Towards Foundation Models for Text-Rich Multimodal Tabular Data

by

Wei Min Loh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2026

© Wei Min Loh 2026

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Eugene Belilovsky Associate Professor, Gina Cody School of Engineering and Computer Science Concordia University
Supervisor:	Pascal Poupart Professor, David R. Cheriton School of Computer Science University of Waterloo
Internal Member:	Yuntian Deng Assistant Professor, David R. Cheriton School of Computer Science University of Waterloo
Internal Member:	Robin Cohen Professor, David R. Cheriton School of Computer Science University of Waterloo
Internal-External Member:	Qinglong Tian Assistant Professor, Department of Statistics and Actuarial Science University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The dissertation is composed of three research works conducted from 2022 to 2026. In all three works, I was the first author – conceptualizing the initial approach, conducting the literature review, implementing the code for the algorithms and models, and designing the experiments. My supervisor, Pascal Poupart, is a co-author of the three works. For each works, I outline the contributions of co-authors.

Chapter 3 presents a comprehensive investigation developed through a research collaboration with Manulife. My co-authors, Sajib Kumer Sinha, Ankur Agarwal, and Pascal Poupart, helped define the scope of the research, identified potential use cases and gave feedback on the project. Pascal Poupart also offered technical advice and edited the manuscript drafts.

Chapter 4 presents work on tabular models that was conducted mostly independently. Jiaqi Shang, a research assistant assigned to this project, was responsible for conducting the literature review and implementing some of the baselines for experiments. Pascal Poupart provided general supervision of the project and edited the manuscript drafts.

Chapter 5 is the outcome of a project in collaboration with the Vector Institute for the development of a reinforcement learning foundation model. My role in the project is producing an encoder for tabular data. Pascal Poupart provided general supervision of the project.

Abstract

Tabular data has been a central data format in statistics for centuries. In the age of machine learning and artificial intelligence, our set of tools has increased in size yet, many areas related to tabular modalities and tasks remain underexplored. Input representation is an important consideration for improving the current state of tabular approaches. Many successful approaches in purely numeric tabular data do not place much emphasis on input representations, and that may be limiting them to the purely numeric regime. On the other hand, the topic of information transfer is also one that practitioners need to address in the real world, especially when facing limited training data and evolving attributes. Information transfer can be in the form of continuously updating a model from a stream of data, or in the form of leveraging common information from a set of related tasks. This dissertation investigates these two areas, aiming to advance the current state of practical tabular models.

The issue of the continual adaptation of a tabular model is addressed by framing it as a multi-armed bandit. The clear advantage of this formulation is that the solution not only adapts to evolving needs at inference time but also has the ability to influence which data points to collect next. In this dissertation, we propose a framework based on Nadaraya-Watson kernel regression and Thompson sampling to continuously improve the existing model without updating model weight at inference time. This framework includes a provable guarantee on the upper bound on error using finite-sample analysis. Empirically, we demonstrated improvements on both the standard benchmark for multi-armed bandits and a newly proposed benchmark for realistic news recommendation tasks.

On the question of input representations, we started with the investigation of the current state of tabular models and identified important desiderata when working with tabular data. To the best of our knowledge, there are very few approaches in the current literature that satisfy the desiderata. We propose a transformer-based architecture, called basis transformers from the ground up, and introduce specialized components suited to the idiosyncrasies of tabular data. This architecture was evaluated on multi-task and related task regression experiments, and demonstrated improvements over gradient boosted decision trees, finetuned large language models, and similar deep tabular models.

Scaling in terms of the amount of data and the number of learnable parameters is the predominant technique for improving learned representations and performance in contemporary machine learning. In the field of tabular models, scaling is particularly challenging due to the heterogeneous formats and data types, and relatively few publicly available sources of tabular datasets because many tables contain proprietary and sensitive information. With extensive engineering and data processing efforts, we developed a multimodal

tabular foundation model, capable of extracting holistic representations and taking column names into account. The experiments demonstrated two main outcomes: the ability to perform zero-shot inference, which remains limited in tabular domains, and improvements over foundation model baselines on text-rich tabular datasets.

Acknowledgements

I would like to thank Pascal Poupart, my supervisor, who granted me the opportunity to embark on this intellectual adventure. These intense five years have enabled me to grow and hone my craft in machine learning, made possible by his guidance, advice, and mentorship.

As a student of a Vector Institute faculty member, it has been a privilege to receive funding as well as access to high-performance clusters, on which many of the experiments in my research heavily depended. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute https://vectorinstitute.ai/partnerships/current-partners/. I would also like to personally thank John Willes for his invaluable advisory support during his time at the Vector Institute, especially when I was facing challenges in large-scale engineering tasks.

This journey was made possible by the support of my parents and brother. It would not have been remotely possible without their emotional and financial support from the other side of the globe. My brother was my primary source of mathematical and scientific knowledge in my childhood, which undoubtedly piqued my interest over the years. It is indeed a blessing to be born into this family.

Jess has been the bedrock of this entire journey. As the only other graduate student I knew in my program who started in Winter 2022, she has been a confidant, an amazing supporter, and a fellow practitioner of machine learning; our late-night calls frequently involve deep talks or deep learning. Thank you for making this journey special and fulfilling.

Dedication

To those who sparked curiosity in my formative years — the nurturers

To those who guided me towards the culmination of my academic pursuits — the mentors

To those who kept me from being consumed by the fire of passion — the protectors

Table of Contents

Examining Committee Membership ii

Author's Declaration iii

Statement of Contributions iv

Abstract v

Acknowledgements vii

Dedication viii

List of Figures xv

List of Tables xviii

1 Introduction 1

1.1 Research Themes 2

1.1.1 Heterogeneity in tabular data 2

1.1.2 Learning style and information transfer 2

1.2 Thesis Overview 4

2 Background **5**
2.1 Multi-Armed Bandits . . . 5
2.1.1 Intuition . . . 5
2.1.2 Formulation . . . 6
2.1.3 Notable Bandits . . . 7
2.1.4 Strategy . . . 7
2.1.5 Regret Bound . . . 8
2.2 Transformers . . . 9
2.2.1 History . . . 9
2.2.2 Attention . . . 10
2.2.3 Transformer Architecture . . . 10
2.2.4 Notable Transformers . . . 13

3 Bandits in the Real World **14**
3.1 Motivation . . . 15
3.2 Related Work . . . 16
3.2.1 Contextual Bandits . . . 17
3.2.2 Other Relevant Bandits . . . 17
3.2.3 Recommender Systems . . . 18
3.3 Algorithm . . . 19
3.3.1 Problem Formulation . . . 19
3.3.2 Concept Drifts and Coupled Arms . . . 20
3.3.3 Nonparametric Estimation . . . 21
3.3.4 Importance Weighted Kernel Regression . . . 22
3.3.5 Embedding Model . . . 24
3.3.6 Computability of Coupling . . . 24
3.3.7 Conditionally Coupled Contextual Thompson Sampling . . . 26
3.3.8 Approximate Bayesian Update . . . 27

3.4 Theoretical Results . . . 29
3.5 Empirical Results . . . 37
3.5.1 Coupled Arms Simulation . . . 37
3.5.2 Contextual Bandit Experiments . . . 38
3.5.3 News Recommendation . . . 42
3.6 Discussion . . . 44
3.7 Conclusion . . . 45

4 A Universal Encoder for Tables **47**
4.1 Terminology . . . 48
4.2 Desiderata . . . 50
4.3 Related Work . . . 51
4.4 Method . . . 53
4.4.1 Vectored Numeric Representation . . . 53
4.4.2 Input Format . . . 55
4.4.3 Architecture . . . 55
4.4.4 Module Implementation . . . 63
4.4.5 Numeric Considerations . . . 64
4.5 Empirical Results . . . 68
4.5.1 Experimental Details . . . 68
4.5.2 Multi-Task Experiments . . . 70
4.5.3 Related Task Evaluation . . . 74
4.6 Discussion . . . 75
4.6.1 Understanding Basis Compression . . . 75
4.6.2 Memory consumption . . . 77
4.6.3 Standardization and one-hot encoding . . . 77
4.6.4 Connections to Bayesian multi-task learning . . . 77
4.7 Conclusion . . . 77

5 A Multimodal Tabular Foundation Model **79**
5.1 Motivation . . . 80
5.2 Related Works . . . 81
5.3 Data Processing . . . 82
5.3.1 Pretraining Data Source . . . 82
5.3.2 Internal Preprocessing . . . 83
5.3.3 Distance Correlation . . . 84
5.3.4 Caching and Storage . . . 85
5.4 Model Pretraining . . . 86
5.4.1 Architecture . . . 86
5.4.2 Pretraining Objective . . . 88
5.4.3 Miscellaneous Pretraining Details . . . 90
5.4.4 Hyperparameters and Pretraining Outcomes . . . 92
5.5 Model Finetuning . . . 94
5.5.1 XGBoost Head . . . 95
5.5.2 Parametric Dimensionality Reduction . . . 95
5.5.3 Hyperparameter Optimization . . . 96
5.6 Downstream Evaluation Results . . . 96
5.6.1 Evaluation Datasets . . . 97
5.6.2 Baselines and Evaluation Metrics . . . 98
5.6.3 Imputation Tasks . . . 99
5.6.4 Standard Regression Tasks . . . 100
5.6.5 Loss Function Ablation Study . . . 104
5.7 Discussion . . . 104
5.7.1 Zero-Shot versus In-Context Learning . . . 104
5.7.2 Pure Numeric Tasks . . . 105
5.7.3 Data quality . . . 106
5.8 Conclusion . . . 106

6 Conclusion and Future Work **108**
6.1 Summary of Contributions . . . 108
6.2 Future Work . . . 109

References **111**

APPENDICES **129**

A Supplementary Material for Chapter 3 **130**
A.1 Generation of Correlated Arms . . . 130
A.2 Correlation Experiment Details . . . 130
A.3 Ablation on RBF Bandwidth in C_3 . . . 132
A.4 MIND Experiment Details . . . 133
A.5 MIND Time Plot . . . 135

B Supplementary Material for Chapter 4 **137**
B.1 Hyperparameters for Scalar versus SMR Ablation . . . 137
B.2 Hyperparameters for Adaptive Loss Reweighing Ablation . . . 138
B.3 Main Experiment Hyperparameters and Configurations . . . 139
B.3.1 Datasets . . . 139
B.3.2 Training Routine . . . 139
B.3.3 Basis Transformer . . . 140
B.3.4 LLMs with TabLLM Routines . . . 141
B.4 Invalid Baselines and Justifications . . . 141
B.4.1 Special Cases . . . 142
B.4.2 General Cases . . . 143
B.5 Complete Multi-Task Results . . . 144
B.5.1 Computation of Scores . . . 144
B.5.2 Scores by Dataset and Algorithm . . . 144
B.6 Table Clustering . . . 149

C Supplementary Material for Chapter 5 **151**

C.1 RelBench Dataset Processing 151

C.2 Evaluation Task Configurations 152

C.2.1 Imputation Task 152

C.2.2 Standard Regression Task 153

List of Figures

2.1 Transformer architecture, with both encoder and decoder structures [151] . 11

3.1 A three-arm example of strong and weak coupling during concept drift of expected reward distributions. 21

3.2 An example of C_3 Beta posterior distribution in continuous spaces. The true μ of both arms for that context is 0.6. 27

3.3 Distance from the anchor arm embedding as a function of correlation ρ with 1.96 sigma error bars over 10 random seeds. 38

3.4 The left figure shows the arm embeddings prior to any fitting while the right figure shows the arm embeddings after training using Algorithm 1. The blue point is the anchor point and the green points are positively correlated to the μ of the blue point while the red points are negatively correlated. . . . 39

3.5 Cumulative regret of the test split of the four datasets with 1.96 sigma error bars over 10 random seeds. Note that in MNIST, LinTS cannot be computed due to numerical issues from the high dimensionality. In MagicTelescope, NeuralUCB and LinTS almost completely overlap because they all repeatedly exploit the same action after the initial steps. 40

3.6 Embedding space of MNIST digits with correct arms chosen where the left shows the embedding vectors of ϕ with an output dimension of 2, and the right shows the embedding vectors of another ϕ with an output dimension of 8 but uses t-SNE [147] for visualization. 41

3.7 Cumulative regret of the MIND dataset with 1 sigma error bars over 10 random seeds. "small" and "large" refer to the relative number of parameters in the two tower models. 43

4.1 The top shows the original input row; the bottom shows a (C, L, D)-tensor padded entry value representation. [P] refers to the pad tokens. The processed row (purple) has $C = 3$ columns (green), each having $L = 3$ tokens (yellow) of size D. $\vec{12}$ denotes the scalar 12 encoded with SMR. 56

4.2 A simplified diagram of a basis transformer block with the tensor dimensions of the inputs and the intermediate latent tensors. 56

4.3 A detailed flow chart showing how a basis transformer works. 65

4.4 Rescaled test R^2 score against log mean target value. Each point represents a dataset, averaged over 5 random seeds. Higher y-values are better. 66

4.5 Test performance with and without adaptive loss reweighing scheme over different degrees of reweighing factor γ. The shaded area represents the region of 1 standard error over 5 random seeds. Higher y-values are better. 69

4.6 Validation MMS-RMSE scores over gradient steps per dataset. The range of y-axis is [0, 1] and the range of x-axis is [0, 200]. The blue line is $\mathcal{BT}$. Lower values are better. 73

4.7 Heatmap of average attention weights over 64 basis queries and each token in BERT-encoded column names. 76

5.1 Illustration of the target column masking in pretrained basis transformers. 87

5.2 Illustration of the regressor head with projection shortcuts. 88

5.3 Loss curves of CS-BCE (left) and label smoothed BCE (right), both with a smoothing coefficient of $\alpha = 0.6$. The vertical dotted lines indicate the threshold of correctness from label smoothing for the corresponding target values. 91

5.4 A complete flow chart from raw tables to model optimization during pretraining. 93

5.5 Training loss (left) and magnitude ratio similarity monitoring trace (right) during the pretraining process of $\mathcal{PBT}$. 94

5.6 Critical difference diagram on the standard regression evaluation with a level of significance of $\alpha = 0.05$ 103

A.1 Visualization of impacts of various correlation ρ and c parameters on the μ of non-anchor arms. Each subplot contains 500 random samples. 131

A.2 Cumulative regret of C_3 with varying σ parameter (for the RBF kernel) on the Cover dataset with 1.96 sigma error bars over 10 random seeds. 132

A.3 Time taken in seconds against timesteps for each algorithm. 135

B.1 An enlarged view of validation MMS-RMSE scores over gradient steps per dataset. The shaded area represents the 1 standard deviation across 5 random seeds from the mean. The range of y-axis is [0, 1] and the range of x-axis is [0, 200]. The blue line is from $\mathcal{BT}$. Lower values are better. 148

List of Tables

3.1 Comparison between algorithms where linear refers to both LinUCB [92], LinTS [3], and SquareCB [46] while neural refers to both NeuralUCB [174] and NeuralTS [171]. n refers to the number of samples seen. 28

4.1 An example table to assist with the explanation of terminology. 48

4.2 Summary of related works and whether they fulfill each desideratum. A ✓means that the algorithm fulfills that desideratum, otherwise it does not. 53

4.3 Number of parameters, and median and inter-quartile range (IQR) of test MMS-RMSE and RMSE across 34 datasets and 5 random seeds for each model. For the data column: (I) refers to numeric + text (dense), (II) refers to numeric only, (III) refers to numeric + one-hot text (sparse). 72

4.4 Mean and standard deviation of success rate of language-capable models over 5 random seeds across 34 datasets. 72

4.5 Median and inter-quartile range (IQR) of test MMS-RMSE and RMSE across 18 related datasets and 5 random seeds for each model. 74

5.1 Additional internal preprocessing rules for T4 datasets. 83

5.2 Summary statistics of cached T4 datasets. 86

5.3 An example RelBench evaluation dataset from Amazon [71]. 98

5.4 Rank results of the imputation tasks. 100
5.5 Rank results of standard regression tasks. 102
5.6 Min-max scaled RMSE results of standard regression tasks. 102
5.7 Rank results of CS-BCE loss function ablation study in standard regression tasks. 104

A.1 Augmenting and diminishing probability schedules by day 134

B.1 Summary of invalid baseline algorithms and their justifications. 144
B.2 Mean MMS-RMSE over 5 random seeds by dataset and algorithm. 145
B.3 Mean RMSE over 5 random seeds by dataset and algorithm. 146
B.4 Mean success rate over 5 random seeds by dataset and algorithm. 147
B.5 Run times for each experiment including setup, training, and evaluations. 149
B.6 File names of the 18 tables from T4. 150

C.1 Baseline hyperparameters for imputation tasks. Hyperparameters omitted from this table use library defaults. 155
C.2 $\mathcal{PBT}$'s XGBoost head hyperparameter search space for the standard regression 156
C.3 Baseline hyperparameter optimization search spaces. 157
C.4 Standard regression configuration 158
C.5 Baseline hyperparameters for standard regression tasks. Hyperparameters omitted from this table use library defaults. 159

Chapter 1

Introduction

In the early 19$^{\text{th}}$ century, there was a well-known dispute between Gauss and Legendre over credit for the method of least squares [137]. Regardless of the outcome of the debate, the method laid the foundation for data-driven approaches in experimental sciences and beyond. At that time, the immediate application of the least squares approach was in astronomy and geodesy [137], as those were domains with substantial observational data with measurement error and inherent noise. Today, it is part of the standard toolkit for practitioners in statistical modeling.

Statistical modelling has roots in data from experimental science in what we now call *tabular data*. Pearson was instrumental in shaping statistical regression, which later became central to supervised learning over measured variables [115]. It was introduced in the context of hereditary and natural selection, where quantitative traits were used to obtain correlation coefficients.

While it may seem that tabular data emerged after the start of the scientific revolution, the tabular representation as a means of organizing records and data was used thousands of years ago [103] by Chinese cartographers and Roman engineers. The tabular format serves as an effective bridge between the human interpretation and the structured representation of many data points. The structured representation fits the demands of mathematical and statistical models, and was therefore adopted early in the age of experimental sciences.

The rapid digitalization of the world and the proliferation of collected data since the 20$^{\text{th}}$ century [67] have opened many opportunities in the field of statistics, leading to the birth of machine learning. Rather than consisting only of numbers organized into columns, collected data can include text and dates, and can consist of many columns. At this scale

and with heterogeneous data types within tables, many traditional statistical tools become insufficient.

Motivated by the historical significance and practical importance of tabular data, this dissertation will address certain key gaps in the current research literature and offer solutions that address these gaps.

1.1 Research Themes

This dissertation is organized around two overarching research themes.

1.1.1 Heterogeneity in tabular data

The interpretation of the term "tabular" in tabular data refers strictly to the manner in which data is organized – as a matrix with column headers. However, many approaches on tabular classification or regression focus only on the numeric subset of tabular data. In many modern applications, tabular data encompasses other data types including text and dates.

In addition, further heterogeneity comes in the form of context. Context is a critical element in tabular data as it adds a higher-level semantic layer to the values. Concretely, the column names provide context to their corresponding values. They are highly informative to human readers, which assists in the communication of what the data is about. In the current literature, however, column names are often not explicitly incorporated into the modeling process.

Therefore, my research aims to develop methods that natively and effectively handle heterogeneity in modern tabular data.

1.1.2 Learning style and information transfer

Tabular datasets are diverse in its structure and complexity, allowing a broad investigation into different strategies for leveraging tabular data. For example, the popular California Housing dataset [113] involving 8 numeric columns is considered a simple problem by today's standards, with respect to predicting median house prices. On the other hand, there are datasets found in RelBench [123] containing realistic tabular data representative

of industrial settings. The two types of tabular data are not directly comparable in terms of the modeling techniques to solve the respective tasks.

Some aspects of interest in my research are continual learning and transfer learning in the context of tabular modality and tasks. Wang et al. [153] defines continual learning as the ability of intelligent systems to learn from a sequence of data and incrementally accumulate knowledge from new learning signals. Weiss et al. [158] explains that transfer learning involves a predictive function leveraging related and useful information from a source domain to improve performance on a target domain.

These aspects are not arbitrary – they address diverse learning requirements. Three types of information transfer are considered.

Continual transfer of knowledge A learner is subjected to an environment where there is a high volume of tabular inputs in a sequential stream. While the input format remains consistent, the learner has to optimize with respect to a reward function that changes over time. The task is to continuously adapt to changes in the dynamics of the system one row of data at a time. While this idea is not unique to tabular data, it is likely one of the more feasible directions when subjected to high volumes due to the generally low feature dimensions.

Small-scale offline transfer of knowledge The objective is to improve the predictive performance of a model in a downstream task. Instead of using the training split of a specific task, the model is trained on the training splits of all related tasks simultaneously. The hypothesis is that knowledge transfer from a pool of related tasks allows a model to draw knowledge from a broader perspective and apply it to the target domain.

Large-scale offline transfer of knowledge As in the small-scale setting, the objective is to improve the predictive performance of a model. However, scaling from a small handful of related tasks to a massive collection of tables obtained from a wide variety of domains could possibly bring forth emergent properties and capabilities that would otherwise be impossible at a small scale. While knowledge transfer from minimally related tasks may be limited, the overall coverage from the sheer scale could improve performance on downstream tasks. Furthermore, when pretrained on that scale, zero-shot predictions could become a viable outcome, enabling a model to work without any task-specific training data.

1.2 Thesis Overview

Thesis Statement: The core contributions of this dissertation revolve around the following thesis statement.

This thesis argues that modern tabular learning requires models and algorithms that treat tables as heterogeneous, context-rich structures rather than purely numeric matrices, and that effective generalization depends on transferring information across changing environments, related tasks and large-scale collections of tables.

The organization of the chapters is briefly described below.

Chapter 2 covers the necessary background knowledge for the subsequent chapters. It contains broad fundamental knowledge on multi-armed bandits and the Transformer architecture. Key milestones in the respective fields are also discussed to highlight recent advancements.

Chapter 3 poses a multi-armed bandit problem in the context of commercial and industrial data. In the current literature, bandit algorithms are often analyzed theoretically, but this chapter examines the practical potential of bandit algorithms. In conjunction with modern tools such as neural networks and classical statistical tools such as Nadaraya-Watson kernel regression, the work in Chapter 3 delivers a framework that combines the best of both worlds. The main contributions are published in Transactions on Machine Learning Research in 2026 [98].

Chapter 4 considers the problem of heterogeneity in tabular data. While many works in the tabular literature assume purely numeric data types, this assumption often does not hold in practical applications. A new design is introduced to natively handle heterogeneous tabular data including textual entries and a variable number of columns, which enables multi-task capabilities and the ability to leverage knowledge from previous tasks for a related task. A preview of the main contributions is available on arXiv [97].

Chapter 5 develops a multi-modal tabular foundation model by extending the work in the previous chapter. It covers the details of the data processing pipeline, large-scale distributed pretraining and the improved architecture. The outcome is a foundation model that has been pretrained on data from over three million unique tables. An emergent outcome is the ability to perform zero-shot predictions at a level better than baseline methods.

Chapter 6 summarizes the contributions of this dissertation and outlines directions for future work.

Chapter 2

Background

Before diving into the contributions and their technical pieces, let us review some background information that might prove useful later on.

2.1 Multi-Armed Bandits

2.1.1 Intuition

A typical introduction to multi-armed bandits goes like this [156]: Imagine you are a gambler surrounded by k slot machines. As would any gambler, your goal is to maximize your monetary returns and this guides your strategy in selecting which of the k slot machines (or arms) to play over time.

Let us consider the two extremes in the space of possible strategies – an *impatient* person would randomly try any of the slot machines, and the moment they obtain a decent payout, they would focus all their remaining time on playing that particular "lucky" slot machine. At the other extreme, there is an *indecisive* person. They would play the k slot machines in an arbitrary order but remain unfazed even when winning big - perpetuating this cycle.

In the bandit literature, the impatient person follows a pure exploitation strategy while the indecisive person follows a pure exploration strategy. The general goal is to balance this exploration-exploitation trade-off. Exploration allows us to gain information about the environment in relation to our actions, but exploitation allows us to accelerate towards the goal of maximizing rewards. Studying bandit strategies requires us to think in terms of

counterfactual reasoning – the "what if's ". When you do not have a good understanding of the environment, in the event of a high reward, it would be natural to think "what if I just got lucky?". However, as you develop a deeper understanding of causality in the environment, the "what if's" become less credible.

With this intuition in mind, we move on to the mathematical formulation of multi-armed bandits.

2.1.2 Formulation

We formulate the stochastic bandit setting [87].

Definition 1. A **stochastic bandit** consists of a countable set of arms $\mathcal{A}$ and a reward distribution associated with each arm $a \in \mathcal{A}$. Across a finite or infinite time horizon, the goal is to maximize the sum of rewards.

There are several implicit assumptions. First, the rewards (and hence the sum of rewards) are random variables. Second, the player does not know the horizon, which could influence the optimality of the strategy. Finally, the player does not have complete information about the distribution of rewards for each arm beforehand and must obtain information about it through sampling.

An important consideration in bandit problems is how performance is quantified, which is conventionally quantified using *regret*.

Definition 2. Suppose the reward distribution for each arm $a \in \mathcal{A}$ has a finite mean μ_a. Let the optimal arm $a^* \in \mathcal{A}$ be the arm that has the highest expected reward

$$\mu^* = \max_{a \in \mathcal{A}} \mu_a \tag{2.1}$$

The **regret** of arm $a \in \mathcal{A}$ is defined to be the difference between μ^* and μ_a.

Leveraging some hindsight information allows us to identify when the player achieved the best possible reward at the time step. Minimizing cumulative regret is equivalent to maximizing the sum of rewards. However, the regret formulation is generally more helpful for theoretical analyses, especially to show whether an algorithm will converge to an optimal policy (sublinear regret) or a local optimum (linear regret).

2.1.3 Notable Bandits

There are several classes of bandits that are worthy of mentioning in the background, due to their applicability in the 21st century.

Contextual bandit refers to a bandit setting where an additional informative input given to the player when deciding which arm to pull [92]. This additional information is called *context*. The introduction of context transforms the reward distribution of an arm into a distribution conditioned on the context.

Combinatorial bandit refers to a bandit where instead of having the choice to pull an arm, the player has to decide a set of arms to pull [26]. This induces a combinatorial problem as there is a unique reward distribution for each combination of arm pulls.

Restless bandit deviates from the original formulation by introducing states of each arm, and each arm undergoes changes according to a transition matrix when pulled [59]. This involves the reward distribution of the pulled arm.

Non-stationary bandit is similar to restless bandits in that the reward distribution of arms changes, but differs in that the reward distribution of all arms varies over time [165]. A consequence is a harsh constraint against exploitation as information gained from exploration will eventually be lost, and will require a higher level of maintained exploration.

2.1.4 Strategy

The general strategy of bandit algorithms is called *greedy in the limit with infinite exploration* (GLIE), an idea thoroughly analyzed by Singh et al. [132]. It states that, for optimal convergence, all state-action pairs need to be explored infinitely often, as the policy gradually becomes greedy. In the limit, the policy is greedy with respect to the estimated reward with probability of 1 [132]. GLIE was introduced for reinforcement learning but it also reflects bandit algorithms as bandits can be seen as an online one-state reinforcement learning environment.

As GLIE suggests, a bandit algorithm will need to start with strong exploration but converge towards a greedy policy. The question is – *how does it algorithmically decide*

the rate of change from exploration to exploitation? Bandit algorithms typically leverage uncertainty quantification tools to address this problem, especially in quantifying *epistemic uncertainty* (variability from lack of information in a model). Uncertainty quantification estimates the degree of uncertainty associated with a model's prediction for a particular input, which is analogous to how much more exploration is needed for the aforementioned input. The beauty of uncertainty quantification is that they offer a principled and grounded statistical approach to quantify uncertainty, or at least a numeric approximation to the intractable variants.

Viewing quantified epistemic uncertainty as another quantity besides rewards forms the basis of modern bandit algorithms [152]. The simplest approach is introduced by Auer et al. [6] called the upper confidence bound (UCB) algorithm. For a finite set of K arms where their current estimates of reward are $\{\hat{\mu}_k : k \in [K]\}$ and the number of times each arm has been pulled is $\{n_k : k \in [K]\}$, pick the arm that maximizes

$$\underset{k}{\operatorname{argmax}}\, \hat{\mu}_k + \sqrt{\frac{2 \log n}{n_k}} \tag{2.2}$$

where n is the current round, or equivalently, the sum of $n_1, \ldots, n_K$. The second term comes from the Chernoff-Hoeffding bound on the sum of rewards, which can be seen as a degree of uncertainty on the mean rewards. The sum of both terms aligns with the idea of *optimism in the face of uncertainty* [112], which can be seen as the other pair of GLIE. It works by adding an exploration bonus to the current estimate of the mean reward [152], incentivizing exploration of arms infrequently pulled.

Observe that, through an asymptotic lens, $\log n$ grows much slower than n_k. This means that as n grows to infinity, GLIE conditions remain satisfied.

2.1.5 Regret Bound

Regret bounds are concerned with the growth rate of error incurred as the number of rounds goes to infinity. This asymptotic quantity is typically the primary goal of theoreticians in the bandit literature.

A useful regret bound should not be linear or worse. It would suggest that the error rate is constant over the number of rounds, questioning the ability of the player to learn. The bound should also never be a constant as it suggests that the player stops exploring at the predefined time. Recall that a bandit algorithm should be sufficiently general to handle a diverse range of problems. A player deciding to stop exploration at a predefined

time could fail on a more difficult problem. Most importantly, GLIE refers to the limit of infinite exploration, which is never practically achieved so an optimal player would never entirely stop exploration.

The UCB algorithm has a (gap-independent) regret bound of $\mathcal{O}(\sqrt{nK\log(n)})$ which is a fairly good bound. For more complex bandit variants such as contextual bandits, the contextual zooming algorithm has a Lipschitz contextual regret bound of $\tilde{\mathcal{O}}(n^{(d+1)/(d+2)})$ [133]. The quantity d refers to the dimension of the context, suggesting that learning becomes more difficult as the dimensionality of the context space increases.

2.2 Transformers

2.2.1 History

Transformers were introduced by Vaswani et al. in the paper titled "Attention is all you need" [151]. It was most strongly associated with natural language processing at the time of publication but became influential in other areas such as computer vision [78] soon after. Before its success in natural language processing, recurrent neural networks [127] and their popular variants such as long short-term memory (LSTM) [68] and gated recurrent units (GRU) [32] were the dominant architectures in the field. Seq2seq [139] refined how previous recurrent neural networks work, with one of the improvements coming from using an encoder-decoder architecture.

A common limitation of those approaches is the poor handling of long-range dependencies across sequences. The community focused its efforts on solving that issue. One notable development is the paper that introduced what we now call Bahdanau attention [8] – a primitive form of attention to assign weights to the most relevant positions of the sequence. This allowed decoders to look up all encoder states without compressing them into a single hidden state. However, at this point, attention was still used alongside a recurrent neural network.

Transformers came about from the realization that attention is not simply an enhancement to sequence-to-sequence tasks, but rather the core mechanism itself, which we will discuss the pivotal improvements below.

2.2.2 Attention

The variant of attention introduced by Vaswani et al. is called *scaled dot-product attention*. The operation takes three matrices (or tensors) Q, K, V, with dimensions $\mathbb{R}^{L_q \times d_k}$, $\mathbb{R}^{L_k \times d_k}$ and $\mathbb{R}^{L_k \times d_v}$, respectively. These matrices are called the *query*, *key* and *value* matrices respectively. The operation involves:

$$\text{SDPA}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \in \mathbb{R}^{L_q \times d_v} \tag{2.3}$$

This is a highly parallel operation since it involves only matrix operations without any recurrence or complex operations. The output of the softmax is sometimes called the *attention weights*, which is useful to determine the relative importance of positions in the sequence. However, attention should learn a variety of patterns and relationships, which leads us to multi-head attention.

$$\text{MHA}(Q, K, V) = \text{concat}\left(O_1, ..., O_h\right) W^O \tag{2.4}$$

where $W^O \in \mathbb{R}^{hd_v \times d_v}$ is the output projection matrix, and each O_i head is the output of scaled dot-product attention with individual projections of each matrix:

$$O_i = \text{SDPA}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.5}$$

By default, attention refers to multi-head attention. There are two specialized types of attention: self-attention and cross-attention. *Self-attention* is the special case in which all three input matrices are the same sequence. This can be seen as learning intra-sequence relationships. *Cross-attention* is the special case when the key and value matrices are the same sequence. This can be seen as learning relationships between two sequences.

Remark. Even when the matrices are the same in self-attention or cross-attention, the linear projections are individually learned which would most likely learn non-trivial attention weights.

2.2.3 Transformer Architecture

While attention is integral to the success of transformers, there are many other crucial components. Unlike recurrent neural networks, which inherently preserve information about sequence order, attention does not encode order by itself. Positional information of the

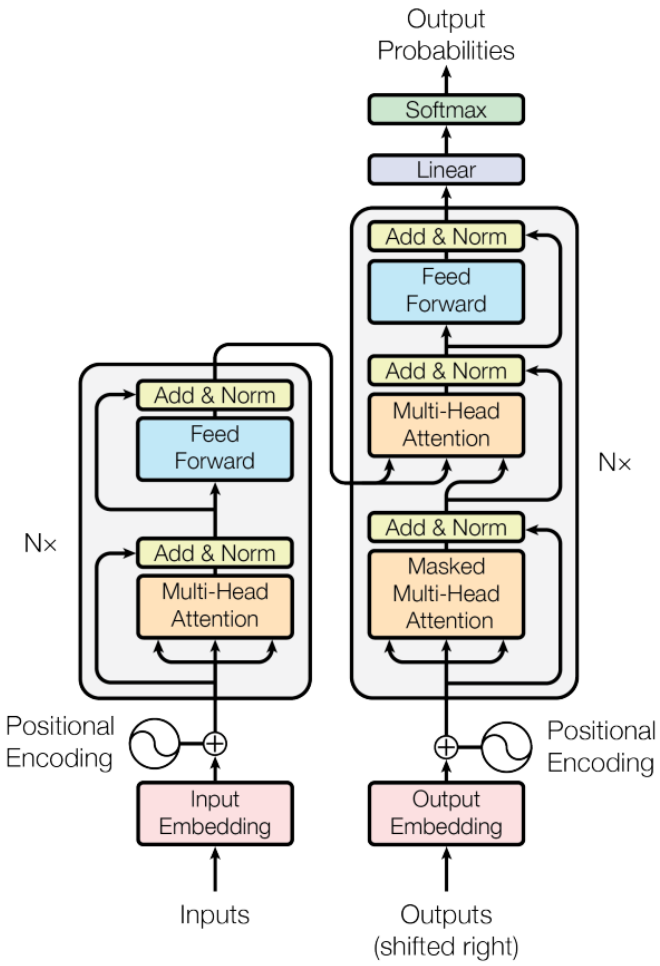


Figure 2.1: Transformer architecture, with both encoder and decoder structures [151]

sequence must be artificially stored in the values of each element in the input sequence. Vaswani et al. use *positional encoding* consisting of sinusoids additively to the input sequence.

$$\mathrm{PE}(\mathrm{pos}, 2i) = \sin\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d}}}\right) \tag{2.6}$$

$$\mathrm{PE}(\mathrm{pos}, 2i+1) = \cos\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d}}}\right) \tag{2.7}$$

$$\tag{2.8}$$

where pos is the position in the sequence and i is the dimension of the vector in the sequence.

The intuition behind positional encoding is to help the model attend to relative positions because of the sinusoidal properties. With a properly trained model, the model should be able to distinguish the positional encoding from the original input sequence. Positional encoding is only necessary when the ordering is crucial. If the input is a set rather than a sequence, positional encodings can be omitted. For example, set transformers [90] utilize a transformer because of its natural permutation invariance.

Attention consists mostly of linear operations. To improve complexity, a fully connected feed-forward neural network is added after every attention operation to learn non-linear relationships from mixtures of vectors. Deep networks suffer from instability within the hidden layers, so layer normalization [7] and residual connections [64] are applied after every attention and fully connected layer.

A transformer can be in one of two forms. A *transformer encoder* utilizes the entire sequence, allowing every position to attend to all positions in the sequence via self-attention. This produces a hidden state for each position of the sequence. A *transformer decoder*, by contrast, also leverages attention but attends only to positions that precede the current position. This is called *causal masking*, which forces the model to focus on only the past to predict the future part of the sequence.

Furthermore, the manner in which a transformer decoder works differs from its encoder counterpart. The encoder digests the input sequence and identifies the context. In neural machine translation, for example, the transformer encoder extracts useful information from a French sentence (source). The transformer decoder receives the English tokens generated thus far and feeds them back as inputs for the next step. While making a prediction for the next word, the transformer decoder can attend to the hidden states of the transformer encoder via cross-attention, grounding its prediction with the original French sentence.

2.2.4 Notable Transformers

Given the basics of a transformer, we discuss two notable variants that are relevant in this dissertation.

The success of the transformer has led to an influx of experimentation on different tasks and domains. In natural language understanding tasks, the transformer encoder manifested as BERT [37], pretrained on a large corpus of unstructured text with self-supervised objectives. By randomly masking certain tokens in the input and making BERT reconstruct the masked tokens using the surrounding context, BERT learns a good token-level representation. Furthermore, BERT has a special token `[CLS]` that is used to signify the aggregation of the sequence, trained by using the hidden states of the `[CLS]` token to predict whether two sentences are successive.

The transformer decoder architecture achieved notable success in natural language generation in the form of GPT [120]. In an unsupervised learning manner, a transformer decoder is pretrained to predict the next token over a large corpus. Due to the flexibility of its output, GPT can be used to solve many tasks including question answering, natural language inference, sentence similarity, and classification.

Chapter 3

Bandits in the Real World

Prior to the development of data-hungry machine learning algorithms, the idea of learning by trial and error fascinated Herbert Robbins [122], who formalized a sequential process of learning in 1952 in a landmark paper titled "Some aspects of the sequential design of experiments". He introduced a statistically grounded framework on how to adapt to future choices based on historical data, as opposed to designing an experiment based on a fixed sampling plan at the start.

Although bandits can theoretically be applied to most types of problems, they are classically used in conjunction with tabular data. Tabular data typically takes on a low-dimensional feature space and is mostly interpretable by both models and humans. In this chapter, we continue applying bandits to tabular domains but also introduce a novel approach to handling a series of significantly harder problems that classical bandit algorithms are unable to solve.

This chapter makes the following contributions:

1. We formulate a practical contextual bandit setting that supports high-dimensional representations, evolving arms, non-stationary rewards, covariate shift, and efficient online updates.

2. We introduce *coupled arms* to characterize how context-dependent arms share similar reward distributions over time.

3. We propose *importance-weighted kernel regression* (IWKR), a nonparametric estimator that mitigates sampling bias and supports linear-time importance-weight updates.

4. We develop *conditionally coupled contextual Thompson sampling* (C_3), which combines learned embeddings, IWKR, and locally constructed Beta posteriors without requiring online neural-network retraining.

5. We prove that C_3 achieves sublinear expected regret of $\mathcal{O}\Big(L^{\frac{d}{d+2}} T^{\frac{d+1}{d+2}} (\log T)^{\frac{1}{d+2}}\Big)$ in the stationary Lipschitz setting.

6. We empirically demonstrate that C_3 learns arm coupling and achieves competitive cumulative regret in simulations, benchmark datasets, and a real-world recommendation setting.

3.1 Motivation

The research area dedicated to studying the bandit algorithm is a highly theoretical one. While the mathematical analyses are elegant, we look towards applying our knowledge of bandits in the real world. To do so, we begin by acknowledging certain gaps that classical bandit algorithms do not adequately address, and we demonstrate them with an e-commerce product recommendation example.

High-dimensional, complex spaces As alluded to in Section 2.1, bandit algorithms are not optimized for both high-dimensional inputs and complex relationships between input features and their corresponding reward distribution. With advancements in natural language processing and computer vision in the early 2020's [143], more information can be encoded as tensors, providing a richer representation of a query, context or environment. For example, estimating whether a customer buys a product would be made easier if a model could capture the semantic meaning of the product category, which is a high-dimensional representation.

Evolving arms Many classical bandit algorithms such as UCB [6] use counting-based methods to optimize for the exploration-exploitation trade-off. A major caveat of counting-based methods and any method that specifically stores the state of unique arms is completely unable to work in problems where the set of arms evolves over time. For example, not all items in the catalogue may be available at every point in time – some may be out of stock, or simply discontinued.

Retraining Classical bandit algorithms are typically lightweight enough that adjusting the parameters of their models may be possible at the arrival of each new data point. This is made possible by the use of a linear model, enabling closed-form optimization solutions or simple numeric approximations. If a bandit algorithm adopts a neural network architecture in its model, it would be impractical to perform gradient-based optimization updates at the arrival of every new data point. For example, the arrival of new data points in an e-commerce platform is incredibly frequent [44] and with the offline batch-wise learning of neural networks, it would be costly and inefficient.

Concept drift Concept drift refers to the change of the conditional distribution of a target variable (e.g. the volume of sales of a product) over time. It can be formally defined as follows: for input variables $X_1, ..., X_T$ and target variables $Y_1, ..., Y_T$ across different times, and true conditional distribution $F(X, Y) = \Pr(Y|X)$, it is said that concept drift occurs if there exists some time t such that $F(X_t, Y_t) \neq F(X_{t+1}, Y_{t+1})$ [100]. Customers in an e-commerce platform would largely undergo some degree of concept drift over a sufficiently long span of time, as a result of trends, emerging technologies, or seasonal effects.

Covariate shift Covariate shift (or data drift) refers to the change in the sampling distribution of data. Formally, for input variables $X_1, ..., X_T$, covariate shift occurs when the distribution generating X_t differs from the distribution generating X_{t+1} for some time t [14]. In the context of e-commerce, this could occur as customer preference changes. Such changes are non-trivial, especially when real-world environments are often part of a complex dynamic system. More significantly, as the policy of the recommendation engine converges to an optimal state, the entropy of the arms sampled decreases and will inevitably result in covariate shift.

The aforementioned five motivating elements are core to the innovation of a system based on bandit theory to solve practical problems.

3.2 Related Work

To better understand what the current literature of bandits offers, we review the related literature.

3.2.1 Contextual Bandits

On the contextual bandit front, LinUCB by Li et al. [92] can be considered one of the pioneering contextual bandit algorithms that demonstrated success through the use of simulated evaluation based on the Yahoo! news dataset. Chu et al. [33] followed up with a rigorous theoretical analysis of a variant of LinUCB. Another popular paradigm is the Bayesian approach, in which Agrawal and Goyal [3] developed a Thompson sampling contextual bandit algorithm with a Gaussian likelihood and prior, assuming a linear payoff function. Unfortunately, there was no empirical evaluation of a practical problem.

With deep neural approaches on the rise, the contextual bandit community has been focusing on algorithms that can learn non-linear reward functions. One of the earlier algorithms was the KernelUCB by Valko et al. [145] which extends LinUCB by exploiting reproducing kernel Hilbert space (RKHS). Similarly, Srinivas et al. [136] generalized Gaussian processes for a contextual bandit setting by introducing the GP-UCB algorithm. While both algorithms attain a sublinear regret, practicality is limited since both have cubic time complexity in terms of the number of samples. After the breakthrough in the theoretical understanding of neural networks, particularly the neural tangent kernel (NTK) by Jacot et al. [75], Zhou et al. [174] developed NeuralUCB which is a neural network-based contextual bandit algorithm with a complete theoretical analysis and a suite of empirical analyses which outperforms many algorithms in tabular dataset benchmarks from OpenML.

More recent advancements include SquareCB [46] which reduces the problem of contextual bandits to an online regression problem. Under mild conditions, SquareCB along with a black-box online regression oracle has optimal bounded regret with no overhead in runtime or memory requirements. While they work on most regression models, the performance is highly dependent on the quality of the selected oracle. Kveton et al. [85] introduced their take on randomized algorithms for contextual bandits. Their novel contributions include additive Gaussian noise for a bandit setting that can be introduced to complex models such as neural networks.

3.2.2 Other Relevant Bandits

Several specialized bandit algorithms may be relevant to our problem. Basu et al. [11] introduced a variant called contextual blocking bandit that handles a variable set of arms but assumes the selected arms of an agent influence the future set of valid arms. Their work revolves around this idea but ultimately differs in having a fixed, finite set of overall arms.

The problem of non-stationary reward distributions in bandits is usually referred to as a restless bandit, which was proposed by Whittle [160]. Wang et al. [155] introduced the Restless-UCB algorithm which provably solves restless bandits, but does not account for context in the environment. Chen and Hou [28] improves upon this by leveraging context and budget constraints. A specialized solution by Slivkins and Upfal [134] assumes that the reward distribution changes gradually and works by continuously exploring while sometimes following the best arm based on the last two observations. However, they assume that all arms are independent and can be modelled as a Brownian motion which is uncommon in practice.

In the realm of sleeping bandits where some arms are occasionally not valid, Slivkins [133] introduced the contextual zooming algorithm that adaptively forms partitions in a similarity space. Operating in a space, as opposed to having fixed arms, allows them to effectively tackle the sleeping bandit problem. While they offer an innovative framework, it is likely that partitions in high-dimensional spaces (e.g. from large language models) would not be tractable.

Kleine Buening et al. [83] proposed a situation that is potentially relevant to many applications from a game-theoretic perspective. Goals of agents could conflict – for example, an arm could be a video from a content creator on a public platform. In such a setting, "clickbait" videos would maximize the reward for the content creator but are seen as a negative phenomenon overall on the platform. Kleine Buening et al. [83] devised an incentive-aware learning algorithm to ensure that the learner obtains the right signal.

3.2.3 Recommender Systems

There is growing interest in applying contextual bandits to recommender systems [10]. On the topic of recommender systems, the more prevalent form of recommendation models in recent years is typically based on neural networks [39]. A popular approach is the two-tower neural network employed by Huang et al. [72] and Yi et al. [169]. Major benefits of such designs include efficient retrieval within the learned embedding space as well as the ability to learn complex relationships between queries and items. Another design called BERT4Rec by Sun et al. [138] leverages the power of transformers in sequential problems to provide recommendations. However, in the mentioned recommender designs, there is no element of exploration, unlike contextual bandits, resulting in a possibly greedy approach that may get stuck in a suboptimal policy. This can also be a major problem when user behaviour changes since all gradient updates to the models are based on historical data alone, and frequent retraining of the models can be expensive.

3.3 Algorithm

We outline the key components, then piece them together towards the end of this section.

3.3.1 Problem Formulation

We focus on contextual Bernoulli bandits where rewards can only take on binary values. Let $c \in \mathcal{C}$ be a context, and $a \in \mathcal{A}$ be an arm. The context space $\mathcal{C} \subseteq \mathbb{R}^{d_c}$ and $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ are assumed to be continuous. However, the number of arms presented at each timestep is assumed to be finite. Arms can be represented in a continuous feature space when they are described by features. If the arms do not contain any informative features, they should be treated as indices and hence represented as one-hot vectors.

The Bernoulli mean parameter $\mu : \mathcal{C} \times \mathcal{A} \to [0, 1]$ is a continuous function representing the probability of the reward being one. For theoretical analyses, μ is assumed to be Lipschitz continuous in $\mathcal{C} \times \mathcal{A}$.

For a given context c, the conditional reward distribution of arm a is

$$R_{c,a} \sim \text{Bernoulli}(\mu(c, a)) \tag{3.1}$$

A more general case that is relevant to the scope of our discussion is to introduce time into the expressions. Time $t \in [0, \infty)$ is assumed to be an independent, monotonically increasing quantity. We assume that μ is Lipschitz continuous with respect to time. While time could be seen as a special instance of context, we view it as a separate entity as previous samples always come from a time period before the current time. Incorporating time as part of the context makes this into a forecasting task, and forecasting is not within the scope of this chapter.

A non-stationary case of the contextual Bernoulli bandit we are investigating has a conditional reward distribution of

$$R_{c,a}^{(t)} \sim \text{Bernoulli}(\mu(c, a, t)) \tag{3.2}$$

The goal is to minimize the cumulative regret for given contexts c_t over all time steps up to horizon T by selecting the arms to pull a_t

$$\text{CumulativeRegret}(T) = \sum_{t=1}^{T} [\mu(c_t, a_t^*, t) - \mu(c_t, a_t, t)] \tag{3.3}$$

where a_t^* is the optimal arm at time t.

3.3.2 Concept Drifts and Coupled Arms

Conditional reward distributions tend to be non-stationary in practice but we can still exploit information about the arms. While μ could change with time, there is inherent structure in *how* μ changes for certain arms. For example, the marginal probabilities of shoppers buying snowboards and skis are not constant across different months, but it is likely that they change together.

It might be tempting to call this idea "correlated" but it refers to a slightly different concept in the context of bandits. Gupta et al. [62] formulated the correlated multi-armed bandit problem where pulling an arm provides some information about another arm that is correlated. This idea is valid in our problem but it does not account for time and how the degree of correlation changes over time.

We introduce the idea of *coupled arms*, where two strongly coupled arms have similar reward distributions over most time periods. On the other hand, two weakly coupled arms only have similar reward distributions at certain time periods. The formulation of correlated bandit [62] introduces a special case of coupled arms.

For two arms a_i, a_j in a Bernoulli bandit with mean rewards $\mu_{i,t}, \mu_{j,t}$ at time t respectively, define their distributional discrepancy by

$$d_{ij} = |\mu_{i,t} - \mu_{j,t}| \tag{3.4}$$

For some tolerance $\varepsilon \geq 0$, arms a_i and a_j are ε-coupled at time t when

$$|\mu_{i,t} - \mu_{j,t}| \leq \varepsilon \tag{3.5}$$

Let $I \subseteq \{1, \ldots, T\}$ be a time interval. Define the empirical temporal coupling strength

$$\Gamma_{ij}(I; \varepsilon) = \frac{1}{|I|} \sum_{t \in I} \mathbb{I}\{|\mu_{i,t} - \mu_{j,t}| \leq \varepsilon\} \tag{3.6}$$

This is the proportion of time points in I during which the two arms are approximately indistinguishable. For a threshold $\rho \in (0, 1]$, arms a_i and a_j are said to be (ε, ρ)-coupled on I if

$$\Gamma_{ij}(I; \varepsilon) \geq \rho \tag{3.7}$$

For a small, fixed ε, if ρ is set close to one, then it can be regarded as *strongly coupled*. For lower but positive values of ρ, it can be considered *weakly coupled*.

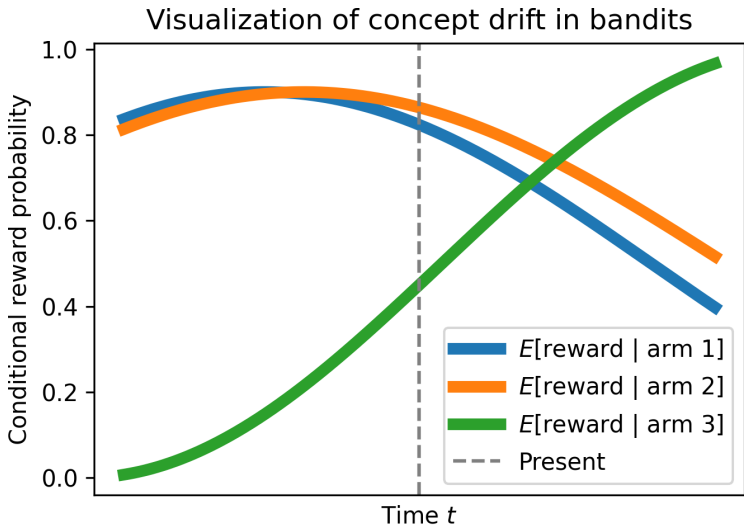


Figure 3.1: A three-arm example of strong and weak coupling during concept drift of expected reward distributions.

Figure 3.1 offers a visualization of how coupled arms are identified. Given the true underlying μ's and assuming that the context is fixed, arms 1 and 2 are said to be strongly coupled, while arms 1 and 3 are said to be weakly coupled. The implication is that pulling arm 1 gives reward information about arms 1 and 2. Recall that the exploration-exploitation trade-off involves a dilemma where exploration gains information about the arms but incurs regret. Leveraging coupled arms effectively simplifies the bandit by "combining" arms 1 and 2 into a single arm, and reduces the exploration requirement, allowing the player to exploit more quickly.

Remark. One of the most common ways to deal with non-stationarity is to forecast the drifts [96]. However, that approach is highly data-intensive and complex, limiting its adoption in bandit algorithms. Our design grounds its beliefs based on the most recent samples and is still exploring arms without strong assumptions on changes to an unknown μ.

3.3.3 Nonparametric Estimation

We will now discuss the model design.

We start with an idea from 1964, called Nadaraya-Watson kernel regression [110] (or NWKR for short). For some query $x \in \mathbb{R}^d$, the assumption is that the labels of training data

points that are close to x are more relevant than distant points. NWKR is a nonparametric regression method that uses a weighted average of targets in a reference dataset $\mathcal{D}_{\text{ref}} = \{(x_i, y_i) : i \in [n]\}$, weighted by a kernel κ.

$$\text{NWKR}(x \mid \mathcal{D}_{\text{ref}}) := \hat{y} = \frac{\sum_{(x_i,y_i)\in\mathcal{D}_{\text{ref}}} \kappa(x, x_i) y_i}{\sum_{(x_i,y_i)\in\mathcal{D}_{\text{ref}}} \kappa(x, x_i)} \tag{3.8}$$

NWKR immediately solves our retraining problem, mentioned in Section 3.1. As it is nonparametric, there is no refitting or finetuning with the arrival of a new data point. Yet, NWKR is an adaptive learning algorithm once $\mathcal{D}_{\text{ref}}$ is augmented with the new data point.

Recall that the issue of covariate shifts in Section 3.1 where solving a bandit problem necessitates a shift in sampling distribution as the policy converges to an optimum. NWKR can be susceptible to such shifts as each point contributes additively to the final regression value.

3.3.4 Importance Weighted Kernel Regression

The goal of kernel regression approaches here is to estimate an unknown function in a subspace in the vicinity of reference samples. If the samples are uniformly sampled across the input space, kernel regression will perform as expected. However, greater non-uniformity can make the regression biased towards the label of points in a neighbouring frequently sampled subspace.

To mitigate such bias, we introduce *importance weights*. A sample is assigned a lower importance weight if it is located in the vicinity of many samples, and a higher importance weight otherwise. The importance weight of a sample in the reference dataset $\mathcal{D}_{\text{ref}}$ is

$$w(x_j) = \frac{1}{\sum_{(x_i,y_i)\in\mathcal{D}_{\text{ref}}} \kappa(x_j, x_i)} \in (0, 1] \tag{3.9}$$

This quantity can be seen as the reciprocal of unnormalized kernel density estimates [30] of each reference sample.

Remark. Note that importance weights are only computed for the reference dataset, not for a query. The calculation involves computing a kernel matrix in which the denominator is never zero since every x_i is in $\mathcal{D}_{\text{ref}}$ and $\kappa(x_j, x_j) = 1$.

We introduce the *importance weight kernel regression* (or IWKR for short)

$$\text{IWKR}(x \mid \mathcal{D}_{\text{ref}}) := \hat{y} = \frac{\sum_{(x_i,y_i)\in\mathcal{D}_{\text{ref}}} \kappa(x, x_i) w(x_i) y_i}{\sum_{(x_i,y_i)\in\mathcal{D}_{\text{ref}}} \kappa(x, x_i) w(x_i)} \tag{3.10}$$

Theorem 1. *Suppose a vector of importance weights $\boldsymbol{w}$ of n samples has been computed. The time complexity of updating the importance weights, given a new sample, is $\mathcal{O}(n)$.*

Proof of Theorem 1. Suppose there are n samples in the reference dataset $\{x_i \in \mathbb{R}^d : i \in [n]\}$. Consider the kernel matrix $K \in \mathbb{R}^{n \times n}$ which holds all pairwise kernel values between every sample. From Equation 3.9, we can deduce that the sum of the i^{th} row of matrix K will be $w(x_i)$, and similarly for the sum of the i^{th} column since K is symmetric. The importance weight of the initial reference dataset $\mathcal{D}_{\text{ref}}$ can be calculated this way which takes on average $\mathcal{O}(n)$ per sample.

Suppose there is already the vector of importance weights for all samples in $\mathcal{D}_{\text{ref}}$ denoted as $\boldsymbol{w}^{(n)} = [w_1^{(n)} \cdots w_n^{(n)}] \in [0,1]^n$. We want to obtain an efficient update equation for $\boldsymbol{w}^{(n+1)}$. To update efficiently, memoization would be useful since $w_i^{(n)}$ itself stores the reciprocal of a sum. During inference, the unweighted kernel value will need to be computed. This result can be stored, and is denoted as $\boldsymbol{z}^{(n+1)} = [\kappa(x_{n+1}, x_1) \cdots \kappa(x_{n+1}, x_n)]$.

There are two steps here: update the existing $w_i^{(n)}$ for $i \in [n]$, and append $w_{n+1}^{(n+1)}$ to the new vector. To obtain $w_i^{(n+1)}$ for $i \in [n]$, the update equation can be expressed as a function of its previous value

$$\begin{aligned} w_i^{(n+1)} &= \frac{1}{\sum_{j=1}^{n+1} \kappa(x_i, x_j)} \\ &= \frac{1}{\kappa(x_i, x_{n+1}) + \sum_{j=1}^{n} \kappa(x_i, x_j)} \\ &= \frac{1}{z_i^{(n+1)} + \frac{1}{w_i^{(n)}}} \end{aligned}$$

which is a constant time operation for each $i \in [n]$ since all of the required values have already been computed. To compute the new importance weight,

$$\begin{aligned} w_{n+1}^{(n+1)} &= \frac{1}{\sum_{j=1}^{n+1} \kappa(x_{n+1}, x_j)} \\ &= \frac{1}{\kappa(x_{n+1}, x_{n+1}) + \sum_{j=1}^{n} \kappa(x_{n+1}, x_j)} \\ &= \frac{1}{1 + \sum_{j=1}^{n} z_j^{(n+1)}} \end{aligned}$$

which is an $\mathcal{O}(n)$ time operation for the new sample. Therefore, the entire update equation for the importance weight given a new sample is a linear time operation. □

Theorem 1 is important in demonstrating that IWKR shares the same time complexity as NWKR. Naively computing $\boldsymbol{w}$ with a new sample without memoization will result in a quadratic time complexity. IWKR mitigates the covariate shift issue with minimal computational overhead but it still leaves many areas of improvement. The input space to IWKR may not be calibrated for κ and does not offer a way to leverage coupled arms.

3.3.5 Embedding Model

The input space of IWKR needs to be calibrated. We opt for a neural network-based model to learn an embedding space $\mathcal{S} \subseteq \mathbb{R}^d$ because of its flexibility to handle any differentiable loss function. The neural network embedding model $\phi : \mathcal{C} \times \mathcal{A} \to \mathcal{S}$ projects every context-arm pair into $\mathcal{S}$. This calls for an update to our notation.

Each context-arm pair is embedded as $s = \phi(c, a)$. The reference dataset is now defined to be a set of embedded context-arm pairs and its corresponding binary reward $\mathcal{D}_{\text{ref}} = \{(s_i, r_i) : i \in [n]\}$. For the rest of this chapter, κ is chosen to be the radial basis function (RBF) kernel, and it operates on the learned embedding space $\mathcal{S}$.

This improves the ability of IWKR in handling high-dimensional, complex spaces. Furthermore, this allows us to handle evolving arms. Suppose there are K arms at time t. Estimating the mean rewards of arms involves a single K-sized batched call.

3.3.6 Computability of Coupling

We contextualize our understanding of NWKR and IWKR in terms of coupling. Recall that if two arms a_1, a_2 are strongly coupled conditioned on context c, they share similar reward distributions across time. From the perspective of discriminative models, we can use the reward of a_1 to predict the reward of a_2 if the context is c. Since the reward prediction is based on IWKR, which is nonparametric, and its weights are influenced by the embedding model ϕ, the parameters of ϕ control the overall reward prediction.

Remark. We emphasize that the degree of coupling is conditional on the context. Consider the following example where the arms are "sell an umbrella" and "sell a raincoat" . Suppose we measure the reward by whether customers buy one of the two products. If the context suggests that it is raining, the reward distribution of both arms is coupled since both products are useful in dealing with rain. However, if the context suggests that it is hot and dry, the reward distribution of both arms is not coupled as umbrellas can be used to provide shade from the sun [129] while a raincoat would make the user warmer and uncomfortable.

To materialize an optimization routine for ϕ to encode coupling information, we consider the following observation. For any two arms a, a' given c and time t, with the IWKR algorithm,

- If $\mu(c, a, t)$ is close to $\mu(c, a', t)$, then moving $\phi(c, a)$ *close to* $\phi(c, a')$ reduces error
- If $\mu(c, a, t)$ is far from $\mu(c, a', t)$, then moving $\phi(c, a)$ *away from* $\phi(c, a')$ reduces error

This observation manifests as an optimization objective

$$\min_{\phi} \mathbb{E}_{(c,a,r)}[\mathcal{L}_{\text{BCE}}(\text{IWKR}(\phi(c, a)|\mathcal{D}_{\text{ref}}), r) + \lambda \mathcal{L}_{\text{ECE}}(\text{IWKR}(\phi(c, a)|\mathcal{D}_{\text{ref}}), r)] \tag{3.11}$$

where $\mathcal{L}_{\text{BCE}}$ is the binary cross entropy loss function and $\mathcal{L}_{\text{ECE}}$ is the expected calibration error [111]. We incorporate calibration as an auxiliary objective to reduce overconfidence which is common in deep neural networks [60]. In a bandit algorithm involving neural networks, calibration is important to avoid biases when facing a lack of data.

Objective 3.11 works for a stationary μ. For the non-stationary case, we want to ensure that the attractive forces are constrained to samples from the same time interval. Otherwise, this would lead to averaging reward information over all time. This can be done by partitioning $\mathcal{D}_{\text{ref}}$ into disjoint sets and IWKR only uses samples from the relevant time interval for a given query. This resolves the fifth and final motivating problem – concept drift.

Algorithm 1 ϕ training process

1: **Inputs**: Training dataset $\mathcal{D} = \{(c_i, a_i, r_i)\}_{i=1}^{n}$, neural network ϕ
2: **for** epoch e **do**
3: Randomly split $\mathcal{D}$ into $\mathcal{D}_{\text{ref}} = (\boldsymbol{c}_{\text{ref}}, \boldsymbol{a}_{\text{ref}}, \boldsymbol{r}_{\text{ref}})$, and $\mathcal{D}_q$
4: **for** $(c, a, r) \in \mathcal{D}_q$ **do**
5: Embed reference $\boldsymbol{s} \leftarrow \phi(\boldsymbol{c}_{\text{ref}}, \boldsymbol{a}_{\text{ref}})$
6: Compute importance weights $\boldsymbol{w}$ for $\boldsymbol{s}$
7: Embed query $s_q \leftarrow \phi(c, a)$
8: $\boldsymbol{s}', \boldsymbol{r}'_{\text{ref}} \leftarrow$ filter $\boldsymbol{s}, \boldsymbol{r}_{\text{ref}}$ so that they are in the same time interval as s_q
9: Estimate mean reward $\hat{\mu}(q)$ using IWKR
10: Compute the sum of losses with $\hat{\mu}$ and r
11: Perform gradient descent on ϕ

The training process is described in Algorithm 1. The randomization in Step 3 forms a self-supervised learning objective by masking certain samples and creating a predictive subtask. $\mathcal{D}_{\text{ref}}$ can be seen as a set of in-context examples and $\mathcal{D}_q$ contains the training samples.

3.3.7 Conditionally Coupled Contextual Thompson Sampling

So far, the focus has been on estimating μ. Here, we will combine the components above into a cohesive bandit algorithm.

Thompson Sampling To incorporate a principled manner of exploration, we adopt Thompson sampling. The conjugate prior for a Bernoulli bandit is a Beta distribution with parameters α and β, where α refers to the count of $r = 1$. Define the following quantities:

$$\eta(s) := \sum_{i=1}^{n} \kappa(s, s_i) \tag{3.12}$$

$$\alpha(s) := \eta(s) \frac{\sum_{i=1}^{n} \kappa(s, s_i) w(s_i) r_i}{\sum_{i=1}^{n} \kappa(s, s_i) w(s_i)} = \eta(s)\hat{\mu}(s) \tag{3.13}$$

$$\beta(s) := \eta(s) \frac{\sum_{i=1}^{n} \kappa(s, s_i) w(s_i)(1 - r_i)}{\sum_{i=1}^{n} \kappa(s, s_i) w(s_i)} = \eta(s)(1 - \hat{\mu}(s)) \tag{3.14}$$

With this, we can sample the posterior $\tilde{\mu}(s) \sim \text{Beta}(\alpha(s), \beta(s))$ from the constructed Beta distribution. The mean simplifies to

$$\mathbb{E}[\tilde{\mu}(s)] = \frac{\alpha(s)}{\alpha(s) + \beta(s)} = \hat{\mu}(s) \tag{3.15}$$

which is exactly IWKR. On the other hand, the information of the frequency the neighbourhood was sampled is still preserved because it can be shown that the total count is still a monotonically increasing function of n,

$$\alpha(s) + \beta(s) = \sum_{i=1}^{n} \kappa(s, s_i) \tag{3.16}$$

The *conditionally coupled contextual Thompson sampling* (C_3) algorithm consists of the embedding model, IWKR and Beta Thompson sampling. The embedding model is trained once with IWKR on historical data. Thompson sampling is used online at every bandit step.

Figure 3.2 shows an illustration of the posterior distribution at two different arms. On the left, we see a scatterplot on a learned embedding space $\mathcal{S}$ and among the historical

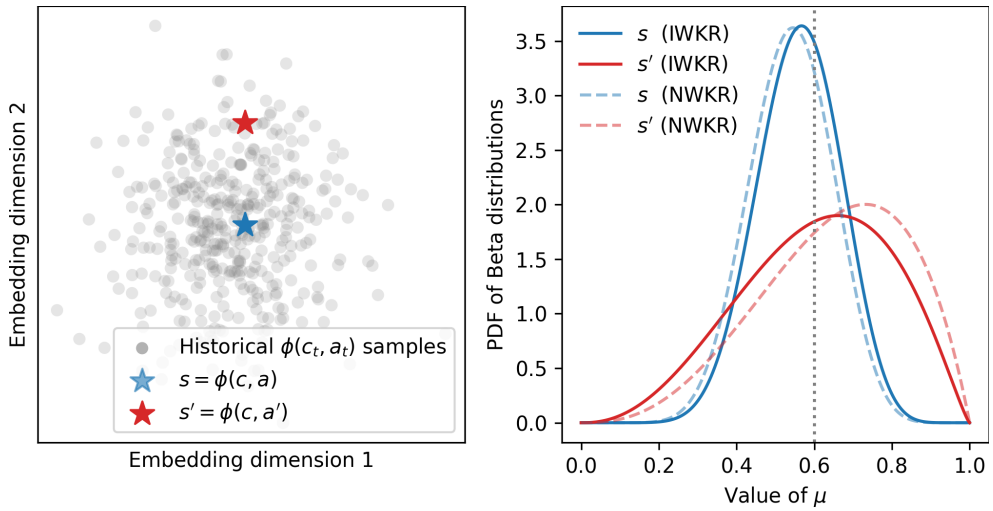


Figure 3.2: An example of C_3 Beta posterior distribution in continuous spaces. The true μ of both arms for that context is 0.6.

samples in $\mathcal{D}_{\text{ref}}$, there are two points corresponding to two distinct arms, given the same context and time. On the right, we see the constructed Beta posterior using both NWKR (dashed lines) and IWKR (solid lines). The blue arm is surrounded by more historical samples so its distribution is more peaked compared to the red arm. For both arms, the IWKR variant has less bias due to the importance weights.

3.3.8 Approximate Bayesian Update

Recall that IWKR is dependent on a reference dataset $\mathcal{D}_{\text{ref}}$. The sampling of the posterior weighs every sample with the RBF kernel relative to some query point. This can be seen as a type of conditioning on a local subspace. After an arm is pulled and the reward is observed, we obtain the triplet $(c_{n+1}, a_{n+1}, r_{n+1})$. Since the algorithm operates on $\mathcal{S}$ and requires the updated importance weights, to conserve time and memory, we can directly store the triplet $(\phi(c_{n+1}, a_{n+1}), r_{n+1}, w_{n+1}^{(n+1)})$ into $\mathcal{D}_{\text{ref}}$. This is an approximate Bayesian update and is important for the online learning element.

Bayesian update typically assumes that every random variable in a sequence is identically distributed. The information gathered is directly stored in the parameter space of some statistical distribution, which will be updated using some closed-form algebraic expression. For C_3, the information is stored in the reference dataset $\mathcal{D}_{\text{ref}}$ embedded on

Table 3.1: Comparison between algorithms where linear refers to both LinUCB [92], LinTS [3], and SquareCB [46] while neural refers to both NeuralUCB [174] and NeuralTS [171]. n refers to the number of samples seen.

Algorithm	Inference time	Update time	Non-linear rewards	Non-stationary tasks	Arm features
C_3	$\mathcal{O}(n)$	$\mathcal{O}(1)$	✓	✓	✓
Linear	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✗	✗	partially
Neural	$\mathcal{O}(1)$	$\mathcal{O}(n)$	✓	✗	✗

$\mathcal{S}$. This allows flexibility to both append and remove samples from $\mathcal{D}_{\text{ref}}$. In problems with concept drift, the conditional reward distribution shifts as a function of time but a typical Bayesian update does not effectively handle this since it might simply average the distributions across time.

To mitigate the issues presented by non-stationarity without frequent retraining, C_3 allows the removal of older samples while appending the latest samples. Time can be seen as a special case of context and since μ is assumed to be Lipschitz continuous, samples nearer in the time dimension would be more relevant.

Algorithm 2 C_3 inference process

Inputs: Reference dataset $\mathcal{D}_{\text{ref}} = (\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{w})$, context c, set of valid arms $\{a_1, ..., a_k\}$, trained embedding model ϕ
for each valid arm index $i \in [k]$ **do**
    Embed queries $s_{q_i} \leftarrow \phi(c, a_i)$
    Compute $\alpha(s_{q_i})$, $\beta(s_{q_i})$ with respect to $\mathcal{D}_{\text{ref}}$
    Sample $\hat{r}_i \sim \text{Beta}(\alpha(s_{q_i}), \beta(s_{q_i}))$
Play best arm $j \leftarrow \text{argmax}_{i \in [k]} \hat{r}_i$
Observe reward r
Append (s_{q_j}, r) to $\mathcal{D}_{\text{ref}}$
Update $\boldsymbol{w}$ in $\mathcal{D}_{\text{ref}}$

The entire inference pipeline can be summarized in Algorithm 2. The update process does not include any gradient updates, unlike many neural contextual bandit algorithms. The properties of C_3 and other contextual bandit algorithms are summarized in Table 3.1.

3.4 Theoretical Results

We validate our design of C_3 with regret analysis.

Theorem 2. *Let the embedding space of ϕ be $S \subset [0,1]^d$. Assume μ is L-Lipschitz. In a stationary bandit scenario, C_3 incurs an expected regret of*

$$\mathbb{E}[R_T] \in \mathcal{O}\left(L^{\frac{d}{d+2}} T^{\frac{d+1}{d+2}} (\log T)^{\frac{1}{d+2}}\right)$$

Lemma 1. *Let $X \sim Beta(\alpha, \beta)$ with $\alpha, \beta \geq 1$, and let $p = \alpha/(\alpha+\beta)$. Then for any $\varepsilon \in (0, 1-p)$,*

$$\Pr(X \geq p + \varepsilon) \leq \exp(-2(\alpha + \beta - 1)\varepsilon^2)$$

Similarly, for any $\varepsilon \in (0, p)$,

$$\Pr(X \leq p - \varepsilon) \leq \exp(-2(\alpha + \beta - 1)\varepsilon^2)$$

Proof of Lemma 1. Let $X \sim \text{Beta}(\alpha, \beta)$ and $Y \sim \text{Binomial}(n, x)$ where $n = \alpha + \beta - 1$. By writing the regularized incomplete beta function representation of the Beta CDF and comparing it to the Binomial tail expression, one can obtain

$$\Pr(X \geq x) = \Pr(Y \leq \alpha - 1)$$

Take $x = p + \varepsilon$. Then $\mathbb{E}[Y] = n(p + \varepsilon)$. Note that $p \leq 1$, thus $\alpha - 1 \leq np$ because

$$\begin{aligned} np &= (\alpha + \beta - 1)\frac{\alpha}{\alpha + \beta} \\ &= \alpha - \frac{\alpha}{\alpha + \beta} \\ &= \alpha - p \\ &\geq \alpha - 1 \end{aligned}$$

Therefore, the event $Y \leq \alpha - 1$ implies $Y \leq np$

$$\Pr(X \geq p + \varepsilon) = \Pr(Y \leq \alpha - 1) \leq \Pr(Y \leq np)$$

Note that $np = n(p+\varepsilon) - n\varepsilon = \mathbb{E}[Y] - n\varepsilon$. Applying the Hoeffding/Chernoff lower-tail bound for Binomial random variables gives

$$\Pr(Y \leq \mathbb{E}[Y] - n\varepsilon) \leq \exp(-2n\varepsilon^2) = \exp(-2(\alpha + \beta - 1)\varepsilon^2)$$

Connecting both gives

$$\Pr(X \geq p + \varepsilon) \leq \Pr(Y \leq np) = \Pr(Y \leq \mathbb{E}[Y] - n\varepsilon) \leq \exp(-2(\alpha + \beta - 1)\varepsilon^2)$$

□

Proof of Theorem 2. Assume that the true mean reward function $\mu : S \to [0, 1]$ is Lipschitz. There exists $L > 0$ such that for all $s, s' \in S$,

$$|\mu(s) - \mu(s')| \leq L\|s - s'\|$$

Cumulative regret is formalized as follows. Let $s_t^* = \text{argmax}_{s \in S_{A_t}} \mu(s)$ be an optimal action, where S_{A_t} is the set of embeddings of all valid arms at time t. The expected cumulative regret is

$$R_T = \sum_{t=1}^{T} \mu(s_t^*) - \mu(s_t)$$

We standardize the notation. The RBF kernel $\kappa_h(\cdot, \cdot)$ has bandwidth $h > 0$. The reference dataset $D_{t-1} = \{(s_i, r_i)\}_{i=1}^{t-1}$ stores all embedding-reward tuples up to time $t - 1$. The values of the Beta parameters $\alpha_{t-1}(s)$ and $\beta_{t-1}(s)$ are obtained by comparing a query embedding $s \in S$ to every embedding in the reference dataset, and their sum is the kernel mass

$$\eta_{t-1}(s) = \sum_{i=1}^{t-1} \kappa_h(s, s_i)$$

The IWKR procedure samples from the constructed Beta distribution

$$\Theta_t(s) \sim \text{Beta}(\alpha_{t-1}(s), \beta_{t-1}(s))$$

and plays $s_t = \text{argmax}_{s \in S_{A_t}} \Theta_t(s)$.

For the analysis of C_3, we use a truncated RBF kernel (compact support) for exact locality.

$$\kappa_h(s, s') := \exp\left(-\frac{\|s - s'\|^2}{2h^2}\right) \mathbb{1}\{\|s - s'\| \leq h\}$$

Let $S \subseteq [0,1]^d$ without loss of generality by rescaling if necessary. Partition $[0,1]^d$ into axis-aligned hypercubes (cells) of length h. Let $\mathcal{G}$ denote the set of all such cells intersecting S. The number of cells satisfies

$$|\mathcal{G}| \leq \left\lceil \frac{1}{h} \right\rceil^d \leq \left(\frac{2}{h} \right)^d \quad \text{for } h \leq 1 \tag{3.17}$$

For each cell $g \in \mathcal{G}$, fix a representative point $x_g \in g \cap S$, i.e. any point in the intersection. Define the cell index of any point $s \in S$ as $g(s) \in \mathcal{G}$, and the best cell representative g^* as

$$g^* = \operatorname*{argmax}_{g \in \mathcal{G}} \mu(x_g)$$

Because $s^* \in S$ lies in some cell $g(s^*)$, the cell representative $x_{g(s^*)}$ is at distance of at most the cell diameter

$$\|s^* - x_{g(s^*)}\| \leq \sqrt{d}h$$

By the Lipschitz property,

$$\mu(s^*) - \mu(x_{g(s^*)}) \leq L\sqrt{d}h$$

Since g^* maximizes $\mu(x_g)$ over g, we have $\mu(x_{g^*}) \geq \mu(x_{g(s^*)})$. Hence,

$$\mu(s^*) - \mu(x_{g^*}) \leq \mu(s^*) - \mu(x_{g(s^*)}) \leq L\sqrt{d}h$$

Decomposing regret gives

$$\begin{aligned} \mu(s^*) - \mu(s_t) &= (\mu(s^*) - \mu(x_{g^*})) + (\mu(x_{g^*}) - \mu(s_t)) \\ &\leq L\sqrt{d}h + (\mu(x_{g^*}) - \mu(s_t)) \end{aligned} \tag{3.18}$$

Up to this point, we have approximated the continuum optimum s^* to the best cell representation at a regret cost of $O(Lh)$. The next step is to bound the second term, which is the regret relative to the best cell representation. To do so, we exploit the locality of the truncated kernel.

For any query s, the kernel sums only involve past points s_i from the same cell $g(s)$. Let the number of times we have played in cell g up to time $t-1$ be

$$N_g(t-1) := |\{i \leq t-1 : g(s_i) = g\}|$$

The cell's respective cumulative binary reward counts are

$$S_g(t-1) := \sum_{i \le t-1 : g(s_i)=g} r_i \qquad\qquad F_g(t-1) := N_g(t-1) - S_g(t-1)$$

Now, fix a query $s \in g$. The kernel mass is

$$\eta_{t-1}(s) = \sum_{i=1}^{t-1} \kappa_h(s, s_i) = \sum_{i:g(s_i)=g} \kappa_h(s, s_i)$$

Moreover, within a cell g, any two points have distance at most $\sqrt{d}h$, hence for $s, s_i \in g$,

$$\exp\left(-\frac{\|s - s_i\|^2}{2h^2}\right) \ge \exp\left(-\frac{d}{2}\right)$$

We call the right hand side $c_d \in (0, 1)$. Therefore,

$$c_d N_g(t-1) \le \eta_{t-1}(s) \le N_g(t-1) \tag{3.19}$$

Define the gap of a cell representative

$$\Delta_g := \mu(x_{g^*}) - \mu(x_g) \ge 0$$

Cells with $\Delta_g = 0$ are optimal within the grid. We want to show that each suboptimal cell g is selected only $O(\log T/\Delta_g^2)$ times in expectation. Fix a suboptimal cell g with $\Delta_g > 0$. Consider the event at time t that IWKR selects a point in cell g. This can happen only if the Beta sample for cell g is unusually high or the Beta sample for the best cell g^* is unusually low.

Concretely, a good event refers to

$$E_g := \left\{ \left| \frac{S_g}{N_g} - \mu(x_g) \right| \le \frac{\Delta_g}{4} \right\}$$

and similarly

$$E_{g^*} := \left\{ \left| \frac{S_{g^*}}{N_{g^*}} - \mu(x_{g^*}) \right| \le \frac{\Delta_g}{4} \right\}$$

By Hoeffding's inequality for Bernoulli sums, conditional on $N_g = n$,

$$\Pr(E_g^c | N_g = n) \leq 2\exp\left(-2n\left(\frac{\Delta_g}{4}\right)^2\right) = 2\exp\left(-\frac{n\Delta_g^2}{8}\right)$$

Let $\Theta_{t,g}$ be the cell-level random variable for the Beta sample corresponding to any query point in cell g at time t. They are all functions of the same within-cell data, only differing in constants via kernel weights. Its parameters satisfy $\alpha+\beta = \eta$ and with Equation 3.19, we have $\eta \geq c_d N_g$.

On event E_g, the empirical mean is close to $\mu(x_g)$ and because the posterior mean is equal to the IWKR mean, which (within a cell) is a weighted average of the r_i's, we can bound the posterior mean deviation by the same scale (absorbing kernel-weight constants into a dimension constant). Denoting $\mathcal{F}_{t-1}$ as the history up to $t-1$,

$$|\mathbb{E}[\Theta_{t,g}|\mathcal{F}_{t-1}] - \mu(x_g)| \leq \frac{\Delta_g}{2}$$

implies

$$\begin{aligned}\Pr\left(\Theta_{t,g} \geq \mu(x_g) + \frac{\Delta_g}{2} \mid \mathcal{F}_{t-1}\right) &\leq \exp\left(-2(\eta-1)\left(\frac{\Delta_g}{2}\right)^2\right) \\ &\leq \exp\left(-\frac{c_d n \Delta_g^2}{2}\right)\end{aligned}$$

when using Lemma 1 with $\varepsilon = \Delta_g/2$ and $\eta \geq c_d n$. Similarly, for the optimal cell g^*, on event E_{g^*} with $N_{g^*} = m$,

$$\Pr\left(\Theta_{t,g^*} \leq \mu(x_{g^*}) - \frac{\Delta_g}{2} \mid \mathcal{F}_{t-1}\right) \leq \exp\left(-\frac{c_d m \Delta_g^2}{2}\right)$$

Now, define the "bad event" that could cause selecting g

$$B_t(g) := \{\Theta_{t,g} \geq \Theta_{t,g^*}\}$$

If g's sample is below $\mu(x_g) + \Delta_g/2$ and g^*'s sample is above $\mu(x_{g^*}) - \Delta_g/2$, event $B_t(g)$ cannot occur because

$$\begin{aligned}\Theta_{t,g} &< \mu(x_g) + \frac{\Delta_g}{2} \\ &= \mu(x_{g^*}) - \frac{\Delta_g}{2} && (\Delta_g = \mu(x_{g^*}) - \mu(x_g)) \\ &< \Theta_{t,g^*}\end{aligned}$$

Therefore,

$$B_t(g) \subseteq \left\{\Theta_{t,g} \geq \mu(x_g) + \frac{\Delta_g}{2}\right\} \cup \left\{\Theta_{t,g^*} \leq \mu(x_{g^*}) - \frac{\Delta_g}{2}\right\}$$

For it to be a useful bound, we want to set a rule where once n exceeds a logarithmic threshold, the probability of selecting g becomes at most $1/T^2$. Let

$$n_g := \left\lceil \frac{16}{c_d \Delta_g^2} \log T \right\rceil$$

Then for all $n \geq n_g$,

$$\begin{aligned}\exp\left(-\frac{c_d n \Delta_g^2}{2}\right) &\leq \exp\left(-\frac{c_d n_g \Delta_g^2}{2}\right) \\ &\leq \exp(-8 \log T) \\ &= T^{-8}\end{aligned}$$

and similarly $\exp(-n\Delta_g^2/8) \leq T^{-2}$ for a suitable constant. Hence, for $n \geq n_g$,

$$\Pr(B_t(g) \mid N_g = n) \leq C_1 T^{-2} \tag{3.20}$$

for some constant C_1 depending only on d.

To bound the expected number of times cell g is selected, we split the sum according to whether $N_g(t-1) < n_g$ or $N_g(t-1) \geq n_g$

$$\begin{aligned}\mathbb{E}[N_g(T)] &= \sum_{t=1}^{T} \Pr(g(s_t) = g) \\ &\leq n_g + \sum_{t=1}^{T} \Pr(g(s_t) = g, N_g(t-1) \geq n_g)\end{aligned}$$

On $\{N_g(t-1) \geq n_g\}$, selecting g implies $B_t(g)$. So,

$$\begin{aligned}\Pr(g(s_t) = g, N_g(t-1) \geq n_g) &\leq \Pr(B_t(g), N_g(t-1) \geq n_g) && \text{(I)} \\ &= \mathbb{E}[\Pr(B_t(g) | \mathcal{F}_{t-1}) \mathbb{1}\{N_g(t-1) \geq n_g\}] && \text{(II)} \\ &\leq \mathbb{E}[C_1 T^{-2} \cdot \mathbb{1}\{N_g(t-1) \geq n_g\}] && \text{(III)} \\ &\leq C_1 T^{-2}\end{aligned}$$

with the following justifications:

(I) $\{g(s_t) = g\} \subseteq B_t(g)$

(II) $\mathcal{F}_{t-1}$ measurable/non-measurable split

(III) result from Equation 3.20

Therefore, with $C_1/T \leq 1$ for $T \geq 3$,

$$\mathbb{E}[N_g(T)] \leq n_g + TC_1T^{-2} = n_g + C_1T^{-1} \leq 2n_g$$

and finally substituting the value of n_g gives

$$\mathbb{E}[N_g(T)] \leq C_2\frac{\log T}{\Delta_g^2}$$

for each suboptimal cell g where C_2 only depends on d.

To summarize, the cumulative regret relative to the best cell is

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{T}\mu(x_{g^*}) - \mu(s_t)\right] &= \sum_{g\in\mathcal{G}}\Delta_g\mathbb{E}[N_g(T)] \\ &\leq \sum_{g:\Delta_g>0}\Delta_g \cdot C_2\frac{\log T}{\Delta_g^2} \\ &= C_2\log T\sum_{g:\Delta_g>0}\frac{1}{\Delta_g} \qquad (3.21)\end{aligned}$$

Next, we bound the worst-case control of the gap-dependent term via an ε-split. For any threshold $\varepsilon > 0$, split cell into: near optimal $\mathcal{G}_{\leq\varepsilon} = \{g : \Delta_g \leq \varepsilon\}$, and clearly suboptimal $\mathcal{G}_{>\varepsilon} = \{g : \Delta_g > \varepsilon\}$. Trivially, regret from near-optimal cells is at most $T\varepsilon$ because each play can be upper bounded by ε regret.

$$\sum_{g:\Delta_g\leq\varepsilon}\Delta_g\mathbb{E}[N_g(T)] \leq \varepsilon\sum_g\mathbb{E}[N_g(T)] = \varepsilon T$$

For clearly suboptimal cells, using Equation 3.21,

$$\begin{aligned}
\sum_{g:\Delta_g>\varepsilon} \Delta_g \mathbb{E}[N_g(T)] &\leq C_2 \log T \sum_{g:\Delta_g>\varepsilon} \frac{1}{\Delta_g} \\
&\leq C_2 \log T \sum_{g:\Delta_g>\varepsilon} \frac{1}{\varepsilon} \\
&\leq C_2 \log T \frac{|\mathcal{G}|}{\varepsilon} \\
&\leq C_3 \log T \frac{h^{-d}}{\varepsilon} && \text{using Equation 3.17}
\end{aligned}$$

for some constant C_3.

Combining results from the ε splits,

$$\mathbb{E}\left[\sum_{t=1}^{T} \mu(x_{g^*}) - \mu(s_t)\right] \leq \varepsilon T + C_3 \log T \frac{h^{-d}}{\varepsilon}$$

Since within one cell, the Lipschitz variation is $O(Lh)$, the grid optimum $\mu(x_{g^*})$ is only meaningful up to Lh, so we set $\varepsilon = Lh$. Upon substitution, the cumulative regret relative to the best cell is

$$\mathbb{E}\left[\sum_{t=1}^{T} \mu(x_{g^*}) - \mu(s_t)\right] \leq LhT + C_3 \log T \frac{h^{-(d+1)}}{L} \tag{3.22}$$

Continuing from Equation 3.18, we can begin to form the overall cumulative regret bound.

$$\begin{aligned}
\mathbb{E}[R_T] &= \sum_{t=1}^{T} \mu(s^*) - \mu(x_{g^*}) + \sum_{t=1}^{T} \mu(x_{g^*}) - \mu(s_t) \\
&\leq TL\sqrt{d}h + \sum_{t=1}^{T} \mu(x_{g^*}) - \mu(s_t) && \text{from Equation 3.18} \\
&\leq C_4 LhT + C_3 \log T \frac{h^{-(d+1)}}{L} && \text{from Equation 3.22 and } C_4 := 1 + \sqrt{d}
\end{aligned}$$

What remains is to optimize the bandwidth with a schedule so as to obtain a useful regret bound. By finding the stationary point,

$$f(h) := C_4 LhT + C_3 \log T \frac{h^{-(d+1)}}{L}$$
$$f'(h) = C_4 LT - C_3 \frac{\log T}{L}(d+1)h^{-(d+2)} = 0$$

we obtain the optimal value of h

$$h = \left(\frac{\log T}{L^2 T}\right)^{\frac{1}{d+2}}$$

Plugging h into the regret bound gives

$$\mathbb{E}[R_T] \leq C L^{\frac{d}{d+2}} T^{\frac{d+1}{d+2}} (\log T)^{\frac{1}{d+2}}$$

This is a standard nonparametric Lipschitz-type rate $T^{\frac{d+1}{d+2}}$ with a mild $(\log T)^{\frac{1}{d+2}}$ factor. □

3.5 Empirical Results

To truly evaluate the efficacy of C_3 in the real world, we conduct three experiments.

3.5.1 Coupled Arms Simulation

The following simulated example demonstrates that ϕ can capture the notion of coupling. Recall coupled arms generalize correlated arms by ensuring that correlation persists over time.

Suppose there is a set of arms $\{a_0, a_1, ..., a_6\}$ without any notion of a context. We call a_0 the *anchor arm* where the corresponding reward distribution is Bernoulli(μ_0). At time t, μ_0 is randomly sampled from a Uniform$(0, 1)$ distribution. Then, the remaining μ_i for the rest of the arms are sampled such that they are either positively or negatively correlated with μ_0. The chosen degree of correlation is fixed for any time for all arms with respect to the anchor arm. The arms are sampled to obtain (a_{it}, r_{it}) pairs to learn μ_i in time t until the end of the episode, where this entire process repeats for time $t + 1$.

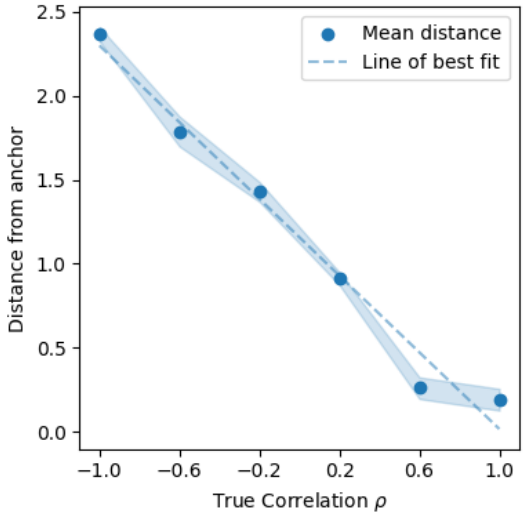


Figure 3.3: Distance from the anchor arm embedding as a function of correlation ρ with 1.96 sigma error bars over 10 random seeds.

Complete details of the generation of coupled arms are described in Appendix A.1. The chosen true correlations for arms $a_1, ..., a_6$ are -1.0, -0.6, -0.2, 0.2, 0.6, 1.0 respectively. For example, since a_1 is strongly negatively correlated to the anchor, if $\mu_0 = 0.9$ then it is very likely that $\mu_1 \approx 0.1$. On the other hand, μ_5 would be in the vicinity of 0.9.

All μ_i's are sampled 200 times where for each time, 100 reward samples are collected. These reward samples are used to train ϕ using Algorithm 1 with $T = 200$ time intervals. A summary of arm embeddings is visualized in Figure 3.3. It follows the expectation that the more coupled a_i is to a_0, the distance to a_0 is lower, and vice versa. To view one example of the spatial positioning of the arm embeddings in a scatter plot, see Figure 3.4.

3.5.2 Contextual Bandit Experiments

This experiment demonstrates the efficacy of C_3 on the classic bandit problem. In the footsteps of the work by Zhou et al. [174], we evaluate using the same datasets from the OpenML repository by [149, 16], namely `shuttle` [80], `MagicTelescope` [20], `covertype` [19] and `mnist` [88]. For this set of experiments, the context space is the input space, and $\mathcal{A}$ is the corresponding label space. The reward is one if the selected arm matches the ground truth label, otherwise zero. Note that this experiment does not fully match the

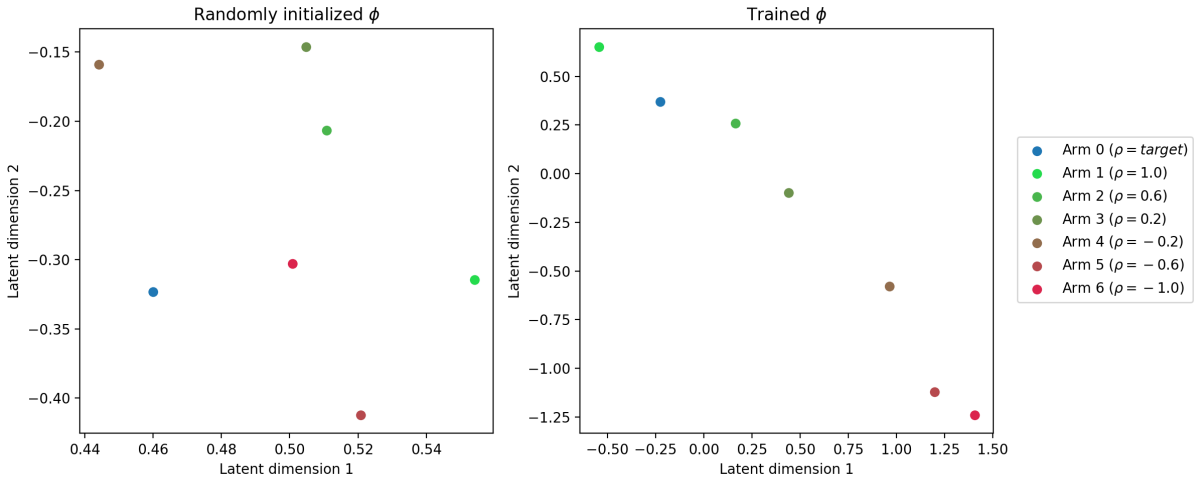


Figure 3.4: The left figure shows the arm embeddings prior to any fitting while the right figure shows the arm embeddings after training using Algorithm 1. The blue point is the anchor point and the green points are positively correlated to the μ of the blue point while the red points are negatively correlated.

standard bandit setting as C_3 targets a more general problem.

C_3 requires historical samples for ϕ to be trained, where historical samples are uncommon for bandit experiments but incredibly common for industry use cases. To ensure a fair comparison, we ensure that other baseline methods, such as LinUCB [92], Thompson sampling for contextual bandits (LinTS) [3], SquareCB [46], NeuralUCB [174], and neural Thompson sampling (NeuralTS) [171], are given the same amount of information. All algorithms are updated using a subset for training and evaluated on a different test subset. This is necessary to avoid contamination when training ϕ. The test split contains 1000 unseen samples. Appendix A.3 contains an ablation study of C_3 with different RBF bandwidth values.

The results are shown in Figure 3.5. C_3 outperforms most of the algorithms in most datasets. C_3 outperforms the next best algorithm for each dataset by 5.7% lower cumulative regret on average. In `shuttle`, `MagicTelescope`, and `covertype`, these problems are of lower dimensionality and more linearly separable hence LinUCB and SquareCB (with a linear regression oracle) perform well. On the other hand, NeuralUCB excels in the MNIST dataset since the multilayer perceptron works well with high-dimensional data and is retrained almost every timestep.

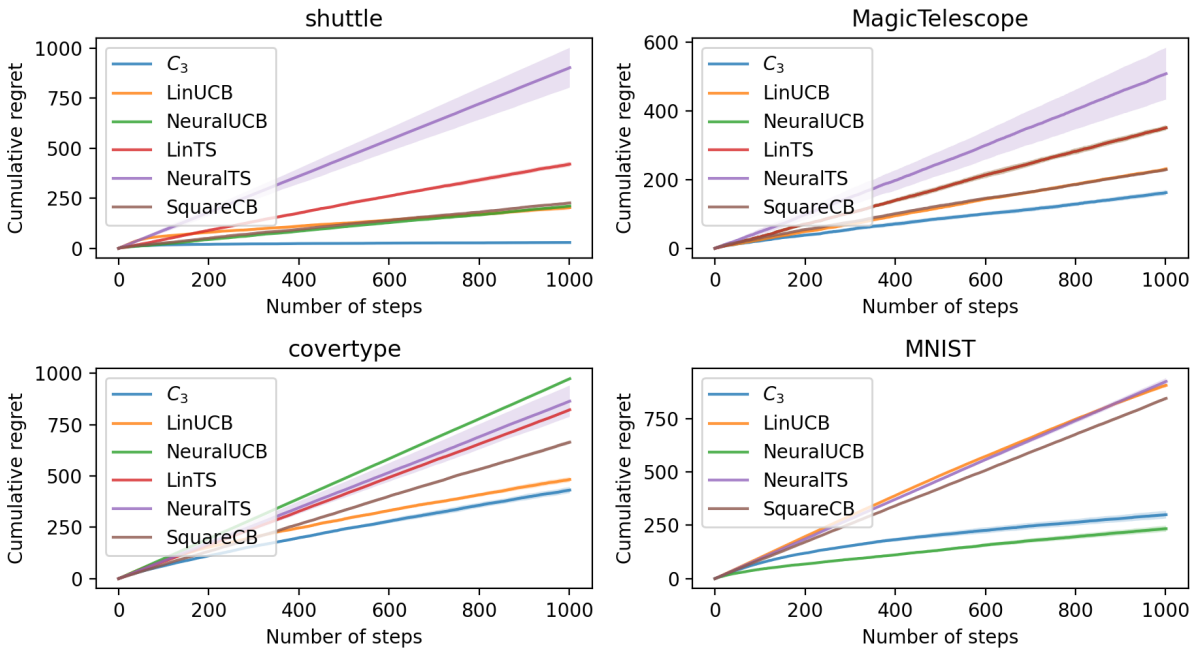


Figure 3.5: Cumulative regret of the test split of the four datasets with 1.96 sigma error bars over 10 random seeds. Note that in MNIST, LinTS cannot be computed due to numerical issues from the high dimensionality. In MagicTelescope, NeuralUCB and LinTS almost completely overlap because they all repeatedly exploit the same action after the initial steps.

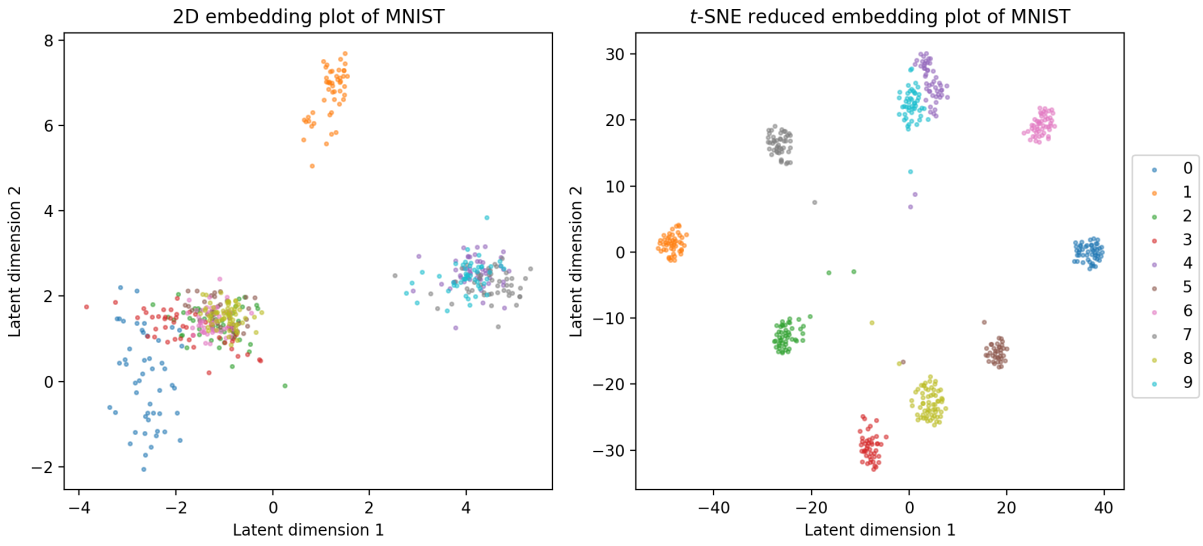


Figure 3.6: Embedding space of MNIST digits with correct arms chosen where the left shows the embedding vectors of ϕ with an output dimension of 2, and the right shows the embedding vectors of another ϕ with an output dimension of 8 but uses t-SNE [147] for visualization.

To visualize the learned embedding space for MNIST, Figure 3.6 shows two possible perspectives of ϕ. Choosing a two-dimensional embedding space $\mathcal{S}$ is suboptimal for MNIST as it is not fully separable, as demonstrated in the left figure. However, choosing a higher-dimensional embedding space, such as 8 dimensions, shows a much better representation as each digit in MNIST is separable than the previous version. The only exception is the four-nine pairs, which are similar visually and a regular multi-layer perceptron with one hidden layer has difficulty separating them. In a practical scenario, one should use a more specialized architecture suitable to the input domain or task, such as a convolutional neural network [89, 47] in this case.

3.5.3 News Recommendation

C_3 will be evaluated on news recommendation which is a realistic example for the non-stationary bandit described in Section 3.1. We use the Microsoft News Dataset (MIND) by [163] for evaluation. The context will be the frequency of a user's historical visits by news category. The arm space is the set of valid news articles to recommend. The reward is whether the user clicks the chosen news article. The objective of an algorithm is to minimize the cumulative regret. For clarity in results, regret is measured relative to the best performing algorithm where at time t, 0 is given to the best performing algorithm and the rest of the algorithms are given their respective relative regret.

We form dense representations of news articles from their titles using the pooler output of a BERT model [37]. We use principal component analysis to reduce the dimensions to 64 to be used as arm embeddings. Whenever a user visits, there is a small collection of possible articles to recommend, up to eight articles, but the valid arm set varies for each user.

In this set of experiments, C_3 will be configured so that every 100 steps, it will randomly remove approximately 20% of samples in $\mathcal{D}_{\text{ref}}$ to account for the concept drift. The hypothesis is that because of the Lipschitz assumption, more recent samples would be more relevant in estimating the mean rewards.

Due to the rigidity of other bandit algorithms, baseline algorithms in Section 3.5.2 could not effectively target the complexities of the news recommendation problem for various reasons. They require a fixed set of arms. Furthermore, LinTS struggles with numeric instability.

However, we add more specialized and more flexible variants of those algorithms. We compare C_3 with one of the most popular recent designs for recommender systems: the two-tower neural network. [72] uses a deep encoder for query information and another deep

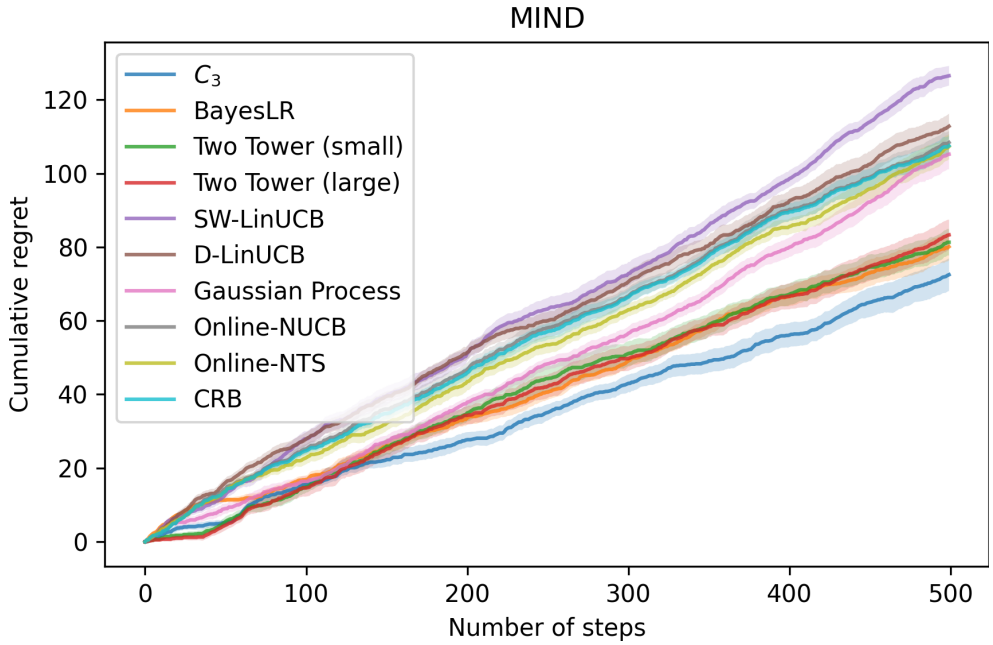


Figure 3.7: Cumulative regret of the MIND dataset with 1 sigma error bars over 10 random seeds. "small" and "large" refer to the relative number of parameters in the two tower models.

encoder for item information then uses the cosine similarity between the two embeddings. A Gaussian process with forgetting is also included [77], where the forgetting is necessary since it cannot handle the entire history and needs to account for the non-stationarity. A similar baseline is using a Bayesian linear regression model that takes a concatenated vector of user and article features and returns the mean and standard deviation of estimated rewards, similar to the Gaussian process. Linear and neural baselines from Section 3.5.2 are modified to handle the non-stationarity elements. We also compare with the contextual restless bandit algorithm [28]. Full experimental details are found in Appendix A.4.

The result can be seen in Figure 3.7. C_3 demonstrates a click lift of 12.4% compared to the next best baseline. Initially, C_3 performs worse than the two-tower approaches (which are static). As the drift of click rate increases in magnitude, C_3 begins to adapt to the drift as it removes older samples in $\mathcal{D}_{\text{ref}}$ while the other algorithms incur regret at the same rate.

3.6 Discussion

Effective Data Utilization Unlike typical contextual bandit settings, C_3 requires the use of historical data to "warm-start" the algorithm. In reality, advertisement campaigns and pricing schemes will involve some degree of human-designed policy at the initial stages which means there could be some data albeit possibly suboptimal. Suboptimality is not a major issue for the training of ϕ since the main objective of ϕ is to learn instantaneous reward distributions and coupling, not optimality. As a result, ϕ can effectively utilize samples from prior campaigns or trials in an off-policy manner. Furthermore, ϕ can be resilient to concept drifts so data from a different time period may still be utilized. Overall, this approach enables transfer learning of dynamics in bandits – distributions change over time but some distributions change together.

Generalization to Embedding Models This work demonstrates that Thompson sampling in an embedding space can serve as an exploration mechanism. However, only a simple multi-layer perceptron is used as an embedding model. There are no restrictions on the model architecture or inputs as long as it is a cost-sensitive regression model. Using a model that has already been pretrained to learn a more complete representation of the input data, such as BERT [37] as done here for textual data, should enable greater sensitivity when learning the dynamics of the environment.

Contrastive Learning Algorithm 1 shows a unique approach to learning an embedding space, one that is calibrated in a sense where it is optimized directly in terms of its usefulness to the prediction of left-out samples. The important element here is the measurability to a task, allowing distance to be more intentional when learning unlike margin-based approaches such as Triplet loss [130].

Explainability A side effect of operating on an embedding model is the ability to visualize the learned embedding space which can be useful for applications that require some transparency or explainability. The mean predictions (IWKR) or the Beta posterior distribution are inherently based on a metric space that can be reduced to a two-dimensional space with t-SNE [147] or UMAP [106]. This allows users to understand the contribution of each reference sample towards the selection of an arm.

Elements of In-Context Learning A reference dataset can be seen as a set of in-context examples – making C_3 a form of in-context learning algorithm. If one chooses to

view it this way, C_3 is a linear-time ICL algorithm in the same classification as DeepONets [31]. Transformer-based ICL approaches are commonly quadratic time algorithms due to the self-attention mechanism [50]. Finally, a Gaussian process is a cubic time algorithm [142] that also uses kernels and reference samples to predict values, doing so in a principled manner but requiring the inversion of the kernel matrix.

Limitations of C_3 The transductive learning aspect of IWKR and importance weight updates can result in substantial numerical instability since it relies on many sum and division operations of floating points. This effectively precludes the use of quantization. Also, as an algorithm that relies on a dataset for inference, it may not scale to millions of points without some form of sampling if speed is crucial in the use case. The hyperparameter tuning of ϕ can be slightly challenging because the RBF kernel used in IWKR can result in vanishing gradients if points are too near or far from some query point, so the choice of bandwidth of the RBF kernel is important. More crucially, the performance of C_3 is dependent on heuristics and whether the learned embedding space is well-calibrated.

3.7 Conclusion

This chapter begins with the sequential problem of selecting arms and observing rewards, starting from tabular data and extending toward settings that jointly involve tabular and textual data. The inspiration of C_3 comes from acknowledging real-world constraints: non-stationarity, high-dimensional inputs, flexible output spaces and lightweight adaptation.

For C_3 to be effective, certain trade-offs are required, and they come in the form of needing some historical data and useful features describing the arms. Fortunately, these requirements are often feasible in practice. To support this point with an example, deep learning has been highly successful in transforming machine learning and artificial intelligence and is mostly attributed to the availability of huge amounts of data, creating an entire field of study on neural scaling laws along the way [66, 29]. Historical data is now more readily available than ever before [175], and this allows us to warm-start bandit algorithms. Arm features can be easily extracted from text (encoded with a language model), heuristics, or empirical statistics.

While there is a large collection of bandit algorithms optimizing for asymptotic regret bounds and, in some cases, handling one or two key generalizations to the type of tasks, they do not sufficiently handle the practical requirements. Although Section 3.4 provides a modest sublinear regret bound, but one has to recall that regret bounds are

asymptotic statements about performance as the number of training points grows – an obvious dichotomy to the goals of practitioners. Nevertheless, C_3 contributes to the broader literature on bandit algorithms by offering an alternative to theory-focused works.

Chapter 4

A Universal Encoder for Tables

Tabular data may include numerical values, column names, categorical values, free-form text and general metadata. Many approaches with tabular inputs discard a significant amount of potentially useful information. This chapter explores ways to leverage tabular data beyond numbers and indexed categories.

The main deliverable of this chapter is the architecture for a universal encoder for tables, which we call *basis transformers*. Basis transformers offer a novel take on specialized tabular models. The design begins from identifying and abiding by several key desiderata that a tabular model should have. This new design philosophy rules out many conventions, such as standardization or one-hot encoding of categorical variables. Two key mechanisms built into the architecture negate the need for those preprocessing techniques. Representing numbers as a sequence of bits in the sign-magnitude representation format [52] is a natural fixed-vector representation for scalar values and avoids the need for standardization since the values are naturally standardized to be either zero or one. The other mechanism is learning a set of latent basis vectors to extract relevant information from text, usually in the form of an arbitrarily long sequence.

The efficacy of this encoder design is validated experimentally and via ablation studies. Understanding the proposed architecture requires an in-depth knowledge of transformers, hence readers are encouraged to review the background material in Chapter 2.2.

This chapter focuses on regression tasks, which align more closely with the proposed innovations, but they could theoretically be extended to handle classification tasks.

The main contributions of this chapter are:

1. We propose *basis transformers* ($\mathcal{BT}$), a universal encoder for heterogeneous tables containing numerical, categorical, textual, and missing values.
2. We introduce two additional desiderata for tabular models: preserving tabular hierarchy and preserving numerical scale and precision.
3. We use sign-magnitude representation for numerical inputs and targets, removing the need for dataset-specific standardization.
4. We introduce *basis compression*, which converts variable-length inputs into fixed-size representations using learned basis queries.
5. We develop an architecture that preserves column-value relationships, models interactions between columns, and repeatedly attends to the original inputs.
6. We propose an adaptive loss-reweighting scheme for multi-task regression.
7. We show that $\mathcal{BT}$ achieves stable multi-task training and strong performance against language-model, tree-based, and deep-tabular baselines.

4.1 Terminology

To assist with understanding the complexity of the structure of a table and its constituents, we provide a systematic definition of tabular data, starting with an example table in Table 4.1.

Table 4.1: An example table to assist with the explanation of terminology.

Age	Continent	Job Title	Income (USD)
24	North America	Sales Associate	53,869.12
`NaN`	Asia	Project Manager (Interim)	110,246.61
41	`null`	Customer Support Representative – Europe	52,632.90

Definition 3. A *column name* refers to the text providing semantic context to the *entries* it corresponds to.

Definition 4. An entry can be either

- a *numeric entry* if its data type is an integer or a real number,
- a *categorical entry* if its data type is from a small set of possible categories (either integers or strings), or
- a *textual entry* if its data type is a string and cannot be guaranteed to be contained within a small set of possible values.

Definition 5. A *missing value* is a special value assigned to indicate the entry is empty or is of an invalid value. `NaN` (Not A Number) is the notation for missing numeric entries, while `null` is the notation for missing textual entries.

For example, in Table 4.1, each of the four phrases in the header of the table is classified as the column name. The "Age" column contains three numeric entries, of which two are integers and one is a missing value. The "Continent" column contains three categorical entries, of which two are strings, and one is a missing value. The "Job title" column has three textual entries. While "Continent" and "Job title" are both string types, we can reasonably assume that there are only seven possible values in continents, while job titles can vary substantially, with the "(Interim)" and "– Europe" suffixes shown as examples of such inconsistencies.

The "Income (USD)" column has three numeric entries with two-decimal real values. In a *regression* task, the goal is to model the relationship between the features, such as age, continent and job titles, and target, such as income. The constraint is that the target must be a numeric entry.

We will formally define a tabular regression task. Let the dataset be $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y} \subseteq \mathbb{R}$. The feature space $\mathcal{X}$ is a product space of columns

$$\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_d \tag{4.1}$$

where each column $\mathcal{X}_j$ may be a real number, integer, categorical, text or a missing value.

Assuming the observations are sampled from some unknown joint distribution $P_{X,Y}$, the goal is to learn a regressor $f : \mathcal{X} \to \mathcal{Y}$ from the hypothesis class $\mathcal{F} \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$.

The objective can be defined as

$$f^* \in \operatorname*{argmin}_{f \in \mathcal{F}} \mathbb{E}_{(X,Y)\sim P_{X,Y}} \left[\ell(f(X), Y)\right] \tag{4.2}$$

where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ is a task-appropriate loss function.

Since $P_{X,Y}$ is likely unknown in practice, we use empirical risk minimization [150] to solve

$$\hat{f} \in \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \tag{4.3}$$

4.2 Desiderata

This work is motivated by several shortcomings in publicly available models in terms of their ability to handle various tasks. While some are clearly superior in terms of their raw performance in standard benchmarks, those benchmarks are selectively narrow in terms of the scope of the problem. This narrowness is partly understandable, as benchmarks must accommodate a wide range of tabular models.

Van Breugel and Van Der Schaar published a position paper on tabular foundation models, accompanied by four desiderata, which are listed below [146].

(**D1**) Handle entries of various types, including numeric, categorical, text, dates, and missing data

(**D2**) Handle heterogeneous feature space, in particular, tables with different sets of columns and categories concurrently

(**D3**) Leverage column names or other metadata

(**D4**) Perceive columns in an equivariant or invariant manner

These desiderata align with our direction, but they could be extended. We propose two additional desiderata as part of our contributions.

(**D5**) Preserve and exploit the inherent hierarchy of tabular structures.

Each table contains many rows; each row contains several entries (columns); each entry could contain a sequence (e.g. text). This separation should remain in the modelling process so that the inductive bias of the algorithm matches the nature of the data. Otherwise, flattening each row into an unstructured sequence of tokens ignores the explicit attribute-value pair structure of a table.

(**D6**) Preserve the scale and precision of numeric values.

Data normalization or standardization is not possible in zero-shot learning and hinders transfer learning across multiple tasks, where values in different ranges should not be made

similar. The most common barrier is the optimization constraint that tabular regression techniques assume that numerical values are normalized in some standard range for stability and numerical reasons [131], including avoiding exploding and vanishing gradients.

In the next few sections, these desiderata will be referenced to determine whether architectures, designs or learning objectives satisfy them.

4.3 Related Work

There is extensive literature on tabular models as they are common and often structured in relatively low-dimensional formats. We focus on related works that are more relevant to our proposed design. To maintain clarity, we focus on the two main desiderata: D1 and D2.

Gradient boosting decision trees (GBDT) have been the de facto method for tabular classification and regression tasks where XGBoost [27] has remained highly competitive for many years [23], famously on Kaggle competitions [168]. This can mainly be attributed to the gradient boosting technique at scale as well as the effective choice of inductive biases. For a relatively fast algorithm, it is incredibly effective and robust. Yet, tasks involving textual data or a variable number of columns (D1, D2) require specialized handling and preprocessing. NODE [117] offers a differentiable tree-like approach that outperforms shallow GBDTs but is significantly slower than gradient boosted trees.

CatBoost introduced two new key ideas. The authors claimed that regular gradient boosting shows target leakage at training time, degrading generalization during test-time [119]. To mitigate this, CatBoost uses ordered boosting which estimates residuals and updates using only models built from earlier examples to simulate a test-time setting. The other contribution is using an ordered target statistic for categorical features, which involves sampling a random permutation of the training set and computing category statistics.

With the recent trends to leverage advances with the transformer architecture or pre-trained language models, neural architectures have become a popular choice. TabNet [5], motivated by the success of the transformer architecture [151], uses an encoder-decoder architecture that can emulate the workings of decision trees and induce interpretability. TabTransformer [73] also uses a transformer with two separate input channels for categorical and continuous features, showing marginal gains over multi-layer perceptrons (MLP) on tabular classification and regression tasks. TabNet and TabTransformer work well in some cases but have to be tailored to one specific task at a time, hence failing D2.

FT-Transformer [53] followed a BERT-like [37] approach with a classification token and improved tokenization strategy. SAINT [135] has a similar design, extending with inter-sample attention and contrastive pretraining but both are unable to handle free-form text (D1) and require latent embeddings per categorical feature (D2). In addition, the space of hyperparameters for those neural approaches is significantly larger than that of tree-based models and is practically more difficult to tune.

TabM returns to the roots of neural networks – the multi-layer perceptron (MLP). TabM is not a foundation model. The authors offer a new perspective on tabular deep learning by constructing a parameter-efficient ensembling technique [54]. It extends the BatchEnsemble mechanism [159] where an ensemble of models shares a backbone weight matrix while each has specific adaptors. They also explored variants by permuting across the ordering of shared backbone weights, adaptors and additional weight components. While each submodel of TabM is weak and often overfits [54], in aggregate, the ensemble is effective in overcoming the shortcomings of MLPs.

TabPFN [69] solves the more difficult task of in-context learning for tabular predictions. They adopted a Bayesian perspective and pretrained on synthetic datasets, showing success in time-constrained settings despite being unable to fully handle free-form text. The synthetic generation is based on structured causal models, which proved to be very effective in emulating the underlying properties of real tabular data. Two models on tabular data and time series are TabBert [114] and UniTTab [101] where temporal dependencies are modelled with a sequence transformer, but UniTTab extended with a sine-cosine numeric encoding and improved handling of heterogeneous types. These models cannot handle values outside the predefined set of categories (D1).

CARTE [79] views tabular data as graphlets. With graph attention networks and vast amounts of data for knowledge graphs, the pretrained model performs well in regression and classification tasks upon finetuning on small amounts of data. Nevertheless, it is reliant on its use of power transforms, and power transforms are specific to a dataset, which limits its ability in cross-table tasks (D2). TabLLM [65] and Tabula-8B [49] view tabular data as a language subtask. By inserting tabular information into a natural language template, pretrained large language models (LLM) are finetuned on datasets and directly used for predictions in the token space. Tabula-8B took it one step further by pretraining a Llama 3-8B model [55] on over 2.1 billion rows of tabular data, making it a foundation model for tabular tasks. When framed as a natural modeling subtask, it requires a large number of parameters, high memory and computational resource consumption, and LLMs sometimes struggle with numbers [43]. Nevertheless, these LLM-based models fully satisfy D1 and D2. We summarize the extent to which the mentioned works satisfy the desiderata in Table 4.2.

Table 4.2: Summary of related works and whether they fulfill each desideratum. A ✓means that the algorithm fulfills that desideratum, otherwise it does not.

Algorithm	D1	D2	D3	D4	D5	D6
GBDT, NODE					✓	✓
TabNet, TabBert, UniTTab, FT-Tr., SAINT		✓		✓	✓	
TabPFN, TabTransformer					✓	
CARTE	✓		✓	✓	✓	
TabLLM and Tabula-8B	✓	✓	✓			✓
Ours	✓	✓	✓	✓	✓	✓

4.4 Method

4.4.1 Vectored Numeric Representation

It is natural for a neural approach to construct a learnable linear map to transform scalar numeric values to a higher-dimensional latent embedding. FT-Transformer adopts this strategy [53]. While this design is valid, it creates an additional chain of dependencies in the model, requiring more data to search for an optimal representation. We propose using a known transformation that is a standard representation introduced in computer science curricula – the sign-magnitude representation (SMR) [52].

Let $v \in \mathbb{R}$ be a scalar value and e_{num} be a sign-magnitude encoder. Let h be the number of *high bits* and ℓ be the number of *low bits*. High bits refer to the coefficients of natural number base-2 exponents, while low bits refer to the coefficients of negative base-2 exponents. We call a_0 the *sign bit*. If v can be expressed in terms of

$$v := (-1)^{a_0} \cdot (a_1 2^{h-1} + a_2 2^{h-2} + \cdots + a_h 2^0 + a_{h+1} 2^{-1} + a_{h+2} 2^{-2} + \cdots + a_{h+\ell} 2^{-\ell}) \quad (4.4)$$

for some coefficients $a_0, a_1, ..., a_{h+\ell} \in \{0, 1\}$, then the encoding of v is

$$e_{\text{num}}(v) = \begin{bmatrix} a_0 & a_1 & \cdots & a_{h+\ell} \end{bmatrix}. \quad (4.5)$$

Sign-magnitude representation is a simple implementation for computer hardware to represent real numbers. Its simplicity is suboptimal as a floating-point number for several reasons that are not relevant here. From the perspective of a representation for a neural architecture, it has several benefits.

Standardized Representation First, it is a binary representation, which directly enables D6 by removing the need for standardization as neural networks work best with vectors of small magnitudes [7]. One might claim that SMR truncates real values if the number of high bits or low bits is insufficient, leading to poor precision. However, it is acceptable or perhaps desirable for neural networks to be insensitive to the extremely large values (e.g. $> 2^{32}$) and the distant decimal places of a value (e.g. 2^{-16}). This means that there is a natural barrier against ranges of values the model was not trained on, making the neural network more robust.

Efficient The operation for computing SMR is highly efficient and can be parallelized. The idea is detailed in Algorithm 3. Additional steps to determine out-of-bound inputs are recommended.

Algorithm 3 Sign-magnitude representation operation

Inputs: Numeric value $v \in \mathbb{R}$, number of high bits h, number of low bits ℓ
Initialize the base-2 exponents $\beta \leftarrow [2^{h-1}, 2^{h-2}, ..., 2^{-\ell}]$
Determine the sign bit
Ensure positivity $v \leftarrow |v|$
$e_{\text{num}}(v) \leftarrow (v \mod \beta)/\beta$ ▷ Broadcasting
Round $e_{\text{num}}(v)$ to the nearest integer
Prepend the sign bit to $e_{\text{num}}(v)$
return $e_{\text{num}}(v)$

Additive Representation SMR is an additive representation. Specifically, flipping a non-sign bit affects the scalar value by a fixed value based on the position of the bit. The additive property is simple for a model to learn the representation. Unlike IEEE-754 [1] where the mantissa is normalized and the exponent is not additive, the representation is more complex for learning.

Resolution This representation splits a scalar into a vector of bits where each non-sign bit specializes in a certain magnitude scale which we call *resolution*. The last bit focuses on small values while the second bit focuses on large values. Resolution-aware property can be useful in cases where D6 is needed. This also works well with attention, e.g. to select positive integer values, a query vector would be one where the last ℓ bits are -1 while the rest of the bits are $+1$.

4.4.2 Input Format

We view a row of a table as an unordered set of column name c_j and entry value v_j pairs $\{(c_j, v_j)\}_{j=1}^{C}$, where C is the number of columns. The set formulation directly fulfills D4, and any model that treats it as a variable-length sequence can handle a row from any table, fulfilling D2. The (c_j, v_j) pairs fulfill D3.

We process all column names c and entry values v in parallel and in the same fashion. We encode numeric values using sign-magnitude representation (SMR) and textual inputs using the hidden state of a pretrained BERT [37] which are then passed into the model, fulfilling D1. We use a learnable linear projection layer to downscale BERT vectors and another learnable linear projection layer to upscale the numeric vectors to embed them in the same representation space. This also helps to differentiate between text and numbers.

If all columns are of numeric types then $L = 1$ since SMR is a single vector. Otherwise, L is set to the maximum length of the BERT embedding vectors and the entries are padded accordingly. Every missing value (NaN) is replaced with a special learnable token of length 1. Therefore, the dimensions of the encoded column names c and encoded values v of a row are:

$$e_{\text{input}}(c) \in \mathbb{R}^{B \times C \times L \times D} \qquad e_{\text{input}}(v) \in \mathbb{R}^{B \times C \times L \times D} \tag{4.6}$$

where B refers to the batch size, C refers to the number of columns, L refers to the maximum sequence length of the entry values, and D refers to the embedding dimension of each token. An example is shown in Figure 4.1.

4.4.3 Architecture

We call our proposed architecture *basis transformers* ($\mathcal{BT}$). The inputs to $\mathcal{BT}$ are the two embedding tensors in Equation 4.6. The output is the SMR of a numeric target. In a top-down manner, we introduce the architecture followed by the details of the novel components in subsequent subsections. A $\mathcal{BT}$ is composed of stacking multiple $\mathcal{BT}$ blocks sequentially, where a module of a $\mathcal{BT}$ block is shown in Figure 4.2.

Basis compression Lee et al. introduced the induced set attention block to overcome the quadratic time complexity in self-attention on input $X \in \mathbb{R}^{n \times d}$ [90]. They defined m inducing points $I \in \mathbb{R}^{m \times d}$ used as a query in a cross attention layer and yielding $H \in \mathbb{R}^{m \times d}$.

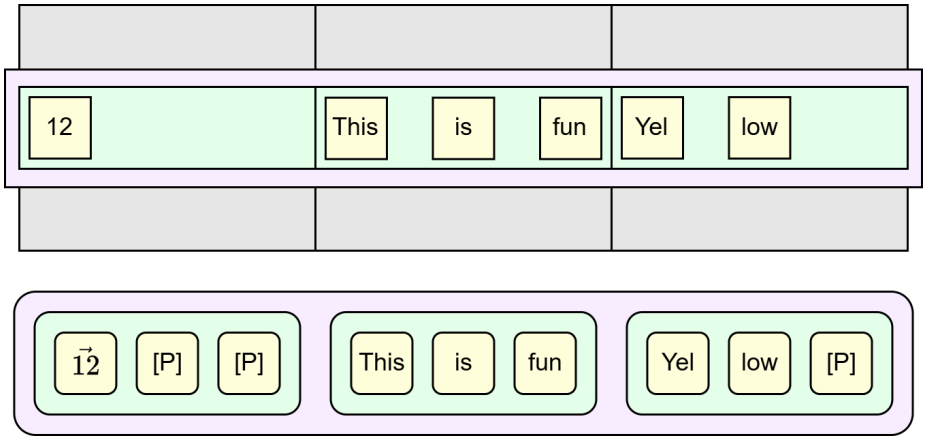


Figure 4.1: The top shows the original input row; the bottom shows a (C, L, D)-tensor padded entry value representation. `[P]` refers to the pad tokens. The processed row (purple) has $C = 3$ columns (green), each having $L = 3$ tokens (yellow) of size D. $\vec{12}$ denotes the scalar 12 encoded with SMR.

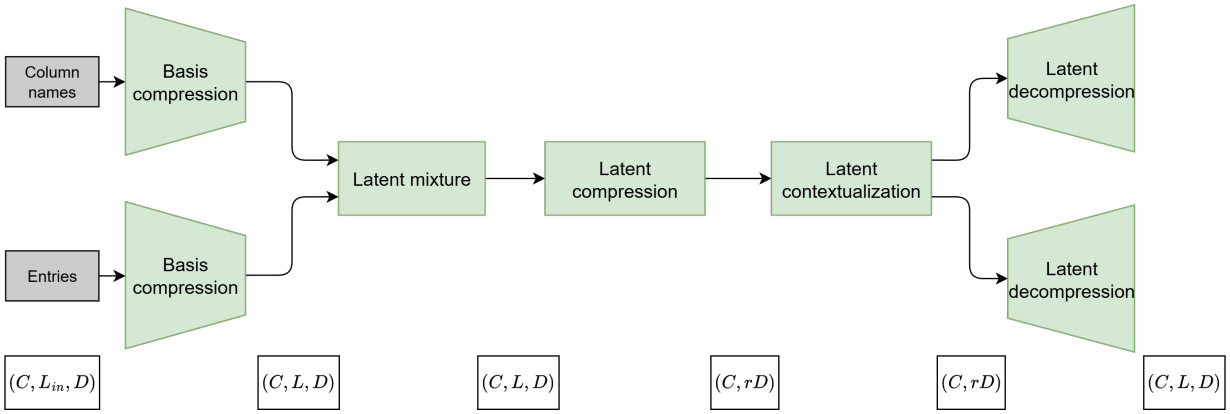


Figure 4.2: A simplified diagram of a basis transformer block with the tensor dimensions of the inputs and the intermediate latent tensors.

I is a learnable matrix. Finally, another cross attention layer is used where X is the query and H is the key-value, producing O as the final output.

$$H = \text{CrossAttn}(I, X) \in \mathbb{R}^{m \times d} \tag{4.7}$$

$$O = \text{CrossAttn}(X, H) \in \mathbb{R}^{n \times d} \tag{4.8}$$

where two arguments to CrossAttn are the query and key-value respectively.

The basis compression module is related to the induced set attention block but with some differences. Basis compression consists of only the first cross-attention layer and the inducing points. Instead of inducing points, we call them *basis queries*. They are used to summarize text and numeric values into a fixed-size sequence in terms of the learned bases. The first $\mathcal{BT}$ block has a set of learnable basis queries of size $\mathbb{R}^{L \times D}$ where it will be broadcast to $\mathbb{R}^{C \times L \times D}$ so that every column receives the same basis queries. This key mechanism enables homogeneity within the model as tables can contain mixed types.

Theorem 3 (Interpolation of covariances and prototypes)**.** *Let $\{(k_i, v_i)\}_{i=1}^n$ be any finite sequence of key-value vectors, and $B = \{b_1, ..., b_m\}$ be a fixed set of learnable basis queries. Define $s_{ri} = \langle b_r, k_i \rangle$ for each basis query b_r. Then the following statements hold.*

1. In the high attention temperature regime $\tau \to \infty$, each basis compressed vector satisfies

$$z_r^{(\tau)} = \bar{v} + \frac{1}{\tau} C_{vk} b_r + \mathcal{O}(\tau^{-2}) \tag{4.9}$$

where $\bar{v}$ is the mean of all v_i's, $\bar{k}$ is the mean of all k_i's, and the empirical key-value covariance matrix

$$C_{vk} = \frac{1}{n} \sum_{i=1}^{n} (v_i - \bar{v})(k_i - \bar{k})^\top$$

2. In the low attention temperature regime $\tau \to 0$, define the maximizer set $M_r = \text{argmax}_{1 \le i \le n} \langle b_r, k_i \rangle$. Then

$$\lim_{\tau \to 0} z_r^{(\tau)} = \frac{1}{|M_r|} \sum_{i \in M_r} v_i \tag{4.10}$$

If the maximizer is unique, i.e. $M_r = \{i_r^*\}$, then

$$\lim_{\tau \to 0} z_r^{(\tau)} = v_{i_r^*} \tag{4.11}$$

and if the unique-max gap is $\Delta_r = s_{ri_r^*} - \max_{i \neq i_r^*} s_{ri}$ and $\|v_i\| \le R$ for all i then

$$\|z_r^{(\tau)} - v_{i_r^*}\| \le 2R(n-1) \exp(-\Delta_r / \tau) \tag{4.12}$$

Theorem 3 suggests there are two characteristics of basis compression. For each basis query, the high attention temperature regime yields a global summary of the input sequence. In this regime, the compressed output approaches the uniform value average, while the first query-dependent correction, $C_{vk}b_r$, captures how the global value summary changes under a small attention bias in the direction of the learned basis query.

At the other end, the low temperature regime makes basis compression act as a localized prototype-like selector, collapsing towards the value vector associated with the most compatible key. The prototype represents questions like "does X contain something like this learned latent prototype?" In practice, basis compression typically operates between these two extremes, producing a soft mixture of global aggregation and localized selection.

Proof of Theorem 3 (high temperature regime). Fix a basis query b_r. Let $\varepsilon = 1/\tau$. Simplify notations with $b = b_r$, $s_i = \langle b, k_i \rangle$, $z^{(\tau)} = z_r^{(\tau)}$.

The attention weights can be written as

$$\pi_i^{(\tau)} = \frac{\exp(\varepsilon s_i)}{\sum_{j=1}^n \exp(\varepsilon s_j)}$$

Since adding the same scalar to every logit does not change the softmax, define the centered scores

$$a_i = s_i - \bar{s}, \qquad \bar{s} = \frac{1}{n}\sum_{j=1}^n s_j \tag{4.13}$$

Using the centered scores results in

$$\sum_{i=1}^n a_i = 0, \qquad \pi_i^{(\tau)} = \frac{\exp(\varepsilon a_i)}{\sum_{j=1}^n \exp(\varepsilon a_j)}$$

The Taylor expansion of the exponential around $\varepsilon = 0$ gives $\exp(\varepsilon a_j) = 1 + \varepsilon a_j + \mathcal{O}(\varepsilon^2)$. Applying this and the sum of a_j's being zero, the denominator of $\pi_i^{(\tau)}$ becomes

$$\sum_{j=1}^n \exp(\varepsilon a_j) = n + \mathcal{O}(\varepsilon^2) \tag{4.14}$$

Hence, the attention weights become

$$\begin{aligned}\pi_i^{(\tau)} &= \frac{1 + \varepsilon a_i + \mathcal{O}(\varepsilon^2)}{n + \mathcal{O}(\varepsilon^2)} \\ &= \frac{1}{n} + \frac{\varepsilon}{n} a_i + \mathcal{O}(\varepsilon^2)\end{aligned}$$

The basis compressed output in its general form

$$z^{(\tau)} = \sum_{i=1}^{n} \pi_i^{(\tau)} v_i$$

becomes

$$\begin{aligned}z^{(\tau)} &= \sum_{i=1}^{n} \left(\frac{1}{n} + \frac{\varepsilon}{n} a_i + \mathcal{O}(\varepsilon^2) \right) v_i \\ &= \bar{v} + \varepsilon \frac{1}{n} \sum_{i=1}^{n} a_i v_i + \mathcal{O}(\varepsilon^2)\end{aligned}$$

Since $a_i = s_i - \bar{s} = \langle b, k_i \rangle - \langle b, \bar{k} \rangle = \langle b, k_i - \bar{k} \rangle$, and the sum of a_i's being zero,

$$\begin{aligned}\frac{1}{n} \sum_{i=1}^{n} a_i v_i &= \frac{1}{n} \sum_{i=1}^{n} v_i \langle b, k_i - \bar{k} \rangle \\ &= \frac{1}{n} \sum_{i=1}^{n} (v_i - \bar{v}) \langle b, k_i - \bar{k} \rangle \\ &= \left[\frac{1}{n} \sum_{i=1}^{n} (v_i - \bar{v})(k_i - \bar{k})^\top \right] b \\ &= C_{vk} b\end{aligned}$$

Therefore, the high-temperature claim is

$$z_r^{(\tau)} = \bar{v} + \frac{1}{\tau} C_{vk} b + \mathcal{O}(\tau^{-2}) \tag{4.15}$$

□

Proof of Theorem 3 (low temperature regime). Assume the same set up with the high temperature regime proof.

Let $s^* = \max_{1 \le i \le n} s_i$ and define the maximizer set $M = \{i : s_i = s^*\}$. Since subtracting the same scalar to every logit does not change the softmax, the attention weights are

$$\pi_i^{(\tau)} = \frac{\exp((s_i - s^*)/\tau)}{\sum_{j=1}^n \exp((s_j - s^*)/\tau)}$$

For every $i \in M$, $s_i - s^* = 0$ so $\exp((s_i - s^*)/\tau) = 1$. On the other hand, for every $i \notin M$, $s_i - s^* < 0$ so $\exp((s_i - s^*)/\tau) \to 0$ as $\tau \to 0$. Therefore, as $\tau \to 0$,

$$\sum_{j=1}^n \exp((s_j - s^*)/\tau) \to |M|$$

and

$$\pi_i^{(\tau)} \to \begin{cases} 1/|M| & i \in M \\ 0 & i \notin M \end{cases}$$

Substituting this into the basis compressed output equation gives

$$\lim_{\tau \to 0} z^{(\tau)} = \frac{1}{|M|} \sum_{i \in M} v_i \tag{4.16}$$

For the special (but common) case where the maximizer is unique, i.e. $M = i^*$, let the unique-max gap be $\Delta = s_{i^*} - \max_{i \neq i^*} s_i > 0$. For any $i \neq i^*$,

$$s_i - s_{i^*} \le -\Delta \tag{4.17}$$

The non-maximal total attention mass satisfies

$$\sum_{i \neq i^*} \pi_i^{(\tau)} = \frac{\sum_{i \neq i^*} \exp((s_i - s_{i^*})/\tau)}{1 + \sum_{j \neq i^*} \exp((s_j - s_{i^*})/\tau)} \tag{4.18}$$

Let A to be the numerator of Equation 4.18. Since we know that $A \ge 0$,

$$\frac{A}{1 + A} \le A$$

and applying Equation 4.17,

$$\begin{aligned}\sum_{i \neq i^*} \pi_i^{(\tau)} &= \frac{A}{1+A} \\ &\leq A \\ &\leq \sum_{i \neq i^*} \exp(-\Delta/\tau) \\ &= (n-1)\exp(-\Delta/\tau)\end{aligned}$$

Now, consider $z^{(\tau)} - v_{i^*}$. As attention weights sum to 1,

$$\begin{aligned}z^{(\tau)} - v_{i^*} &= \left(\sum_{i=1}^{n} \pi_i^{(\tau)} v_i\right) - v_{i^*} \\ &= \sum_{i=1}^{n} \pi_i^{(\tau)} (v_i - v_{i^*})\end{aligned}$$

If $\|v_i\| \leq R$ for all i then $\|v_i - v_{i^*}\| \leq 2R$. Therefore,

$$\begin{aligned}\|z^{(\tau)} - v_{i^*}\| &\leq \sum_{i \neq i^*} \pi_i^{(\tau)} \|v_i - v_{i^*}\| \\ &\leq 2R \sum_{i \neq i^*} \pi_i^{(\tau)} \\ &\leq 2R(n-1)\exp(-\Delta/\tau) \qquad (4.19)\end{aligned}$$

□

Latent mixture This module merges information between compressed column names c_j and their respective entries v_j using cross attention. A number, category and even text in a row on its own can be highly ambiguous. On the other hand, the interpretation of "15" under the "age" column could be "young". This is an intra-column operation, i.e. the cross attention is done between the column name and entry value of a single column, for all columns in parallel. Information is processed in accordance with the intended tabular structure, satisfying D5. Furthermore, the output is a set of encoded (c_j, v_j) pairs, following the formulation in Section 4.4.2.

Latent compression This module compresses a sequence of vectors into a single vector by concatenating the vectors and then projecting them to a lower-dimensional subspace, so that the next module can operate on a high-level representational space.

Cross attention is an order equivariant operation with respect to the query and an order invariant operation with respect to the key-value (i.e. input sequence). The order of basis queries is fixed by construction, so the output of basis compression is invariant to the order of the input sequence. Latent compression exploits this property by concatenating the output vectors of basis compression in sequence, then projecting them to a lower-dimensional subspace.

Latent contextualization This module processes inter-column relationships to extract high-level details needed for the task using self-attention layers. The number of self-attention layers is a hyperparameter dependent on the complexity of the task, similar to depth in decision trees. This is the most canonical process in a tabular predictive model to transform the inputs into a prediction.

Latent decompression This module produces two new sets of basis queries of dimension $\mathbb{R}^{C\times L\times D}$ for the next block, enabling the chaining of multiple blocks sequentially – one as the basis queries for the column name inputs, and another as the basis queries for the entries. A table can have many columns, so the model has to "look" at the input row multiple times, each time viewing from a different perspective. The final $\mathcal{BT}$ block will only contain one latent decompression module, yielding a sequence of size $\mathbb{R}^{C\times L\times D}$ where mean aggregation will be performed over columns to get a tensor of size $\mathbb{R}^{L\times D}$. Finally, it is flattened and downscaled to a vector of size D representing the prediction.

The multiple "looks" design is inspired by the U-Net [126] in the latent diffusion model [125]. To guide the diffusion process in aligning with the input textual prompt, it is believed that it is insufficient to pass in the textual prompt once at the input of the U-Net, even with residual connections. In addition to passing the textual prompts at the beginning, multiple cross-attention layers are included throughout the U-Net so that the same textual prompts, which are unchanged, can be attended to.

In $\mathcal{BT}$, we believe it is especially important to allow multiple "looks" at the input because basis compression is used which may result in a significant loss of information at every $\mathcal{BT}$ block. The intuition is: after the first block gets a partial look at the input, it produces basis queries for the next block to focus on certain elements of the table that might have been more useful. By chaining multiple $\mathcal{BT}$ blocks, where each block gets a

slice of information (if there are textual entries), it forms a more refined representation of the current row of data by building upon the previous belief from the previous blocks. Even if there are no textual entries, which implies every $\mathcal{BT}$ block receives full information on the numeric entries, this chaining process acts as an ensemble of predictors.

4.4.4 Module Implementation

This section outlines the implementation details of the architecture. An *independent dimension* is one where there is no information exchange between elements across that dimension, and operations are done in parallel. For example, batch size is an independent dimension.

Basis compression This module is a pre-layer normalization transformer layer specified by Xiong et al. [166] without the self-attention layers, and using basis queries $q_x \in \mathbb{R}^{C\times L\times D}$ in the cross attention layer with residuals from the basis queries. In the first $\mathcal{BT}$ block, the shared learnable basis queries $q_x \in \mathbb{R}^{L\times D}$ are duplicated (broadcasted) in the first dimension from 1 to C. For input sequence (column name or entries) $X \in \mathbb{R}^{C\times L_x\times D}$,

$$z_x = \texttt{BasisComp}(q_x, X) \in \mathbb{R}^{C\times L\times D} \tag{4.20}$$

The cross attention is done over the L dimension. C is an independent dimension.

Latent mixture This module consists of cross attention (over the L dimension), and layer normalizations. In the cross attention layer, the compressed column embedding is the query while the compressed value embedding is the key-value of each column. There is a residual connection passing from the query to the output. C is an independent dimension. For some compressed column embedding $z_{\text{col}} \in \mathbb{R}^{C\times L\times D}$ and compressed value embedding $z_{\text{val}} \in \mathbb{R}^{C\times L\times D}$,

$$z = \texttt{LMix}(z_{\text{col}}, z_{\text{val}}) \in \mathbb{R}^{C\times L\times D} \tag{4.21}$$

Latent compression This module involves reshaping ($\mathbb{R}^{C\times L\times D}$ to $\mathbb{R}^{C\times LD}$), followed by layer normalization and a linear projection ($\mathbb{R}^{C\times LD}$ to $\mathbb{R}^{C\times rD}$) where $r \in \mathbb{N}$ is a hyperparameter. C is an independent dimension.

$$\bar{z} = \texttt{LComp}(z) + \bar{z}_{\text{prev}} \in \mathbb{R}^{C\times rD} \tag{4.22}$$

In practice, r is smaller than L to focus on important parts of the sequence. The output of latent compression is considered a high-level bottlenecked representation of the input as each column is represented with a single high-dimensional vector. If this is not the first $\mathcal{BT}$ block, it receives a residual connection from the output of the latent contextualization module from the previous block.

Latent contextualization To promote the learning of relationships between columns, multiple self-attention blocks (a self-attention layer over the C dimension, a shallow MLP, and layer normalizations) are stacked sequentially. This module does not change dimensions.

$$\bar{z}_{\text{ctx}} = \texttt{LCont}(\bar{z}) \in \mathbb{R}^{C \times rD} \tag{4.23}$$

Latent decompression The purpose of this module is to generate two new sets of basis queries for the next block. The intermediate latent tensor is first upscaled (from $\mathbb{R}^{C \times rD}$ to $\mathbb{R}^{C \times LD}$) with a linear projection layer then reshaped to $\mathbb{R}^{C \times L \times D}$. Finally, to specialize the tensor into column name and entry entities, it is passed through two identical but distinct sets of MLP and layer normalization.

$$q_{\text{col}}, q_{\text{val}} = \texttt{LDecomp}(\bar{z}'_{\text{ctx}}) \in \mathbb{R}^{2 \times C \times L \times D} \tag{4.24}$$

Figure 4.3 provides a visual aid to understand the $\mathcal{BT}$ architecture.

4.4.5 Numeric Considerations

Working with SMR SMR is used for inputs and outputs. The output being a vector of binary values transforms the regression problem into a multi-label classification problem, hence we use the sum of binary cross entropy losses across all the $1 + h + \ell$ dimensions of the prediction. Therefore, we do not need any form of normalization or scaling, satisfying D6 as well as potentially unlocking zero-shot capabilities.

An implication of this is that each resolution is weighted equally in the loss function, resulting in a stable training process whereas mean squared error loss could grow intractably large and is sensitive to different scales in values. Targets likely differ in ranges of values in multi-task settings. For example, datasets with targets in $[10^0, 10^1]$ would be systematically underfitted compared to datasets with targets in $[10^3, 10^6]$ when using conventional

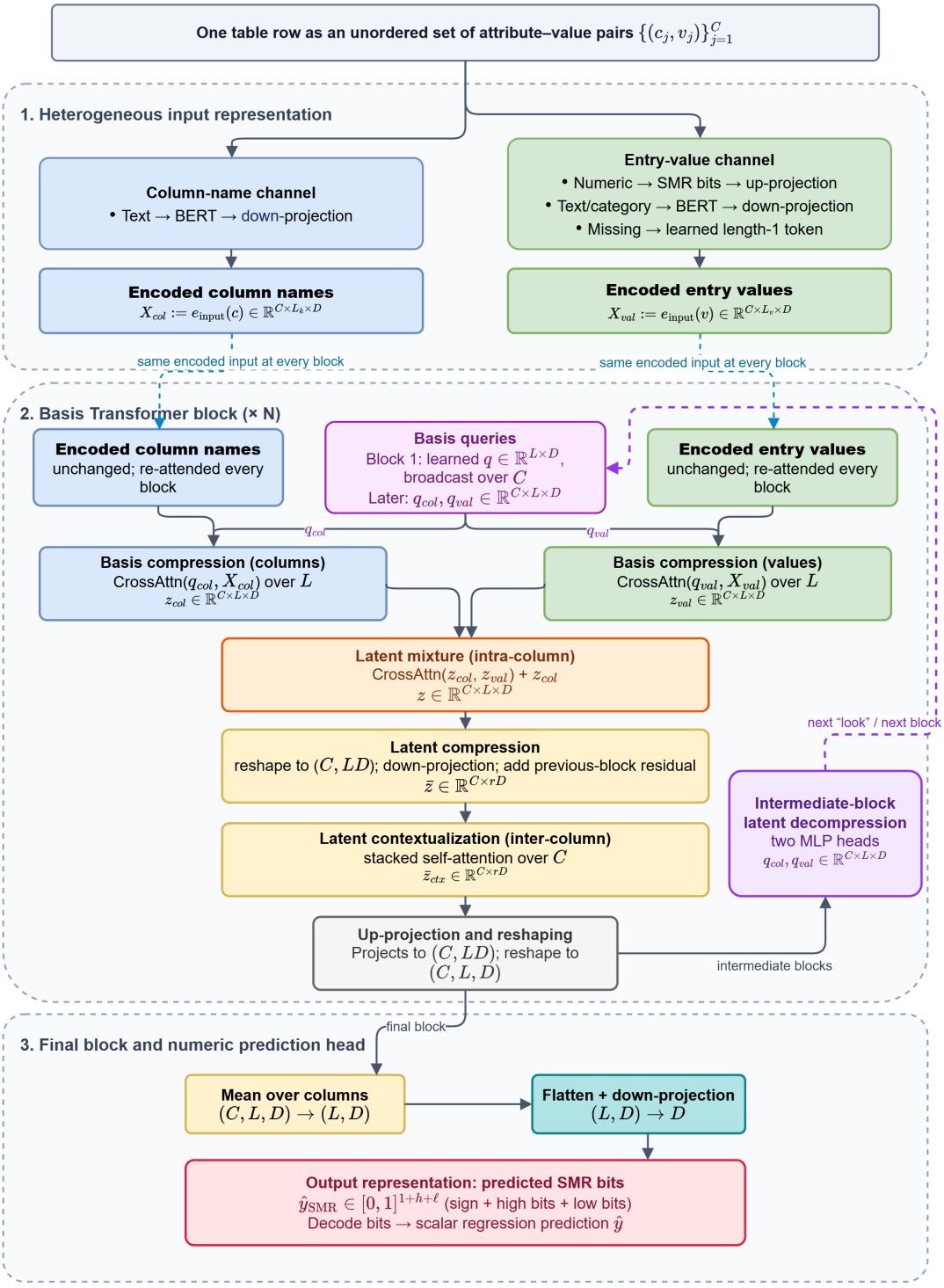


Figure 4.3: A detailed flow chart showing how a basis transformer works.

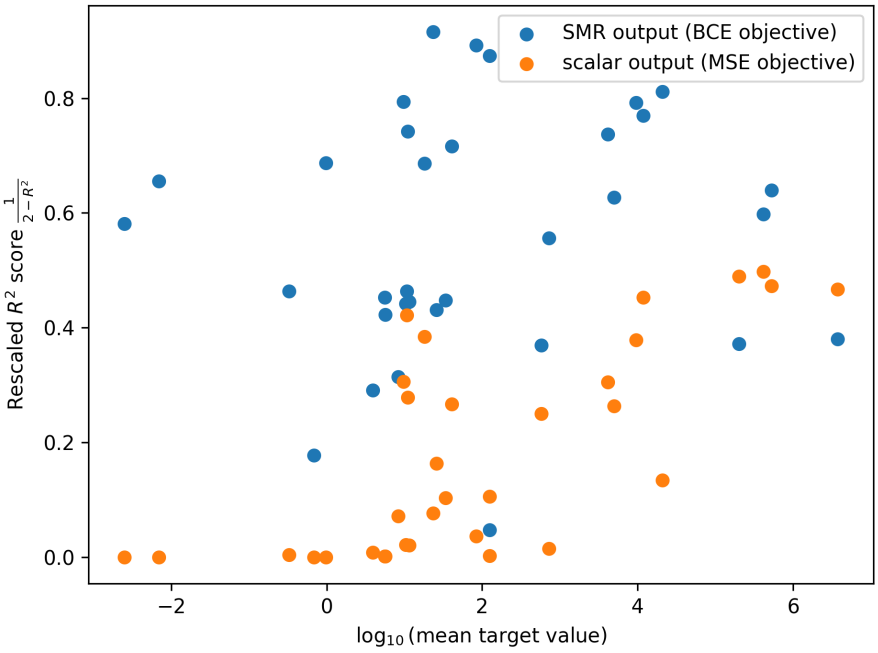


Figure 4.4: Rescaled test R^2 score against log mean target value. Each point represents a dataset, averaged over 5 random seeds. Higher y-values are better.

regression losses. To see this phenomenon, an ablation study using MSE and BCE losses is conducted below.

We trained two identical models, except that one of the models has an output head replaced so that it produces a single scalar without any activation functions, i.e. the standard regression head. The exact hyperparameters can be found in Appendix B.1. This model is subjected to the same datasets and training routine, besides using mean squared error instead of binary cross entropy. The results for both are presented in Figure 4.4.

We use the R^2 score because of the regularity of the bounds. The range of R^2 is $(-\infty, 1]$, making it difficult to observe any structural patterns. Instead, we rescale it to $(0, 1]$ using the $\frac{1}{2-R^2}$ transformation. This is the Normalized Nash-Sutcliffe Efficiency (NNSE) metric [104], closely related to R^2. The mean target values of each dataset vary significantly, so we use a log transformation on the x-axis.

Besides the SMR variant being better in raw performance, we can observe a clear relationship between the average target values and the test performance of the scalar variant when trained using mean squared error. Once the mean target value passes 10^5, the scalar variant has competitive performance comparable to the SMR variant but before that, it consistently underperforms.

There are works that convert a regression problem into a binned regression objective (a special case of multi-class classification) by predicting the interval containing the target [86] but it is unable to produce a point-wise prediction.

Learning Augmentation In a multi-task regime where some tasks are inherently more challenging than others, we designed an *adaptive loss reweighing scheme* to focus on hard examples for regression tasks during mini-batch training. For prediction $\hat{y} \in \mathbb{R}$ (decoded from SMR) and target $y \in \mathbb{R}$, we compute a bounded heuristic $g : \mathbb{R} \times \mathbb{R} \to (0, 1]$ on how well the magnitude of prediction is

$$g(y, \hat{y}) = \frac{\min\{|\hat{y}|, |y|\} + \varepsilon}{\max\{|\hat{y}|, |y|\} + \varepsilon} \text{ for a small } \varepsilon > 0 \tag{4.25}$$

A hard example would have $g(y, \hat{y})$ close to 0. The new loss $\tilde{L}(y, \hat{y})$ over a mini-batch of n samples based on the original loss function $L(y, \hat{y})$ is

$$\tilde{L}(y_i, \hat{y}_i) = [(1 - g(y_i, \hat{y}_i)) \cdot (1 - 2\gamma) + \gamma] \, L(y_i, \hat{y}_i) \tag{4.26}$$

where $\gamma \in [0, 0.5]$ controls the aggressiveness of loss reweighing by rescaling $1 - g(y, \hat{y})$ to a range of $[\gamma, 1 - \gamma)$. Choosing $\gamma = 0.5$ disables any adjustments, while $\gamma = 0$ enables the strongest adjustment. While this scheme ignores the sign of a prediction, in practice, most models can predict the signs well especially $\mathcal{BT}$ which has a dedicated bit for sign prediction.

There are several regression metrics that could theoretically be used to reweigh the loss such as R^2, normalized Nash-Sutcliffe Efficiency (NNSE) [61] and Robinson's Agreement Coefficient (RAC) [124]. These regression metrics are rescaled to a bounded interval using a measure of variance. Using variance to standardize the metric is not suitable for a mini-batch process, especially in a multi-task learning regime. The variance of targets wildly fluctuates across tasks and across mini-batches, leading to inaccurate signals during backpropagation.

Our approach measures performance as a ratio of prediction to target or a ratio of target to prediction, whichever is smaller. This ensures that the measure is bounded to $[0, 1]$, and does not have a dependency on statistics of other elements in the mini-batch.

The result of an ablation study of various γ is shown in Figure 4.5. The exact hyper-parameters can be found in Appendix B.2. While $\gamma = 0.45$ performs the best, in the full experiment, we used a stronger adjustment factor of $\gamma = 0.2$ because there are 34 tasks while there are only three tasks here.

4.5 Empirical Results

4.5.1 Experimental Details

The experiments use the OpenML-CTR23 benchmark [45], consisting of 34 regression datasets. The entry values consist of numbers, textual categorical values, and free-form text. No standard preprocessing is performed, such as one-hot encoding, standard scaling of mean and standard deviation, and dropping rows with missing values (NaNs). Each dataset is split into training, validation, and test sets. 5 random seeds are used for each experiment. For each seed, the checkpoint evaluated on the test set is selected based on the best mean validation score.

The primary evaluation metric is the root mean squared error (RMSE) of min-max scaled (MSS) targets and predictions, which we call MMS-RMSE. Min-max scaling refers to: identifying the minimum and maximum value of targets in a particular test dataset, $y_{\min}$

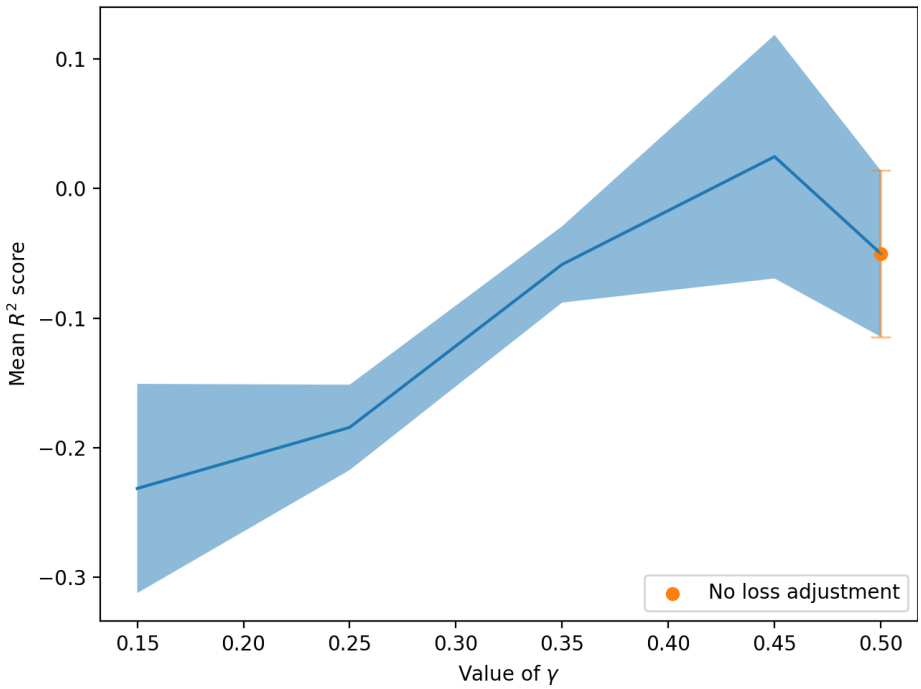


Figure 4.5: Test performance with and without adaptive loss reweighing scheme over different degrees of reweighing factor γ. The shaded area represents the region of 1 standard error over 5 random seeds. Higher y-values are better.

and $y_{\max}$, then scaling both the predictions $\hat{y}$ and targets y with the linear transformation

$$\hat{y} \leftarrow \frac{\hat{y} - y_{\min}}{y_{\max} - y_{\min}} \qquad y \leftarrow \frac{y - y_{\min}}{y_{\max} - y_{\min}} \in [0, 1] \tag{4.27}$$

After min-max scaling, RMSE is computed. This is necessary since each dataset has a different range of targets, and we want to normalize them so that the aggregation of scores is not biased towards datasets with large target values. We also include the raw RMSE.

4.5.2 Multi-Task Experiments

One model is trained on the entire OpenML dataset simultaneously. After every 200 gradient steps, the model is validated on each dataset. Finally, the best-performing checkpoint is tested on each dataset.

Baselines One large group of baselines used for comparison with $\mathcal{BT}$ are large language models as prescribed by TabLLM [65]. We use the best-performing serialization technique they achieved, which is the template `The <COLUMN_NAME> is <VALUE>.`. The LLMs are finetuned on rows that are serialized with the template. For a fair comparison, we limit the number of parameters of baselines to be similar to the size of $\mathcal{BT}$. There are two classes of LLMs that we use as baselines: sequence-to-sequence language models (Flan-T5 [34], LaMini-T5 [164], BART-large [91]), and causal language models (Pythia [15], Cerebras-GPT [38]).

We also considered comparisons to Qwen, TinyLlama, TABULA-8B and CARTE, but we had to exclude them due to data contamination issues (see Appendix B.4). We also had to exclude several algorithms because they either could not handle free-form text, could not deal with heterogeneous rows, required specialized information or were not reproducible (see Appendix B.4).

The weights of $\mathcal{BT}$ were randomly initialized while the LLMs were pretrained on their respective pretraining tasks, and the checkpoints were obtained from Huggingface [162]. Hyperparameters and configurations are described in Appendix B.3.

Modified Tabular Baselines Given the circumstances, specialized large language models for tabular tasks are the only truly comparable baselines for our work. However, if one permits modifications to popular baselines to produce predictions, despite violating their assumptions and conditions, we can get a more holistic picture.

To accommodate multi-task learning where one model handles multiple tables, we pad every feature vector to the same number of columns with `NaN` (or 0 for TabPFN). For XGBoost and CatBoost, we conducted two sets of experiments: numeric-only (every feature vector has 82 elements from only numeric columns) and numeric-text (every feature vector has 190 elements). For TabPFN, it was excessively slow with numeric-text vectors and did not finish after several days of execution, so only numeric-only vectors were used. Furthermore, TabPFN has a hard limit of 10,000 training samples so we had no choice but to subsample to abide by this limit.

Some recent works on deep tabular models include FT-Transformer [53], SAINT [135], TabTransformer [73] and TabM [54]. Similar to classic models, they require column padding to function. However, in addition, these deep approaches learn a tokenized embedding of each non-numeric feature. We modified them so that they stores a learnable set of embedding over the union of all non-numeric features. The embedding vectors are learned during the training process.

There are some additional details:

1. XGBoost and CatBoost results are obtained with grid search over 27 hyperparameter permutations (max_depth, reg_lambda, eta) and the best based on the validation set is used.
2. Max tree depth is limited to 16 due to API limits.
3. TabPFN is not fine-tuned because it works as an ICL model and our evaluation does not have in-context learning based experiments.

Results Table 4.3 reports central tendency and spread of MMS-RMSE and RMSE. $\mathcal{BT}$ outperforms in central tendency. To view the complete breakdown of results by datasets and runtimes, they are shown in Appendix B.5.

The LLMs do not always abide by the serialization template. During evaluation, the generated outputs by the LLMs are parsed using a lenient regular expression to extract the numeric values, ignoring: excessive white spaces, a lack of an EOS token, or a missing period. The proportion of predictions that can be parsed is called *success rate*, which is reported in Table 4.4. Sequence-to-sequence models do better than causal models.

Another important point is the stability of training, shown in Figure 4.6. After every 200 gradient steps, each model is evaluated once on all validation datasets. There are

Table 4.3: Number of parameters, and median and inter-quartile range (IQR) of test MMS-RMSE and RMSE across 34 datasets and 5 random seeds for each model. For the data column: (I) refers to numeric + text (dense), (II) refers to numeric only, (III) refers to numeric + one-hot text (sparse).

Model	Data	Parameters (M)	MMS-RMSE (↓)	RMSE (↓)
$\mathcal{BT}$ (ours)	(I)	251	**0.195** ± 0.128	**18.125** ± 1630.480
Pythia410m	(I)	410	0.475 ± 0.663	37.410 ± 4116.078
Cerebras590m	(I)	590	0.295 ± 0.265	20.505 ± 2321.217
FlanT5-base	(I)	248	0.260 ± 0.200	28.360 ± 1507.803
LaMiniT5-223M	(I)	223	0.245 ± 0.145	24.290 ± 1840.678
BART-large	(I)	406	0.430 ± 0.427	36.200 ± 5796.245
XGBoost	(II)	-	0.300 ± 1.740	34.855 ± 1158.780
XGBoost	(III)	-	0.320 ± 2.430	43.915 ± 263.760
CatBoost	(II)	-	25.000 ± 83.948	1412.635 ± 4258.080
CatBoost	(III)	-	15.715 ± 72.480	1345.585 ± 1852.318
TabPFN	(II)	-	6.455 ± 125.797	623.470 ± 1332.810
FT-Transformer	(III)	0.2	0.565 ± 46.295	1015.850 ± 3587.760
SAINT	(III)	0.4	2.400 ± 125.565	2802.950 ± 3395.168
TabTransformer	(III)	1.6	11.780 ± 135.683	1174.785 ± 2745.360
TabM	(III)	2.8	307.385 ± 2152.908	24345.230 ± 38230.565

Table 4.4: Mean and standard deviation of success rate of language-capable models over 5 random seeds across 34 datasets.

	$\mathcal{BT}$	Pythia410m	Cerebras590m	FlanT5-base	LaMiniT5-223M	BART-large
Mean	**1.00**	0.95	0.92	**1.00**	**1.00**	0.98
Std. dev.	0.00	0.12	0.21	0.00	0.00	0.06

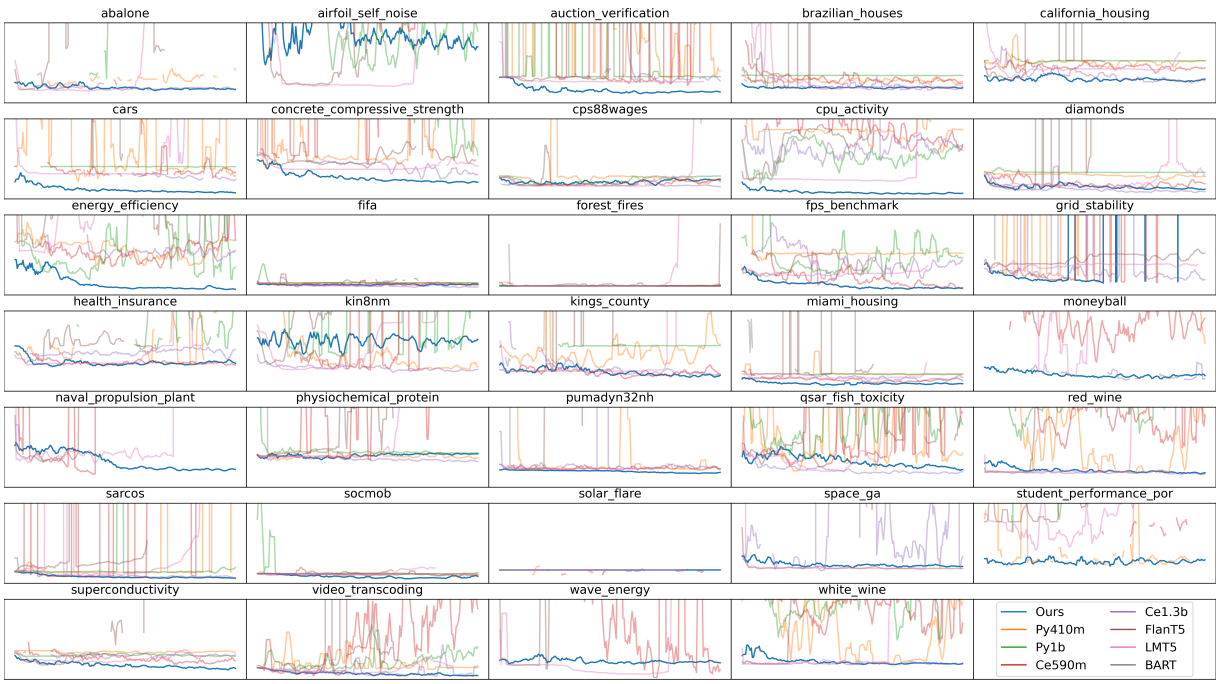


Figure 4.6: Validation MMS-RMSE scores over gradient steps per dataset. The range of y-axis is [0, 1] and the range of x-axis is [0, 200]. The blue line is $\mathcal{BT}$. Lower values are better.

Table 4.5: Median and inter-quartile range (IQR) of test MMS-RMSE and RMSE across 18 related datasets and 5 random seeds for each model.

Model	MMS-RMSE (↓)	RMSE (↓)
$\mathcal{BT}$ (ours)	**0.210** ± 0.020	**12.630** ± 0.488
Cerebras590m	0.295 ± 0.143	18.230 ± 9.183
FlanT5-base	0.235 ± 0.038	13.850 ± 0.632
LaMiniT5-223M	0.250 ± 0.033	14.665 ± 1.173
XGBoost	0.230 ± 0.030	14.555 ± 2.073
CatBoost	0.220 ± 0.045	17.791 ± 8.921
FT-Transformer	0.645 ± 0.088	38.350 ± 0.888
SAINT	0.615 ± 0.080	36.200 ± 1.275
TabTransformer	0.670 ± 0.090	39.425 ± 0.995
TabM	0.685 ± 0.100	40.405 ± 0.980

40,000 gradient steps for each model. Some lines from the baselines are discontinuous because they fail to conform to the prescribed template at some points, hence they cannot be evaluated. $\mathcal{BT}$ is consistently stable when evaluated on unseen samples. Despite being trained from scratch, it acquired a decent understanding of the tasks within 200 gradient steps (i.e. the left-most blue point in every subplot). Figure B.1 in Appendix B.5 contains an enlarged landscape view and information on variability.

4.5.3 Related Task Evaluation

To demonstrate the importance of multi-task learning in a tabular setting, we want to show that approaches that can natively handle multiple tables at once benefit from cross-table transfer learning, compared to single-table approaches. We identified 18 tables from the T4 dataset [49] that are closely related (socioeconomic and demographic data). Full table extraction details are outlined in Appendix B.6. $\mathcal{BT}$ and the better-performing LLMs from Table 4.3 are trained on the training splits of all 18 tables, then evaluated on the 18 test splits simultaneously. The other models, being unable to handle multiple tables, are trained separately on each of the 18 tables. Each of the 18 trained models was evaluated on their corresponding test splits. The results are shown in Table 4.5.

Among the baselines, CatBoost achieves the strongest performance. $\mathcal{BT}$ likely outperformed CatBoost because $\mathcal{BT}$ could leverage learned information from the other 17 tables

that are related. CatBoost, despite being the superior algorithm in a single table, is unable to scale and leverage information from related tables. Baseline deep tabular approaches are among the worst-performing ones as they require more training data than classic tree-based models and they cannot handle cross-table training.

4.6 Discussion

4.6.1 Understanding Basis Compression

Basis compression is done in linear time with respect to the input sequence length. It homogenizes columns into fixed-sized sequences, making it easier to handle. Upon gradient-based optimization, the set of basis queries q becomes oriented towards the most important parts of the embedding space of all input sequences. The choice of q influences the degree of attention towards certain keywords, values, or a mixture of both. This emulates a "soft" split in decision trees. For example, when predicting house prices from free-form text, q could contain vectors such as "beautiful" and "pests". This ensures that words of higher mutual information with the targets get a higher mixture proportion in the result.

To demonstrate this point empirically, we loaded the trained weights of BT (without any additional finetuning or adjustments). To make a clear illustration that BT can capture important features, we chose the column names from the `auction_verification` dataset for two reasons. There is some redundancy in column names here (e.g., "process", "property") so that we can show the basis queries ignore tokens that are less informative. Secondly, there are a small but sufficient columns to be used as a concrete example.

The BERT tokens of each column name are listed in each cell in Figure 4.7. For example, the first row represents the first column name `process.b1.capacity` (including special tokens) broken down by the BERT tokenizer. The colour (one-to-one corresponding to attention weight values) is averaged over the 64 basis queries. For example, the highest value in the heatmap (brightest yellow cell) is 0.282, which corresponds to the average attention weight between each basis query and `b` token in `process.b2.capacity`.

From Figure 4.7, we can see that the basis queries have been trained to give a higher attention score to important tokens that differentiate themselves from other columns. For example, basis queries tend to give lower weights to the symbol "`.`" but higher weights to `price` and `product` since they are unique.

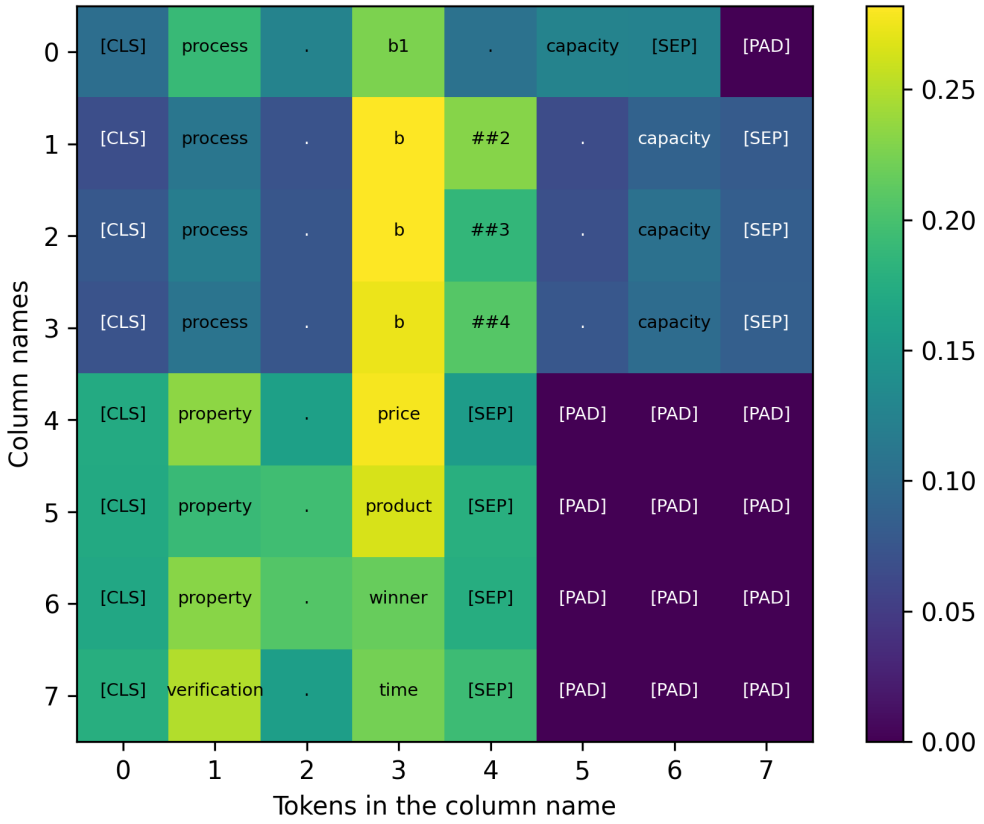


Figure 4.7: Heatmap of average attention weights over 64 basis queries and each token in BERT-encoded column names.

4.6.2 Memory consumption

A limitation of $\mathcal{BT}$ is the memory consumption, attributed to its 4D input tensor representation. As transformers require padding for parallelism during mini-batch training, a higher-dimensional tensor leads to greater memory inefficiency. This is especially the case where the variance in sequence length L is high as $\mathcal{BT}$ handles numeric values ($L = 1$), and textual values ($L \gg 1$).

4.6.3 Standardization and one-hot encoding

Methods that require standardization and one-hot encoding of raw text entries pose a fundamental barrier to true generalization. Such methods make neural network optimization significantly easier due to improved stability and simplicity. However, standardization restricts transfer learning and effectively sets a limit to zero-shot learning. One-hot encoding makes strong assumptions about a specific task that will likely not be transferable to other tasks. A core contributions of this work is the development of mechanisms that circumvent the need for standardization and one-hot encoding to remove the ceiling on neural networks.

4.6.4 Connections to Bayesian multi-task learning

Multi-task tabular regression is challenging because a model has to: (1) infer the task from a single row, (2) digest information from the row given the task, and (3) calibrate the prediction onto an unbounded range of values given input information and task. The Bayesian perspective of multi-task learning with a similar process has been explored [17, 161, 140] primarily on Gaussian processes, but the architecture of $\mathcal{BT}$ abides by some key premises. It has been noted that not all tasks are equally similar in practice [9, 35]; there are inherent structures within certain groups of tasks. Basis queries align with this where a task is implicitly defined by the set of column names and entry values, whereupon decomposed into a sequence that can be expressed in terms of common identifiers potentially across all tasks.

4.7 Conclusion

This chapter covers a multimodal design for tabular data specialized towards satisfying the six desiderata in Section 4.2. The architecture deviates from the standard approaches typ-

ically used for regression, such as standardization, scalar output head and mean-squared error objectives. They are simultaneously replaced with the sign-magnitude numeric representation, negating the need for standardization and improving neural network optimization stability. Furthermore, another key idea is to leverage semantic information from text that is frequently ignored by many modelling approaches. Because text and high-dimensional sequences are difficult to handle directory, the proposed architecture uses basis compression to make their use practical for regression tasks.

Chapter 5

A Multimodal Tabular Foundation Model

The term "foundation model" was coined by the Center for Research on Foundation Models, a part of the Stanford Institute for Human-Centered Artificial Intelligence [21]. While the term itself is new, the concept is a natural progression of the field of deep learning and its ecosystem. Deep learning is inherently data hungry [118] and deep neural networks typically scale well relative to non-neural approaches, driving the demand for more high-quality data. Data collection has become an integral part of the ecosystem, which enabled and fueled further research.

Foundation models are usually discussed in the context of natural language processing [21] or computer vision. More recently, there have been attempts to develop foundation models for:

- Graph neural networks [95],
- Protein folding [18],
- Time series [93], and
- Tabular learning [70]

The purpose of a foundation model is to learn from massive amounts of raw data, usually on a surrogate task (called *pretraining*). Starting from a pretrained model, the model undergoes further parameter adjustment using data specific to the final task of

interest (called *downstream task*). The pretraining process allows the model to determine the optimal representation of high-level features [21]. If done properly, either the model would perform better via transfer learning[1], or will require less task-specific data, or both.

Previously, Chapter 4 presented a multi-task tabular framework for regression tasks. A multi-task regime can be seen as an intermediary between a standard training routine and a foundation model. This chapter will take the basis transformer to new heights as a multimodal tabular foundation model via a specialized pretraining process and improved basis transformer architecture. The goal is to produce a foundation model capable of capturing semantic information from a row of a table that would be useful for downstream tasks.

The contributions of this chapter are:

- We propose modifications to the basis transformer architecture to be better suited as a foundation model.
- We develop a scalable preprocessing and caching pipeline for the T4 dataset.
- We introduce a target selection strategy for self-supervised learning with distance correlation heuristic.
- We introduce a sparse and smooth variant of the binary cross entropy loss for numeric targets.
- We combine learned representations with dimensionality reduction and an XGBoost regression head.
- We demonstrate strong performance on zero-shot imputation and multimodal regression benchmarks.

5.1 Motivation

One of the main lessons from Chapter 3 is to leverage additional useful signals in solving the cold-start problem. Now, every task has its own dataset that may differ in structure and organization. Fortunately, Chapter 4 offers a unique architecture and learning routine that has the flexibility to handle any tabular data. What it needs is to be scaled up and pretrained on datasets with a suitable objective. This motivates the present chapter.

[1] Leveraging knowledge learned from one task that can be applied to another related task. [21]

The motivation of this chapter is to produce a powerful representation of multimodal tabular data. Multimodal in this context refers to the ability to handle multiple data types such as integer, real numbers, text, etc. The representation must be robust and precise so that downstream tasks can easily extract useful signals from heterogeneous tables. Furthermore, it should produce a hierarchical representation simultaneously, as opposed to a specialized model that produces a row representation and another producing a column representation.

5.2 Related Works

Tabular foundation models are increasingly established in the literature. Arguably, the most notable tabular foundation model is TabPFN [69]. The latest generation of TabPFN is TabPFN-2.5 at the time of writing [57]. All generations of TabPFN are trained purely on synthetic data [57] and are therefore instances of prior fitted networks [109]. The novelty of TabPFN comes from the synthetic data generation process, based on structured causal models. The idea is to inject prior distributions that are sufficiently related to real tabular data, and the authors deemed structured causal models to be empirically effective. TabPFN, as an in-context learning (ICL) model, uses a transformer to condition on in-context examples to make a prediction on an unseen sample without finetuning.

TabDPT[2] is another foundation model that operates using in-context examples [102]. However, there are many differences. The training process is driven by self-supervised learning on real datasets. Self-supervision is not based on random sampling but rather on a retrieval-based system that samples rows that are similar in feature space [102]. As for the model architecture, both features and labels are encoded into the same embedding space. The encoded features and labels of the context are summed together, and are then concatenated with the encoded features of the query. The result is passed into a transformer, producing both regression and classification outputs.

Mitra focuses on its synthetic data generation process as the key to a good tabular foundation model [173]. They proposed three properties of a good prior: (1) performance on real data, (2) diversity to avoid overfitting on itself, and (3) distinctiveness to avoid overlaps with other priors. The ICL-based foundation model uses a similar approach to TabPFN whereby self-attention is done across rows and columns. Experimentally, they show that having a better mix of synthetic priors led to Mitra being more sample efficient in terms of the number of in-context examples [173]. Another outcome in Mitra is superior

[2]DPT stands for discriminative pretrained transformer.

finetuning performance. For example, the authors claimed TabPFNv2 does not show significant improvements when finetuned or ensembled, while Mitra shows strong gains from finetuning.

CARTE, which stands for "context aware representation of table entries", views tabular entities from a graphlet perspective [79]. The abstract representation of a row in a table is the center node in a star-like graph. The edges represent the column names, while the neighbouring nodes represent the corresponding cell values (entries). Textual entities are initialized with a language model, while the nodes of numeric values are the product of their value with the embedding of the corresponding column name. The foundation model itself is a graph attention network that modulates the importance of neighbours. Unlike usual tabular foundation models, CARTE is pretrained on a large knowledge base that contains factual information of various entities with a contrastive objective. To solve downstream tasks, CARTE applies power transform [157] on numeric entries and finetunes a small part of the pretrained network.

TabSTAR is a tabular foundation model designed for tabular datasets that include text fields [4]. By having a trainable text encoder and special target-aware tokens, the model can learn joint representations of the numeric and textual entries in unison without dataset-specific output heads. The architecture involves mapping every feature into a textual representation (column name and values), then embedding them into the same space. Numeric entries are fused with their corresponding column name embedding vectors. All features go through self-attention to learn cross-feature interactions and finally the embedding of the target columns is mapped to a real value or a probability distribution [4]. TabSTAR is pretrained on 350 datasets using a standard supervised learning objective.

The related works from Chapter 4 are also useful in this chapter.

5.3 Data Processing

5.3.1 Pretraining Data Source

Tabular models constrained to numeric and categorical types enjoy the benefits of using synthetic data generation. For models that attempt to incorporate textual semantics directly into the prediction process, there have been no successes in using synthetic data generation with text to the best of our knowledge.

The TabLib dataset contains 627 million tables collected from the Internet, primarily from Common Crawl and GitHub [41]. Despite its large volume of tabular data, it can be

Table 5.1: Additional internal preprocessing rules for T4 datasets.

No.	Action	Justification
1	Drop columns with bytes data type	Ambiguous data type that cannot be handled
2	Drop rows where all entries are `NaN`	Not useful for any downstream task
3	Drop tables without any numeric entries	Regression requires at least one numeric column
4	Convert Python `datetime` objects into ISO formatted string	Python objects are not universal representations
5	Drop tables with extremely long textual entries	Such tables would significantly slow down pretraining

difficult to work with due to the heterogeneity of the data format and the dataset exhibits quality limitations. As a result, Gardner et al. [49] created the T4 dataset, containing 4 million tables, which is a filtered variant of TabLib. Approximately 97.91% of the tables were removed based on heuristics and summary statistics. The T4 dataset consists of over 2.1 billion rows and occupies 2 TB of storage when compressed.

There are 18 exclusion rules in T4 [49]. The notable ones for tables include: non-English tables, every entry is of the same type, and parsing errors. Exclusion rules for rows or columns include: long column names, numeric column names, and constant features.

5.3.2 Internal Preprocessing

In this chapter, we use the processed tables from T4, and not the rest of the processing such as target column selection or encoding techniques. Despite the extensive preprocessing by Gardner et al., there were a few considerations that remained unaddressed. We summarize our preprocessing algorithm in Table 5.1.

Given our computational resource constraints, it is impossible to store and pretrain on 2.1 billion rows of heterogeneous tabular data. The two options are: reduce the number of tables, or subsample rows from each table. We chose the latter, as it is more important to provide coverage of different tasks and table structures for a pretraining step. Furthermore, we anticipate diminishing returns from increasing the number of rows in a single table. The maximum number of rows in any table is 100. Any table in T4 exceeding that threshold is subsampled to 100 rows randomly.

To avoid having to call a language model during pretraining, all textual entries are encoded with BERT beforehand during preprocessing. Since we do not need a powerful textual representation, we opt for a distilled version of BERT by Turc et al. [144] with a hidden dimension of 128 and 10 self-attention blocks. While there are more modern variants of transformer encoder models for natural language understanding tasks, we could not find any at the start of the work that were sufficiently small and lightweight. Any missing values in textual entries were explicitly replaced with the BERT embedding of the string `[NaN]`.

Candidate indicators are Boolean variables for each column in every table, indicating whether the corresponding column can be selected as a regression target. A candidate column must fulfill both criteria: (A) the column must contain numeric entries, (B) the numeric entries must be floating point values, or integers, where the number of unique values must be less than 5%. The second expression in (B) is necessary to avoid making an ID column a target, as IDs are inherently arbitrary.

5.3.3 Distance Correlation

Distance correlation was introduced by Székely et al. [141] as a new measure of dependence between random vectors with properties that are relevant in our pretraining process. We provide a brief overview of distance correlation below.

Let $\mathcal{R}(X, Y)$ be the distance correlation between random vectors X and Y. Two important properties are:

1. $\mathcal{R}(X, Y) \in [0, 1]$
2. $\mathcal{R}(X, Y) = 0$ if and only if X and Y are independent.

In contrast to Pearson correlation, distance correlation can determine independence and measure non-linear dependence. However, unlike Pearson correlation, distance correlation does not measure the direction of dependence. To compute sample distance correlation on observations $(X_1, Y_1), ..., (X_n, Y_n)$, first determine pairwise distances

$$a_{ij} = \|X_i - X_j\|, \quad b_{ij} = \|Y_i - Y_j\| \tag{5.1}$$

Then perform double-centering.

$$A_{ij} = a_{ij} - \bar{a}_{i\cdot} - \bar{a}_{\cdot j} + \bar{a}_{\cdot\cdot} \tag{5.2}$$
$$B_{ij} = b_{ij} - \bar{b}_{i\cdot} - \bar{b}_{\cdot j} + \bar{b}_{\cdot\cdot} \tag{5.3}$$

The sample squared distance covariance is

$$\mathcal{V}_n^2(X,Y) = \frac{1}{n^2}\sum_{i,j=1}^{n} A_{ij}B_{ij} \tag{5.4}$$

Similarly, compute the sample squared distance variances $\mathcal{V}_n^2(X,X)$ and $\mathcal{V}_n^2(Y,Y)$. Finally, the sample distance correlation is [141]

$$\mathcal{R}_n^2(X,Y) = \begin{cases} \dfrac{\mathcal{V}_n^2(X,Y)}{\sqrt{\mathcal{V}_n^2(X,X)\mathcal{V}_n^2(Y,Y)}} & \text{if } \mathcal{V}_n^2(X,X)\mathcal{V}_n^2(Y,Y) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

A benefit of distance correlation is that X and Y do not need to share the same number of dimensions. Hence, this allows distance correlation to be computed between every pair of columns during preprocessing, including distance correlation between numeric entries (scalars) and textual entries (vectors from the `[CLS]` embedding from BERT). This allows a heuristic to estimate column dependencies in a quick and efficient manner.

For each table, we compute pairwise distance correlation between all columns, yielding a symmetric matrix. For each column, we take the largest 50% of values in the column, excluding the diagonals, then compute the mean. The resultant vector is a heuristic for the aggregate column dependencies of each column in the table.

5.3.4 Caching and Storage

The preprocessed and encoded data are stored as a Python [148] pickle format, which is then further compressed with DEFLATE lossless data compression algorithm [36], reducing the total storage required by 10-15%. In addition, BERT embedding tensors are stored as 16-bit floating point representations instead of the usual 32-bit representation. The items stored in the files are:

- Column name BERT embedding tensors (and their respective padding masks)
- Textual entry embedding tensors (and their respective padding masks)
- Numeric values (raw and SMR encoded)
- Candidate indicators
- Mean distance correlation by column

Table 5.2 contains key summary statistics for the cached T4 datasets.

Table 5.2: Summary statistics of cached T4 datasets.

Summary statistics	**Value**
Number of tables	3,488,957
Number of rows	342,346,044
Maximum number of rows in a table	100
Maximum BERT sequence length	64
Proportion of T4 dataset used	∼16%
Total storage used	2.2 TB

5.4 Model Pretraining

5.4.1 Architecture

The architecture is mostly similar to the one discussed in Chapter 4 Section 4.4.3. However, the architecture differs in several aspects. To avoid ambiguity, we denote the improved basis transformer proposed in this chapter as $\mathcal{PBT}$ which simply stands for a *pretrained basis transformer*.

Modifications are necessary to accommodate the needs of a foundation model so that it can be sufficiently expressive in its representation. We outline the details of these modifications below.

Target Column Masking In $\mathcal{BT}$, the regressor head receives the mean of latent tensors from each column as input, and does not explicitly receive the column name of the target. This is generally sufficient for a multi-task objective as the targets can be inferred from the training set. To pretrain $\mathcal{PBT}$ and enable zero-shot inference, the column name of the target should be incorporated as part of the input. Since basis transformers accept pairs of column names and entries, the entry corresponding to the target column is masked with the special token `[TARGETMASK]`. Figure 5.1 visualizes the flow.

Input Transforms The linear projections in $\mathcal{BT}$ that are used to downscale BERT [37] vectors and upscale numeric vectors are now replaced by a one-hidden-layer MLP followed by a linear projection with layer normalization [7]. With a highly diverse range of texts in over three million tables and constrained by a frozen text encoder, there is a need for more expressive textual representation in line with the tabular tasks.

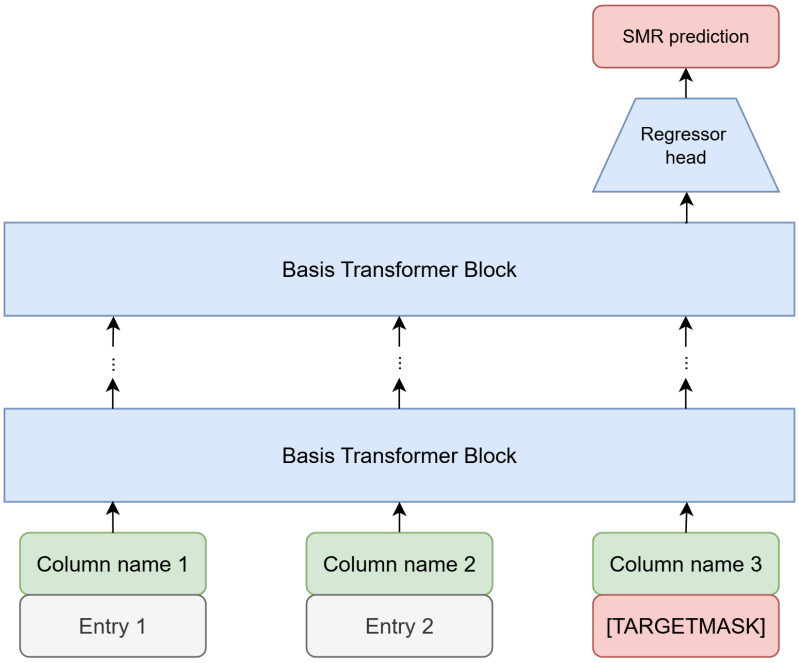


Figure 5.1: Illustration of the target column masking in pretrained basis transformers.

Residual Connections Recall that in $\mathcal{BT}$, there are residual connections from the high-level bottlenecked representation of the previous block to the latent compression model in Equation 4.22. We observe marginal improvements when residual connections are scaled to a smaller value by a factor of 0.8. This phenomenon is also observed and studied by Zhang et al..

Regressor Head Recall that the regression output of $\mathcal{BT}$ is an SMR encoding of the predicted regression target. This can be a fairly complex representation to produce, hence the regressor requires more complexity to model this conversion. However, deep feed forward neural networks that decrease in hidden dimensions cannot have residual connections. To mitigate this, we add *projection shortcuts*, which are simply residual connections that involve linear projection and layer normalization to ensure good gradient flow during back-propagation. Figure 5.2 shows the interaction of layers in the regressor head.

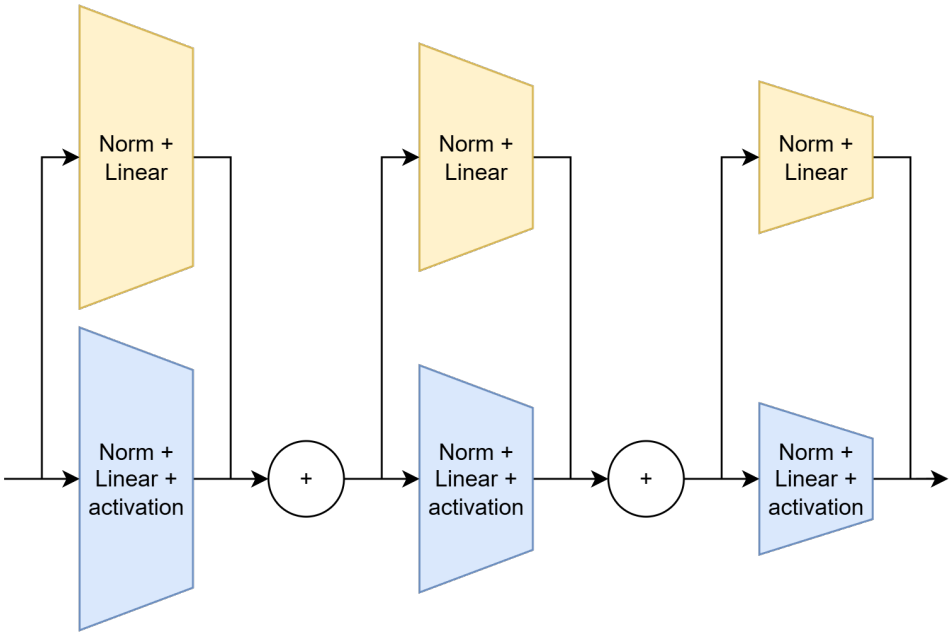


Figure 5.2: Illustration of the regressor head with projection shortcuts.

5.4.2 Pretraining Objective

Target selection The pretraining objective for $\mathcal{PBT}$ is similar to the masked language model (MLM) objective in BERT [37]. The MLM pretraining scheme involves random masking of tokens or replacement with a random token, which could lead to multiple tokens being masked. This is reasonable as there are some redundancies in languages. In $\mathcal{PBT}$, however, we mask exactly one column for two reasons: (1) A column is significantly more information dense than a token in natural language, so having multiple columns masked greatly reduces the amount of information in the row, and (2) the quality of T4 dataset is not sufficiently high and many numeric features could be unrelated to the rest of the columns.

The manner in which columns are selected as targets is not uniformly random. The cached values of distance correlation in Section 5.3.3 influence the probability in which columns are selected as targets on a table-by-table basis. The desired selection conditions are:

(1) Cap distance correlation score to a maximum of γ_U as the those targets might correspond to excessively simple tasks

(2) If all distance correlation scores are below γ_L, make the column with the highest score the target

(3) If there exists some distance correlation score above some threshold γ_L, never make any column below γ_L the target

(4) Distance correlation is converted to a categorical distribution via a softmax operation with temperature τ

The details of the selection process are included in Algorithm 4.

Algorithm 4 Target column selection algorithm

Inputs: Distance correlation vector $\overline{\mathcal{R}} \in [0,1]^C$, candidate indicators $\chi \in \{0,1\}^C$, low score threshold γ_L, high score threshold γ_U, softmax temperature τ
for $i \in \chi$ **do**
    if $\chi_i = 0$ **then**
        Set $\overline{\mathcal{R}}_i \leftarrow -\infty$ ▷ Disallow non-candidates
Clip values in $\overline{\mathcal{R}}$ to a maximum of γ_U ▷ (1)
if $\max(\overline{\mathcal{R}}) > \gamma_L$ **then**
    Set values in $\overline{\mathcal{R}}$ below γ_L to $-\infty$ ▷ (2)
else
    Set values in $\overline{\mathcal{R}}$ below $\max(\overline{\mathcal{R}})$ to $-\infty$ ▷ (3)
return $\text{softmax}\left(\frac{\overline{\mathcal{R}}}{\tau}\right)$ ▷ (4)

Optimization objective Recall that $\mathcal{PBT}$ is trained with SMR encoded targets which are vectors of bits, implying a binary cross entropy objective. However, this leads to poor performance in practice. Label smoothing improves performance as loss values are lower and stabilizes training. We introduce a variant of binary cross entropy with label smoothing called *clipped smoothed binary cross entropy* (CS-BCE).

Let $y \in \{0,1\}$ be a binary target, $p \in (0,1)$ the predicted probability and $\alpha = [0,1]$ the smoothing coefficient. First, we define the label-smoothed target.

$$q_y = (1-\alpha)y + \frac{\alpha}{2} = \begin{cases} \frac{\alpha}{2} & y = 0 \\ 1 - \frac{\alpha}{2} & y = 1 \end{cases} \tag{5.6}$$

CS-BCE can then be defined piecewise as

$$\mathcal{L}_{\text{CS-BCE}}(y, p; \alpha) = \begin{cases} 0 & y = 1 \text{ and } p \geq 1 - \frac{\alpha}{2} \\ 0 & y = 0 \text{ and } p \leq \frac{\alpha}{2} \\ \text{BCE}(q_y, p) - \text{BCE}(q_y, q_y) & \text{otherwise} \end{cases} \tag{5.7}$$

where $\text{BCE}(q, p) = -q \log p - (1 - q) \log(1 - p)$ is the standard binary cross entropy loss.

Regular label smoothing of binary cross entropy has an undesirable convexity in its loss surface, which is shown on the right graph of Figure 5.3. For example, suppose the target is 1 but the model predicts 0.9. During backpropagation, the model is optimized to adjust its prediction towards 0 despite already being on the correct side of the prediction threshold. This adjustment is unnecessary and potentially detrimental.

The improvement in CS-BCE comes from "sparsifying" the loss by clipping the loss on either side of the minimum value. Models are no longer penalized for having higher confidence than intended on correct predictions, as seen on the left graph of Figure 5.3. The sparsity of the loss also reduces the degree of internal covariate shift [74], especially on models producing many binary values in tandem, as parameters responsible for correct predictions do not experience gradients stemming from that predicted value. A minor improvement is to shift the loss values accordingly so that the correct predictions always have a loss of zero, which is helpful for monitoring and interpreting the loss values.

5.4.3 Miscellaneous Pretraining Details

Monitoring Metric Over the course of pretraining $\mathcal{PBT}$, the model is validated multiple times to identify the checkpoint that best generalizes to unseen data. As discussed in Chapter 4, multiple tasks, tables and columns have different scales of target values, making it difficult to compute an aggregate of predictive performance. We use Equation 5.8 as it measures relative agreement in magnitude so we call this *magnitude ratio similarity* (MRS).

$$\text{MRS}(y, \hat{y}) = \frac{\min\{|\hat{y}|, |y|\} + \varepsilon}{\max\{|\hat{y}|, |y|\} + \varepsilon} \text{ for a small } \varepsilon > 0 \tag{5.8}$$

MRS is bounded between 0 and 1, where 1 indicates perfect agreement in magnitude. It is a unitless, symmetric and scale-invariant measure, making it suitable to measure across various datasets with different scales of targets. For example, if the MRS is 0.5, this implies

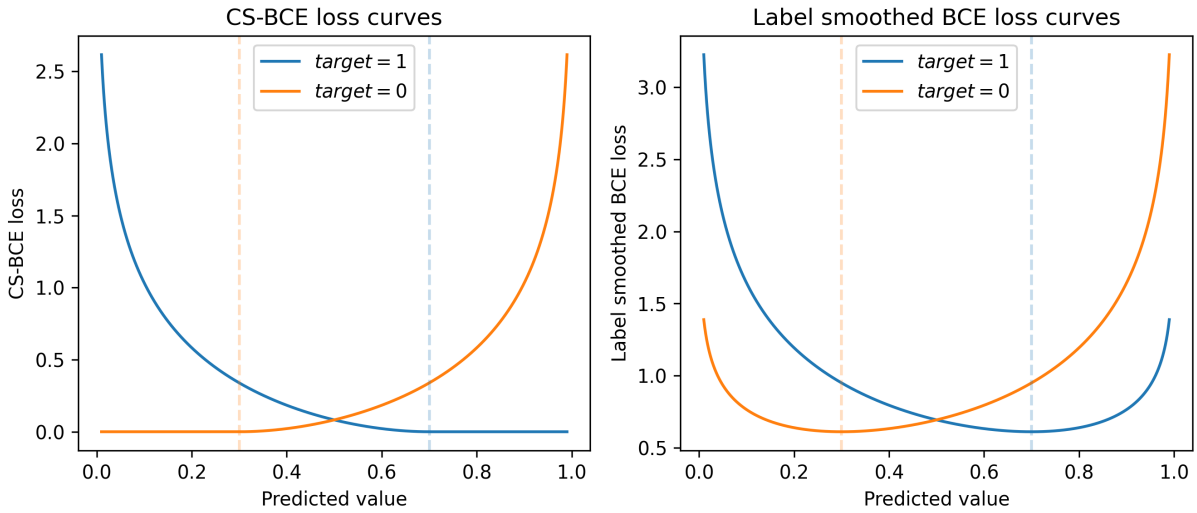


Figure 5.3: Loss curves of CS-BCE (left) and label smoothed BCE (right), both with a smoothing coefficient of $\alpha = 0.6$. The vertical dotted lines indicate the threshold of correctness from label smoothing for the corresponding target values.

that the predicted value is either double or half the true value. We focus on the magnitude only as it makes the interpretation clearer, and $\mathcal{PBT}$ is generally performant in predicting the sign bit.

The mean of MRS is computed across the entire validation split, and the checkpoint with the highest value is chosen as the final model.

Learning rate scheduler To improve the stability of pretraining, we use a cosine learning rate schedule [99]. We find that periodic warm restarts were not empirically helpful, so we excluded that component. We did, however, use a short warmup schedule before moving to the cosine learning rate schedule.

Mixed precision training Mixed precision training [108] is used to leverage features of modern GPU accelerators that enable faster 16-bit matrix multiplication. As 16-bit numbers take half the space of 32-bit numbers, the batch size is theoretically larger, although it would not be doubled. In our configuration of Huggingface's `accelerate` interface [58], we opted for the bfloat16 [24] representation to reduce the risk of gradient underflow or overflow that could happen with the float16 representation.

Internally, the master copy of weights is preserved in float32 representation but the forward pass, loss and gradient computations are done in bfloat16 representation. They are upcasted to the higher float32 representation to update weights before being downcasted to the bfloat16 representation for the next step [108].

Connecting data processing, caching and pretraining A flow chart is provided in Figure 5.4 that summarizes the end-to-end pipeline, starting from the data processing of raw tables to the model optimization.

5.4.4 Hyperparameters and Pretraining Outcomes

Encoding hyperparameters The high bits and the low bits in the SMR are chosen to be 48 and 15 respectively. Including the sign bit, the total numeric dimension is 64.

Architectural hyperparameters $\mathcal{PBT}$ consists of 5 $\mathcal{BT}$ blocks stacked sequentially. There are 4 initial basis queries. The chosen latent vector dimension in sequences is $D = 192$. Recall that the latent compression module in Figure 4.2 maps a sequence with shape (L, D) to a single latent compressed vector with shape (rD). r is chosen to be 5. Each latent contextualization module has 2 self-attention layers, except for the last $\mathcal{BT}$ block which has 4 self-attention layers.

All multihead attention modules have 32 attention heads and an MLP expansion factor of 4. The regressor head (as described in Section 5.4.1) is a 4-layer feedforward neural network with projection shortcuts. The regressor expects a dimension of $rD = 960$ and goes through changes in dimension in this order: 2048, 1024, 256, and finally 64. The final dimension represents the SMR output of the prediction. The entire $\mathcal{PBT}$ model has 159,067,136 parameters.

Training setup The entire pretraining is done on 4 compute nodes hosted on a high-performance cluster. Each compute node has 32 CPU cores, 160 GB of RAM and four A40 48GB GPUs. With each process having a GPU and a training batch size of 32, the effective batch size is 512 rows of tabular data. We use Huggingface's accelerate library [162] to enable distributed training.

We define *strides* to be the number of training gradient updates before validating. $\mathcal{PBT}$ was trained with 335 strides, with each stride consisting of 2000 forward passes and gradient updates. Equivalently, this means that $\mathcal{PBT}$ was validated 335 times. In total, $\mathcal{PBT}$ was

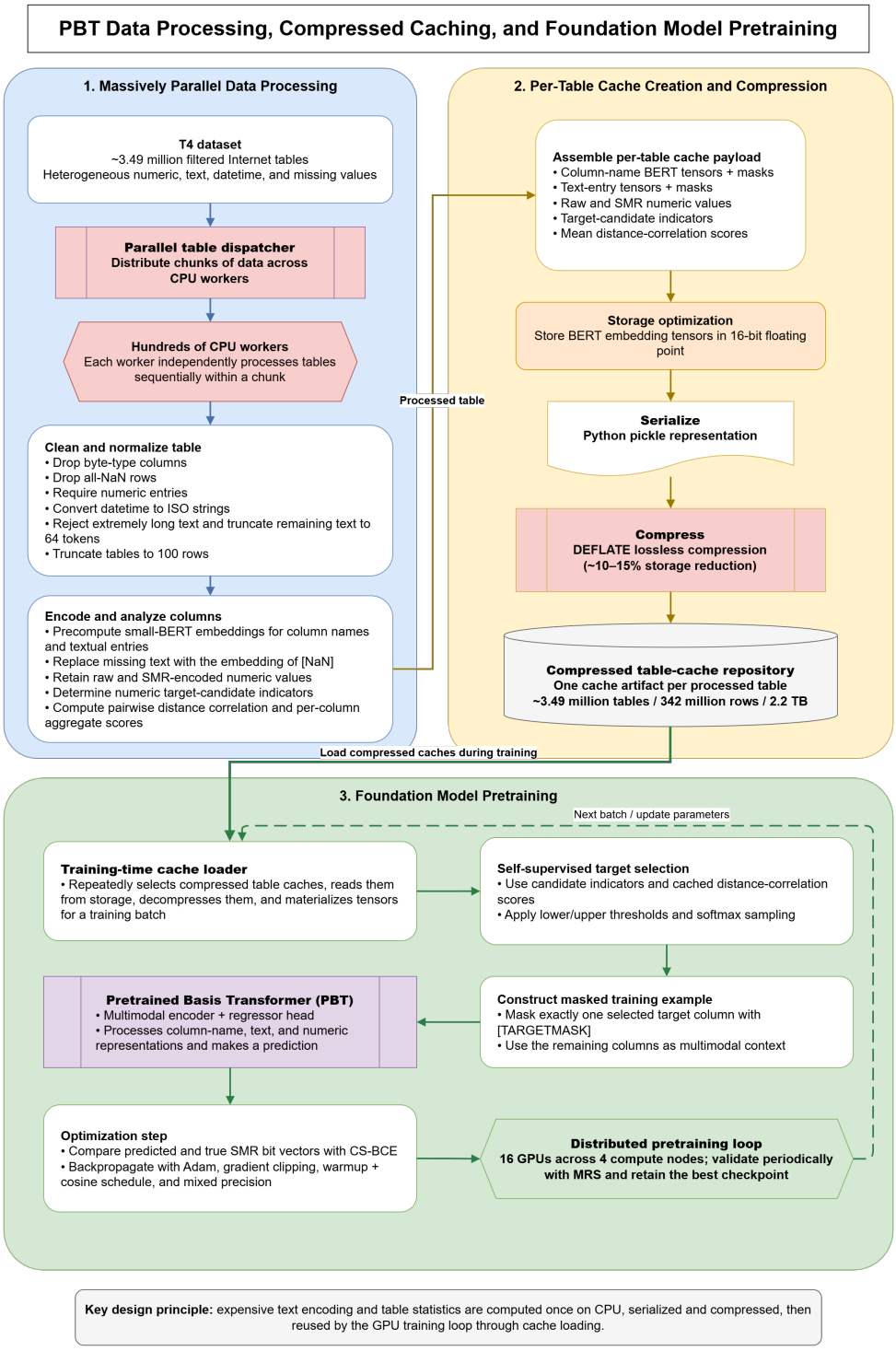


Figure 5.4: A complete flow chart from raw tables to model optimization during pretraining.

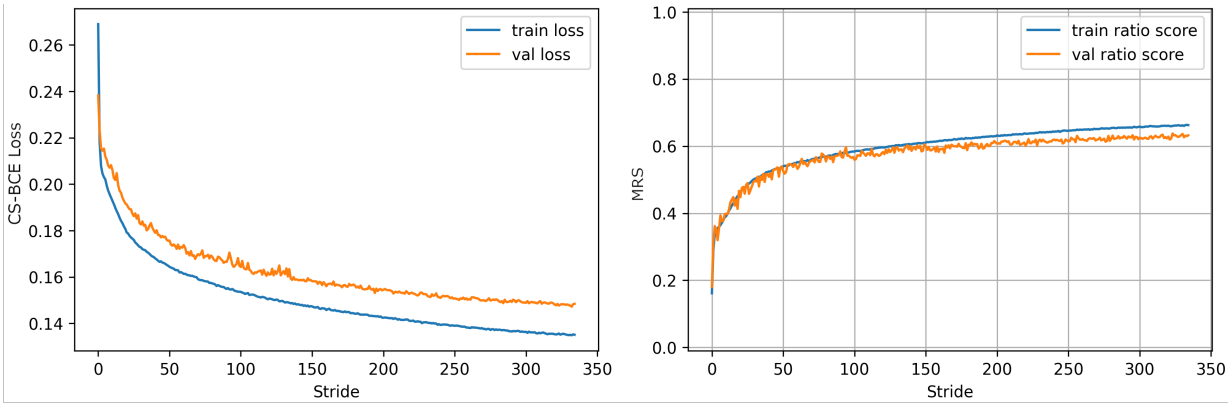


Figure 5.5: Training loss (left) and magnitude ratio similarity monitoring trace (right) during the pretraining process of $\mathcal{PBT}$.

trained on 343,040,000 rows of data. As mentioned in Table 5.2, there are 342,346,044 rows so this equates to approximately one epoch of training.

Recall that the target column selection strategy in Section 5.4.2 is based on distance correlation. We set γ_U to 0.75 and γ_L to 0.3. The softmax temperature τ is set to 0.3.

Optimization hyperparameters The CS-BCE loss function uses $\alpha = 0.8$, which is a very aggressive label smoothing.

We use the Adam optimizer [81] with a L2 weight decay of 1×10^{-4}. For training stability, the gradient norms are clipped to a maximum of 1.0. We also use a learning rate scheduler with 10,000 warmup steps to an initial rate of 2×10^{-4} and cosine annealing such that the learning rate reaches 5×10^{-5} by the final gradient update.

Pretraining Outcome The entire pretraining process took 262 hours, or approximately 11 days. The loss and MRS values are shown in Figure 5.5.

5.5 Model Finetuning

This section pertains to specializing a pretrained $\mathcal{PBT}$ for a specific task and dataset. Finetuning a $\mathcal{PBT}$ is not strictly necessary – it improves performance, but it loses zero-shot capabilities.

5.5.1 XGBoost Head

The regressor head in Figure 5.2 is crucial in enabling cross-table predictions during pretraining. This section narrows down the scope to a single or smaller set of tasks and tables, opening the door for appropriate adjustments to the regressor head.

Here, we propose replacing the regressor head directly with an XGBoost regressor [27]. In theory, it can be any model, but we actively avoided any in-context learning model due to the computational overhead of the context window during inference time. The XGBoost regressor is fitted with the same input to the regressor head, which is the latent vector from the last basis transformer block corresponding to the target column, and scalar values of the target.

The pretraining process aims to learn an optimal representation as well as a mapping to the target. However, on a narrower problem, the SMR output of the regressor head is excessively complex, and the inductive bias of the XGBoost or smaller designs should fit the task better. This combines two complementary strengths: the multimodal representational power of $\mathcal{PBT}$, and the predictive performance of XGBoost.

To specialize the representation on a specific task or table, the $\mathcal{PBT}$ can be finetuned end-to-end with the original regressor head, before producing the latent vectors for the XGBoost to be fitted.

5.5.2 Parametric Dimensionality Reduction

The latent vector from $\mathcal{PBT}$ is high-dimensional and directly fitting the XGBoost model on those vectors is not recommended for three reasons:

- High-dimensional input likely requires a deep tree and hence a longer time to fit
- There is a higher risk of multicollinearity which can lead to redundancy and instability
- There is a higher risk of overfitting

A common way to reduce dimensionality is to use principal component analysis (PCA) [2]. There are some drawbacks to PCA in our situation. The goal of PCA is variance preservation independent of a task. Using PCA also interrupts the end-to-end framework and does not participate directly in the finetuning process.

Let us focus on the last linear projection layer in $\mathcal{PBT}$ before producing the latent vector. It has weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and bias $b \in \mathbb{R}^{d_{\text{out}}}$, and transformation is as follows:

$$z = Wh + b \tag{5.9}$$

where z is the output latent vector and h is a vector used to create z.

We propose reparameterizing W into two matrices $U_k \in \mathbb{R}^{d_{\text{out}} \times k}$ and $V_k \in \mathbb{R}^{k \times d_{\text{in}}}$ where k is some predefined rank k. The two matrices are randomly initialized. We attempted to use singular value decomposition (SVD) but doing so yielded poorer performance compared to random initialization. The reparameterization results in a two-step process

$$\hat{z} = V_k h \tag{5.10}$$

$$z \approx U_k \hat{z} + b \tag{5.11}$$

where now $\hat{z}$ is the reduced output latent vector.

For finetuning, we want the latent vector to be specialized towards the downstream task. However, XGBoost is a non-differentiable model, hence we retain the original $\mathcal{PBT}$ regressor head to propagate gradients through the rest of the model. In practice, this does not require many epochs to finetune the latent space. After this is done, the XGBoost head will be fitted using the latent vectors $\hat{z}$ and the $\mathcal{PBT}$ parameters will be frozen.

5.5.3 Hyperparameter Optimization

For a finetuned version of $\mathcal{PBT}$, there are now two separate components: the multimodal feature extractor of $\mathcal{PBT}$, and the XGBoost regression head fitted on a compressed latent representation $\hat{z}$. The benefit of this modularity is $\mathcal{PBT}$ being a sufficiently expressive neural network that can produce useful representations without too many finetuning steps, and XGBoost, being a lightweight prediction model given the compressed latent representation, can undergo intensive hyperparameter optimization to refine its prediction.

For efficient search of XGBoost hyperparameters, we opt for the tree-structured Parzen estimator (TPE) algorithm [12] that is implemented in the Hyperopt package [13].

5.6 Downstream Evaluation Results

We evaluate $\mathcal{PBT}$ and relevant baseline models across multimodal tabular datasets on two tasks: imputation tasks and standard regression tasks. Imputation tasks involve predicting

numeric entries that are missing in a zero-shot manner. However, some baseline models, especially the ICL-variants, are not capable of zero-shot predictions so they are evaluated in a one-shot setting. Standard regression tasks involve training or finetuning on a training split, choosing the best hyperparameter using the validation split, and then reporting the results on the test split.

5.6.1 Evaluation Datasets

Pure tabular models are commonly evaluated using the popular OpenML benchmarks [149, 16] such as OpenML-CTR23 [45], or the more modern variants such as TabZilla [105] and AMLB [51]. Those are excellent datasets for the narrower scope of tasks addressed by those models. However, we do not use those datasets for a few reasons.

While there have been recent efforts in selecting a "harder" subset of tabular datasets, the focus has been on the relationship between the features and the targets. The motivation of $\mathcal{PBT}$ is tackling the difficulties of ingesting the features in real world data; $\mathcal{PBT}$, at this time, is not designed to compete on the clean, mostly numeric datasets. Another reason is that some tabular foundation models are directly pretrained on the evaluation benchmarks – TabDPT being one of them [102]. The risk of data leakage is arguably more likely than in other domains because reliable evaluation benchmarks are relatively scarce. Finally, $\mathcal{PBT}$ aims to fill the gaps of research and commercial needs, which neither of the mentioned benchmarks offers.

The T4 dataset has been introduced in Section 5.3.1. When preparing for the pretraining of $\mathcal{PBT}$, we excluded 149 tables to be used as the test split. This will be used for table imputation tasks as it contains tables with no obvious targets. A model will be asked to fill in the deliberately masked numeric entries, given another row as a reference, either used as in-context examples or used for training. $\mathcal{PBT}$ will not be given a row as reference as the intention is to demonstrate zero-shot capabilities.

Table imputation alone is not sufficient and while the T4 dataset offers diversity in tasks and challenges in weakly correlated columns, it is not ideal to be used as a source of regression evaluation datasets. We opt instead for the RelBench dataset by Robinson et al. [123]. RelBench is a convenient interface to a collection of realistic, production-level datasets. Production-level implies integer IDs, timestamps, extensive free-form text with the possibility of typos or artifacts in the 44 tables that we obtain from RelBench. There is one target column for every table, and they are generally appropriate for the context of the table. Additional RelBench information can be found in Appendix C.1. RelBench will be used for what we call the "standard regression" task.

Table 5.3: An example RelBench evaluation dataset from Amazon [71].

product_id	verified	review_text	summary	customer_name	rating
31892	True	A small book with a large amount of information. The women described in this book are giants to my	The Adams women	Ladyhawke	4.0
331515	False	As usual, the plot takes plenty of twists and turns. Laugh out loud! Characters are easy to follow	Lot's of fun!	Amazon Customer	5.0
405776	True	Great condition!	Five Stars	Amazon Customer	5.0
345922	False	Welcome to Buckneck (complete with a straight-from-the-South story to its name), a small town in We	Cozy mystery with warm distinct characters	E. Marks	4.0

An example of a RelBench dataset is shown in Table 5.3. When tasked to predict the `rating`, the `review_text` column is crucial as a feature but should not be assumed to be the only important feature. Here, we emphasize the presence of free-form text but there are also cross-column relationships to exploit, which may involve merging information between a free-form text column and a numeric column.

5.6.2 Baselines and Evaluation Metrics

Strong contenders in the tabular domain are chosen as baselines, grouped into one of the three categories: tabular foundation models, deep tabular models and non-neural models. The tabular foundation models are:

1. CARTE [79]
2. Mitra [173]
3. TabDPT [102]
4. TabPFN-2.5 [57]
5. TabSTAR [4]

The non-neural models are:

1. Linear regression (from scikit-learn [116])
2. Multi-layer perceptron (from scikit-learn [116])
3. XGBoost [27]
4. CatBoost [119]

The implementations of many algorithms were obtained from popular sources such as scikit-learn [116] and AutoGluon [42]. The rest were obtained from their respective GitHub repositories. Baseline algorithms, without a specialized inference pipeline, cannot handle text features. In this case, ordinal encoding is applied to transform textual features into numeric features.

To compare different methods during imputation, each method is scored using root mean squared error (RMSE) and subsequently ranked where 1 indicates the best performer for that table. To evaluate methods as a whole, the mean and standard error of the rank are used. The exact computation will be shown in detail in each subsection.

On the other hand, to compare different methods for the standard regression, we also use ranks based on RMSE. However, we also use a standardized variant of RMSE called min-max scaled RMSE, introduced in Chapter 4 Equation 4.27. This metric is suitable in standard regression and not zero/one-shot imputation because the predictions are significantly more consistent and competitive so the difference in performance is discernible here. Zero and one-shot imputation can be quite challenging for the other methods so the difference in performance is not as interpretable.

5.6.3 Imputation Tasks

The primary goal of imputation is to demonstrate the zero-shot capabilities of $\mathcal{PBT}$. The baseline algorithms are given one sample to either train or to be used as in-context samples. A numeric column is randomly selected to be the target column to be imputed. The rest of the rows in the table will be used as the test dataset. Additional details can be found in Appendix C.2. The mean and median number of rows in the test splits are 619.28 and 665 respectively. For each dataset, five columns are used as target columns. In total, this experiment involves

$$619 \frac{\text{mean entries}}{\text{table}} \times 149 \text{ tables} \times 5 \text{ random columns} \times 9 \text{ methods} = 4{,}150{,}395$$

Table 5.4: Rank results of the imputation tasks.

Method	**Mean rank** (↓)	**Standard error of rank**
$\mathcal{PBT}$ (ours)	**4.12**	0.13
Linear regression	4.71	0.06
CatBoost	4.73	0.05
XGBoost	4.73	0.05
TabPFN	4.75	0.07
Mitra	4.85	0.12
TabDPT	4.89	0.07
TabSTAR	4.98	0.13
MLP	7.25	0.10

evaluation points.

Let $\mathcal{M} = \{m_1, \ldots, m_M\}$ denote the set of evaluated models. For each dataset, five columns are independently masked at random. Let $\mathcal{C} = \{c_1, \ldots, c_T\}$ denote the complete set of such masked columns pooled across all datasets. For each masked column $c \in \mathcal{C}$, let $s_{m,c} \in \mathbb{R}$ be the RMSE attained by model m when predicting the held-out values of that column. We rank the models on each masked column independently

$$r_{m,c} \;=\; \mathrm{rank}(s_{m,c} \,|\, \{s_{m',c}\}_{m' \in \mathcal{M}}) \,, \tag{5.12}$$

where lower RMSE corresponds to a smaller (better) rank. The aggregate score for model m is the pooled mean over all masked columns,

$$\bar{r}_m \;=\; \frac{1}{T} \sum_{c \in \mathcal{C}} r_{m,c}, \qquad \mathrm{SE}(\bar{r}_m) \;=\; \frac{\sigma_m}{\sqrt{T}}, \tag{5.13}$$

where σ_m is the sample standard deviation of $\{r_{m,c}\}_{c \in \mathcal{C}}$. The mean and standard error of ranks are shown in Table 5.4.

Note that CARTE does not work in a one- or two-shot regime due to numeric instability. Hence, it is omitted from the imputation evaluation.

5.6.4 Standard Regression Tasks

Standard regression is the most common way tabular models are evaluated. This involves using a training and validation dataset to fit a model, specializing it towards a particular

task. The training dataset provides the means to optimize the learnable parameters, while the validation dataset provides the means to optimize the hyperparameters.

These tasks are found in the RelBench dataset, where every dataset has a designated target column. The mean and median number of rows in the test splits are 5608.72 and 6251 respectively. This experiment involves:

$$56077 \ \frac{\text{mean entries}}{\text{table}} \times 44 \text{ tables} \times 5 \text{ seeds} \times 10 \text{ methods} = 123{,}369{,}400$$

evaluation rows.

All tabular foundation models are finetuned with the complete training dataset, including the ICL-based foundation models. Notably, TabPFN, through its Python library, offers a well-documented and convenient wrapper to finetune on a dataset. The non-foundation models are trained starting from randomly initialized weights. All approaches use hyperparameters optimized as described in Section 5.5.3. However, for $\mathcal{PBT}$, only the hyperparameters of the XGBoost head are optimized with the Hyperopt library. The $\mathcal{PBT}$ component is only finetuned for 1 epoch, according to the procedure described in Section 5.5.2. Additional details can be found in Appendix C.2.

For each method m and dataset d, RMSE is measured under K random seeds $\mathcal{S} = \{s_1, \ldots, s_K\}$ ($K = 5$ in our experiments), yielding scores $s_{m,d,k}$. Ranks are computed independently for each (d, k) pair across methods,

$$r_{m,d,k} \ = \ \text{rank}(s_{m,d,k} \,|\, \{s_{m',d,k}\}_{m' \in \mathcal{M}}) \,, \tag{5.14}$$

and aggregated by pooling over both datasets and seeds,

$$\bar{r}_m \ = \ \frac{1}{NK} \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{S}} r_{m,d,k}. \tag{5.15}$$

The rank and MMS-RMSE scores are shown in Tables 5.5 and 5.6. In addition, to show statistical significance in $\mathcal{PBT}$'s performance uplift, a ranked-based post hoc analysis was conducted. Each dataset served as a block while each model represented a treatment group. Since the comparison involved multiple models evaluated using the same experimental settings, the Conover posthoc procedure was used to perform pairwise comparisons among the models.

To support the interpretation of the statistical results, a critical difference diagram was generated in Figure 5.6 to visualize the average ranks of the models, and to depict whether the differences are statistically significant, based on the posthoc p-value matrix. The horizontal connecting bars represent models that could not be statistically distinguished at the selected significance levels.

Table 5.5: Rank results of standard regression tasks.

Method	**Mean rank (↓)**	**Standard error of rank**
$\mathcal{PBT}$ (ours)	**2.60**	0.05
TabPFN	3.16	0.05
XGBoost	3.44	0.05
CatBoost	4.54	0.06
TabDPT	4.63	0.08
CARTE	5.23	0.09
TabSTAR	5.75	0.08
Mitra	6.74	0.05
LinReg	8.91	0.01
MLP	10.00	0.00

Table 5.6: Min-max scaled RMSE results of standard regression tasks.

Method	**Mean MMS-RMSE (↓)**	**Standard error of MMS-RMSE**
$\mathcal{PBT}$ (ours)	**0.017**	0.003
XGBoost	0.025	0.004
TabPFN	0.030	0.005
TabDPT	0.032	0.005
CatBoost	0.052	0.009
Mitra	0.064	0.011
TabSTAR	0.089	0.016
CARTE	0.092	0.016
LinReg	0.130	0.020
MLP	0.742	0.020

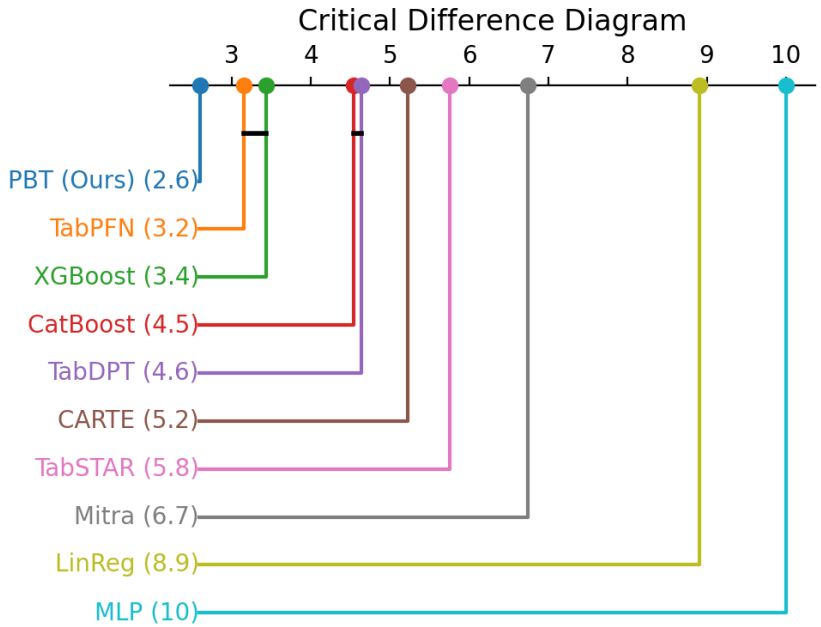


Figure 5.6: Critical difference diagram on the standard regression evaluation with a level of significance of $\alpha = 0.05$

Table 5.7: Rank results of CS-BCE loss function ablation study in standard regression tasks.

Method	**Mean rank (all)** (↓)	**Mean rank (pair)** (↓)
PBT (with CS-BCE)	**3.02** ± 0.06	**1.42** ± 0.02
PBT (without CS-BCE)	3.14 ± 0.05	1.58 ± 0.02
TabPFN	3.79 ± 0.06	N/A
XGBoost	4.14 ± 0.06	N/A
CatBoost	5.34 ± 0.07	N/A
TabDPT	5.41 ± 0.08	N/A
CARTE	5.94 ± 0.10	N/A
TabSTAR	6.59 ± 0.09	N/A
Mitra	7.72 ± 0.05	N/A
LinReg	9.91 ± 0.01	N/A
MLP	11.00 ± 0.00	N/A

5.6.5 Loss Function Ablation Study

To verify the efficacy of the loss function introduced in Section 5.4.2, we pretrained an identical copy of $\mathcal{PBT}$ but with a regular label-smoothed binary cross entropy. The smoothing coefficient and all other pretraining configurations are kept the same. The non-CS-BCE variant of $\mathcal{PBT}$ is evaluated in a standard regression setting, and ranked in two ways – against only the regular $\mathcal{PBT}$, and against all approaches. The results are shown in Table 5.7.

5.7 Discussion

5.7.1 Zero-Shot versus In-Context Learning

There is a fundamental difference in design philosophy between $\mathcal{PBT}$ and TabPFN – a deliberate design choice as $\mathcal{PBT}$ is meant to fill the gaps in the space of tabular models. TabPFN performs incredibly well in a low-data regime [57] due to its prior. However, the low-data regime still requires at least some observed examples to function.

$\mathcal{PBT}$, on the other hand, offers an alternative. While inherently unable to leverage in-context examples, it can produce meaningful representations either when the test-time

data format is inconsistent or when a large finetuning dataset is available. Let us explore both cases separately.

Inconsistent test-time data format Regimes involving evolving data formats could be applicable in a retrieval-based use-case and services with noisy or inconsistently formatted data. For example, retrieval applications could benefit since $\mathcal{PBT}$ acts as a tabular encoder to query for similar data in a vast data bank. Also, services with fuzzy data have to be robust to incoming data format, and $\mathcal{PBT}$ can act as an adaptor. Initially, $\mathcal{PBT}$ should be finetuned with a diverse set of training data revolving around a central theme such as housing prices. However, during deployment, $\mathcal{PBT}$ can encode or even directly produce a prediction from a tabular query despite having never observed that input format.

Large finetuning dataset In regimes with a large finetuning dataset, $\mathcal{PBT}$ and ICL-based models such as TabPFN cater to different needs. Beyond the question of predictive performance, they would excel in different settings. If the large finetuning dataset changes frequently, the ICL-based model would be the better choice because such models do not require weight updates, which is usually a compute-intensive endeavour.

However, if the large finetuning datasets are not frequently adjusted, $\mathcal{PBT}$ may prove to be the better option. Recall that ICL-based approaches use extensive self-attention operations to condition on ICL examples when predicting, and there is generally a high inference time cost when the dataset is large. For instance, TabPFN-2.5 officially supports up to 50,000 data points [57] but could support a larger dataset. The limit is set based on the computational overhead of self-attention. $\mathcal{PBT}$, on the other hand, can transfer relevant information via finetuning once in an offline setting. At deployment, there is no overhead as a function of the size of the finetuning dataset.

5.7.2 Pure Numeric Tasks

$\mathcal{PBT}$ underperforms in pure numeric tasks because of the current scale of training, and the design was never intended to support pure numeric tasks. There are two aspects to consider.

Inductive biases The architecture focuses on the fusion between numeric data and textual embedding vectors. The deficits of deep learning in tabular tasks have been thoroughly explored by Grinsztajn et al. [56]. $\mathcal{PBT}$ likely suffers from the smoothness bias mentioned

in their paper, and do most neural network-based approaches. Tree-based approaches do not face such issues and are generally seen to be effective.

Pretraining Pretraining makes a significant difference for neural approaches. To mitigate the smoothness issues, a massive dataset is required to modulate the latent space within the neural network. TabPFN is by far the most successful in pretraining, leveraging massive amounts of carefully curated synthetic datasets [57].

In a pure numeric regime, tree-based approaches and TabPFN excel due to their own strengths. The strengths of $\mathcal{PBT}$ do not necessarily transfer to a pure numeric regime at this time.

5.7.3 Data quality

While $\mathcal{PBT}$ was pretrained on a large amount of data, it could have been improved with higher quality data. Concretely, the T4 dataset [49], which is derived from TabLib [41], lacks "completeness" – the coverage within a topic/domain in terms of columns or spread of entry values so that meaningful information can be learned. For example, there may be many education datasets but they are record-based datasets with no meaningful target columns hence there are limited learnable relationships between pairs of columns. Furthermore, within a particular domain such as healthcare, there is insufficient diversity of datasets offering different perspectives. If the topic relates to tuberculosis, there is only one format offering insights into that topic, even if there are multiple datasets without duplicates. Overall, $\mathcal{PBT}$ could be improved with better curation of tabular data, with proper oversight in terms of privacy and safety.

5.8 Conclusion

Leveraging the strengths of basis transformers from Chapter 4 and scaling them to a foundation model by pretraining can be a promising solution to dealing with heterogeneous text-rich multimodal tabular data. Existing approaches fundamentally lack certain characteristics, especially free-form textual entries and zero-shot capabilities simultaneously. It has been demonstrated and statistically verified that a pretrained basis transformer excels in a specialized but common case of tabular tasks.

While $\mathcal{PBT}$ offers a fundamentally different way to scale tabular models, it can still be improved by incorporating more lessons from other methods. This could come in the

form of an in-context learning variant, or a task-independent row or table encoder variant – expanding the flexibility $\mathcal{PBT}$ can offer. Finally, a major promising but tangential direction is the synthetic generation of tabular data with an emphasis on realistic textual entries. TabPFN (up to version 2.5) has proven the efficacy of synthetic pretraining and this natural extension remains an important direction for future work.

Chapter 6

Conclusion and Future Work

In this final chapter, we provide a summary of contributions to the field of machine learning, tabular learning, and multi-armed bandits, followed by a discussion of possible directions for future work.

6.1 Summary of Contributions

First, we identified the key limitations encountered when applying existing bandit algorithms in many real-world applications, particularly those involving non-stationarity and high-dimensional inputs. While neural network-based bandit algorithms have been studied, we proposed an efficient approach that does not require frequent weight updates and natively adapts to new samples and evolving reward functions. Our proposed bandit algorithm is accompanied by a regret analysis to establish theoretical guarantees. More importantly, in both simple and realistic scenarios, the algorithm outperforms baselines on most datasets.

Second, we explored the current state-of-the-art in tabular models and identified limitations in extending these models to heterogeneous multimodal tabular data. The desiderata for tabular foundation models proposed by Van Breugel and Van Der Schaar, which we extended, form the backbone of our design philosophy. We introduced two important mechanisms: a vectored numeric representation based on the sign-magnitude representation, and a compression approach based on cross attention. The former allows training on multiple datasets without being adversely affected by differences in scale. The latter enables the homogenization of structured data, a necessary component for handling inherently heterogeneous data, and we provide theoretical insights into its capabilities. Overall,

we proposed an architecture called *basis transformer* with the aforementioned innovations that performs competitively against existing tabular models in multi-task and related task regression problems.

Third, we argue that basis transformers are well suited to scaling on multimodal tabular data, leading to the development of a foundation model based on this architecture. Using a large existing collection of datasets, the model underwent self-supervised pretraining guided by the distance correlation heuristic and a modification of the binary cross entropy loss, which improved the robustness and efficacy of the pretraining process. For finetuning, we proposed a model-agnostic adapter-like framework to efficiently specialize the pretrained model for a downstream task. The experimental results of the pretrained and finetuned basis transformers demonstrate improved performance over state-of-the-art baselines on tasks with text-rich heterogeneous tabular data.

6.2 Future Work

In this section, we provide broader directions that extend beyond the scope of each individual chapter.

A natural extension is the inclusion of classification tasks. It can be deceptively challenging from the perspective of multi-task and zero-shot learning. A regression task, by definition, guarantees that the target space is a one-dimensional numeric output. On the other hand, classification involves a set of valid target labels that may differ from task to task. This inconsistency can be challenging to model when there are several tasks simultaneously or when zero-shot capabilities are required. There are two possible ways to address this issue. The more conventional approach is to model the alignment of latent vectors with the set of valid target labels, forming a logit, which is then passed to a softmax layer to produce probabilities. A more natural extension to our work involves placing all target labels in a special column and the model is tasked with producing a regression output representing the logit. The row with the target label that corresponds to the highest logit represents the predicted class.

Synthetic data generation, as demonstrated by TabPFN [57], could be the key to elevating basis transformers in representation learning and predictive power. While the generation of purely numeric tables has been thoroughly explored within the research community, the realistic generation of tables containing coherent text remains limited. Because basis transformers require proper textual context, the current tools cannot readily support the pretraining process of basis transformers.

While the problems that the basis transformer was tasked with solving contain numeric values and free-form text, the design is not limited to those two data types. A major innovation in computer vision in 2021 involved viewing images as a sequence of tokens and using Transformer-based architectures [40]. In the same spirit, basis transformers can take any sequence of vectors as input. If the hypothesis is that an image can provide useful context for a table, this presents a promising research direction to pursue.

Finally, connecting the insights from all three chapters in this dissertation raises the question about the feasibility of applying foundation models in a bandit algorithm. Multi-armed bandits are typically concerned with balancing the exploration-exploitation trade-off and adapting to new information. A foundation model provides a more refined representation for downstream modeling, usually by contextualizing inputs using data observed during pretraining. Combining bandit algorithms and foundation models could forge a promising new direction in the bandit literature.

References

[1] Ieee standard for floating-point arithmetic. Standard IEEE Std 754-2008, IEEE Computer Society, New York, NY, USA, August 2008. URL https://web.archive.org/web/20160806053349/http://www.csee.umbc.edu/~tsimo1/CMSC455/IEEE-754-2008.pdf.

[2] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[3] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*, pages 127–135. PMLR, 2013.

[4] Alan Arazi, Eilam Shapira, and Roi Reichart. TabSTAR: A tabular foundation model for tabular data with text fields. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FrXHdcTEzE.

[5] Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6679–6687, 2021.

[6] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4(May):83–99, 2003.

[10] Yikun Ban, Yunzhe Qi, and Jingrui He. Neural contextual bandits for personalized recommendation. In *Companion Proceedings of the ACM Web Conference 2024*, pages 1246–1249, 2024.

[11] Soumya Basu, Orestis Papadigenopoulos, Constantine Caramanis, and Sanjay Shakkottai. Contextual blocking bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 271–279. PMLR, 2021.

[12] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[13] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[14] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(9), 2009.

[15] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[16] Bernd Bischl, Giuseppe Casalicchio, Taniya Das, Matthias Feurer, Sebastian Fischer, Pieter Gijsbers, Subhaditya Mukherjee, Andreas C Müller, László Németh, Luis Oala, et al. Openml: Insights from 10 years and more than a thousand papers. *Patterns*, 6(7), 2025.

[17] Christopher M Bishop and Markus Svensén. Bayesian hierarchical mixtures of experts. *arXiv preprint arXiv:1212.2447*, 2012.

[18] Andreas Bjerregaard, Peter Mørch Groth, Søren Hauberg, Anders Krogh, and Wouter Boomsma. Foundation models of protein sequences: A brief overview. *Current Opinion in Structural Biology*, 91:103004, 2025.

[19] Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50K5N.

[20] R. Bock. MAGIC Gamma Telescope. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C52C8B.

[21] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[22] Thomas Bonnier. Revisiting multimodal transformers for tabular data with text fields. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1481–1500, 2024.

[23] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*, 2022.

[24] Neil Burgess, Jelena Milanovic, Nigel Stephens, Konstantinos Monachopoulos, and David Mansell. Bfloat16 processing for neural networks. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pages 88–91. IEEE, 2019.

[25] Allan Carroll, Avalon, Ben Hamner, events4you, glhf, and Greg Melton. Event recommendation engine challenge. https://kaggle.com/competitions/event-recommendation-engine-challenge, 2013. Kaggle.

[26] Nicolo Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

[27] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[28] Xin Chen and I-Hong Hou. Contextual restless multi-armed bandits with application to demand response decision-making. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pages 2652–2657. IEEE, 2024.

[29] Xue-Wen Chen and Xiaotong Lin. Big data deep learning: challenges and perspectives. *IEEE access*, 2:514–525, 2014.

[30] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187, 2017.

[31] Shao-Ting Chiu, Junyuan Hong, and Ulisses Braga-Neto. Deeposets: Non-autoregressive in-context learning of supervised learning operators. In *Neurips 2024 Workshop Foundation Models for Science: Progress, Opportunities, and Challenges*, 2024.

[32] Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1724–1734, 2014.

[33] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

[34] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25 (70):1–53, 2024.

[35] Hal Daumé III. Bayesian multitask learning with latent hierarchies. *arXiv preprint arXiv:0907.0783*, 2009.

[36] Peter Deutsch. Rfc1951: Deflate compressed data format specification version 1.3, 1996.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[38] Nolan Dey, Gurpreet Gosal, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, Joel Hestness, et al. Cerebras-gpt: Open compute-optimal language

models trained on the cerebras wafer-scale cluster. *arXiv preprint arXiv:2304.03208*, 2023.

[39] Zhenhua Dong, Zhe Wang, Jun Xu, Ruiming Tang, and Jirong Wen. A brief history of recommender systems. *arXiv preprint arXiv:2209.01860*, 2022.

[40] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[41] Gus Eggert, Kevin Huo, Mike Biven, and Justin Waugh. Tablib: A dataset of 627m tables with context. *arXiv preprint arXiv:2310.07875*, 2023.

[42] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[43] Guhao Feng, Kai Yang, Yuntian Gu, Xinyue Ai, Shengjie Luo, Jiacheng Sun, Di He, Zhenguo Li, and Liwei Wang. How numerical precision affects mathematical reasoning capabilities of llms. *arXiv preprint arXiv:2410.13857*, 2024.

[44] Lei Feng. Data analysis and prediction modeling based on deep learning in e-commerce. *Scientific programming*, 2022(1):1041741, 2022.

[45] Sebastian Felix Fischer, Matthias Feurer, and Bernd Bischl. Openml-ctr23–a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023.

[46] Dylan Foster and Alexander Rakhlin. Beyond ucb: Optimal and efficient contextual bandits with regression oracles. In *International Conference on Machine Learning*, pages 3199–3210. PMLR, 2020.

[47] Kunihiko Fukushima, Sei Miyake, and Takayuki Ito. Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (5):826–834, 1983.

[48] Weihao Gao, Zheng Gong, Zhuo Deng, Fuju Rong, Chucheng Chen, and Lan Ma. Tabkanet: Tabular data modeling with kolmogorov-arnold network and transformer. *arXiv preprint arXiv:2409.08806*, 2024.

[49] Josh Gardner, Juan C Perdomo, and Ludwig Schmidt. Large scale transfer learning for tabular data via language modeling. *arXiv preprint arXiv:2406.12031*, 2024.

[50] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598, 2022.

[51] Pieter Gijsbers, Marcos LP Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. Amlb: an automl benchmark. *Journal of Machine Learning Research*, 25(101):1–65, 2024.

[52] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM computing surveys (CSUR)*, 23(1):5–48, 1991.

[53] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34:18932–18943, 2021.

[54] Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Sd4wYYOhmY.

[55] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[56] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35:507–520, 2022.

[57] Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, et al. Tabpfn-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

[58] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

[59] Sudipto Guha, Kamesh Munagala, and Peng Shi. Approximation algorithms for restless bandit problems. *Journal of the ACM (JACM)*, 58(1):1–50, 2010.

[60] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[61] Hoshin Vijai Gupta and Harald Kling. On typical range, sensitivity, and normalization of mean squared error and nash-sutcliffe efficiency type metrics. *Water Resources Research*, 47(10), 2011.

[62] Samarth Gupta, Shreyas Chaudhari, Gauri Joshi, and Osman Yağan. Multi-armed bandits with correlated arms. *IEEE Transactions on Information Theory*, 67(10): 6711–6732, 2021.

[63] Ivan Guz, nightbat, and Wendy Kan. Avito context ad clicks. https://kaggle.com/competitions/avito-context-ad-clicks, 2015. Kaggle.

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

[65] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR, 2023.

[66] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[67] Martin Hilbert. Digital technology and social change: the digital transformation of society from a historical perspective. *Dialogues in clinical neuroscience*, 22(2): 189–194, 2020.

[68] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[69] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

[70] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

[71] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

[72] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2553–2561, 2020.

[73] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

[74] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[75] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[76] AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, Ddl Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. Mistral 7b. arxiv 2023. *arXiv preprint arXiv:2310.06825*, 2024.

[77] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On bayesian upper confidence bounds for bandit problems. In *Artificial intelligence and statistics*, pages 592–600. PMLR, 2012.

[78] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

[79] Myung Jun Kim, Leo Grinsztajn, and Gael Varoquaux. CARTE: Pretraining and transfer for tabular learning. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller,

Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 23843–23866. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/kim24d.html.

[80] Ross D. King, Cao Feng, and Alistair Sutherland. Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence an International Journal*, 9(3):289–333, 1995.

[81] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[82] Tassilo Klein, Clemens Biehl, Margarida Costa, Andre Sres, Jonas Kolk, and Johannes Hoffart. SALT: Sales autocompletion linked business tables dataset. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024. URL https://openreview.net/forum?id=UZbELpkWIr.

[83] Thomas Kleine Buening, Aadirupa Saha, Christos Dimitrakakis, and Haifeng Xu. Strategic linear contextual bandits. *Advances in Neural Information Processing Systems*, 37:116638–116675, 2024.

[84] Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*, 2022.

[85] Branislav Kveton, Manzil Zaheer, Csaba Szepesvari, Lihong Li, Mohammad Ghavamzadeh, and Craig Boutilier. Randomized exploration in generalized linear bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 2066–2076. PMLR, 2020.

[86] Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. A comprehensive analysis of deep regression. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2065–2081, 2019.

[87] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[88] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[89] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[90] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

[91] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[92] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[93] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 6555–6565, 2024.

[94] Carlos García Ling, ElizabethHMGroup, FridaRim, inversion, Jaime Ferrando, Maggie, neuraloverflow, and xlsrln. H&m personalized fashion recommendations. https://kaggle.com/competitions/h-and-m-personalized-fashion-recommendations, 2022. Kaggle.

[95] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. Towards graph foundation models: A survey and beyond. *arXiv preprint arXiv:2310.11829*, 2023.

[96] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in neural information processing systems*, 35:9881–9893, 2022.

[97] Wei Min Loh, Jiaqi Shang, and Pascal Poupart. Basis transformers for multi-task tabular regression. *arXiv preprint arXiv:2506.06926*, 2025.

[98] William Loh, Sajib Kumer Sinha, Ankur Agarwal, and Pascal Poupart. A practical algorithm for feature-rich, non-stationary bandit problems. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=tRbwfej9uY.

[99] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

[100] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

[101] Simone Luetto, Fabrizio Garuti, Enver Sangineto, Lorenzo Forni, and Rita Cucchiara. One transformer for all time series: Representing and training with time-dependent heterogeneous tabular data. *arXiv preprint arXiv:2302.06375*, 2023.

[102] Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=pIZxEOZCId.

[103] Francis T Marchese. Exploring the origins of tables for information visualization. In *2011 15th International Conference on Information Visualisation*, pages 395–402. IEEE, 2011.

[104] THIBAULT Mathevet, CLAUDE Michel, Vazken Andréassian, CJIP Perrin, et al. A bounded version of the nash-sutcliffe criterion for better model assessment on large sets of basins. *IAHS PUBLICATION*, 307:211, 2006.

[105] Duncan C. McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=CjVdXey4zT.

[106] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[107] Wes McKinney et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

[108] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

[109] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

[110] Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[111] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[112] Kento Ochi and Moto Kamiura. Overtaking method based on variance of values: resolving the exploration–exploitation dilemma. *Procedia Computer Science*, 24:126–136, 2013.

[113] R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

[114] Inkit Padhi, Yair Schiff, Igor Melnyk, Mattia Rigotti, Youssef Mroueh, Pierre Dognin, Jerret Ross, Ravi Nair, and Erik Altman. Tabular transformers for modeling multivariate time series. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3565–3569. IEEE, 2021.

[115] Karl Pearson. Iv. contributions to the mathematical theory of evolution. iii. regression, heredity, and panmixia. *Proceedings of the Royal Society of London*, 59 (353-358):69–71, 1896.

[116] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[117] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

[118] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM computing surveys (CSUR)*, 51(5):1–36, 2018.

[119] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[120] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[121] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[122] Herbert Robbins. Some aspects of the sequential design of experiments. 1952.

[123] Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan E Lenssen, Yiwen Yuan, Zecheng Zhang, et al. Relbench: A benchmark for deep learning on relational databases. *Advances in Neural Information Processing Systems*, 37:21330–21341, 2024.

[124] William S Robinson. The statistical measurement of agreement. *American sociological review*, 22(1):17–25, 1957.

[125] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[126] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[127] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[128] Yoan Russac, Claire Vernade, and Olivier Cappé. Weighted linear bandits for non-stationary environments. *Advances in Neural Information Processing Systems*, 32, 2019.

[129] William Sangster. *Umbrellas and their history.* DigiCat, 2022.

[130] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[131] Murali Shanker, Michael Y Hu, and Ming S Hung. Effect of data standardization on neural network training. *Omega*, 24(4):385–397, 1996.

[132] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308, 2000.

[133] Aleksandrs Slivkins. Contextual bandits with similarity information. In *Proceedings of the 24th annual Conference On Learning Theory*, pages 679–702. JMLR Workshop and Conference Proceedings, 2011.

[134] Aleksandrs Slivkins and Eli Upfal. Adapting to a changing environment: the brownian restless bandits. In *COLT*, pages 343–354, 2008.

[135] Gowthami Somepalli, Avi Schwarzschild, Micah Goldblum, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. In *NeurIPS 2022 First Table Representation Workshop*, 2022.

[136] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[137] Stephen M Stigler. Gauss and the invention of least squares. *the Annals of Statistics*, pages 465–474, 1981.

[138] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[139] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[140] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

[141] Gábor J Székely, Maria L Rizzo, and Nail K Bakirov. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):2769–2794, 2007.

[142] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

[143] Amirsina Torfi, Rouzbeh A Shirvani, Yaser Keneshloo, Nader Tavaf, and Edward A Fox. Natural language processing advancements by deep learning: A survey. *arXiv preprint arXiv:2003.01200*, 2020.

[144] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models. arxiv 2019. *arXiv preprint arXiv:1908.08962*, 1908.

[145] Michal Valko, Nathaniel Korda, Rémi Munos, Ilias Flaounas, and Nelo Cristianini. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv:1309.6869*, 2013.

[146] Boris Van Breugel and Mihaela Van Der Schaar. Position: Why tabular foundation models should be a research priority. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 48976–48993. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/van-breugel24a.html.

[147] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[148] Guido Van Rossum et al. Python programming language. In *USENIX ATC*, 2007.

[149] Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2): 49–60, 2014.

[150] Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

[151] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[152] Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*, pages 437–448. Springer, 2005.

[153] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

[154] Ruiyu Wang, Zifeng Wang, and Jimeng Sun. Unipredict: Large language models are universal tabular classifiers. *arXiv preprint arXiv:2310.03266*, 2023.

[155] Siwei Wang, Longbo Huang, and John Lui. Restless-ucb, an efficient and low-complexity algorithm for online restless bandits. *Advances in Neural Information Processing Systems*, 33:11878–11889, 2020.

[156] Richard Weber. On the gittins index for multiarmed bandits. *The Annals of Applied Probability*, pages 1024–1033, 1992.

[157] Sanford Weisberg. Yeo-johnson power transformations. *Department of Applied Statistics, University of Minnesota. Retrieved June*, 1(2003):8, 2001.

[158] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[159] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Sklf1yrYDr.

[160] Peter Whittle. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A):287–298, 1988.

[161] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022, 2007.

[162] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. arxiv 2019. *arXiv preprint arXiv:1910.03771*, 10, 2020.

[163] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 3597–3606, 2020.

[164] Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. Lamini-lm: A diverse herd of distilled models from large-scale instructions. *CoRR*, abs/2304.14402, 2023. URL https://arxiv.org/abs/2304.14402.

[165] Qingyun Wu, Naveen Iyer, and Hongning Wang. Learning contextual bandits in a non-stationary environment. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 495–504, 2018.

[166] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

[167] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[168] Xulei Yang and Jie Ding. A computational framework for iceberg and ship discrimination: Case study on kaggle competition. *IEEE Access*, 8:82320–82327, 2020.

[169] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 269–277, 2019.

[170] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

[171] Weitong ZHANG, Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural thompson sampling. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tkAtoZkcUnm.

[172] Xiao Zhang, Ruoxi Jiang, William Gao, Rebecca Willett, and Michael Maire. Residual connections harm generative representation learning. *arXiv preprint arXiv:2404.10947*, 2024.

[173] Xiyuan Zhang, Danielle C. Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Cuixiong Hu, Huzefa Rangwala, George Karypis, and Bernie Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=t8YRsWY6HM.

[174] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pages 11492–11502. PMLR, 2020.

[175] Matt Zwolenski and Lee Weatherill. The digital universe: Rich data and the increasing value of the internet of things. *Journal of Telecommunications and the Digital Economy*, 2(3):47–1, 2014.

APPENDICES

Appendix A

Supplementary Material for Chapter 3

A.1 Generation of Correlated Arms

Suppose μ_0 has been sampled from the Uniform$(0, 1)$ distribution. Now, suppose we want to generate the reward distribution for arm i with a correlation ρ_i with μ_0. The mean parameter for arm i is generated as follows:

$$\mu_i \overset{\text{sample}}{\sim} \text{Beta}(2c(\rho_i(\mu_0 - 0.5) + 0.5), 2c(\rho_i(0.5 - \mu_0) + 0.5)) \tag{A.1}$$

where $c \geq 1$ is a hyperparameter that controls the variance of the reward distribution of the correlated arm; the higher c is, the lower the variance. In our experiments, the value of c is chosen to be 50. Examples of sampled correlated μ_i's can be found in Figure A.1.

A.2 Correlation Experiment Details

The training dataset consists of 200 random but correlated values of μ_i's, and 100 samples of arm-reward pairs with arms being uniformly sampled. ϕ is a multilayer perceptron that takes a 7-dimensional vector (one-hot encoded for each arm including the anchor arm), has a single hidden layer of size 256 and an output dimension of 2. Each set of weights is interleaved with a Softplus activation layer. The bandwidth parameter of the RBF kernel is chosen to be $\sigma = 1$.

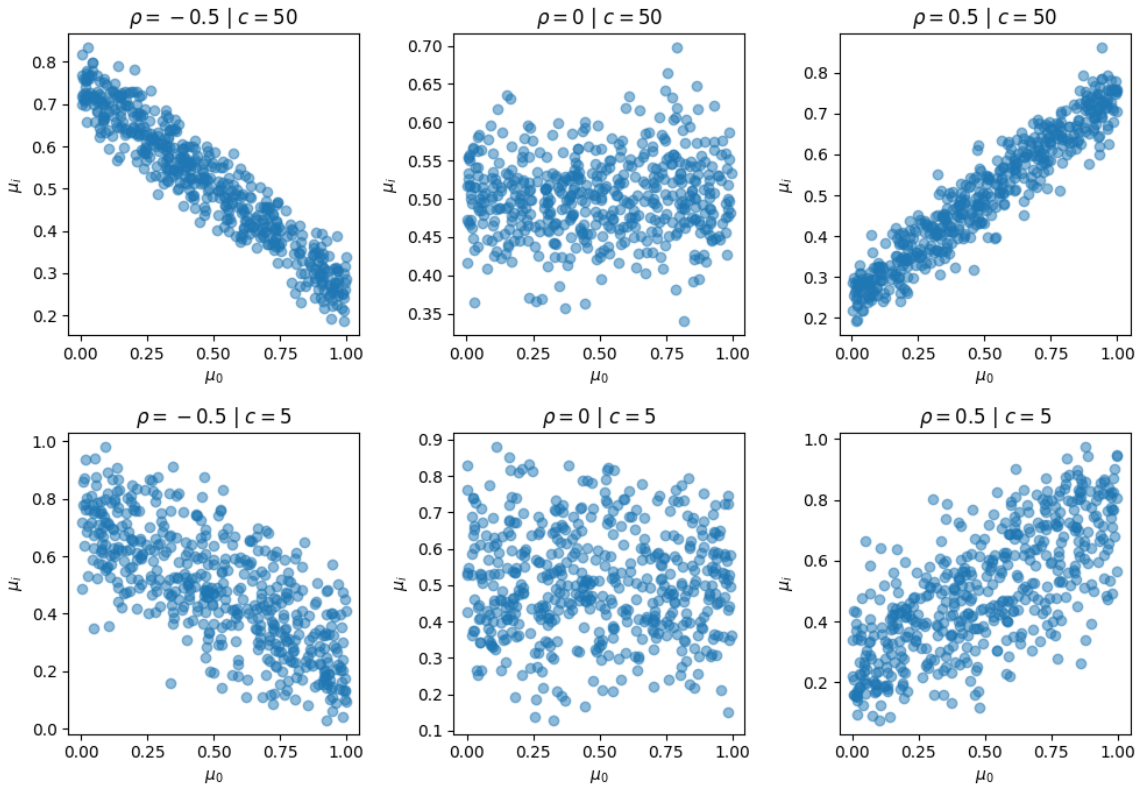


Figure A.1: Visualization of impacts of various correlation ρ and c parameters on the μ of non-anchor arms. Each subplot contains 500 random samples.

The loss function for ϕ is chosen to be $\mathcal{L}(\hat{\mu}, r) = \mathcal{L}_{\text{BCE}}(\hat{\mu}, r) + 5\mathcal{L}_{\text{ECE}}(\hat{\mu}, r)$, where the ECE loss uses 5 bins. For each of the 4 training epochs, 50% of the entire training dataset is sampled to be used for training, of which 20% will be used as the reference dataset and the remaining 80% contains the queries. The learning rate is 10^{-3} (Adam optimizer with default configurations) with an exponential decay rate of 0.99 per epoch. Since no validation dataset is used, the resultant model of the final epoch will be used.

A.3 Ablation on RBF Bandwidth in C_3

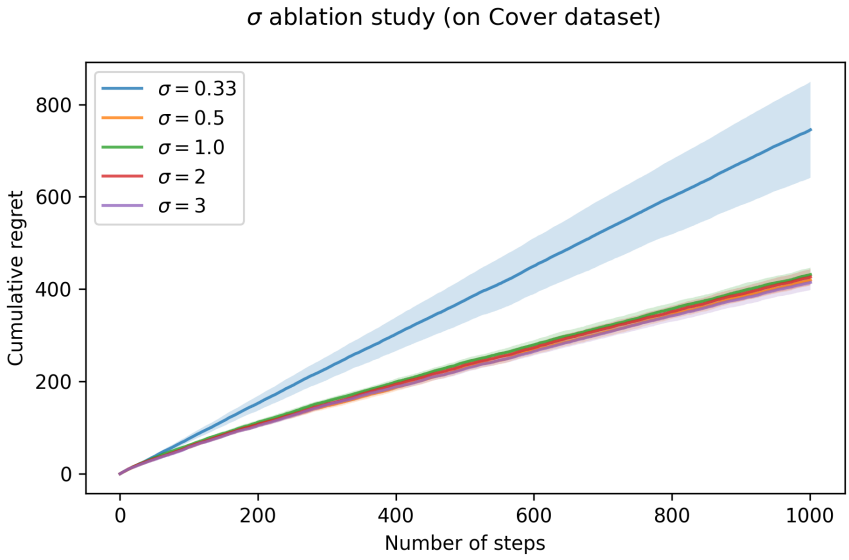


Figure A.2: Cumulative regret of C_3 with varying σ parameter (for the RBF kernel) on the Cover dataset with 1.96 sigma error bars over 10 random seeds.

To demonstrate the effects of varying the RBF kernel, we conduct an ablation study to identify: (1) possible failure cases, and (2) potential robustness across parameters. In experiments in the main paper, the RBF bandwidth σ was set to $\sigma = 1$. We vary σ by 2 and 3 times larger and smaller, leading to 4 new combinations: $0.33, 0.5, 2, 3$. There are two groups of results in Figure A.2. The obvious outlier is $\sigma = 0.33$ which essentially could not learn – a failure case. We postulate this is because the effective "search" radius is too small and did not leverage (or condition on) neighbouring points to make an estimate. The other group, the rest of the σ's, showed invariance to the choice of σ as they are statistically

identical in performance. This is an advantage as this simplifies the hyperparameter tuning process as long as σ is not too small.

A.4 MIND Experiment Details

Prior to the experiments, the titles of all news articles are converted to BERT embeddings. The specific pretrained model for BERT from Huggingface [162] is `bert-base-uncased`. The resultant vectors that summarize the entire news article are of size 768 so we used PCA to downsample to 64 dimensions and saved the embedding vectors as a file.

The test dataset provided by MIND does not include labels [163] so we could not use that split. Instead, we combine the training and validation datasets that span 7 days. These datasets contain information on article impressions and clicks, i.e. what the users see and which ones do they interact with. Since there are many of them, we decided to be selective and remove instances with nine or more articles shown and sample 20% of the entire dataset. This limits the initial amount of information given to the models and forces the evaluation to be based on incremental learning and exploration strategies.

The first three days were selected to be used as training and validation datasets – 80% training and 20% validation. This implies that reference datasets during the training of ϕ will be partitioned into $T = 3$ partitions. The remaining four days will be used as a pool of test data. Given computational constraints, especially over 10 random seeds, we pick only 500 points to be tested. However, the chosen 500 points are selected in a way that takes into account the date. Specifically, every j^{th} (fixed) entry is selected as test points such that the first and last points are close to the start of the fourth day and end of the seventh day respectively.

To demonstrate stronger concept drift so that its effect can be seen in fewer test samples, we gradually modify the click rate so that the sports category will have a higher click rate over time and all other categories will have a lower click rate in the same time span. An optimal agent should recognize this change and adapt to it.

We augment the click rates of the sports category and diminish the click rate of every other category. For augmenting, for any sports instance which is not a click, we sample a Bernoulli random variable (1 means click, 0 means no click) with probability $\Pr(X = 1) = p_i$ and assign that value to be click value. Similarly, for diminishing, for any non-sports instance which is a click, we sample a Bernoulli random variable with probability $\Pr(X = 0) = p_i$ and assign that value to be the click value. The probabilities by day are shown in Table A.1.

Table A.1: Augmenting and diminishing probability schedules by day

Day	Probability
9	0.00
10	0.00
11	0.10
12	0.15
13	0.20
14	0.25
15	0.30

ϕ was trained with the loss function $Ł(\hat{\mu}, r) = Ł_{\text{BCE}}(\hat{\mu}, r) + 0.01Ł_{\text{ECE}}(\hat{\mu}, r)$, where the ECE loss uses 5 bins. ϕ is initialized to be a multilayer perceptron with input dimension of 82 (18 news categories for context and 64 dimensional arm features), a single hidden layer of dimension 512 and an output dimension of 128. The bandwidth for the RBF kernel is chosen to be 0.6. ϕ is trained for 20 epochs with a batch size of 32. The learning rate is set to 10^{-3} (Adam optimizer with default configurations) with an exponential decay rate of 0.9. At each epoch, only 10% of the training data is used for training, where 1% of them are used as reference samples while the rest are used as query points. The ϕ chosen for testing is the one that attains the lowest validation loss.

For the Bayesian linear regression model, the α parameter, which controls the degree of exploration through the coefficient of the standard deviation of reward, is set to 1.96. λ, the parameter for numerical stability and regularization, and σ, the standard deviation of the residuals, are set to 1.

Two towers refer to the context encoder and arm encoder, which are both mappings to the same embedding space. The dot product between the context vector and arm vector represents the logits which are then passed to a softmax layer. The two towers implementation has two variants: small and large. For the small variant, the context encoder layers are $[18, 64, 32]$ and the arm encoder layers are $[64, 128, 32]$. For the large variant, the context encoder layers are $[18, 64, 64]$ and the arm encoder layers are $[64, 256, 64]$. The linear layers of both variants are interleaved with a ReLU activation layer. The small variant was trained for 5 epochs while the large variant was trained for 40 epochs. The learning rates are 8×10^{-3} (Adam optimizer with default β_1, β_2) and the batch sizes are 32. For each variant, the model chosen for testing is the one that attains the lowest validation loss.

The modifications to the linear baseline algorithms are sliding window (SW) and discounting (D). Sliding window means that they are refitted with the most recent subset

(50000 samples), while discounting uses a weighted linear regression [128] with a γ of 0.95. For the neural baselines, the online modification is changing the buffer into a sliding window (2000 samples). Note that the neural models are initially trained with the entire training size, as the sliding window buffer is only used during test time.

The Gaussian process, due to its cubic time complexity with respect to the number of samples, is restricted to a subsampling of 0.05. The forgetting element works by replacing the oldest sample with the newly seen sample, keeping the buffer the same size. The contextual restless bandit (CRB) requires heavy modification. Since the arms vary with almost no repeating arms, the state space is incredibly sparse and a transition dynamic that is unknown with minimal samples. Our problem does not have a complex budget constraint so the linear programming component is ignored.

A.5 MIND Time Plot

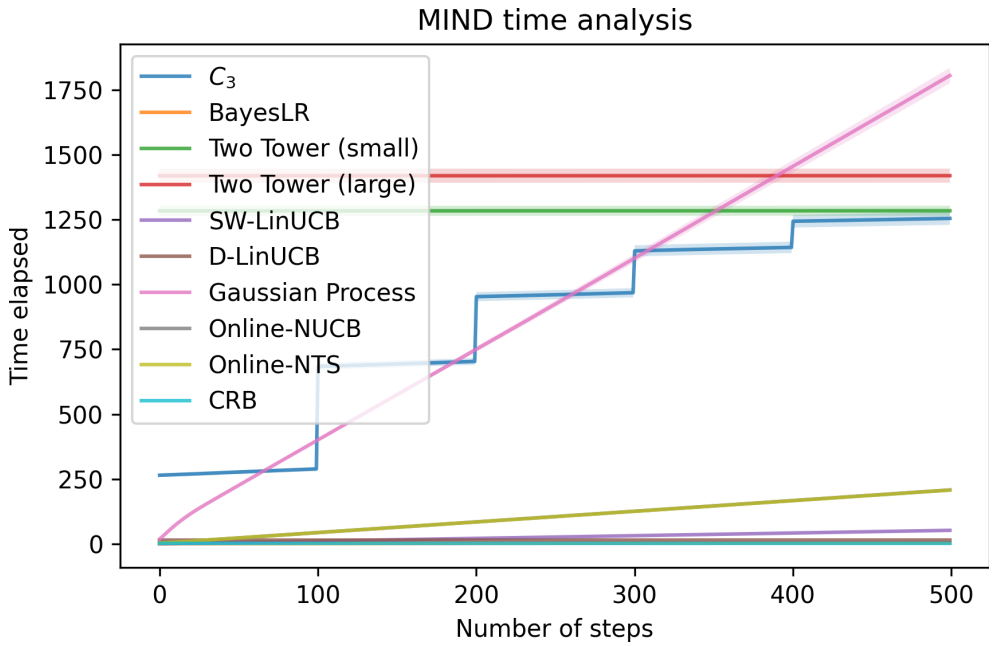


Figure A.3: Time taken in seconds against timesteps for each algorithm.

Figure A.3 shows a measure of time efficiency for each algorithm. The plot for some algorithms do not start at zero as they require some amount of warmup. The most prominent

is the two tower approaches as they are relatively large models. The stepwise pattern in C_3 is attributed to an unoptimized approach of dropping samples in its buffer. As mentioned in Section 3.5.3, 20% of its samples are dropped every 100 steps. This implementation could be improved in future work.

Appendix B

Supplementary Material for Chapter 4

B.1 Hyperparameters for Scalar versus SMR Ablation

The text encoder used is a smaller distilled BERT [37]: `bert_uncased_L-8_H-256_A-4` by Turc et al. [144] hosted on Huggingface [162]. The reason for using a smaller encoder is because we believe the semantic space across all text in the 34 datasets is relatively small. The embedding dimension of this BERT is 256. The sign-magnitude representation has 29 bits for natural number exponents (i.e. $\{2^{28}, ..., 2^0\}$), 14 bits for negative exponents (i.e. $\{2^{-1}, ..., 2^{-14}\}$) and one sign bit. This was chosen to fit the range of values in the dataset. Five random seeds were used: 0, 1, 2, 3, and 4.

These are the important hyperparameters of $\mathcal{BT}$:

- Embedding dimension: 144
- Number of blocks: 4
- Number of attention heads: 8
- Number of basis queries: 64
- $r = 6$

- Number of self-attention blocks: 9
- Batch size: 64× number of GPUs
- Dropout rate: 0.0
- Maximum norm: 1.0
- Learning rate: 1×10^{-4}
- Weight decay: 1×10^{-2}
- Exponential decay of learning rate multiplier: 0.985
- Adaptive loss reweighing factor $\gamma = 0.2$

The optimizer used is the AdamW optimizer with default β values. The loss function is binary cross entropy.

B.2 Hyperparameters for Adaptive Loss Reweighing Ablation

To show the efficacy of the reweighing scheme on a smaller scale experiment, we pick three datasets: abalone, concrete, and white wine. We used a small $\mathcal{BT}$ model:

- Embedding dimension: 32
- Number of blocks: 1
- Number of attention heads: 8
- Number of basis queries: 96
- $r = 2$
- Number of self-attention blocks: 5
- Batch size: 128× number of GPUs
- Dropout rate: 0.4

- Maximum norm: 1.0
- Learning rate: 2×10^{-4}
- Weight decay: 5×10^{-2}
- Exponential decay of learning rate multiplier: 0.99

The number of parameters is 422,037. The sign-magnitude representation has 14 bits for natural number exponents (i.e. $\{2^{13}, ..., 2^0\}$), 6 bits for negative exponents (i.e. $\{2^{-1}, ..., 2^{-6}\}$) and one sign bit. We also used a smaller BERT model (bert_uncased_L-12_H-128_A-2) with an embedding dimension of 128.

B.3 Main Experiment Hyperparameters and Configurations

B.3.1 Datasets

One dataset (Geographical Origin of Music) was removed because the baselines cannot efficiently handle the large number of columns (68). The size of each of the 34 datasets varies considerably. We want to ensure that the evaluation (validation and test splits) is consistent throughout, and also subject the models to some degree of data imbalance – a highly common issue for tabular modality. The test and validation set size is 20% of the number of rows of the smallest dataset, and is randomly sampled; the training set size is the remaining rows.

B.3.2 Training Routine

We do not use the notion of epochs because there are too many samples. Instead, we introduce the notion of *strides*: the training gradient steps before evaluating. There are two quantities of interest: the number of strides (essentially the number of validation passes) and the stride size (number of gradient steps). All models in the multi-task experiment use 200 strides, and a stride size of 200.

B.3.3 Basis Transformer

The text encoder used is a smaller distilled BERT [37]: `bert_uncased_L-8_H-256_A-4` by Turc et al. [144] hosted on Huggingface [162]. The reason for using a smaller encoder is because we believe the semantic space across all text in the 34 datasets is relatively small. The embedding dimension of this BERT is 256. The sign-magnitude representation has 29 bits for natural number exponents (i.e. $\{2^{28}, ..., 2^0\}$), 14 bits for negative exponents (i.e. $\{2^{-1}, ..., 2^{-14}\}$) and one sign bit. This was chosen to fit the range of values in the dataset. Five random seeds were used: 0, 1, 2, 3, and 4.

These are the important hyperparameters of $\mathcal{BT}$:

- Embedding dimension: 144
- Number of blocks: 4
- Number of attention heads: 8
- Number of basis queries: 64
- $r = 6$
- Number of self-attention blocks: 9
- Batch size: 64× number of GPUs
- Dropout rate: 0.0
- Maximum norm: 1.0
- Learning rate: 1×10^{-4}
- Weight decay: 1×10^{-2}
- Exponential decay of learning rate multiplier: 0.985
- Adaptive loss reweighing factor $\gamma = 0.2$

The optimizer used is the AdamW optimizer with default β values. The loss function is binary cross entropy.

B.3.4 LLMs with TabLLM Routines

LLMs face memory (VRAM) issues due to the attention operation across the entire serialized row. As a result, a smaller batch size and gradient accumulation need to be used. Furthermore, since LLMs are pretrained, to be fair to $\mathcal{BT}$, we reduce the batch size but keep the number of gradient steps the same. LLMs would have already had a good representation learned so it would not need to see that many samples but it still has the same number of finetuning steps as $\mathcal{BT}$. The finetuning processes using the Huggingface [162] interface which offers a convenient method to handle sequence-to-sequence and causal language models.

In general, the batch size is 16× the number of GPUs, which is often achieved with gradient accumulation. However, the T5 variants consume relatively much more memory than the other LLMs and could only have a batch size of 8× the number of GPUs. The learning rates and other related hyperparameters are the default or recommended ones listed in Huggingface or their respective papers.

Tabular data often contains a lot of numeric data, and this is definitely true in the OpenML-CTR23 benchmark [45]. If we naively serialize the data, most LLMs would not be able to fit a batch of data into the GPU memory buffer due to the many digits in numeric entry values. To solve this issue while retaining as much relevant information, we round numbers to 5 significant digits. Numbers with less than 5 significant digits are not affected.

Recall that not all LLMs can successfully produce a prediction that fits the template. In cases like this, we compute the scores excluding those failure cases – in other words, we do not penalize LLMs for such failures. However, in enterprise or important settings, this might be considered a major failure.

B.4 Invalid Baselines and Justifications

There are several possible baseline algorithms that fail to meet one or more criteria. We will first discuss the three special cases, followed by the general cases that most other works fall under.

B.4.1 Special Cases

Qwen2.5 [167] is a fairly recent LLM that is competitive with many state-of-the-art LLMs. Our experiments with Qwen showed an unprecedented superior performance – some datasets reached above 0.8 validation R^2 score when trained on less than 8% of training data. After training on 17% of the samples, it has almost perfect predictions (close to 0 mean squared error) on 5 tasks (validation set). Upon inspection, their technical report mentioned that they trained with tables as part of their efforts to improve support to a larger domain. This confirms our suspicion that Qwen, unfortunately, could not be used as a baseline algorithm due to data contamination. This is also the case for TinyLlama [170] which was pretrained on The Stack dataset [84] which uses CSV data and attained impossibly high validation score upon being finetuned on small amounts of training data. It attained perfect predictions on one task (validation set) after training on 3% of the samples. By 12% of the training samples, it has attained perfect predictions on 6 tasks (validation set).

Tabula-8B [49] follows a very similar direction as TabLLM [65]. The primary difference is TABULA-8B was pretrained on over 2.1 billion rows of tabular data [49], while the models in TabLLM were only pretrained on language tasks. The OpenML CTR-23 dataset [45] is a very popular and common tabular dataset. In Table 1 of Gardner et al. [49]'s paper, it specifically mentions that 34.2% of OpenML-CTR23 datasets has been potentially leaked. To avoid data contamination, this is the main reason TABULA-8B was not included as a baseline algorithm.

CARTE [79] has a novel design that views a row of a table as a graphlet. We were eager to compare our works with CARTE, but we faced serious implementation difficulties when applying it to a multi-task regression setting as a result of their design. Unlike other works that use LLMs for tabular tasks, numeric values are represented by taking the embedding of its corresponding column name embedding and scaling it by the scalar value. In the paper, they mentioned that numeric values are preprocessed beforehand using a power transform. The multi-task experiments require a model to be fitted with multiple datasets simultaneously, implying inconsistent ranges of target values. In addition to power transform not being a suitable preprocessing step, we faced floating point overflow issues when running multi-task experiments with a wide range of targets on the CARTE code base. When removing the power transformation preprocessing altogether, the training process begins but is plagued with NaNs due to very large loss values from datasets such as wave_energy. The issue is not rectified even when dividing the value of all targets by a large constant. At that point, we decided not to go forward with more invasive modifications of their works at the risk of overriding the core of their works.

B.4.2 General Cases

There are four categories of design deficits in terms of requirements for the multi-task experiments.

(A) Inability to handle free-form textual data in entries

(B) Inability to handle heterogeneous rows (inconsistent columns)

(C) Requires additional specialized information for each dataset

(D) No complete, reproducible code

(A) is a common issue with many tabular models as many assume tabular data are mostly numeric. The OpenML CTR-23 dataset [45], however, contains tables that have more than 50% textual columns. (B) This is a key requirement for multi-task learning. Many models are adapted to handle only fixed-size inputs, effectively ruling out being a multi-task or foundation model as they cannot generalize to multiple datasets. (C) While this is generally not an issue in tabular tasks, in the case of multi-task learning, this is highly restrictive. If a model requires auxiliary information such as the set of categorical features for certain columns, or a standard scaler for numeric values of certain columns, it diminishes the capacity of a multi-task or zero-shot model. (D) Without complete code provided by authors of baseline algorithms, it would be impossible to implement and experiment with the exact competitive configuration demonstrated by the authors.

The summary of invalid baseline algorithms is shown in Table B.1. Unlike Table 4.2, Table B.1 is not about fulfilling desiderata but rather a concrete inability to implement for our experiments.

Table B.1: Summary of invalid baseline algorithms and their justifications.

Algorithm	(A)	(B)	(C)	(D)
Gradient boosting decision trees (e.g. XGBoost [27])	✓	✓		
TabNet [5]		✓	✓	
NODE [117]	✓	✓		
TabPFN [69]	✓	✓		
TabTransformer [73]	✓	✓		
TabKANet [48]	✓	✓	✓	
TabBert [114]	✓		✓	
UniTTab [101]	✓		✓	✓
UniPredict [154]				✓
FT-Transformer [53]	✓		✓	
SAINT [135]	✓		✓	
Tabular-Text Transformer [22]			✓	

B.5 Complete Multi-Task Results

B.5.1 Computation of Scores

Recall that all experiments are done over 5 random seeds. Each experiment consists of 34 datasets and 8 algorithms. RMSE and MMS-RMSE are computed using the mean over random seeds. Central tendencies and spreads in Table 4.3 are effectively statistics of the mean scores. That means the reported statistics are in fact median of means and IQR of means. For brevity and clarity, we treat the mean over random seeds as the actual score.

However, in the following subsections, we expand to show the means of both metrics over the random seeds.

B.5.2 Scores by Dataset and Algorithm

In Table B.2, for some algorithms and for some datasets, the performance exceeds a range that makes it difficult to include in a compact table. As such, we denote mean MMS-RMSE score worse than 100 by `Poor`. The tables have their dataset names truncated to 8 characters.

Table B.2: Mean MMS-RMSE over 5 random seeds by dataset and algorithm.

	Ours	Pythia410m	Cerebras590m	FlanT5-base	LaMiniT5-223M	BART-large
abalone	0.25	0.32	0.66	0.26	0.22	0.71
airfoil	1.02	4.07	0.55	0.33	0.28	0.29
auction	0.14	0.35	0.26	0.32	0.29	0.34
brazilia	0.18	0.36	0.30	0.25	0.20	0.27
californ	0.33	0.49	0.38	0.27	0.39	0.73
cars	0.11	0.60	0.44	0.47	1.19	1.60
concrete	0.27	0.55	0.35	0.60	0.49	0.57
cps88wag	0.28	0.37	0.20	0.23	0.22	0.25
cpu acti	0.06	0.76	0.82	0.76	0.19	0.87
diamonds	0.14	0.37	0.21	0.13	0.16	0.36
energy e	0.09	0.73	0.47	0.66	0.76	0.71
fifa	0.10	0.21	0.17	0.19	0.17	7.52
forest f	0.20	0.22	0.20	0.20	0.20	0.20
fps benc	0.09	0.63	0.24	0.36	0.30	0.41
grid sta	0.19	2.11	0.17	0.34	0.28	0.43
health i	0.41	0.60	0.40	0.52	0.42	0.65
kin8nm	0.49	0.34	0.25	0.26	0.25	0.27
kings co	0.17	0.44	0.14	0.24	0.24	0.66
miami ho	0.17	0.36	0.19	0.24	0.24	1.43
moneybal	0.17	1.53	0.34	0.95	0.83	12.73
naval pr	0.22	5.35	0.17	0.35	0.36	6.15
physioch	0.47	0.46	0.34	0.47	0.47	0.53
pumadyn3	0.17	6.49	0.21	0.25	0.25	0.32
qsar fis	0.29	0.40	0.43	0.43	0.20	0.61
red wine	0.26	1.03	1.29	0.26	0.26	0.65
sarcos	0.13	0.23	0.10	0.33	0.19	0.25
socmob	0.12	0.20	0.17	0.20	0.20	0.21
solar fl	0.21	0.23	0.21	0.21	0.21	0.25
space ga	0.28	1.09	1.12	0.21	0.23	Poor
student	0.23	1.40	0.71	0.74	0.51	0.79
supercon	0.19	0.43	0.39	0.19	0.26	0.43
video tr	0.08	0.19	0.09	0.16	0.12	0.13
wave ene	0.25	9.40	1.41	0.20	0.15	6.52
white wi	0.24	0.97	1.12	0.22	0.22	0.58

Table B.3: Mean RMSE over 5 random seeds by dataset and algorithm.

	Ours	Pythia410m	Cerebras590m	FlanT5-base	LaMiniT5-223M	BART-large
abalone	4.1e+00	5.4e+00	1.1e+01	4.3e+00	3.7e+00	1.2e+01
airfoil	3.0e+01	1.2e+02	1.6e+01	9.9e+00	8.5e+00	8.6e+00
auction	5.6e+03	1.4e+04	1.1e+04	1.3e+04	1.2e+04	1.4e+04
brazilia	2.8e+03	5.7e+03	4.7e+03	3.9e+03	3.2e+03	4.3e+03
californ	1.5e+05	2.2e+05	1.7e+05	1.2e+05	1.7e+05	3.3e+05
cars	4.4e+03	2.2e+04	1.7e+04	1.8e+04	3.8e+04	6.1e+04
concrete	1.7e+01	3.4e+01	2.2e+01	3.8e+01	3.1e+01	3.6e+01
cps88wag	5.5e+02	7.3e+02	4.1e+02	4.5e+02	4.3e+02	4.9e+02
cpu acti	6.1e+00	7.5e+01	8.1e+01	7.5e+01	1.9e+01	8.6e+01
diamonds	2.0e+03	5.2e+03	3.0e+03	1.9e+03	2.3e+03	5.1e+03
energy e	3.3e+00	2.6e+01	1.7e+01	2.4e+01	2.7e+01	2.6e+01
fifa	1.7e+04	3.5e+04	2.9e+04	3.2e+04	2.9e+04	1.1e+06
forest f	3.6e+01	4.0e+01	3.6e+01	3.6e+01	3.6e+01	3.6e+01
fps benc	1.9e+01	1.4e+02	5.2e+01	8.0e+01	6.6e+01	8.9e+01
grid sta	3.0e-02	2.9e-01	2.0e-02	5.0e-02	4.0e-02	6.0e-02
health i	2.0e+01	3.0e+01	2.0e+01	2.6e+01	2.1e+01	3.3e+01
kin8nm	5.5e-01	3.7e-01	2.8e-01	2.8e-01	2.7e-01	3.0e-01
kings co	2.2e+05	5.5e+05	1.7e+05	3.0e+05	3.0e+05	8.5e+05
miami ho	2.5e+05	5.2e+05	2.7e+05	3.5e+05	3.4e+05	2.0e+06
moneybal	8.2e+01	7.3e+02	1.6e+02	4.5e+02	4.1e+02	6.1e+03
naval pr	1.0e-02	2.7e-01	1.0e-02	2.0e-02	2.0e-02	3.0e-01
physioch	9.7e+00	9.4e+00	6.9e+00	9.5e+00	9.5e+00	1.1e+01
pumadyn3	3.0e-02	1.2e+00	4.0e-02	4.0e-02	4.0e-02	6.0e-02
qsar fis	2.2e+00	3.0e+00	3.2e+00	3.2e+00	1.5e+00	4.5e+00
red wine	9.2e-01	3.7e+00	4.5e+00	9.3e-01	9.3e-01	2.2e+00
sarcos	1.2e+01	2.1e+01	8.8e+00	3.1e+01	1.7e+01	2.3e+01
socmob	2.6e+01	4.4e+01	3.6e+01	4.5e+01	4.4e+01	4.5e+01
solar fl	1.1e+00	1.1e+00	1.1e+00	1.1e+00	1.1e+00	1.2e+00
space ga	2.4e-01	9.4e-01	9.9e-01	1.8e-01	2.0e-01	4.2e+04
student	3.8e+00	2.2e+01	1.2e+01	1.2e+01	8.2e+00	1.3e+01
supercon	2.4e+01	5.2e+01	4.8e+01	2.3e+01	3.2e+01	5.3e+01
video tr	1.2e+01	2.6e+01	1.1e+01	2.2e+01	1.9e+01	1.9e+01
wave ene	1.4e+05	5.6e+06	8.0e+05	1.2e+05	8.9e+04	3.8e+06
white wi	9.3e-01	3.6e+00	4.4e+00	8.7e-01	8.6e-01	2.3e+00

Table B.4: Mean success rate over 5 random seeds by dataset and algorithm.

	Ours	Py410m	Ce590m	FlanT5	LaMiniT5	BART
abalone	1.00	0.86	0.15	1.00	1.00	1.00
airfoil	1.00	1.00	1.00	1.00	1.00	1.00
auction	1.00	1.00	0.96	1.00	1.00	1.00
brazilia	1.00	1.00	1.00	1.00	1.00	1.00
californ	1.00	1.00	1.00	1.00	1.00	1.00
cars	1.00	1.00	0.94	1.00	1.00	1.00
concrete	1.00	1.00	1.00	1.00	1.00	1.00
cps88wag	1.00	1.00	1.00	1.00	1.00	0.87
cpu acti	1.00	1.00	0.99	1.00	1.00	1.00
diamonds	1.00	1.00	0.99	1.00	1.00	1.00
energy e	1.00	1.00	1.00	1.00	1.00	1.00
fifa	1.00	1.00	1.00	1.00	1.00	1.00
forest f	1.00	1.00	1.00	1.00	1.00	1.00
fps benc	1.00	0.74	0.99	1.00	1.00	0.97
grid sta	1.00	1.00	1.00	1.00	1.00	1.00
health i	1.00	0.48	0.96	1.00	1.00	1.00
kin8nm	1.00	1.00	1.00	1.00	1.00	1.00
kings co	1.00	1.00	1.00	1.00	1.00	1.00
miami ho	1.00	1.00	1.00	1.00	1.00	1.00
Moneybal	1.00	1.00	1.00	1.00	1.00	1.00
naval pr	1.00	1.00	1.00	1.00	1.00	0.77
physioch	1.00	1.00	0.99	1.00	1.00	1.00
pumadyn3	1.00	1.00	1.00	1.00	1.00	1.00
QSAR fis	1.00	1.00	0.97	1.00	1.00	1.00
red wine	1.00	1.00	0.98	1.00	1.00	0.82
sarcos	1.00	1.00	1.00	1.00	1.00	1.00
socmob	1.00	1.00	1.00	1.00	1.00	0.92
solar fl	1.00	1.00	0.63	1.00	1.00	1.00
space ga	1.00	1.00	1.00	1.00	1.00	1.00
student	1.00	0.62	0.16	1.00	1.00	1.00
supercon	1.00	1.00	0.74	1.00	1.00	1.00
video tr	1.00	0.70	1.00	1.00	1.00	1.00
wave ene	1.00	1.00	1.00	1.00	1.00	1.00
white wi	1.00	1.00	1.00	1.00	1.00	0.87
mean	1.00	0.95	0.92	1.00	1.00	0.98

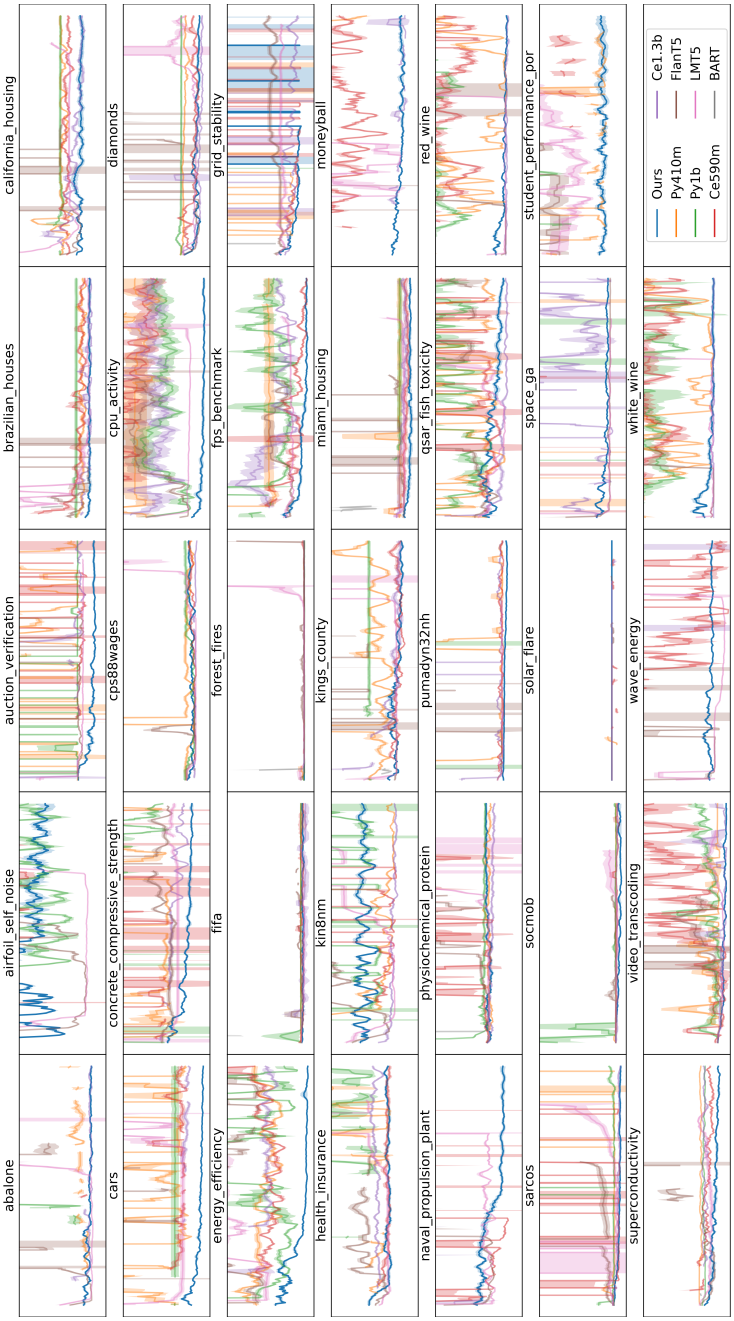


Figure B.1: An enlarged view of validation MMS-RMSE scores over gradient steps per dataset. The shaded area represents the 1 standard deviation across 5 random seeds from the mean. The range of y-axis is [0, 1] and the range of x-axis is [0, 200]. The blue line is from $\mathcal{BT}$. Lower values are better.

Table B.5: Run times for each experiment including setup, training, and evaluations.

Ours	Py410m	Ce590m	FlanT5	LMT5	BART
21h 35m	13h 23m	36h 49m	21h 55m	21h 00m	17h 19m

Note that in $\mathcal{BT}$, we do not cache the BERT text encoders. We expect better runtimes if we were to cache BERT vectors before running the experiments because our bottleneck is calling the BERT model and padding the input tensors every mini-batch.

B.6 Table Clustering

The goal is to extract a cluster of tables that are linked in their topics in a scalable fashion. The T4 dataset [49] consists of around 4 million unique tables, making manual selection almost impossible. The URL to the T4 dataset is https://huggingface.co/datasets/mlfoundations/t4-full. To solve this, we employ the help of LLMs. The steps are outlined as follows:

1. Basic filtering – we remove tables that do not have any numeric columns, no suitable target columns or contain very long column names
2. Load `Mistral-7B-Instruct-v0.3` [76]
3. Topic prompting – Fed in only the column names of the first chunk of data, prompting `Mistral-7B` to identify the topic of each table
4. Load `paraphrase-MiniLM-L6-v2` [121]
5. Topic embedding – Embed the obtained topics with `paraphase-MiniLM`
6. Compute pairwise cosine similarity
7. Set a cosine similarity threshold of 0.9 and select table pairs that fulfill the threshold
8. Apply K-Means clustering
9. Manually extract cluster and corresponding tables

18 tables that fall under the general theme of socioeconomic/demographic data were extracted. We ensured that every table has between 500 and 5000 rows, between 3 and 25 columns and at least 40% text-based columns for diversity. The train, validation and test splits are 50:25:25. All tables share a common column "age" which is set as the target column. The input columns are different for each table.

The directory of the tables is `chunk-0000` of the T4 dataset. The file names of the 18 tables are listed below:

Table B.6: File names of the 18 tables from T4.

Table name
3ygAL4aTumFSQDjkTGthqYeZeYcLtCkMeFxzqnz6OGk=__8f618902-e1a7-11ee-ab82-0a94effb1a4d.parquet
3ygAL4aTumFSQDjkTGthqYeZeYcLtCkMeFxzqnz6OGk=__e294bce8-e198-11ee-a82a-0a94efb504bd.parquet
-QNqDbw_IyA2vTYkwofja5ZUEGIikjSLwUBc3LiZcXc=__90a5e60a-e1a7-11ee-ab82-0a94effb1a4d.parquet
-QNqDbw_IyA2vTYkwofja5ZUEGIikjSLwUBc3LiZcXc=__e3de454c-e198-11ee-a82a-0a94efb504bd.parquet
5Rdowy95hJN-nj4i5gJ9Kj_hIQX-nwKcgmAPGc_5xIo=__504af19e-e199-11ee-8870-0a94efb504bd.parquet
5Rdowy95hJN-nj4i5gJ9Kj_hIQX-nwKcgmAPGc_5xIo=__521b2142-e199-11ee-8870-0a94efb504bd.parquet
C1o1fCr8jnjj5d_gt5oFYp9EOTTKIURdhX613MicyII=__5280c136-e19a-11ee-b65a-0a94efb504bd.parquet
C1o1fCr8jnjj5d_gt5oFYp9EOTTKIURdhX613MicyII=__cb23a73a-e1a8-11ee-892a-0a94effb1a4d.parquet
EszmLxWUbGTsp76vXQJ7MtCJHmUN3Nx8jC_swSDI-6g=__5c45271c-e1a8-11ee-94d6-0a94effb1a4d.parquet
EszmLxWUbGTsp76vXQJ7MtCJHmUN3Nx8jC_swSDI-6g=__e3eb23a6-e199-11ee-86ad-0a94efb504bd.parquet
b4ApPeM1bmXXkQbGDXY9NJFLR8WqPBwOYCV7xLXQlpQ=__275b23ea-e1a7-11ee-884e-0a94effb1a4d.parquet
b4ApPeM1bmXXkQbGDXY9NJFLR8WqPBwOYCV7xLXQlpQ=__a2227e98-e198-11ee-a4d0-0a94efb504bd.parquet
nuul9kC0qomBqF_Pt_g01GqAB2Nrs05tXclUdXpv1JE=__548445e4-e199-11ee-b11d-0a94efb504bd.parquet
nuul9kC0qomBqF_Pt_g01GqAB2Nrs05tXclUdXpv1JE=__cd0c1894-e1a7-11ee-b194-0a94effb1a4d.parquet
ed_eyWM0UyLwcfa5Yj8gvS1meFmQnDheJkkehMr6t2s=__19d3599e-e1a8-11ee-ac2c-0a94effb1a4d.parquet
ed_eyWM0UyLwcfa5Yj8gvS1meFmQnDheJkkehMr6t2s=__a368c842-e199-11ee-b7aa-0a94efb504bd.parquet
gtHK4IIt0si_lh-VhHV8uZdPig6zYVNmqVdoX_dNMLU=__8eb282ae-e1a7-11ee-ab82-0a94effb1a4d.parquet
gtHK4IIt0si_lh-VhHV8uZdPig6zYVNmqVdoX_dNMLU=__e1e00a78-e198-11ee-a82a-0a94efb504bd.parquet

Appendix C

Supplementary Material for Chapter 5

C.1 RelBench Dataset Processing

RelBench contains multiple separate databases from difference sources such as Stack Exchange Data Dump and Amazon Review Data Dump. Those are processed and deanonymized production databases with plenty of information.

The databases from the following sources [123]:

- Amazon [71]
- H&M [94]
- Event recommendation [25]
- Avito [63]
- Sales Autocompletion Linked Business Tables (SALT) [82]
- RateBeer [123]

Each of these sources contains multiple massive tables, typically over millions of rows of data. Some sources were omitted because they do not have a regression task. Tables from the sources were extracted via the RelBench API and reorganized into a Pandas

dataframe [107]. As the name suggests, relational database operations are used to merge tables into cohesive units as methods in this chapter do not model the relational nature of the databases.

The tables are preprocessing as follows. Bytes columns are dropped and datatime objects are replaced with their corresponding ISO date time format. Rows where all entries have `NaN`s are dropped as they are nonsensical in the context of evaluation. Similarly, any row whose target column is `NaN` is removed as a score cannot be computed.

The tables are randomly subsampled to contain at most 62,500 rows of data. Given that the training-validation-test split is 80:10:10, each table having 62,500 implies that the training split is 50,000. This is not an arbitrary value – it is the official limit for TabPFN-2.5 [57]. Furthermore, these datasets tend to be very wide in the number of columns so the computational resources cannot effectively cannot handle the millions of rows in general.

C.2 Evaluation Task Configurations

This appendix provides the full configuration details for the two downstream evaluations reported in the main text: imputation task and standard regression task.

C.2.1 Imputation Task

Data For each dataset the following preprocessing is applied:

1. The `Unnamed: 0` index column, if present, is dropped.
2. All numeric columns are cast to `float`
3. Up to `n_masks` $= 5$ numeric columns are sampled uniformly (without replacement) as imputation targets

Per-target inference loop For each selected target column c:

1. Remaining NaNs in c are dropped from the ground-truth slice
2. Root-mean-squared error (RMSE) is computed against the ground truth

Configuration Given the zero-shot or one-shot nature of this task, there are not many configurations. $\mathcal{PBT}$ performs a direct zero-shot inference. Baselines do not provide a flexible implementation for zero-shot inference are:

1. Mitra
2. TabDPT
3. TabPFN-2.5
4. All non-neural models

Instead, they were provided one random row (not from the test set) to condition the prediction on. The seeds used are: 0, 1, 2, 3, 4. The hyperparameters of baseline algorithms are summarized in Table C.1.

Compute Each dataset run is launched as a single SLURM job with the resource request: 1 node, 1 NVIDIA A40 GPU, 8 CPU cores, 64 GB RAM.

C.2.2 Standard Regression Task

Data Each dataset is processed independently as follows:

1. Cast numeric columns to `float` and drop rows whose target (the last column) is NaN
2. Split into train/val/test with the ratios 0.8/0.1/0.1 with a global seed
3. Replace the target column in the test frame with zeros so that models never observes the ground truth at inference time

$\mathcal{PBT}$ fine-tuning The finetuning process involves the following:

1. Load the pretrained $\mathcal{PBT}$ model
2. Replace the last linear layer with a low rank bottleneck, as outlined in Section 5.5.2

3. Run `num_finetuning_epoch` $= 1$ epoch of supervised fine-tuning on the training partition with the BCE loss on the signed-magnitude binary target representation, using AdamW ($\eta = 8 \times 10^{-5}$, weight decay 10^{-5}) under `bfloat16` mixed precision via HuggingFace `Accelerate`

4. Switches to evaluation mode and extracts the latents for the train, val, and test partitions.

5. Concatenates each row's $\mathcal{PBT}$ latent with the original numeric features of that row (excluding the target column) before handing them to XGBoost

XGBoost head and hyperparameter optimization. A GPU-accelerated XGBoost regressor (`tree_method="hist"`, `device="cuda"`, objective `"reg:squarederror"`) is wrapped in a TPE search via `hyperopt.fmin` [13], with the validation RMSE as the objective. The search runs until `max_hyperopt_time` $= 900\,\mathrm{s}$ (15 minutes per dataset) elapses. The best validation model is then evaluated on the test partition. The search space is given in Table C.2.

As for the baselines, they are also given 900 seconds of search time. The hyperparameter spaces of each baseline are found in Table C.3. CARTE is evaluated without additional hyperparameter search. TabPFN undergoes hyperparameter search via the `TunedTabPFNRegressor` interface, which does not reveal the hyperparameter space.

Configuration. The full configuration for $\mathcal{PBT}$ is summarized in Table C.4. The hyperparameters of baseline algorithms are summarized in Table C.5.

Compute. Each array element is a single SLURM job requesting: 1 node, 1 NVIDIA A40 GPU, 8 CPU cores, and 64 GB RAM.

Table C.1: Baseline hyperparameters for imputation tasks. Hyperparameters omitted from this table use library defaults.

Method	Hyperparameter	Value
Linear regression (LR)	imputer fitting set	test
MLP	imputer fitting set	test
	hidden layer sizes	$[200, 100]$
	shuffle	true
	max iterations	200
XGBoost	# estimators	200
	max depth	12
CatBoost	iterations	200
	depth	12
TabPFN	# estimators	1
Mitra	patience	5
	fine-tune steps	100
	# estimators	1
	warmup steps	0
TabM	default by TabM authors	
TabDPT	normaliser	quantile-uniform
	clip σ	4.0
	missing indicators	false
	# ensembles	1
	context size	2048
TabSTAR	max epochs	4
	global batch	128

Table C.2: $\mathcal{PBT}$'s XGBoost head hyperparameter search space for the standard regression

Hyperparameter	**Distribution**	**Range / Step**
`max_depth`	quantized uniform (int)	$[4, 8]$, step 1
`n_estimators`	quantized uniform (int)	$[2000, 6000]$, step 100
`eta`	uniform	$[0.01, 0.05]$
`reg_lambda`	uniform	$[0, 1]$
`gamma`	uniform	$[0, 1]$
`colsample_bytree`	uniform	$[0.2, 0.6]$

Table C.3: Baseline hyperparameter optimization search spaces.

Hyperparameter	**Distribution**	**Range / choices**
XGBoost		
`max_depth`	quantised uniform (int)	[4, 18], step 2
`reg_lambda`	uniform	[0, 4]
`n_estimators`	categorical	{100, 200, 300}
CatBoost		
`max_depth`	quantised uniform (int)	[4, 16], step 2
`reg_lambda`	uniform	[0, 4]
`n_estimators`	categorical	{100, 200, 300}
Linear regression		
`alpha`	uniform	[0, 5]
MLP		
`hidden_layer_sizes`	categorical	{[100], [200], [300]}
`alpha`	uniform	$[10^{-5}, 10^{-3}]$
`max_iter`	quantised uniform (int)	[50, 250], step 5
Mitra		
`lr`	uniform	$[10^{-5}, 10^{-3}]$
`warmup_steps`	categorical	{750, 1000, 1250}
`metric`	categorical	{mse, mae, rmse}
TabDPT		
`clip_sigma`	uniform	[1, 10]
`feature_reduction`	categorical	{pca, subsample}
TabSTAR		
`lora_lr`	uniform	$[10^{-5}, 10^{-3}]$
`lora_alpha`	uniform	[1, 3]
`lora_dropout`	uniform	[0.01, 0.2]
TabM		
`k`	quantised uniform (int)	[16, 32], step 4
`n_blocks`	quantised uniform (int)	[1, 5], step 1

Table C.4: Standard regression configuration

Field	**Value**
Splits (per dataset)	
`train_ratio`	0.8
`val_ratio`	0.1
`test_ratio`	0.1
Finetuning	
`batch_size`	32
`reduced_rank`	512
`num_finetuning_epoch`	1
`max_hyperopt_time`	900 s
Finetuning optimizer	
Optimizer	AdamW
Learning rate	8×10^{-5}
Weight decay	10^{-5}
Mixed precision	`bfloat16` via `Accelerate`
Loss	BCE on signed-magnitude binary targets
Reproducibility	
`seeds`	[0, 1, 2, 3, 4]

Table C.5: Baseline hyperparameters for standard regression tasks. Hyperparameters omitted from this table use library defaults.

Method	Hyperparameter	Value
Linear regression (LR)	imputer fitting set	train
MLP	imputer fitting set	train
	hidden layer sizes	[100]
	shuffle	true
	max iterations	200
XGBoost	# estimators	200
	max depth	12
CatBoost	iterations	200
	depth	12
TabPFN	# estimators	1
Mitra	patience	5
	fine-tune steps	100
	# estimators	1
	warmup steps	0
TabM	default by TabM authors	
TabDPT	normaliser	quantile-uniform
	clip σ	4.0
	missing indicators	false
	# ensembles	1
	context size	2048
TabSTAR	max epochs	15
	global batch	128
CARTE	scoring	squared error
	# models	10
	batch size	64
	max epoch	15
	early stopping patience	5
	parallel jobs	5