: https://arxiv.org/abs/2303.13731.

[27] Perusha Moodley et al. *Multi-State-Action Tokenisation in Decision Transformers for Multi-Discrete Action Spaces*. 2024. arXiv: 2407.01310 [cs.LG]. URL: https://arxiv.org/abs/2407.01310.

[28] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. "Causal confusion in imitation learning". In: *Advances in neural information processing systems* 32 (2019).

[29] Russ Tedrake. *Underactuated Robotics. Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. 2023. URL: https://underactuated.csail.mit.edu.

[30] Jemin Hwangbo et al. "Control of a Quadrotor With Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 2.4 (2017), pp. 2096–2103. DOI: 10.1109/LRA.2017.2720851.

[31] Jongjin Park et al. "Object-aware regularization for addressing causal confusion in imitation learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 3029–3042.

[32] Justin Fu et al. *D4RL: Datasets for Deep Data-Driven Reinforcement Learning*. 2021. arXiv: 2004.07219 [cs.LG].

[33] Yingcong Li et al. "Mechanics of Next Token Prediction with Self-Attention". In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. Ed. by Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li. Vol. 238. Proceedings of Machine Learning Research. PMLR, 2024, pp. 685–693. URL: https://proceedings.mlr.press/v238/li24f.html.

[34] Juergen Schmidhuber. "Reinforcement Learning Upside Down: Don't Predict Rewards–Just Map Them to Actions". In: *arXiv preprint arXiv:1912.02875* (2019).

[35] Rupesh Kumar Srivastava et al. *Training Agents using Upside-Down Reinforcement Learning*. 2021. arXiv: 1912.02877 [cs.LG].

[36] Qinqing Zheng, Amy Zhang, and Aditya Grover. *Online Decision Transformer*. 2022. arXiv: 2202.05607 [cs.LG].

[37] David Brandfonbrener et al. "When does return-conditioned supervised learning work for offline reinforcement learning?" In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 1542–1553.

[38] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[39] Keiran Paster, Sheila McIlraith, and Jimmy Ba. “You can’t count on luck: Why decision transformers and rvs fail in stochastic environments”. In: *Advances in neural information processing systems* 35 (2022), pp. 38966–38979.

[40] Kuang-Huei Lee et al. “Multi-game decision transformers”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27921–27936.

[41] Ziqi Zhang et al. “Context-former: Stitching via latent conditioned sequence modeling”. In: *arXiv preprint arXiv:2401.16452* (2024).

[42] Kaizhe Hu et al. “Decision transformer under random frame dropping”. In: *arXiv preprint arXiv:2303.03391* (2023).

[43] Shengchao Hu et al. “Graph decision transformer”. In: *arXiv preprint arXiv:2303.03747* (2023).

[44] Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. *Can Wikipedia Help Offline Reinforcement Learning?* 2022. arXiv: 2201.12122 [cs.LG].

[45] Kerong Wang et al. *Bootstrapped Transformer for Offline Reinforcement Learning*. 2022. arXiv: 2206.08569 [cs.LG].

[46] Sachin G Konan, Esmaeil Seraj, and Matthew Gombolay. “Contrastive decision transformers”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 2159–2169.

[47] Yaru Hao et al. “Self-attention attribution: Interpreting information interactions inside transformer”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 14. 2021, pp. 12963–12971.

[48] Chong Li et al. “Interpreting and exploiting functional specialization in multi-head attention under multi-task learning”. In: *arXiv preprint arXiv:2310.10318* (2023).

[49] Paul Michel, Omer Levy, and Graham Neubig. “Are sixteen heads really better than one?” In: *Advances in neural information processing systems* 32 (2019).

[50] Tan Nguyen et al. “Improving Transformer with an Admixture of Attention Heads”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 27937–27952.

[51] Jian Li et al. *Multi-Head Attention with Disagreement Regularization*. 2018. arXiv: 1810.10183 [cs.CL].

[52] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[53] Wangchunshu Zhou et al. “Scheduled DropHead: A Regularization Method for Transformer Models”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1971–1980. DOI: 10.18653/v1/2020.findings-emnlp.178. URL: https://aclanthology.org/2020.findings-emnlp.178.

[54] Zewei Sun et al. “Alleviating the Inequality of Attention Heads for Neural Machine Translation”. In: *Proceedings of the 29th International Conference on Computational Linguistics*. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, Oct. 2022, pp. 5246–5250. URL: https://aclanthology.org/2022.coling-1.466.

[55] Timo Lohrenz et al. *Relaxed Attention: A Simple Method to Boost Performance of End-to-End Automatic Speech Recognition*. 2021. arXiv: 2107.01275 [eess.AS].

[56] Timo Lohrenz et al. *Relaxed Attention for Transformer Models*. 2022. arXiv: 2209.09735 [cs.LG].

[57] Catherine Olsson et al. “In-context learning and induction heads”. In: *arXiv preprint arXiv:2209.11895* (2022).

[58] Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. arXiv: 1801.01290 [cs.LG].

[59] Quoc V Le, Alex J Smola, and Stéphane Canu. “Heteroscedastic Gaussian process regression”. In: *Proceedings of the 22nd international conference on Machine learning*. 2005, pp. 489–496.

[60] Ozan Sener and Vladlen Koltun. “Multi-Task Learning as Multi-Objective Optimization”. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 525–536.

[61] Debabrata Mahapatra and Vaibhav Rajan. “Multi-Task Learning with User Preferences: Gradient Descent with Controlled Ascent in Pareto Optimization”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 6597–6607. URL: https://proceedings.mlr.press/v119/mahapatra20a.html.

[62] Jianlin Cheng, Zheng Wang, and Gianluca Pollastri. “A neural network approach to ordinal regression”. In: *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE. 2008, pp. 1279–1284.

[63] Yunhao Tang and Shipra Agrawal. *Discretizing Continuous Action Space for On-Policy Optimization*. 2020. arXiv: 1901.10500 [cs.LG].

[64] Greg Brockman et al. “Openai gym”. In: *arXiv preprint arXiv:1606.01540* (2016).

[65] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. “Offline reinforcement learning with implicit q-learning”. In: *arXiv preprint arXiv:2110.06169* (2021).

[66] Yang You et al. “Large batch optimization for deep learning: Training bert in 76 minutes”. In: *arXiv preprint arXiv:1904.00962* (2019).

[67] Davoud Ataee Tarzanagh et al. “Max-margin token selection in attention mechanism”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 48314–48362.

[68] Heejune Sheen et al. “Implicit Regularization of Gradient Flow on One-Layer Softmax Attention”. In: *arXiv preprint arXiv:2403.08699* (2024).

[69] Jie Yan, Quan Liu, and Lihua Zhang. “Offline Reinforcement Learning Based on Next State Supervision”. In: *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2024, pp. 5775–5779. DOI: 10.1109/ICASSP48485.2024.10446781.

[70] Lan Wu et al. “Offline Reinforcement Learning with Generative Adversarial Networks and Uncertainty Estimation”. In: *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2024, pp. 5255–5259. DOI: 10.1109/ICASSP48485.2024.10446266.

[71] Anurag Ajay et al. *Is Conditional Generative Modeling all you need for Decision-Making?* 2023. arXiv: 2211.15657 [cs.LG].

[72] Russ Tedrake. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. Course notes for MIT 6.832. 2023. URL: https://underactuated.csail.mit.edu.

[73] Ramin Ghorbani, Marcel J.T. Reinders, and David M.J. Tax. “RESTAD: Reconstruction and Similarity Based Transformer for Time Series Anomaly Detection”. In: *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. 2024, pp. 1–6. DOI: 10.1109/MLSP58920.2024.10734755.

[74] Hojjat Salehinejad et al. “Contrastive Representation of Channel State Information for Human Body Orientation Recognition in Interaction with Machines”. In: *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*. 2023, pp. 1–6. DOI: 10.1109/MLSP55844.2023.10285895.

[75] Anders Gjølbye et al. “Speed: Scalable Preprocessing of EEG Data for Self-Supervised Learning”. In: *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2024, pp. 1–6.

[76] Yiqi Wang et al. “A trajectory is worth three sentences: multimodal transformer for offline reinforcement learning”. In: *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*. Ed. by Robin J. Evans and Ilya Shpitser. Vol. 216. Proceedings of Machine Learning Research. PMLR, 2023, pp. 2226–2236. URL: https://proceedings.mlr.press/v216/wang23d.html.

[77] Xuenan Xu et al. “Investigating Passive Filter Pruning for Efficient CNN-Transformer Audio Captioning”. In: *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2024, pp. 1–6.

[78] Sungkyun Chang et al. “YourMT3+: Multi-Instrument Music Transcription with Enhanced Transformer Architectures and Cross-Dataset STEM Augmentation”. In: *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2024, pp. 1–6.

[79] Umberto Cappellazzo et al. “Parameter-efficient transfer learning of audio spectrogram transformers”. In: *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2024, pp. 1–6.

[80] Teerapat Jenrungrot et al. “Lmcodec: A low bitrate speech codec with causal transformer models”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[81] Pulkit Agrawal et al. *Learning to Poke by Poking: Experiential Learning of Intuitive Physics*. 2017. arXiv: 1606.07419 [cs.CV].

[82] Deepak Pathak et al. *Zero-Shot Visual Imitation*. 2018. arXiv: 1804.08606 [cs.LG].

[83] Rich Caruana. “Multitask learning”. In: *Machine learning* 28 (1997), pp. 41–75.

[84] Weixia Zhang et al. “Continual learning for blind image quality assessment”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3 (2022), pp. 2864–2878.

[85] Simon Vandenhende et al. “Multi-task learning for dense prediction tasks: A survey”. In: *IEEE transactions on pattern analysis and machine intelligence* 44.7 (2021), pp. 3614–3633.

[86] Zhizhong Li and Derek Hoiem. “Learning without forgetting”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2935–2947.

[87] Weixia Zhang et al. “Task-specific normalization for continual learning of blind image quality models”. In: *IEEE Transactions on Image Processing* (2024).

[88] Simeng Sun et al. “GraphIQA: Learning distortion graph representations for blind image quality assessment”. In: *IEEE Transactions on Multimedia* 25 (2022), pp. 2912–2925.

[89] Ronan Collobert and Jason Weston. “A unified architecture for natural language processing: Deep neural networks with multitask learning”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 160–167.

[90] Xiaodong Liu et al. “Multi-task deep neural networks for natural language understanding”. In: *arXiv preprint arXiv:1901.11504* (2019).

[91] Vincenzo Moscato et al. “Multi-task learning for few-shot biomedical relation extraction”. In: *Artificial Intelligence Review* 56.11 (2023), pp. 13743–13763.

[92] Jiasheng Si et al. “Biomedical argument mining based on sequential multi-task learning”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 20.2 (2022), pp. 864–874.

[93] Siddharth Sigtia et al. “Multi-task learning for speaker verification and voice trigger detection”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 6844–6848.

[94] Shinji Watanabe et al. “ESPnet: End-to-end speech processing toolkit”. In: *arXiv preprint arXiv:1804.00015* (2018).

[95] Sebastian Ruder. "An overview of multi-task learning in deep neural networks". In: *arXiv preprint arXiv:1706.05098* (2017).

[96] Yu Zhang and Qiang Yang. "A Survey on Multi-Task Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (2022), pp. 5586–5609. DOI: 10.1109/TKDE.2021.3070203.

[97] David G. T. Barrett and Benoit Dherin. *Implicit Gradient Regularization*. 2022. arXiv: 2009.11162 [cs.LG]. URL: https://arxiv.org/abs/2009.11162.

[98] Sanjeev Arora et al. "Implicit regularization in deep matrix factorization". In: *Advances in neural information processing systems* 32 (2019).

[99] Suriya Gunasekar et al. "Implicit regularization in matrix factorization". In: *Advances in neural information processing systems* 30 (2017).

[100] Samuel L Smith et al. "On the origin of implicit regularization in stochastic gradient descent". In: *arXiv preprint arXiv:2101.12176* (2021).

[101] Zhongwang Zhang and Zhi-Qin John Xu. "Implicit regularization of dropout". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.6 (2024), pp. 4206–4217.

[102] Michael Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. 2020. arXiv: 2009.09796 [cs.LG]. URL: https://arxiv.org/abs/2009.09796.

[103] Abdelghani Ghanem, Philippe Ciblat, and Mounir Ghogho. *Multi-Objective Decision Transformers for Offline Reinforcement Learning*. 2023. arXiv: 2308.16379 [cs.LG]. URL: https://arxiv.org/abs/2308.16379.

[104] Jonathan Baxter. "A model of inductive bias learning". In: *Journal of artificial intelligence research* 12 (2000), pp. 149–198.

[105] Simon Vandenhende et al. "Multi-task learning for dense prediction tasks: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* 44.7 (2021), pp. 3614–3633.

[106] Tianhe Yu et al. "Gradient surgery for multi-task learning". In: *Advances in neural information processing systems* 33 (2020), pp. 5824–5836.

[107] Zhao Chen et al. "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks". In: *International conference on machine learning*. PMLR. 2018, pp. 794–803.

[108] Bo Liu et al. "Conflict-averse gradient descent for multi-task learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18878–18890.

[109] Aviv Navon et al. "Multi-task learning as a bargaining game". In: *arXiv preprint arXiv:2202.01017* (2022).

[110] Suriya Gunasekar et al. “Characterizing implicit bias in terms of optimization geometry”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1832–1841.

[111] Navid Azizan and Babak Hassibi. “Stochastic gradient/mirror descent: Minimax optimality and implicit regularization”. In: *arXiv preprint arXiv:1806.00952* (2018).

[112] Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. “Implicit regularization of discrete gradient dynamics in linear neural networks”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[113] Blake Woodworth et al. “Kernel and rich regimes in overparametrized models”. In: *Conference on Learning Theory*. PMLR. 2020, pp. 3635–3673.

[114] Chulhee Yun, Shankar Krishnan, and Hossein Mobahi. “A unifying view on implicit bias in training linear neural networks”. In: *arXiv preprint arXiv:2010.02501* (2020).

[115] Zhiyuan Li, Tianhao Wang, and Sanjeev Arora. “What Happens after SGD Reaches Zero Loss?–A Mathematical Framework”. In: *arXiv preprint arXiv:2110.06914* (2021).

[116] Shahar Azulay et al. “On the implicit bias of initialization shape: Beyond infinitesimal mirror descent”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 468–477.

[117] Mo Zhou and Rong Ge. “Implicit regularization leads to benign overfitting for sparse linear regression”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 42543–42573.

[118] Jeff Z HaoChen et al. “Shape matters: Understanding the implicit bias of the noise covariance”. In: *Conference on Learning Theory*. PMLR. 2021, pp. 2315–2357.

[119] Jianqing Fan, Zhuoran Yang, and Mengxin Yu. “Understanding implicit regularization in over-parameterized single index model”. In: *Journal of the American Statistical Association* 118.544 (2023), pp. 2315–2328.

[120] Daniel Soudry et al. “The implicit bias of gradient descent on separable data”. In: *Journal of Machine Learning Research* 19.70 (2018), pp. 1–57.

[121] Mor Shpigel Nacson et al. “Convergence of gradient descent on separable data”. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 3420–3428.

[122] Ziwei Ji and Matus Telgarsky. “Directional convergence and alignment in deep learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17176–17186.

[123] Kaifeng Lyu and Jian Li. “Gradient descent maximizes the margin of homogeneous neural networks”. In: *arXiv preprint arXiv:1906.05890* (2019).

[124] Ziwei Ji and Matus Telgarsky. “Gradient descent aligns the layers of deep linear networks”. In: *arXiv preprint arXiv:1810.02032* (2018).

[125] Samy Jelassi, Michael Sander, and Yuanzhi Li. "Vision transformers provably learn spatial structure". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 37822–37836.

[126] Hongkang Li et al. "A theoretical understanding of shallow vision transformers: Learning, generalization, and sample complexity". In: *arXiv preprint arXiv:2302.06015* (2023).

[127] Yuchen Li, Yuanzhi Li, and Andrej Risteski. "How do transformers learn topic structure: Towards a mechanistic understanding". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 19689–19729.

[128] Samet Oymak et al. "On the role of attention in prompt-tuning". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 26724–26768.

[129] Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. "Trained transformers learn linear models in-context". In: *arXiv preprint arXiv:2306.09927* (2023).

[130] Yu Huang, Yuan Cheng, and Yingbin Liang. "In-context convergence of transformers". In: *arXiv preprint arXiv:2310.05249* (2023).

[131] Eshaan Nichani, Alex Damian, and Jason D Lee. "How transformers learn causal structure with gradient descent". In: *arXiv preprint arXiv:2402.14735* (2024).

[132] Siyu Chen et al. "Training dynamics of multi-head softmax attention for in-context learning: Emergence, convergence, and optimality". In: *arXiv preprint arXiv:2402.19442* (2024).

[133] Yu Huang et al. "Transformers provably learn feature-position correlations in masked image modeling". In: *arXiv preprint arXiv:2403.02233* (2024).

[134] Davoud Ataee Tarzanagh et al. "Transformers as support vector machines". In: *arXiv preprint arXiv:2308.16898* (2023).

[135] Bhavya Vasudeva, Puneesh Deora, and Christos Thrampoulidis. "Implicit bias and fast convergence rates for self-attention". In: *arXiv preprint arXiv:2402.05738* (2024).

[136] Yuandong Tian et al. "Scan and snap: Understanding training dynamics and token composition in 1-layer transformer". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 71911–71947.

[137] Christos Thrampoulidis. "Implicit bias of next-token prediction". In: *arXiv preprint arXiv:2402.18551* (2024).

[138] Zhongwang Zhang and Zhi-Qin John Xu. *Implicit regularization of dropout*. 2023. arXiv: 2207.05952 [cs.LG]. URL: https://arxiv.org/abs/2207.05952.

[139] Scott Fujimoto, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods". In: *International conference on machine learning*. PMLR. 2018, pp. 1587–1596.

[140] Scott Emmons et al. “RvS: What is Essential for Offline RL via Supervised Learning?” In: *arXiv preprint arXiv:2112.10751* (2021).

[141] Anurag Ajay et al. “Is Conditional Generative Modeling all you need for Decision Making?” In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=sP1fo2K9DFG.

[142] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

[143] Alexander L Strehl, Lihong Li, and Michael L Littman. “Reinforcement Learning in Finite MDPs: PAC Analysis.” In: *Journal of Machine Learning Research* 10.11 (2009).

[144] Sergey Levine et al. “Offline reinforcement learning: Tutorial, review, and perspectives on open problems”. In: *arXiv preprint arXiv:2005.01643* (2020).

[145] Vladimir Naumovich Vapnik, Vlamimir Vapnik, et al. “Statistical learning theory”. In: (1998).

[146] Mohamed El Amine Seddik et al. *How Bad is Training on Synthetic Data? A Statistical Analysis of Language Model Collapse*. 2024. arXiv: 2404.05090 [cs.LG]. URL: https://arxiv.org/abs/2404.05090.

[147] Shi Fu et al. *Towards Theoretical Understandings of Self-Consuming Generative Models*. 2024. arXiv: 2402.11778 [cs.LG]. URL: https://arxiv.org/abs/2402.11778.

[148] Shengyi Jiang, Jingcheng Pang, and Yang Yu. “Offline imitation learning with a misspecified simulator”. In: *Advances in neural information processing systems* 33 (2020), pp. 8510–8520.

DOCTORAL SCHOOL

Titre: Méthodes d'apprentissage supervisé pour l'apprentissage par renforcement hors ligne

Mots clés: Apprentissage par renforcement hors ligne, Transformers de décision, Modélisation de séquences, Optimisation multi-objectifs, Régularisation implicite, Mécanismes d'attention

Résumé: L'apprentissage par renforcement hors ligne vise à apprendre des politiques à partir de données statiques sans interaction avec l'environnement. Cette thèse améliore les approches par modélisation de séquences via de nouvelles architectures et analyses théoriques. Nous proposons les Multi-Objective Decision Transformers (MO-DT) et Trust Region Decision Transformers (TRDT) qui induisent des motifs d'attention diversifiés, ainsi que le Return-Guided Decision Translator (RGDT) utilisant une architecture encodeur-décodeur. Notre analyse théorique sur des modèles simplifiés révèle que l'apprentissage multi-tâches encourage implicitement le désaccord entre gradients et établit des bornes de complexité d'échantillonnage. Empiriquement, nos méthodes atteignent des performances compétitives sur les benchmarks D4RL de locomotion, avec TRDT améliorant le Decision Transformer jusqu'à 31% sur certaines tâches. Ce travail démontre l'efficacité des approches supervisées pour le RL hors ligne dans le domaine du contrôle continu.

Title: Supervised Learning Methods for Offline Reinforcement Learning

Keywords: Offline Reinforcement Learning, Decision Transformers, Sequence Modeling, Multi-Objective Optimization, Implicit Regularization, Attention Mechanisms

Abstract: Offline reinforcement learning learns policies from static datasets without environment interaction. This thesis advances sequence modeling approaches through novel architectures and theoretical insights. We introduce Multi-Objective Decision Transformers (MO-DT) and Trust Region Decision Transformers (TRDT) that induce diverse attention patterns, plus Return-Guided Decision Translator (RGDT) with encoder-decoder architecture. Theoretical analysis on simplified models reveals multi-task training implicitly encourages gradient disagreement. We establish sample complexity bounds for sequence modeling. Empirically, our methods achieve competitive performance on D4RL locomotion benchmarks, with TRDT improving Decision Transformer by up to 31% on certain tasks. This work demonstrates principled supervised learning effectively addresses offline RL challenges in continuous control domains.

Institut Polytechnique de Paris
91120 Palaiseau, France