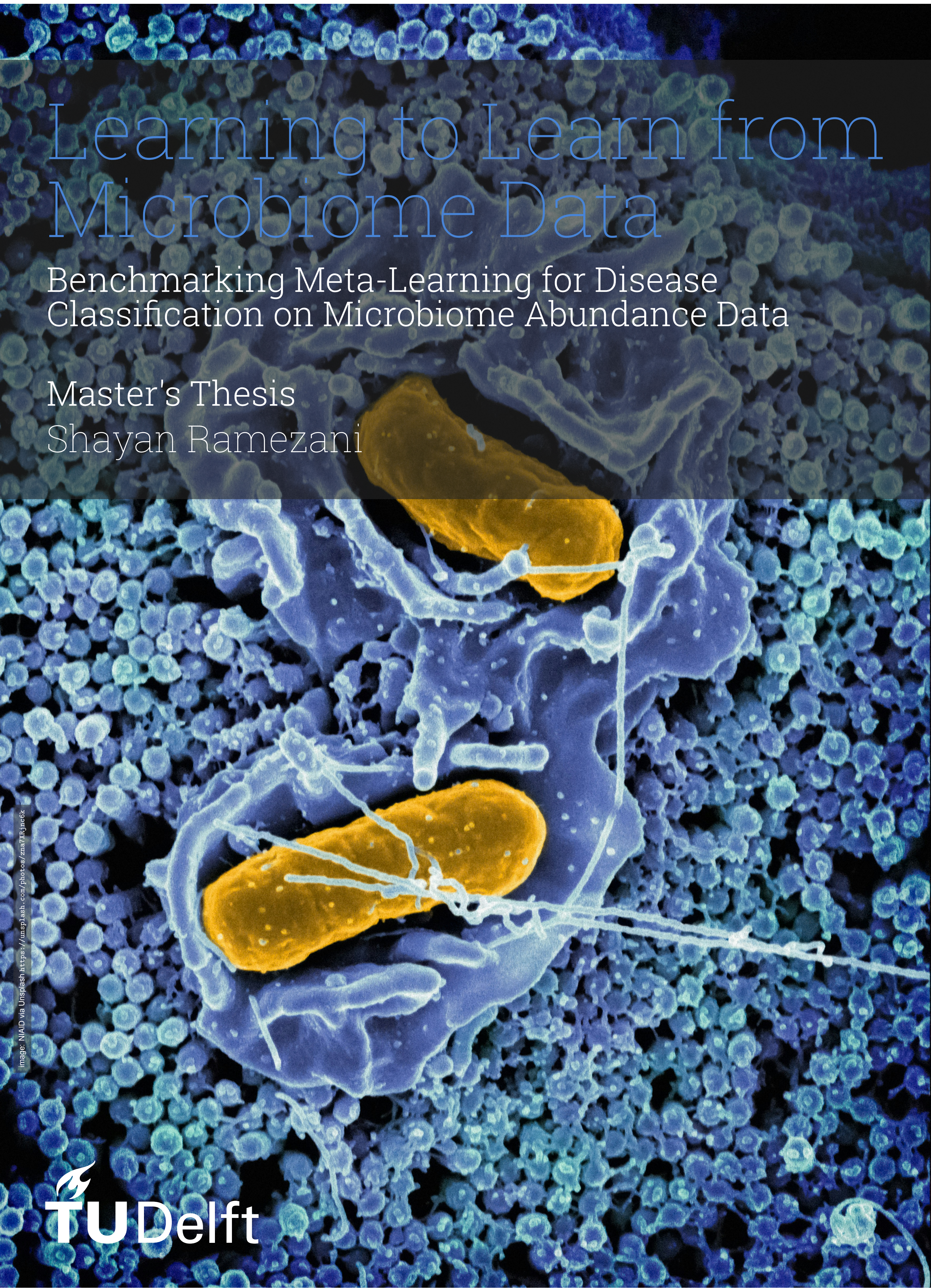


Learning to Learn from Microbiome Data

Benchmarking Meta-Learning for Disease Classification on Microbiome Abundance Data

Master's Thesis

Shayan Ramezani

TU Delft

Learning to Learn from Microbiome Data

Benchmarking Meta-Learning for Disease Classification on Microbiome Abundance Data

by

Shayan Ramezani

Thesis Advisor:	Thomas Abeel
Daily Supervisor:	Jasmijn Baaijens
Daily Co-Supervisor:	Chengyao Peng & Bianca Cosma
Project Duration:	October 2024 - July 2025
Faculty:	Faculty of Electrical Engineering, Mathematics & Computer Science, Delft

TU Delft

Contents

1 Abstract

The human gut microbiome has emerged as a key player in health and disease, yet machine learning on microbiome data remains challenging due to its high dimensionality, sparsity, compositionality, and inter-study heterogeneity. Although classical and deep learning methods have demonstrated promise, they often require extensive labeled data, which is rarely available in microbiome research. In this thesis, we investigate whether meta-learning can address these challenges by enabling better generalization from small, heterogeneous microbiome datasets. Specifically, we benchmark Prototypical networks (Protonets), a metric-based, few-shot meta-learning algorithm, against strong classical baselines (Random Forests, XGBoost, and Multi-layer Perceptrons) for disease classification tasks across a selected number of gut microbiome studies. We introduce a unified benchmarking pipeline that standardizes preprocessing, dimensionality reduction, task construction, and evaluation across studies. A leave-one-study-out cross-validation strategy simulates realistic deployment scenarios where only a few labeled samples are available from a new cohort. Our experiments explore the impact of support set size and dimensionality reduction via principal component analysis. Results show that although Protonets offer a conceptually appealing approach for few-shot learning, they consistently underperform compared to Random Forests in classification accuracy. Statistical analyses confirm the significance of this performance gap, and embedding visualizations reveal limited class separation in the learned feature space. These findings suggest that, under the evaluated conditions, classical models like Random Forests remain the more robust choice for microbiome classification in low-data regimes. By offering a rigorous and reproducible evaluation, this work lays the foundation for further exploration of meta-learning in microbiome research and highlights both the potential and current limitations of learning to learn in this complex domain.

2 Introduction

Over the past two decades, research has increasingly recognized the importance of the microbiome (communities of microorganisms in an environment) for human health. In particular, the gut microbiome has been associated with a wide range of conditions, including neurological disorders and brain function[11, 66, 10], anxiety and depression[20], behavioral traits[9], metabolic conditions such as obesity[74, 73], cancer[64], and many other relations[28]. This rapidly growing body of literature underscores the relevance of the gut microbiome in both basic research and clinical applications.

2.1. Data Challenges and Current ML Approaches

With the rapid growth of publicly available microbiome data, there is an unprecedented opportunity to deepen our understanding of these microbial communities and unlock their potential. While classical machine learning methods have great potential for analyzing microbiome data, its application is challenging because:

- small sample sizes coupled with high dimensionality often lead to overfitting;
- the heterogeneity of data makes it challenging to combine datasets to obtain larger datasets;
- because labeled datasets are scarce, not all ML methods are effective;
- the data are compositional, meaning only relative, not absolute, abundances matter[23]. This violates common statistical assumptions like independence between features and normality[3].

An important step in mitigating these issues is preprocessing. But the lack of preprocessing standards in the field[87], with even absence of reliable and unbiased techniques for some data qualities, make this challenging. This can be for example due to biases introduced by normalization and transformation steps[86], and batch effects arising from differences in sequencing protocols and laboratory conditions[87, 46].

2.2. From Transfer Learning to Meta-Learning

Deep learning (DL) has surpassed classical machine learning (ML) methods in many fields, achieving strong results in vision, language, and speech, which show the potential of dealing with some of the complex problems with regards to microbiome data[88]. However, learning from a small dataset becomes a bigger challenge when DL models are used, due to struggles to generalize[1]. Therefore, a promising solution is to learn a more general model that utilizes extensive public datasets, through which the model borrows statistical strength across studies due to exposure to more data.

This appears feasible because, despite substantial variation in microbiome composition across studies (often due to differences in environments), certain microorganisms tend to appear together in consistent patterns across very different environments[26]. For instance, studies have found that these patterns hold across countries and laboratories[4], that a small group of microbes are responsible for many of the shared connections across environments[43], and that species found in multiple environments often play similar roles and have similar characteristics[57]. Many more examples of such cross-environment patterns can be found in the recent literature[53, 65].

Transfer learning seems promising for learning this cross-study patterns[30] and multiple studies in the field have looked into this[8, 55, 7], where transfer learning is used to pre-train a model on a plethora of data from different context, before applying the model to a new context with fine-tuning.

But recent evidence suggests that few-shot meta-learning, algorithms that learn to adapt quickly to new tasks with minimal additional data, can outperform standard fine-tuning in transfer learning when the target context offers only tens of examples[42]. Due to the limited amount of data in individual microbiome studies, looking at few-shot learning methods makes sense. This is the main reason for looking at meta-learning models in this study.

Meta-learning goes beyond transfer learning and the related multitask learning, enabling better performance when data is scarce or the environment is changing[75, 1]. Yet, systematic investigations of meta-learning for microbiome classification remain scarce[29].

To probe that open question we adopt Prototypical networks[67], a lightweight, metric-based meta-learner that is particularly well-matched to microbiome data. Protonets require only a handful of labelled examples to form class ”prototypes”, train with fast first-order updates, and yield embeddings whose centroids can be inspected as representative community profiles. This combination of data efficiency, computational modesty, and interpretability makes them an ideal first test-bed for the high-dimensional, small-sample microbiome regime.

2.3. Research Objectives

The goal of this thesis is to benchmark the performance of a few-shot meta-learning method (namely Prototypical networks) against classical machine learning approaches for disease classification using microbiome data. The focus is on realistic cross-study scenarios where only a few labelled samples are available from a new cohort. This setting reflects practical challenges in microbiome research, such as high dimensionality, data scarcity, and strong inter-study variability.

Concretely, the main research question is:

> *Does meta-learning with Prototypical networks outperform classical machine learning models in few-shot microbiome classification?*

Related questions in answering the above question are:

- How do conventional baseline models perform in general?
- How does the amount of labeled data (number of shots) affect classification performance on microbiome data with the models studies?
- How does dimensionality reduction using principal component analysis affect classification performance on microbiome data with the models studied?

2.4. Scope and Contributions

To date, systematic meta-learning evaluations for microbiome classification remain scarce. We evaluate a representative algorithm, Prototypical networks, on taxonomic abundance tables drawn from 36 gut-microbiome studies. Contributions include:

1. **A unified experimental pipeline** that standardizes preprocessing, task generation and evaluation across heterogeneous studies.
2. **A comparative benchmark** of meta-learning methods against strong tabular baselines (e.g. Random Forests, XGBoost, Multi-layer perceptrons) under different shot regimes.
3. **An empirical analysis** of the inner workings of Prototypical networks on the dataset used and data qualities that may affect its performance.

By moving beyond conventional supervised learning approaches, we aim to clarify whether meta-learning delivers a practical edge for microbiome disease classification.

2.5. Thesis Structure

We start in Chapter 3 by introducing the reader to important microbiome and machine learning topics that are important for this study. In Chapter 4, we give a review of related existing literature on using machine learning in microbiome analysis. Chapter 5 dives into specifics about the data used for experimentation, the methodology for processing the data, and the setup for models and training. Chapter 6 presents the experimental results and their analysis, and discusses the findings and some key takeaways. In the last chapter, we end with the conclusion and discuss limitations and possible future works.

3 Background on Microbiome Data

Microorganisms exist in complex communities (microbiomes) that inhabit virtually every environment, from soil and oceans to the human body. These microbial communities play fundamental roles in ecosystem function and host health. For example, the human microbiome, trillions of microbes living in and on our bodies, contributes to our immune system development, digestion, and metabolism[2]. Although most of these microbes are benign or beneficial, there are some that can cause diseases [2]. Mounting evidence has linked imbalances or shifts in the microbiome to a wide range of health conditions, both for humans and other living organisms[32]. In addition, they influence any other ecosystem they occur in. These findings underscore the ecological and biomedical importance of microbiomes and motivate intense research efforts in microbiome science.

3.1. High-Dimensional Taxonomic Data from High-Throughput Sequencing Techniques

Microbiome data are typically generated by high-throughput DNA sequencing of environmental or clinical samples to profile the community's composition. The two predominant approaches are amplicon sequencing (marker-gene surveys) and shotgun metagenomic sequencing[21, 29].

- **16S rRNA Amplicon Sequencing**: This widely used method (also called DNA metabarcoding) targets a specific genetic marker present in all bacteria and archaea (the 16S rRNA gene[21]). The result is a set of 16S gene reads that can then be clustered into Operational Taxonomic Units (OTUs) based on sequence similarity or resolved into exact Amplicon Sequence Variants (ASVs) with denoising algorithms to keep rare members of the community[29]. Each OTU/ASV roughly corresponds to a bacterial taxon. This produces a table of taxonomic abundances: counts (or frequencies) of each OTU/ASV per sample. Amplicon sequencing is cost-effective and requires relatively few reads per sample (tens of thousands) to capture community composition[59, 37], making large cohort studies more feasible. However, it has limitations: the 16S marker provides limited taxonomic resolution (closely related species may be indistinguishable), it cannot detect organisms lacking the target marker such as viruses or many fungi (although the ITS region may help here [21]) and it yields no direct functional gene information[59, 37]. Moreover, PCR amplification can introduce biases and errors, which must be carefully managed[37].
- **Shotgun Metagenomic Sequencing**: In shotgun sequencing, total DNA from the sample is randomly fragmented and sequenced without targeting a specific gene. This produces a comprehensive catalogue of genes present. Shotgun metagenomics yields higher resolution profiles: it can detect microbes at strain level and recover functional genes in addition to taxonomic composition[36]. It is relatively unbiased by primer choice since it does not rely on a single marker[37]. A sufficiently deep shotgun sequence can quantify organisms across the tree of life (bacteria, archaea, viruses, eukaryotes) and characterize the functional capacity of the community. The trade-offs are cost and complexity: shotgun data typically require a much larger number of reads per sample (on the order of millions) to adequately capture low-abundance community members.

Another practical challenge is that in host-associated samples (e.g. blood, tissue swabs), host DNA often dominates the read pool[37]. Additionally, shotgun sequencing cannot distinguish live versus dead microbes or activity levels, it shows what genes are present, but not which are expressed[36]. Despite these challenges, shotgun metagenomics provides a richer dataset that includes both taxonomic and functional profiles, and recent work has shown that even shallow shotgun sequencing can outperform 16S surveys in resolution and reproducibility at comparable cost[37].

Other omics layers, transcriptomics, proteomics, metabolomics, and phenomics, offer complementary perspectives on microbiome function; however, they fall outside the scope of this thesis.

After sequencing, data from either approach undergoes bioinformatics processing to produce an abundance table, a matrix of microbial features (OTUs, species, or genes) by samples. The end result is a high-dimensional feature vector for each sample, with each feature representing a microbial taxon (or gene) and its abundance in that sample.

Having turned raw sequences into high-dimensional abundance matrices, we now ask why those matrices so often confound standard statistics and machine-learning models.

3.2. Challenges in Analyzing Microbiome Data

Microbiome data derived from sequencing have several distinctive characteristics that pose challenges for analysis:

High-Dimensional and Small Sample Sizes A single microbiome sample can contain hundreds to thousands of distinct taxa, yielding feature tables with very high dimensionality. The number of features (e.g. OTUs or species) often far exceeds the number of samples[3]. For instance in MGnify[60], one of the largest microbiome databases, only 65 out of more than 5,000 studies have a sample size larger than 1,000[1]; with the feature counts routinely exceeding the sample size. One reason for this can be the cost and complexity of sample collection and sequencing. Only a minority of studies include more than a few hundred samples, and in less-studied environments or rare disease contexts, sample sizes can be lower.

The combination of small sample sizes and high-dimensional data, is problematic as models can inadvertently learn noise or sample-specific patterns rather than true generalizable signals[1]. These mean that microbiome datasets lie in a vast feature space which encourages machine learning models to overfit and limits their statistical power to detect patterns. Due to these, special care (e.g. dimensionality reduction or feature selection) is often required to deal with these high-dimensional; data[3].

Sparse Data In addition to high-dimensionality and small sample sizes, most entries in the data matrix (often $> 90\%$) are zero, since many taxa are absent in a given sample (each sample contains only a subset of the possible communities). This sparsity further aggravates overfitting

Class Imbalance Microbiome datasets frequently suffer from imbalanced classes – for instance, a disease cohort might have far fewer positive cases than negatives, or certain environments (e.g. gut microbiomes) are over-represented in public data while others are under-represented. Imbalanced data can bias classifiers toward the majority class and make it hard to learn minority class signals. If not optimized for the correct metric, the model may even not learn biological signals at all. Altogether, these make it difficult to train robust predictive models and to generalize findings broadly.

These also mean that performance metrics should not be taken as absolute values just as they are and should be compared to some understandable and simple baseline, e.g. a model with random outputs. For example, in imbalanced datasets where the positive class is the majority class, a classifier that always predicts the positive label can already attain an apparently respectable $F1$ score. In these cases, it is useful to include a majority-class dummy (=positive-class dummy) baseline against which all subsequent models are compared.

[1]Last checked on 20th of June 2024.

Unfortunately, despite the importance of such baselines, many microbiome classification studies omit them entirely, reporting performance metrics without demonstrating that their models outperform trivial baselines. Two recent meta-analyses illustrate the problem. Duvallet et al.[16] and Lee et al.[39] both benchmark Random Forests yet omit a trivial dummy baseline, leaving open whether their models captured biology or merely class ratios. This problem is bigger in the former study as the data is more imbalanced.

These highly-cited, high-quality papers (and many other [14, 81, 84]) illustrate a common issue in the field: benchmarking sophisticated models (e.g. RF, LASSO) without first confirming they beat a dummy baseline, even when class imbalance is clearly present.

Compositionality Microbiome data are compositional in nature[23, 3], meaning that only the relative abundances of features are meaningful rather than absolute counts. Sequencing assays typically report each sample's microbial profile as proportions that sum to a constant (100% or an arbitrary total read count) [23]. This data live on a simplex (a constrained sum space), not in unconstrained Euclidean space, which complicates downstream modeling that assumes data in Euclidean space.

This induces inherent dependencies among features (an increase in one taxon's relative abundance necessarily means a decrease in another, even if both absolute abundances could have increased). Compositional data violate assumptions of standard statistical techniques that treat features as independent absolute quantities[23]. Ignoring compositionality can lead to misleading results, such as spurious correlations driven by the closure effect (the constant-sum constraint). Best practices therefore recommend transforming or analyzing data using compositional data analysis methods (e.g. log-ratio transformations) that respect the geometry of proportions [23, 86].

Biological and Technical Heterogeneity Across Studies Microbiome composition varies widely with host species, body site, lifestyle, geography, and even among nominally healthy individuals, two human-gut profiles may share only a minority of taxa. Such ecological diversity (illustrated by the low Jaccard similarities between gut studies in Figure 3.1) can overshadow disease signals and is further amplified by sample-level differences in sequencing depth [27], assembly and gene-prediction workflows [61], and natural variation between samples [41, 36]. On top of these biological factors, laboratory protocols, sequencing platforms, and bioinformatic pipelines introduce systematic artifacts (batch effects) that often explain more variance than the phenotype of interest [62]. When datasets from multiple batches are combined, each batch behaves like a separate domain, blocking cross-study generalization unless specialized correction methods are applied [62, 40]. Nevertheless, handling these effects remains a major challenge as algorithms are mainly developed for RNA-seq analysis[25], often requiring careful study design (e.g. case-control studies within a single cohort) or advanced statistical correction.

Because each preprocessing decision can amplify or mask the challenges just described, the next section audits some of commonly used methods.

3.3. Current Workflows Lack a Standardized Pipeline

Preparing microbiome data for analysis involves several steps, each of which can impact downstream results. However, there are no standards in the field for preprocessing the data, with even absence of reliable and unbiased techniques for some specific steps [87, 25, 78, 46, 24, 44]; but common practices have emerged for data preprocessing and feature extraction from sequencing data, which include:

Normalization Because sequencing depths vary (some samples have more total reads than others), normalization is required to make samples comparable. A simple approach is converting counts to relative abundances (proportions), effectively dividing each sample's counts by its total reads. However, proportions induce the compositional effects noted above. Another common practice is rarefaction, which subsamples each sample's reads down to an equal count[87]. Rarefaction can mitigate depth differences but discards data and may introduce its own bias[87]. More advanced normalization methods include cumulative sum scaling, total sum scaling, trimmed mean of M-value or other approaches to adjust for uneven sequencing depth and compositionality[87]. There is ongoing debate about the

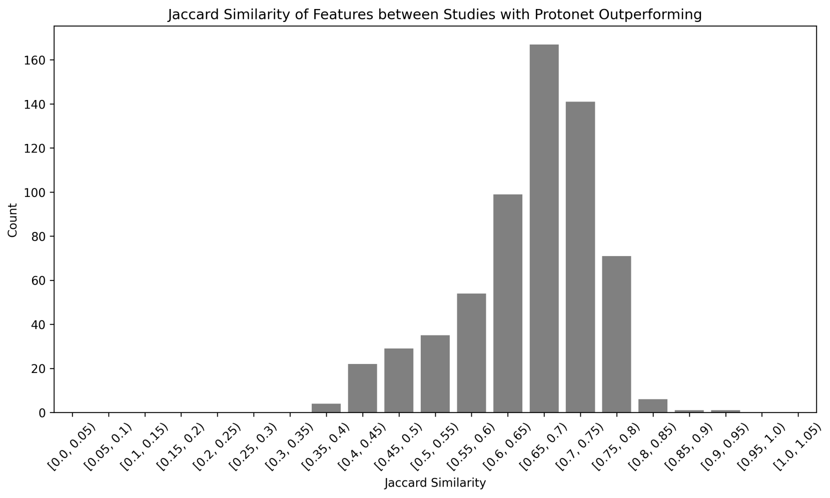


Figure 3.1: Jaccard similarity count of the feature space of the different human gut microbiome studies in the dataset curated by Sun et al.[68]. The count basically indicates how often the range of Jaccard similarity occurs between different pairs of studies.

best normalization as no single method is universally optimal, and each can impact the downstream analysis differently [87, 46, 79].

Compositional Transformation After normalization, further transformations are often applied to make the data more suitable for modeling and to get rid of compositionality of data. If the right transformation is used, this step is believed to help in ”reducing biases and recovering true biological signals”[86]. The CLR transformation (taking log-ratios relative to the geometric mean of all features in a sample) is a popular choice grounded in compositional data theory[23]. However, CLR and many of the other transformations add bias to the data and alternatives are being introduced[86]. According to Zhang et al. [86], log-ratio transformations were not developed for sparse data like *microbiome data*, as they cannot handle zeros and no universal solution has been found on how to treat the zeros [86, 79], making them less appropriate for applications within the microbiome field. Zhang et al. [86] give novel transformations, e.g. Centric Arcsine Constrast (CAC) and Additive Arcsine Contrast (AAC), which shows enhanced performance in these sparse scenarios and when the rest of the data points are mainly extremely small.

Feature Engineering Researchers may incorporate ecological knowledge, for example, combining features at higher taxonomic levels (summarizing by genus or family) to reduce dimensionality, or computing diversity indices and other derived features[17]. Functional considerations matter as well: microbiome communities exhibit pronounced functional redundancy, meaning that taxonomically distinct assemblages can carry out overlapping metabolic roles. Puente-Sánchez et al.[57] states that pairs of species found in multiple environments tend to be more similar in what they do and how they are related (higher functional and phylogenetic redundancy). This suggests that having this kind of similarity helps them survive when their environment changes.

Another important step can be feature selection, identifying a subset of taxa that are most relevant to the prediction task (e.g. potential biomarkers of disease). Methods such as differential abundance analysis or regularized regression are used to reduce thousands of features to a manageable number of informative microbes[29]. In microbiome machine learning studies, it's not uncommon to perform dimensionality reduction (like principal component analysis) or even use unsupervised techniques (like clustering or autoencoders) to extract latent features that capture variance in the community[83, 29, 52].

Handling Batch Effects and Confounders As noted, technical or study-specific variations can distort microbiome measurements, obscuring genuine biological signals. If data from multiple runs or studies are combined, preprocessing may include steps to correct batch effects (e.g. using statistical methods such as ComBat or quantile regression adjustments)[40, 87]. Similarly, metadata variables (like subject age, diet, or DNA extraction kit) that could confound analysis should be regressed out or balanced between comparison groups if possible. Careful experimental design and computational batch correction are essential to ensure that downstream classifiers learn true biological patterns rather than artifacts of sequencing order or lab origin.

In summary, preprocessing transforms raw sequences into an analysis-ready feature matrix while attempting to mitigate noise and biases. However, there is no established pipeline in microbiome research: different studies employ different quality filters, clustering thresholds, normalization techniques, and so on. In fact, some steps lack a clearly superior method, and new best practices are still being developed (for instance, how best to normalize compositional data, correct batch effects, or transform compositional data). This lack of standardization means that machine learning pipelines must often be tailored and carefully optimized for each dataset or problem domain.

With the data finally distilled into an analysis-ready matrix, we turn to the algorithms best suited to its high-p, low-n, compositional quirks.

3.4. Machine Learning Algorithms

The microbiome field increasingly relies on supervised learning to convert high-dimensional abundance tables into clinically useful predictions. We benchmark three widely used learners, Random Forests (robust non-linear splits and built-in feature importance), XGBoost (state-of-the-art on tabular data), and a shallow multi-layer perceptron (baseline deep model), against meta-learning approaches. Here, we will introduce the reader to the general concepts of meta-learning.

3.4.1. Meta-Learning

Meta-learning aims to learn how to learn: instead of training a model for a single task as in conventional deep learning approaches, it trains an algorithm that can rapidly adapt to new tasks. This means that these models will not learn to classify samples explicitly, but exploit the training data with the final goal of being able to learn to classify unseen samples faster from only a few examples [72]. Such rapid-adaptation paradigms are attractive when a new microbiome study may arrive with fewer than ~ 50 labelled samples.

As in deep learning, there is a train and test stage during meta-learning. In one stage meta-training is performed to learn how to learn tasks faster and better on the labeled training dataset; and in the second stage meta-testing is performed for evaluation purposes, where adaptation to the new task of the test set with the prior knowledge of the meta-trained model is supposed to help in performing better in the downstream task. One difference from other deep learning methods comes from this adaptation step, where the algorithms gets the chance to learn about the new task before inference.

This means that the dataset is always first split into a train and test set for use in the meta-training and meta-testing part respectively.

A typical example is Model-Agnostic Meta-Learning (MAML) [18], an algorithm that has to learn to better initialize a model, visualized inFigure 3.2. In the meta-training stage, the inner loop initializes the model in the algorithm based on a few data points from a task (called the support set), after which it performs inference on a different set of data point from the same task (called the query set). In the outer loop, the loss value from the inference on the query set gets back-propagated to update the model, so the next run the initialization can be better. So, in every iteration the model sees two non-overlapping datasets for the same task.

So in general, during meta-training, ”task-wise training” happens in two nested loops:

1. **Inner loop — task-level adaptation**: Given a small support set from one task, the model performs a brief adaptation step (e.g. a few gradient updates or the computation of task-specific statistics), producing task-specialized parameters $\theta'_{\mathcal{T}}$ from the shared meta-parameters θ. These parameters are then evaluated on the task's *query* set to obtain the loss $\mathcal{L}_{\mathcal{T}}$.
2. **Outer loop — meta-level optimization**: The loss $\mathcal{L}_{\mathcal{T}}$ is back-propagated through the entire adaptation path to update the shared meta-parameters θ, so that the model can adapt more

quickly and perform better on future, unseen tasks.

During meta-testing, a small part of the test data drawn from a previously unseen task is used to get a well initialized model that will do inference on the rest of the unseen test data.

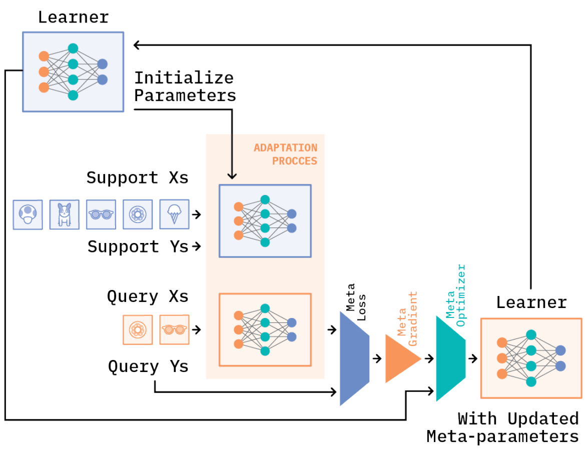


Figure 3.2: A typical example of the inner working of meta-learning algorithms. (Reproduced from [22])

In the general concept of meta-learning, every task consists out of some selected classes from the full training data. The number of classes selected (N) and the size of the support set (k), give the well-known N-way k-shot learning scenario of few-shot learning; with k giving the shot number which is usually small.

Throughout the learning process, also different but related tasks can be used, but the assumption is that there is an underlying distribution from which all these tasks are sampled [75].

Families of methods

Meta-learning research has produced many different techniques in the last decade and choosing the right method can be more art than science. These techniques are usually divided into three different categories [75, 33]:

- **model-based (black-box) methods** equip the learner with an additional memory or controller (e.g., LSTM controller, external memory) so it can store and retrieve information across tasks and learn to learn purely through its own forward pass.
- **metric-based methods** learn an embedding or distance function such that classification can be done by simple comparisons in that learned feature space
- **optimization-based methods** focus on learning an efficient way to optimize model parameters. These methods essentially train the model's starting point or training procedure to be well-suited for few-shot adaptation.

Optimization-based techniques offer the same advantages as model-based techniques but also outperform them when dealing with larger training set [31, 33]; and when it comes to generalizability to more distant tasks [33] (which is studied for MAML in [19]), they are a good choice. However, optimization-based approaches have the disadvantage of being computationally expensive compared to the other techniques.

Also Metric-based techniques outperform model-based techniques in supervised learning [33] (which is shown for graph-neural networks in [22]) too. Metric-based techniques are in addition simpler to optimize and are extremely data efficient, and therefore form a good starting points for our research where data scarcity is an important factor.

For an in-depth comparison of the advantages and disadvantages of the different methods, the reader is referred to table I in [75].

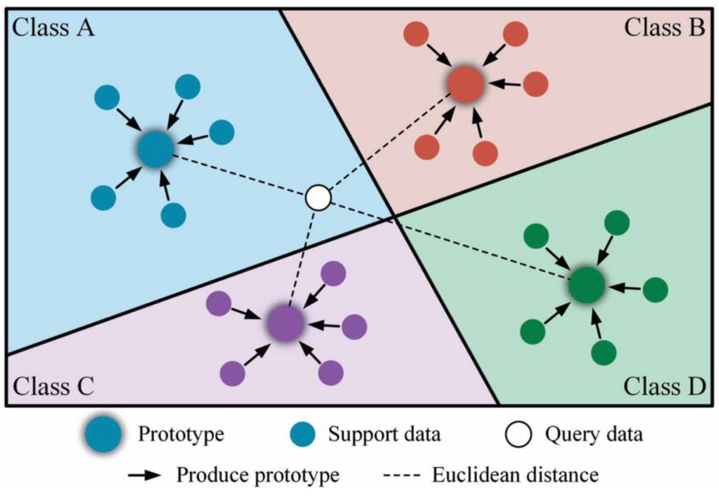


Figure 3.3: Visual representation of how Protonets work. Reprinted from [85].

A typical metric-based technique is Prototypical Networks (Protonets) [67], which have received much attention in the few-shot learning scenario[51] . In short, Protonets define a class representation (called prototypes) in a learned embedding space and classify unseen examples by calculating a distance metric to these prototypes. According to the authors of the original paper, Protonets are a ”promising approach for few-shot learning”.

Due to Protonet’s simplicity and data efficiency, their use of embedding spaces where taxa can be embedded on, and metric use that can easily be connected to the concept of similar taxa, focus will be laid on this approach.

Inner workings of Prototypical networks

Supervised meta-learning with Protonets basically consists of the two stages described for meta-learning. Figure 3.3 visualizes the inner workings of Protonets specifically, with the following two nested loops:

- **Inner loop (adaptation and classification)**: for each task, the support samples are embedded and aggregated per class to form **prototypes** (class cluster representations). Query samples from the same task are embedded and classified by the nearest prototype.
- **Outer loop (meta-update)**: The classification loss of the query samples from the inner loop is back-propagated to update the embedding parameters, so that future tasks can be solved better from only a few support examples.

During meta-testing, the same steps are taken essentially. First, the support set is used to obtain the prototypes. Afterwards, query samples (which during testing comprise the entire held-out data minus the support samples) are used for evaluating the performance by mapping them to the representation space and calculating the loss and other metrics based on the labels predicted, without any backpropagation.

4 Related Work

Robust biological insight demands robust evidence. Thanks to concerted international efforts, public repositories now hold tens of thousands of microbiome samples that offer unpresented opportunities for discovery. Yet the very properties reviewed in the background, e.g. high dimensionality, sparsity, compositionality, and batch effects, undermine both classical statistics and modern machine learning.

4.1. Classical Machine Learning on Microbiome Tables

With the recent advancements in machine learning and the surprising successes, such as XGBoost models that predict myocardial infarction risk from $> 500,000$ UK Biobank participants[47] or sparse LASSO scores helping in identifying key DNA variants for common diseases across multiple ancestries[58], it has become clear that it can be useful in extracting knowledge from large datasets. These methods have the advantage of inferring non-linear relationships in high-dimensional data that classical statistics cannot capture[44].

The go-to classical machine learning methods for classification purposes with microbiome data are SVMs, decision trees, random forests (RFs), boosted trees, logistic regression, and simple neural networks [69]. Of these, RFs are the most widely used in the microbiome classification tasks and have shown good performance [69, 52]. Although traditional and basic ML methods applied to specific microbiome datasets have been the mainstream way of conducting research, there is evidence that they fail to capture governing patterns and representations regarding microbiome data that can help in having more accurate analyses of the data [12].

Building strong models is a big challenge and it is believed that ”no single machine learning pipeline can universally accommodate all predictive modeling scenarios” [52]. Papoutsoglou et al.[52] suggests elaborate experimentation to find the optimal methods including that for preprocessing, but this is time-consuming and inaccessible for non-ML experts. A recent audit of 100 microbiome ML papers found that 71% failed to report final hyperparameters and 90% did not perform any hyperparameter search, indicating that too many studies still rely on default settings and ad-hoc preprocessing [15].

4.2. Deep Learning and the Rise of Pre-Trained Models

More recently, in many fields deep learning has become the go-to method when using ML methods to achieve a better and more general performance. Specifically, large language model research has inspired many different architectures and methods [83]. But on the architectural part, these methods have their shortcomings in dealing with microbiome data and its unique challenges regarding high dimensionality, sparsity, and heterogeneity[83].

The recent advancements in deep learning also have shown a great potential for pre-trained models (PTMs)[77, 76], which can, inter alia, fully exploit unannotated data [77]. Examples of recent applications exceed human language generation/understanding and image recognition to include models dealing with DNA sequences [34], crop mapping [82], chemical data [63] and many more.

More specifically, due to the successes in other fields like language processing, computer vision, and protein folding, there has been motivation for transfer learning where pre-trained models are created for microbiome data to capture the fundamental patterns and are fine-tuned for downstream tasks afterwards[13]. One argument is that when training these models, many datasets can be combined in one way or another and tackling the data challenges becomes easier, e.g. sample size increases and matches what the model needs for the high dimensionality of the data. Another important argument for transfer learning is related to its usefulness in unsupervised/self-supervised learning, through which due to availability of much more unlabeled data than labeled data better performance can be achieved. Unsurprisingly, several groups have attempted to look into pre-training models for microbiome data.

4.2.1. Representative Efforts on Pre-trained Models

Different attempts on creating pre-trained models show varying results.

Çiftcioğlu & Nalbanoglu [8] try to learn a latent representation with embedding networks on gut taxonomic tables before training a separate classical classifier model on the embedded data [8]. The authors acknowledge that the small pre-training cohort prevents generality beyond gut samples and so does not provide a foundation model, which they consider to be possible in the field.

Pope et al.[55] pre-train paired transformer generator/discriminator networks on masked taxonomic V4 16S reads, then fine-tune the discriminator as a dimensionality reducer for disease classification [55]. However, the model does not show impressive results, which can be attributed to other factors unrelated to the strength of the model. One of the reasons can be the limited dataset size. This can lead to pre-training not having learned the governing fundamental patterns of the data. In addition, the focus of this research on taxonomic data based on ”16S data of the V4 region” can be seen as a limitation as classification from this may not be able to achieve high performance no matter the amount of the data, due to lower taxonomic resolution and lack of standardized protocols [55].

Chong et al. [7] develop a pre-trained model, called EXPERT, that adapts its knowledge to the specific context of the downstream task in three steps: (1) reinitializing only the contextual layers of the pre-trained model according to a specific context, (2) training the contextual layers with the data $\mathcal{D}$ in the specific context, and (3) further fine-tuning by training the whole model on the data $\mathcal{D}$. However, the method presented is a supervised learning method, not leveraging the abundant unlabeled data available, missing an opportunity to enhance performance and generalizability by incorporating unsupervised or semi-supervised approaches.

Common Limitations

Across these studies, different constraints recur:

1. **Insufficient pre-training scale** – fewer samples than typical language or protein PTMs.
2. **Transfer-learning ceiling** – fine-tuning on dozens of target samples often fails to adapt weights adequately [42].
3. **Opaque or inconsistent preprocessing** – the preprocessing steps in these studies are not always clear, while preprocessing microbiome data is crucial.

4.3. Few-Shot Scenarios & Meta-Learning as Alternative

Recent studies indicate that few-shot meta-learning, where algorithms rapidly adapt to new tasks using only a handful of additional examples, can outperform traditional transfer learning methods when the target task offers merely tens of samples [42]. Because individual microbiome datasets are typically small, framing the problem in a few-shot setting is natural. Also, in the field of computer vision, meta-learning has already proven its worth in such small data scenarios where the dimensionality is also large [54, 31]. This rationale encourages our focus on meta-learning approaches in the present study.

In addition, transfer learning typically requires the source and target tasks to be related[75] while meta-learning is better able to adapt to less related target tasks.

Altogether, meta-learning explicitly targets scenarios with scarce data or shifting conditions, often delivering superior performance compared to standard transfer learning or alternatives like multitask learning [75, 1]. Nevertheless, systematic evaluations of meta-learning for microbiome classification are lacking to the best of our knowledge[29].

5 Methodology

In this chapter, the steps taken to produce the results are explained. We start out by first discussing how the dataset is obtained and processed before learning, and mention some data characteristics. Afterwards, important aspects of our meta-learning implementation is explained, we dive into the different useful strategies to train classical ML methods, and some experimental protocols for further analysis are explained. All the related results will be shown and discussed in the next chapter.

Reproducible code and data reside in the accompanying GitHub repository: `https://github.com/sr1998/thesis`.

5.1. Dataset Acquisition & Preprocessing

The dataset used for the experiments is the species-level taxonomic abundance data from [68] [1], that the authors obtained by combining abundance data from multiple studies each investigating a different disease. Each study can have its samples categorized as either healthy or diseased.

5.1.1. Preprocessing Pipeline for our Compositional, Sparse Microbiome Data

Microbiome data is inherently compositional and sparse, so preprocessing to handle these characteristics is essential and can be done before data-splitting as they do not leak label information across studies. An overview of the preprocessing pipeline is shown in Figure 5.1.

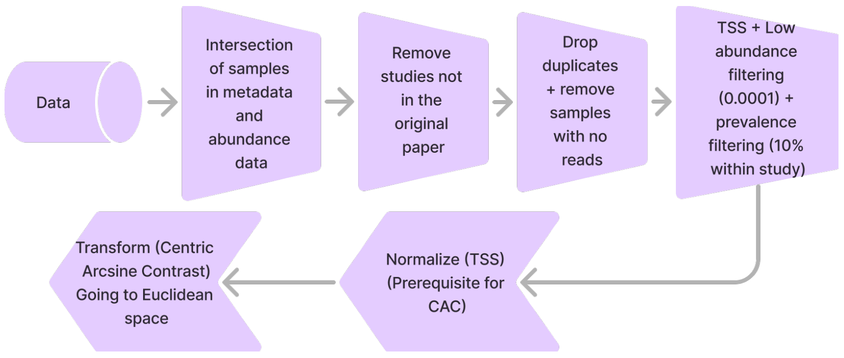

Data
Intersection of samples in metadata and abundance data
Remove studies not in the original paper
Drop duplicates + remove samples with no reads
TSS + Low abundance filtering (0.0001) + prevalence filtering (10% within study)
Normalize (TSS) (Prerequisite for CAC)
Transform (Centric Arcsine Contrast) Going to Euclidean space

Figure 5.1: Preprocessing pipeline for the Sun et al.[68] data. The raw dataset contains samples that occur in the metadata file but not in the abundance data file and vice versa; thus the intersection is kept. Studies `LiS_2021a` and `LiS_2021b` are removed. Samples are then cleaned, normalized, abundance/prevalence-filtered, again normalized, and finally transformed into Euclidean space.

As mentioned before, preprocessing standards in the field still need to be developed. The choices

[1] Exact version used here can be downloaded from `https://github.com/yexianingyue/GM_common_diseases/commit/9e2ca93a2bf1c48f96c5748c4704055c362061e0`

made here are driven by what is required for subsequent steps, such as normalization being essential before applying the transform step, and by commonly adopted best practices.

In prose, the steps for preprocessing the abundance data are:

1. **Sample intersection**: Keep only samples present in `sample.group` *and* `mpa4_species.profile`. Duplicated sample IDs are deduplicated by keeping the first occurrence.
2. **Remove some studies**: Two studies, namely `LiS_2021a` and `LiS_2021b`, were removed from the dataset downloaded as they did not occur in the analysis of the paper by sun et al. [68].
3. **Zero-read feature removal**: Globally drop taxa whose total read count across all samples is 0.
4. **Total sum scaling (TSS)**: TSS converts counts to relative abundances so that the abundance threshold in the next step is interpretable.

$$x_{ij}^{\text{TSS}} = \frac{x_{ij}}{\sum_k x_{ik}} \tag{5.1}$$

5. **Per-study abundance & prevalence masking**: For every study and every taxon, set $X_{s,t} = 0$ if abundance $< 1e4$ *or* prevalence within that study $< 10\%$.
6. **Redo TSS**: To renormalize after masking
7. **Centred arcsine contrast (CAC)**: The compositional data transformation step at the end is a crucial step in the preprocessing pipeline, as it brings the data into Euclidean space where deep learning operations make sense. Here, Centric Contrast transformation CAC is preferred "for more complex data structure where no single reference part is appropriate, maintaining full dimensionality"[86]. For the experiments in this research, CAC has been applied in the preprocessing, given by[2]:

$$\text{transformed\_data} = arcsin(\sqrt{\text{original\_data}}) \times \begin{bmatrix} 1-\frac{1}{p} & -\frac{1}{p} & \dots & -\frac{1}{p} \\ -\frac{1}{p} & 1-\frac{1}{p} & \dots & -\frac{1}{p} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{p} & -\frac{1}{p} & \dots & 1-\frac{1}{p} \end{bmatrix}_{p\times p} \tag{5.2}$$

Regarding the metadata provided by sun et al. [68], the important columns for the experiments here are `[Sample, Group, Project_1]`. These columns represent the `sample name`, `sample group (Control or Disease)`, `project source (author + year)`, respectively. So, we can remove all the other columns.

5.1.2. Dataset Statistics & Class Imbalance

After applying the preprocessing pipeline, the total number of studies left in the data is 36, with the study sizes differing between 70 and 400. However, as will be indicated later, not all studies are used for all experiments. The focus of the experiments will be to apply binary classification to separate healthy samples (assigned value 0 in experiments) from diseased samples (assigned value 1 in experiments).

The disease groups per study can be more specific than just healthy or diseased, e.g. the diseased samples for HIV can be grouped into INR, IR and VU groups. See Figure 5.2 for a look at how these groups are distributed according to the original paper. In here, our focus will however just be on binary classification as the original paper has done [68].

An important characteristic of this dataset is the imbalance of samples. This can be seen by observing the difference in counts of healthy and diseased samples in Figure 5.2. Here, the positive disease class is in most studies the majority class.

Usually in ML, training data is balanced so the models do not learn the bias present in the distribution of data but truly learn biological signals. This is necessary as with time, the distribution of classes may change. Test sets are always kept imbalanced to represent the real world data and have strict evaluations.

[2]Original implementation can be found here: `https://github.com/bioscinema/Microbiome_Transformation/blob/50b828b65210b2259905b3f07d152ca674c47da7/Real_Data_Analysis/Overlap_Skewness.R#L149`

In one of our experiments, an analysis of balancing is done and further analysis is based on the conclusion from this. This experiment is shown in Subsection B.1.3. The balancing in this experiment is done by oversampling the minority class with SMOTE[5], carried out before training on the train set. Testing is always done on the imbalanced test data.

In addition, experiments also give the performance of a `Dummy` classifier, that always predicts *diseased*, since diseased samples are more important in these disease classification tasks, are considered as the positive class, and in most studies are the majority class $(27/34)$. This provides a naive lower bound under the naturally existing class imbalance in most studies. Importantly, this baseline also reflects how metrics such as $F1$-score, which are biased towards the positive class, can be deceptively high even for trivial models. Including the `Dummy` classifier thus helps gauge whether any improvement by more complex models actually stems from learning biological signal, rather than just exploiting the label imbalance. Especially in few-shot scenarios, where the risk of overfitting is heightened, this simple but informative reference point provides critical context for interpreting model performance.

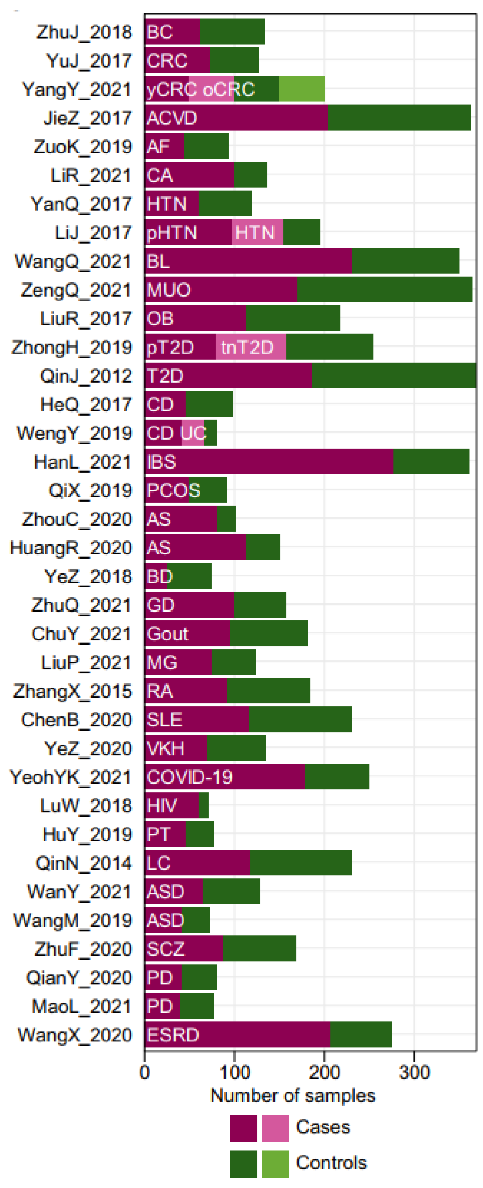


Figure 5.2: Distribution of diseased vs. healthy (=control) individuals for the studies in the dataset curated by Sun et al. [68] (adapted from [68]).

5.1.3. Dimensionality Reduction Strategies

In Chapter 3, we have already indicated that dimensionality reduction can be helpful in dealing with high-dimensional, sparse data. Principal component analysis (PCA) gives a common, quick, model-free estimate of the intrinsic dimensionality of a dataset. Therefore, we analyze classification accuracy under three data representations:

1. the **full feature set**;
2. a **PCA projection** in which samples are expressed in the first *k* principal-component (PC) scores (where k is a hyperparameter optimized for);
3. a **PCA-guided feature subset** that helps preserve taxonomic interpretability (where number of features to select is a hyperparameter).

This last method is a simple unsupervised way to prune taxa, where the taxa are ranked by how strongly they load onto the first c PCs and only the most "important" ones are retained. Formally, each taxon j receives an *importance score*:

$$\varphi_j = \text{importance}(j) = \sum_{i=1}^{c} |l_{ji}| w_i, \tag{5.3}$$

- where l_{ji} is the loading of taxon j on PC i, indicating how much the taxon contributes to the PC;
- and w_i is an optional weight proportional to the variance explained by that PC (which can be scaled).

Taxa are sorted by φ_j and kept until the *feature-importance coverage*

$$\text{coverage}(m) = \frac{\sum_{j=1}^{m} \varphi_{(j)}}{\sum_{j=1}^{p} \varphi_{(j)}}$$

reaches a chosen threshold τ. Note that this coverage is defined over features, not over PCs; hence it behaves very differently from the familiar "retain 99 % of variance" rule used when selecting c.

All dimensionality-reduction models are fit inside the training fold only and then applied to the corresponding test data, so no information leaks from test to train data. In Subsection B.2.2, we reason for whether dimensionality reduction is at all useful before the main experiments shown in Subsection 6.3.2.

5.2. Few-Shot Meta-Learning with Prototypical Networks

In this research, we apply the widely used Prototypical networks (Protonets) for the microbiome classification of interest. Here, the specific way of feeding the data into our model is explained.

5.2.1. Episodic Study-Wise Data Splitting

For training and evaluating Protonets, we need a meta-train and meta-test dataset. In here, we also apply hyperparameter optimization, which means the meta-train dataset is further split into a train and validation set during this process.

We follow a leave-one-study-out protocol, where the test set is the data from one of the studies in the complete dataset. The validation set is also obtained by using the same leave-one-study-out approach on the meta-train data.

The training is done on a task basis following the steps depicted in Figure 5.3. .

During training, the query set size is the same as the support set size. However, during evaluation time the model:

1. Receives the k-shot support set of the held-out study and adapts the model.
2. Predicts the labels of the query set (which is the rest of the dataset here), and records the metrics.

No further gradient steps are taken after the adaptation during evaluation.

Both the selected validation studies and the support set of the validation and test studies are preselected and stored for equal comparison and reproducible outcomes.

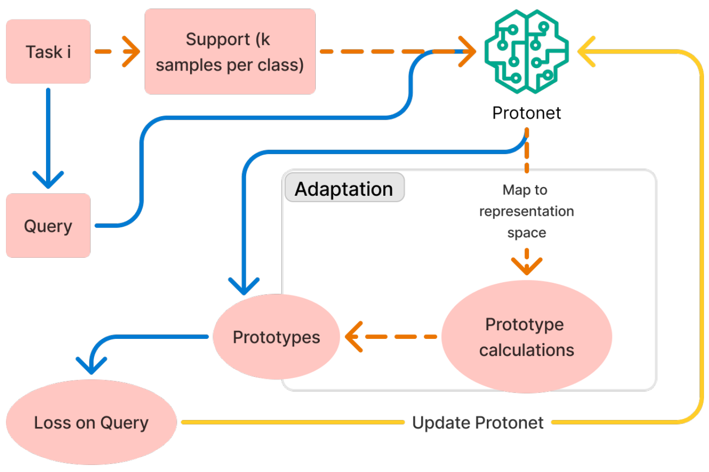


Figure 5.3: High-level steps taken to train Protonet on a selected task. The orange path is followed first, adapting the model to the support set. The blue path is followed afterwards, getting the predictions of the query set and calculating the loss; followed by the yellow path using the loss to improve the model.

Task construction and few-shot notation

In this work, each task corresponds to an individual microbiome study. Because for every study the sample labels are simply either healthy or diseased, the class dimension is fixed at $N = 2$, giving a 2-way k-shot setup. For every training episode, we first sample one study from the meta-training pool, draw k labelled examples per class to form the support set (size = $2 \times k$), and place another k labelled samples per class from the same study in the query set.

Dataset & episode sampler implementation

To turn the raw abundance tables into few-shot tasks we combine two small, self-contained PyTorch utilities—`MicrobiomeDataset` and `BinaryFewShotBatchSampler`—that encapsulate per-study balancing/oversampling and the support/query split required by meta-learning. Here is a summary of the implementation details:

Design goal	**How it is achieved in code**
Study-wise task (one task = one study)	`MicrobiomeDataset` groups the samples by the study they belong to. Every training/testing episode is generated by the sampler picking one study and then drawing samples from that study for both the support and query set. A preselected support set can be supplied for the test/validation study so the support and query set is always the same for reproducible evaluation data. This means, the exact same data points are used for support/query during evaluation on test/validation study when the experiment is repeated.
Class balance inside every task	During training, for each chosen study in a learning episode exactly k samples per class are taken for the support set and query set. If a study's minority class is too small to provide at least one non-overlapping support and query set ($< 2\times k$), the whole study is dropped beforehand. During training, the sampler advances through the larger class circularly, so the smaller class is effectively oversampled until both contribute the same number of episodes and all the samples from the larger class have been sampled (either for support or query) once, guaranteeing balanced k-shot support and query sets within every task. At evaluation time, the unseen study is visited exactly once: its first k samples per class become the support set, and the entire remainder forms the query set.
Shuffling strategy	The study list and within each study, the class-specific samples, are shuffled once every epoch when training and only once at the beginning when evaluating (so evaluation support and query sets stay the same). Shuffling for evaluation is skipped when a preselected support set is provided.
Label encoding & loss re-weighting	Labels are one-hot encoded before training. Global class counts over the whole training corpus are used to derive inverse-frequency weights, which are fed into the loss function for optimization. This makes the training more similar to the classical models benchmarked against, as we will see.

5.2.2. Study-Specific Gaussian Jitter Data Augmentation

Limited data has been a challenge in many different applications of deep learning and we know that in general labelled microbiome datasets are limited. One of the main solutions has been to augment the data by various methods, e.g. image manipulations in computer vision, where they improve performance and generalizability.

This has inspired us to look into using a simple augmentation method as a first step to increase generalizability performance of Protonets with data augmentation. For abundance tables already mapped into Euclidean space we adopt an lightweight augmentation analogue: *add zero-mean Gaussian noise whose scale reflects the natural dispersion of each study*.

Since microbiome sequencing data are compositional and the sequencing process involves stochastic sampling, variations in count data are expected. Moreover, many of the zero counts in abundance tables are likely to be technical zeros (undetected taxa due to limited sequencing depth or sub-sampling during preprocessing) rather than true biological absences. Taking this into account, we can see that adding some noise to the data is acceptable and worth looking into.

One concern has been that due to this noise, labels of samples may not be valid anymore. This is the reason why the noise added is a hyperparameter to be optimized.

Algorithm
Here, we first calculate the standard deviation (sd) of each feature per study:

$$\sigma_{study_i} = sd(X_{study_i}) \in \mathbb{R}^d \tag{5.4}$$

where σ is the column-wise standard deviation of the preprocessed training samples belonging to study i and d is the feature dimensionality. In addition, this σ-vector is multiplied by a tunable jitter fraction f to obtain the final noise scale $\tilde{\sigma}_{study_i} = f\sigma_{\mathbf{study_i}}$. The reason for calculating σ study-wise is because the data of each of these studies comes from a different research that causes study specific batch effects and heterogeneity.

The standard deviation of each study in the dataset is used to add some noise (a fraction of this standard deviation) to the data points in the Euclidean space on-the-fly during training, meaning that at every `__getitem__`[3] call the original sample x is perturbed as

$$\tilde{x}_{study_i} = x_{study_i} + \epsilon_{study_i} \qquad \epsilon_{study_i} \sim \mathcal{N}(0, diag(\tilde{\sigma}_{study_i})^2) \tag{5.5}$$

Because the noise is generated at runtime, a single epoch already sees as many independent variants as there are gradient steps, without increasing disk usage.

The fraction of noise added is taken to be a hyperparameter in the experiments which can also be zero, essentially skipping this augmentation step if it hurts performance. In addition, the augmentation is switched off in validation and test phases.

5.3. Classical ML Methods & Suggested Training Strategies

The meta-learning method here is benchmarked against common classical ML models. The main machine learning classifiers used in microbiome research are random forest classifiers. In addition to random forests, we include multi-layer perceptrons and XGBoost classifiers as classical baselines models.

In addition to these classical ML models, a `Dummy` classifier is used as a baseline for them all. The `Dummy` classifier used in the experiments here is one that always outputs the positive class, which in this research represents diseased samples. As the $F1$-score is positive-class focused, an always-positive prediction can look very good for the imbalanced datasets here.

In this section, the different strategies for training these classical models are outlined, which are named the *simple baseline*, *k-shot baseline*, and *meta-learning inspired baseline*. To make the comparison to meta-learning fair, we want a strong baseline model but also one that is as similar as possible with regards to training. The next chapter compares the different strategies in performance and based on this brings focus to only one of them.

5.3.1. Simple Baseline Method

The first baseline method, which is the simplest, is to select only one study (a study gives multiple samples of healthy vs diseased) and to train the baseline model on part of the data of this study and test on the rest of the data. Nested cross-validation is used as visualized in Figure 5.4. Here, the assumption is made that there is no k-shot scenario and $80/20$ `ShuffleSplit`[4] is used for obtaining both the outer train/test set and the inner train/validation sets.

5.3.2. K-Shot Baseline Method

The second type of baseline is trying to imitate only the meta-testing loop where the test set is obtained from one study. As depicted in Figure 5.5, the pipeline is comparable to the pipeline for the simple baseline method with the exception that instead of doing the $80/20$ split, we use k examples per class from the dataset being split for training. This means that the train set will contain only a limited number of examples.

[3]Referring to the `__getitem__` of the torch.utils.data.Dataset class. See `https://docs.pytorch.org/docs/stable/data.html#torch.utils.data.Dataset`

[4]See `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.ShuffleSplit.html`

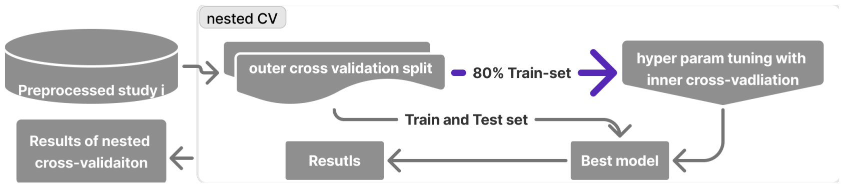

nested CV
Preprocessed study i
outer cross validation split
80% Train-set
hyper param tuning with inner cross-vadliation
Train and Test set
Results of nested cross-validaiton
Resutls
Best model

Figure 5.4: Simple baseline benchmarking pipeline with nested cross-validation.

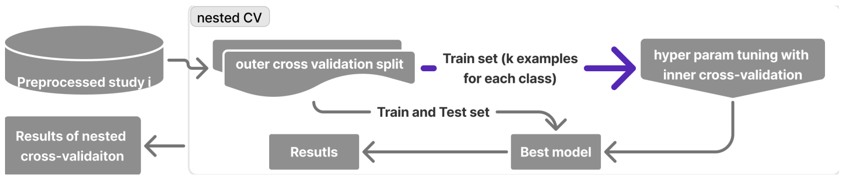

nested CV
Preprocessed study i
outer cross validation split
Train set (k examples for each class)
hyper param tuning with inner cross-validation
Train and Test set
Results of nested cross-validaiton
Resutls
Best model

Figure 5.5: k-shot baseline benchmarking pipeline with nested cross-validation imitating the meta-testing step of meta-learning.

5.3.3. Meta-Learning Inspired Baseline Method

This baseline is designed to let classical tabular models benefit from roughly the same information that a meta-learner sees, while keeping their conventional ”train-once, predict-once” workflow.

First, we apply the usual leave-one-study-out partitioning: one entire study is held out as the test study, all remaining studies form the training pool.

Inside that training pool, we further perform a study-level cross-validation loop to select hyperparameters: on each inner fold we withhold one complete study as a validation study and fit the model on the others.

Where the k examples per class come in.

For every fold, both in the inner (train/validation) loop and later in the outer (train/test) loop, we sample k labelled support examples per class from the held-out study and append them to the global training table.

Both the selected validation studies and the support set of the validation and test studies are preselected and stored to ensure fair comparison across models and reproducibility.

Concretely:

- **Hyperparameter search:** When the validation study is left out, its k samples per class are concatenated to the multi-study training data used to fit the model whose hyperparameters we are about to evaluate. The remaining $n_{\text{val}} - 2k$ samples of that study are kept strictly for validation. Thus, the model trains on all non-validation studies of the train set and the support set of the validation study, and we check generalization on the unseen remainder, the query set in meta-learning terms.
- **Final model fitting:** After choosing the best hyperparameters, we repeat the same procedure with the (previously unseen) test study: its support set is added to the full training table and the rest of its data becomes the test set on which we report performance.

This means that for the meta-learning inspired baseline, the k-shot support examples are always part of the model’s training data. Nothing special happens at inference time; once the model has been fit, it outputs predictions for the remaining data of the test study exactly as an ordinary classifier would.

Key distinction from meta-learning In a meta-learning episode, the support set examples are not folded into a big batch with other studies; instead they form a dedicated support set. The meta-learner carries out a fast adaptation step only on those k samples per class, adjusts its internal parameters or prototypes, and then predicts the labels of the episode’s query set. The adaptation is itself a tiny training run, but it is:

1. **Task-specific**: parameters are nudged in a way that is unique to that study’s distribution.

2. **Discarded afterwards** — the meta-learner reverts to its meta-trained parameters before tackling the next task.

Thus, both the meta-learning inspired pipelines and the meta-learning pipeline see the same k labelled support examples per class from the test study in addition to the whole training data, yet they use them very differently: the baseline sees all the training data including the support set at once for a global fit, whereas the meta-learner uses the support set for adaptation once only before evaluation.

For even fairer comparisons, in addition to cross-validation, the same train/val and support set splits are used for both meta-learning inspired methods and Protonets.

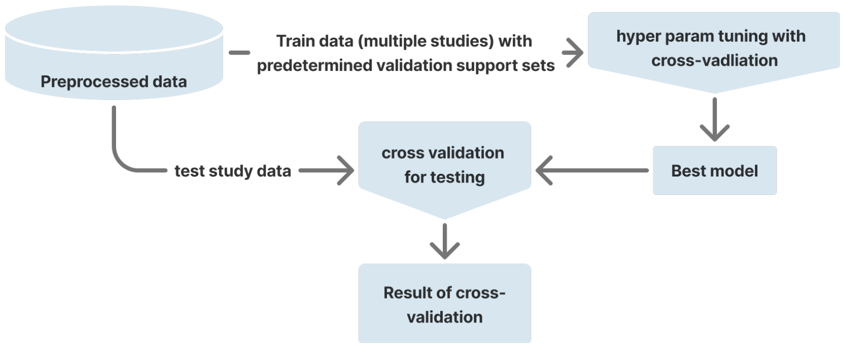

Preprocessed data
Train data (multiple studies) with predetermined validation support sets
hyper param tuning with cross-vadliation
Best model
test study data
cross validation for testing
Result of cross-validation

Figure 5.6: Meta-learning inspired baseline

For all these methods, hyperparameter tuning has been conducted and for the details, the reader is referred to Appendix A for specific details on this.

5.4. Evaluation Metrics & Embedding Analysis Protocols

To judge whether the Protonet embedding network actually adds task-relevant structure beyond the raw abundance vectors, we combine a reproducible embedding extraction pipeline with both qualitative and quantitative evaluations. First, we obtain the embeddings of the paired support and query samples that are used in meta-testing. We then project these embedded representations, alongside their corresponding raw input data, into two dimensions with a uniform UMAP configuration chosen to be fold- and study-agnostic. This side-by-side view allows us to visualize how class geometry changes when the network is applied, while keeping preprocessing, neighbourhood size and random seed strictly controlled.

Experimental protocol

Here, we outline the protocol used to do the analysis of the Protonet embeddings in Subsection 6.5.2.

Embedding source For each outer CV fold on which the Protonet's embedding network is trained, the network maps every point in the test set to a d-dimensional embedding determined by the study specific hyperparameter configuration. For every fold we collect the two disjoint subsets from the meta-testing folds:

- **support set** — the vectors used to compute class prototypes for adaptation to the test study;
- **query set** — the held-out samples from the test study on which Protonet is evaluated.

Fold selection To avoid a figure explosion we show one "representative" fold per study: the fold whose Protonet test $F1$ is closest to the test study's median $F1$ across all outer folds.

Preprocessing Before fitting UMAP, we z-score each feature (mean 0, SD 1) using statistics from the combined support $\cup$ query set and then apply both this standardization and the subsequent UMAP projection to that same pooled set of points.

Projection targets We create two parallel projections per study and fold:

1. **Input space** — the raw input feature vectors that were supplied to the embedding network.
2. **Embedding space** — the d-dimensional Protonet embeddings after scaling.

Both projections use the same unsupervised algorithm, enabling a like-for-like visual comparison of how much structure the embedding network adds beyond the input space.

UMAP settings Each data matrix (input and embedding) is reduced to 2-D using UMAP [45] with

$$k = \lceil \sqrt{N} \rceil, \quad \text{min\_dist} = 0.05, \quad \text{metric} = \text{euclidean}, \quad \text{random\_state} = 42, \tag{5.6}$$

where N is the total number of points (support $+$ query). Choosing $k \propto \sqrt{N}$ is a heuristic that keeps the effective neighborhood radius comparable across studies of very different size. The UMAP model is fitted on (support $\cup$ query) and applied to the same set.

Complementing these plots, we compute a small set of geometry-aware and projection-faithfulness metrics to quantify any gains (or losses) the embedding introduces:

- **Silhouette coefficient (Silh)**: A number between -1 and $+1$ that captures how compact each class is relative to its distance from other classes, in the feature space before the UMAP embeddings.
  - **Higher is better**: values above 0.5 indicate well-separated, globular clusters; values near or below zero signal serious overlap.
- **Trustworthiness (T) & Continuity (C)**: Two scores in the range $[0, 1]$ that tell us how faithfully the low-dimensional plot preserves the high-dimensional neighbourhood structure.
  - **Values above** 0.9 mean the projection is a good cartoon of the original geometry; drops below 0.8 warn that points have been torn apart or squashed together.

6 Results & Discussion

To judge whether meta-learning is a promising avenue for researchers to look further into, it is necessary to benchmark it against models usually used for classifying diseases from microbiome abundance data. Our goal is to characterize how Prototypical networks perform, across data scarce regimes, study heterogeneity, and alternative representation spaces, relative to classical machine learning classifiers.

The main machine learning classifiers used in microbiome research are random forest (RF) classifiers. In addition to RFs, we include multi-layer perceptrons and XGBoost classifiers as classical baselines models.

The chapter is organized as follows. Section 6.1 establishes the best training settings for training classical ML baselines on the gut-microbiome studies, ensuring a fair method for comparison with meta-learning. Section 6.2 provides a concise overview of the full benchmarking pipeline and evaluation protocol. Section 6.3 shows further analysis of the classical models and the Prototypical network meta-learner with regards to shot number and dimensionality reduction. In Section 6.4, the two methods are placed side-by-side on identical data splits, and paired F_1-score differences are subjected to statistical-significance testing to highlight where meta-learning is better, worse, or indistinguishable. The remainder of the chapter tries to analyze this underperformance with a look at the Protonet embeddings and feature similarity of studies.

More detailed visualizations and some extra analysis are supplied in Appendix B.

6.1. Fair Comparison with Meta-Learning Inspired Baseline — with Random Forests on Top

We have introduced multiple strategies for training classical models fairly, since comparing these classical ML models to meta-learning models can easily end up being unfair. Our goal in this section is to decide on a fair baseline strategy that can be used for further analysis and comparison with meta-learning, establishing a baseline that is both strong and as similar as possible to the meta-learning method regarding the way it is trained and tested. The strategies that we will look at are: the *simple baseline*, *k-shot baseline*, and *meta-learning inspired baseline*.

6.1.1. Comparing Classical Training Strategies: Simple, K-Shot, and Meta-Learning Inspired

Here, we analyze which baseline training strategy and model is most suitable for the benchmarking against Protonets.

The meta-learning inspired baseline appears to be the most conceptually similar to meta-learning, as the data used for training and testing is exactly the same. By adding k shots per class from the test study to the classical model's training data, we give it the opportunity to exploit study-specific information exactly once during ordinary fitting. This make the classical model's training more comparable to that of meta-learning models, so that comparisons between the two are as fair as possible. At the same time, because the classical model still ingests all training samples simultaneously, we preserve the standard strengths of tree-based and neural baselines (robustness to noisy features, simple optimization, etc.).

However, aggregating data from multiple studies may introduce too much noise for classical models to learn effectively. Therefore, we benchmark the meta-learning inspired baseline against the k-shot baseline to assess whether such data aggregation negatively impacts performance. Our expectation is that this is not the case and that this k-shot setting puts the model at a big disadvantage as there is almost no data to learn from.

We also include the simple baseline, as it reflects how classical machine learning models are typically used in the field. This method however gives classical models the advantage of seeing many examples from the test study compared to the other methods.

Table 6.1 shows this comparison of the different strategies with different ML models. We can conclude that as expected the k-shot baseline is being outperformed by the meta-learning baseline and that the simple baseline outperforms all baselines in absolute metric values.

However, comparing the simple baseline to the other baselines and meta-learning is unfair due to the way the data is split in this strategy. Therefore, the performance of the positive-class `Dummy` classifier for the different baselines (with meta-learning baseline and k-shot baseline having the same `Dummy`) can be analyzed to see how much the performance improves by using the classical models.

Table 6.1: Performance of baseline training strategies across four classifiers. The given Cells report the `mean±std` $F1$-scores over outer CV folds. Because the Simple baseline uses a random $80/20$ split, it is not directly comparable to the other baselines that test on the query set of the test study. Improvements over the `Dummy` classifier therefore serve as the fairest indicator of a model's added value in this table.

Strategy/Model	`Dummy`	**RF**	**XGBoost**	**MLP**
Meta-learning inspired baseline	0.727±0.10	0.705±0.09	0.673±0.10	0.676±0.12
K-shot baseline	0.727±0.10	0.616±0.12	0.620±0.13	0.597±0.22
Simple baseline	0.720±0.12	0.730±0.15	0.740±0.16	0.737±0.16

Looking at the different strategies, we can conclude that due to the comparable setup of the meta-learning inspired method (with regards to how training is done in meta-learning) and the good performance it shows, it is the right choice for benchmarking against meta-learning.

Subsection B.1.1 shows in-depth results related to comparing different strategies.

Now, looking only at the meta-learning inspired baseline using different classification models, we observe from Table 6.1 and Figure 6.1 that RFs outperform XGBoost and multi-layer perceptrons, which is confirming the field's tendency of using RFs as they show good performance.

Nevertheless, the performance of RF in general is not outstanding, as the `Dummy` classifier is showing comparable results. The `random` classifier is an additional "dummy" classifier that has truly random predictions. This shows that the `Dummy` classifier used in all the experiments has been chosen specifically due its specific strength in the imbalanced data case we have and forms a good dummy baseline for all the other models. More detailed results can be found in Subsection B.1.2.

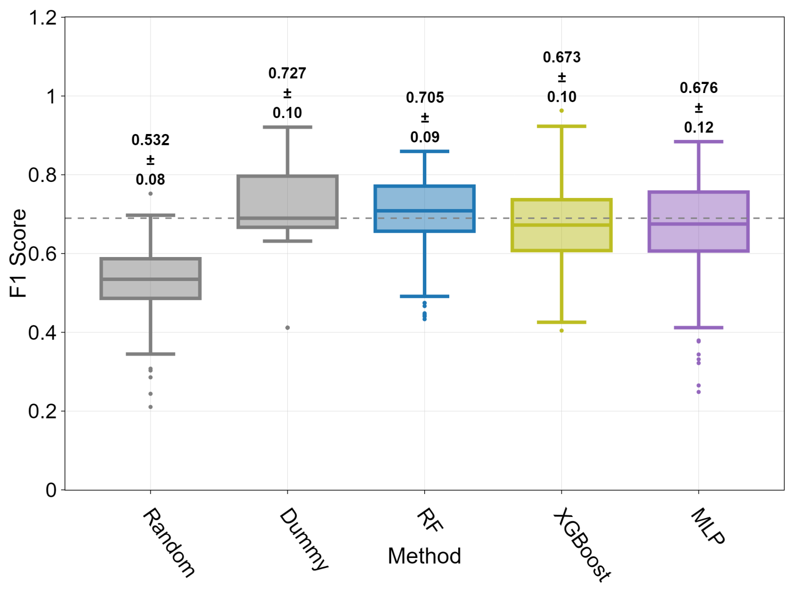


Figure 6.1: Performance of meta-learning inspired baseline method using different classification models for $k = 10$. The corresponding `mean±std` values of are given on top of each box.

Some concern arises from this as many microbiome research focusing on the same kind of classification as here do not report any ”dummy” classifier performance. This is a common issue in the field as discussed in Chapter 3, for examples see [14, 16, 39, 81, 84].

Due to the imbalanced data found in the field and the bias of the metrics like `F1`, `precision`, and `recall` towards the positive-class, high performance scores do not by themselves mean strong performance; making the results reported in many papers hard to interpret, leading to overestimation of model effectiveness, and thus in dire need for an understandable baseline to compare results to. Hence, including a strong, appropriate dummy classifier is critical for future benchmarking studies.

Based on the discussion above, the remainder of this chapter focuses on the meta-learning inspired baseline that uses an RF classifier to simplify comparisons. In addition, whenever useful the performance of the `Dummy` classifier is given to show how well models are learning true biological signals and the performance of the `random` classifier is given to show that learning is happening in general.

6.2. A Unified, Cross-Study Benchmarking Workflow

One of the contributions we have had in mind throughout this project was the setup of an useful benchmarking pipeline, with all the necessary step: data-splitting, training and evaluation. We have introduced three different strategies of training the classical models and in the previous subsection concluded that the meta-learning inspired baseline is what we will be using for benchmarking against meta-learning. For this method and meta-learning itself, we have designed a training and evaluation pipeline.

Figure 6.2 gives a visual overview of the benchmarking pipeline used for the meta-learning inspired baseline and meta-learning methods. We start with the data being split into a meta-train and meta-test set, with only one study in the meta-test set. This study-wise splitting is chosen as it simulates how in the real world a trained model can be shared and used on an unseen study by adaptation and inference.

The next step is to find the right configuration (hyperparameters) to train the model with. For this, the training set is again split in the leave-one-study-out fashion to obtain an inner train and validation set.

This is where meta-learning inspired method and meta-learning diverge paths for the first time. The meta-learning inspired baseline model is trained on the train data plus the support set of the validation

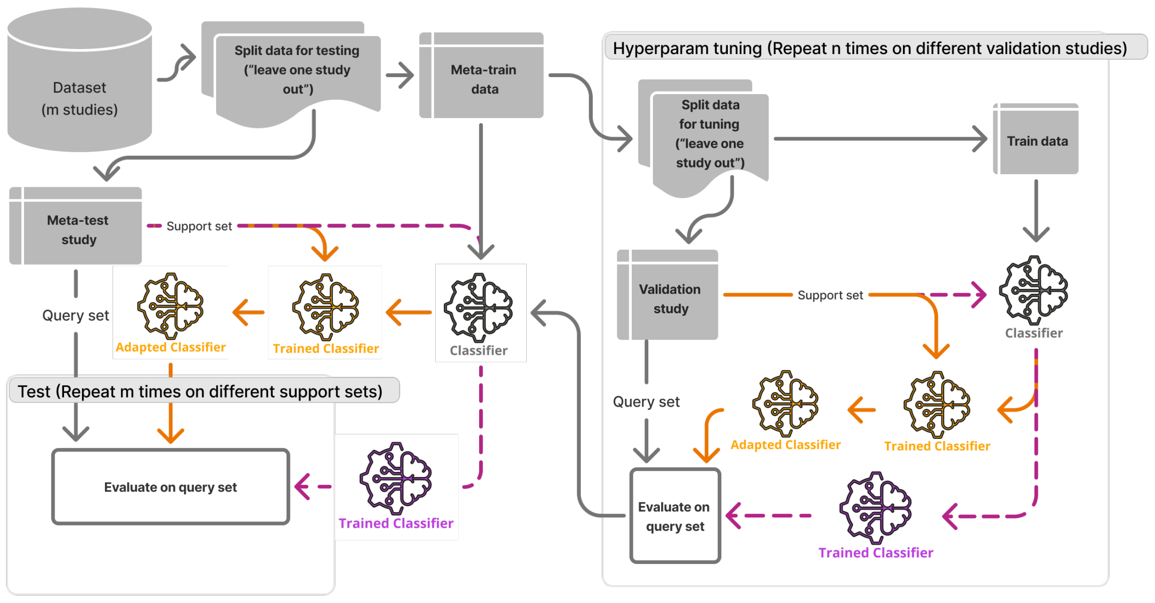


Figure 6.2: High-level overview of the benchmarking pipeline in this thesis. The gray path indicates the shared data path for both meta-learning inspired method and meta-learning itself, while the purple path belongs to meta-learning inspired baseline and the orange path to meta-learning.

data, while the meta-learning method first trains on the train data and afterwards does an adaptation step with the support set from the validation data. The two paths converge when evaluation is done on the left out query set.

During the hyperparameter tuning phase, each set of hyperparameters are validated on multiple validation studies to get more reliable performance estimates, and at the end the best hyperparameter set is used for testing the model

In this testing phase, the two methods diverge paths for a second time. The meta-learning inspired baseline model with the tuned hyperparameters is trained on the meta-train data plus the support set from the meta-test data, while the meta-learning model with tuned hyperparameters is first trained on the meta-train data and afterwards adapted on the support set from the meta-test data. The two paths again converge for testing on the left out query set.

This testing phase is repeated multiple times on the meta-test study with different support sets to obtain more reliable performance estimates.

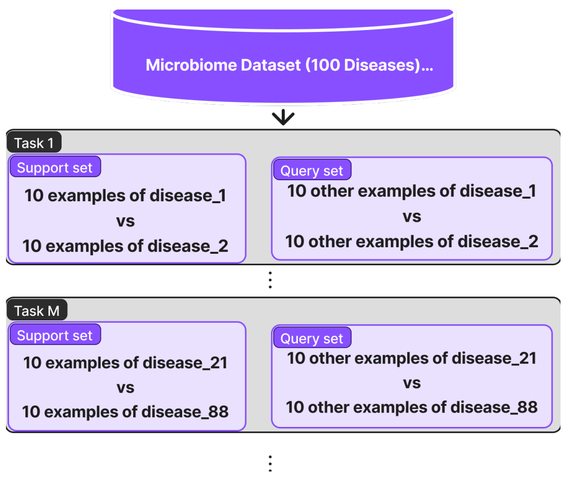
6.2.1. Study-Centric Leave-One-Study-Out Data Splitting

In conventional meta-learning, each task is assembled by first sampling a few classes and then drawing the support and query examples per class from a single, mixed pool of data. That class-centric split is illustrated in Fig.6.3a. To respect batch boundaries in microbiome datasets due to differing sequencing protocols and populations, and to emulate the real scenario of deploying a model to an unseen cohort, we adopt a study-centric leave-one-study-out scheme.

Because of this leave-one-study-out scheme, meta-testing tests the meta-learning algorithm on one unseen study. The unseen study is treated as *one* single held-out task: the model receives its k-shot support set for adaptation and is evaluated on the rest of the data (query set) from the study, reflecting real-world deployment where an analyst can label only a handful of samples from a new cohort.

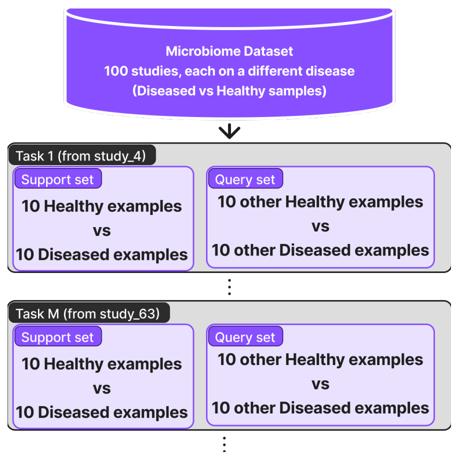
During meta-training, the goal is to learn in a comparable way as meta-testing. The difference here is that the query set is also following the k-shot logic, where k examples per class are sampled each time for the query set.

Figure 6.3 contrasts this study-centric split with the conventional class-centric split in meta-learning, highlighting that what is normally a ”class” becomes an entire study in our setup and each study forms a binary classification problem.



Microbiome Dataset (100 Diseases)...
Task 1
Support set
10 examples of disease_1
vs
10 examples of disease_2
Query set
10 other examples of disease_1
vs
10 other examples of disease_2
Task M
Support set
10 examples of disease_21
vs
10 examples of disease_88
Query set
10 other examples of disease_21
vs
10 other examples of disease_88

(a) Standard data splitting strategy for general meta-learning methods if it were to be applied to our dataset. Tasks are constructed by sampling two classes from a shared pool of samples, and selecting support and query examples from each.



(b) Our approach replaces class-centric task construction with a study-centric one. Each task corresponds to a full study and involves binary classification between healthy and diseased samples. This strategy respects study boundaries and better reflects the real-world deployment scenario of adapting a model to an unseen cohort using a k-shot support set.

Figure 6.3: Comparison between conventional class–centric meta-learning splits and the study-centric splitting strategy used in this work. Visualization of 2-way 10-shot scenario learning during meta-training.

A Note on the Choice of k in the Experiments

In meta-learning, the support set size is an important parameter that influences the performance a lot. To frame the data-scarce reality of microbiome research we benchmark different shot numbers: $k = 2$, $k = 5$, $k = 10$, and $k = 25$. The smaller values represents a genuinely few-shot situation, while the largest one tests whether modestly increasing the labelled budget improves adaptation. Requiring non-overlapping support and query subsets means that not every study qualifies for all regimes due to them being smaller than required:

- 34 of 36 studies meet the 10-shot criterion where each class in the studies should contain at least 20 (10×2) labelled samples;
- and only 20 of 36 studies contain at least 50 (25×2) labelled samples required for the 25-shot split.

6.2.2. Design Choices Ensure Leak-Free, Balanced, and Comparable Data Splits

In this section and in Section 5.2, the exact ways of feeding the data into the models have been described. Altogether, they help us achieve the following points:

- **No hidden information leakage**: In every training episode of the Protonet, only the support is used for the adaptation in the inner loop, and the query set is used for loss calculation and optimization. The same is the case for the baseline method, although the support set is used differently, namely added to the whole training data.
- **Minimal hyperparameters for data sampling**: Apart from the optional `jitter_fraction`, all other behaviour (cycling, shuffling, caching) is deterministic and data-driven, which keeps the search space related to data operations small. The shot number is decided for every experiment beforehand.
- **Data split consistency between models**: With the use of preselected support sets, the testing of different models between the meta-learning method and meta-learning inspired methods can be comparable during testing.
- **Balance across classes within a study**: Because we are looking at few-shot scenarios, splits trained on should be balanced for Protonets (2-way, k-shot). To ensure consistency across tasks,

balancing is used by oversampling. The data can still be shown to be imbalanced by using reweighting as this is necessary for a fair comparison.

6.3. Which Factors Actually Drive Performance?

To better understand what impacts classification performance on microbiome abundance data, we jointly analyze the role of the size of the support set (k-shot) and dimensionality reduction for both baseline and meta-learning models. These two factors reflect core challenges in microbiome research: high-dimensionality and low sample sizes.

Regarding a higher number of shots, the expectation is that the methods will have an improved performance due to the fact that the support set is larger. It is also expected that this also reduces the variability in performance between different cross-validation folds.

Dimensionality reduction methods such as PCA and PCA-guided feature selection offer ways to address the curse of dimensionality and reduce computational cost, while k-shot analysis mirrors real-world constraints where only a limited number of labeled samples per study are available. In this section, we systematically compare how these two factors influence performance of both classical ML models (with the meta-learning inspired RF baseline) and meta-learning models (Protonets), using consistent data splits and evaluation protocols to enable fair comparison. By bringing these analyses together, we aim to clarify whether either approach benefits from dimensionality reduction and whether learning improves with increased data in the support set.

Our expectation is that dimensionality reduction will help with decreasing computational complexity by reducing the number of features trained on while offering the possibility of improved model performance due to noise reduction. One reason from literature is that this should be possible due to the existence of functional redundancies in microbiome data.

Our expectation is that dimensionality reduction will help with decreasing computational complexity by reducing the number of features trained on while offering the possibility of improved model performance due to noise reduction. This should, inter alia, be possible due to the existence of functional redundancies, whereby taxonomically different microbes perform overlapping metabolic roles, so collapsing the feature space is unlikely to discard signals essential for classification.

By isolating the factors analyzed here, we establish a solid reference point and a set of practical insights for future research.

6.3.1. Effect of Support Set Size: Protonets Improve, RFs Plateau

As explained before, the shot number (k) is the support set size of each class. Comparing the influence of different k is important, as more data is expected to reduce the noise learned and improve performance. But in the microbiome field, the amount of data to learn from is hard to increase by a lot, which is our motivation for testing small changes in the data. As indicated, in the 25-shot setting the number of studies tested on is only 20 which means that comparing to other shot numbers should be done on the same 20 test studies. Thus, the groups visualized only contain 20 studies each.

Figure 6.4 shows the performance of the meta-learning inspired baseline and Protonets for different shot numbers.

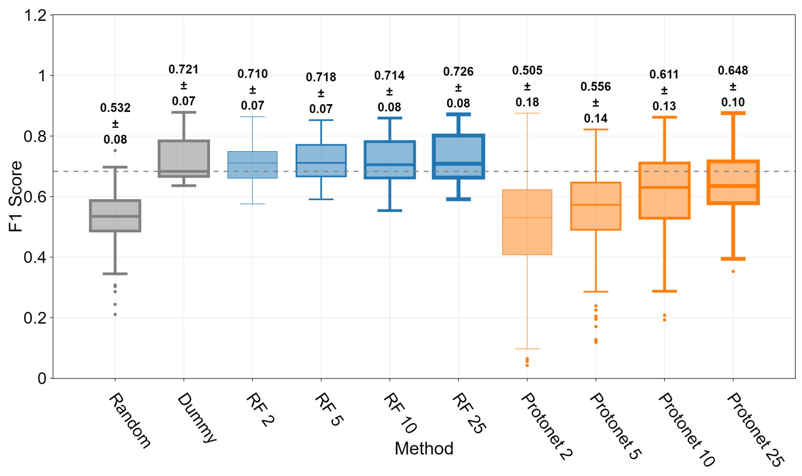


Figure 6.4: CV F1-scores for the meta-learning inspired `Dummy` and RF methods, and Protonets for different k-shots for the same test-studies. The corresponding `mean±std` values are given on top of each box. Number of shots is indicated on the x-axis, with thicker outlines corresponding to higher shot numbers. Each group only contains the 20 test studies tested in the 25-shot setting.

It can be seen that the small increase in k does not change the performance of RF baseline models drastically: the mean CV performance increased from 0.710 to 0.726. However, Protonets show better performance when the shot number increases (namely an increase from 0.505 to 0.648), showing promise for larger datasets. In addition, the variance of the performance is reducing, indicating that more support examples help the model learn more stable and generalizable representations.

We also observe that at really low shot settings (e.g. around 2 shots), Protonets perform worse than even the `random` classifier: mean CV performance of 0.505 vs 0.532 respectively, with a much larger variance for Protonets. This means that adapting on a small number of support set examples is noisy and can often results in harmful inductive bias, pushing the model below chance-level performance.

Since performance differences across shot numbers are minor for the baseline case and small for the 10- and 25-shot case, meta-learning starts performing well around 10-shots, and we have more studies to test on in the 10-shot case, we focus on the 10-shot setting for the remainder of this chapter. And we ignore the 2 and 5-shot cases, as Protonets basically perform like a random classifier.

Our hypothesis still is that higher-shot scenarios will increase performance and this can be tested with more data.

6.3.2. PCA-Based Approaches Improve Classification With Fewer Features

Due to dimensionality reduction being helpful in dealing with high-dimensional, sparse data, we analyze its effect on the performance of meta-learning inspired method and Protonets. For the exploratory rationale behind our PCA settings, see Subsection B.2.2; we focus here on their effect on classification accuracy.

From the discussion in the background and the appendix, dimensionality reduction of microbiome data seems an interesting, promising avenue for the field to reduce computational load and improve performance as microbiome data usually has many more features than samples. Reducing computational load is even more important for meta-learning, as deep learning methods need more resources to train. For this reason, comparing the performance of the methods with different dimensionality reduction techniques can give insights into what additional steps in the preprocessing pipeline can benefit the performance.

Here, we experiment with dimensionality reduction and report the experimental results for the three representations discussed in Section 5.1:

- **Full feature set** – original data;

- **PCA projection** – strong dimensionality reduction, loses direct taxon semantics. Number of principal components is a hyperparameter;
- **PCA-guided feature subset** – interpretable compromise; number of features to select is a hyperparameter (can be at most 1500 so feature reduction has some significance) and number of PCs has not much effect on this and is set to 500 to explain more of the variance of the data.

This setup helps us to find whether dimensionality reduction can help achieve a better classification performance, mainly by getting rid of noise in data, and checking whether computational load can be reduced.

Figure 6.5 shows the performance of the baseline and Protonet models with different dimensionality reduction techniques.

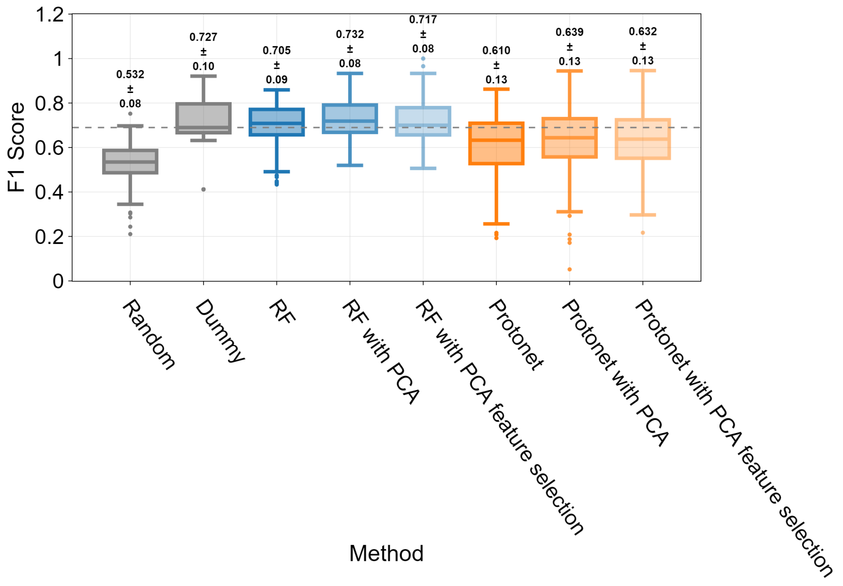


Figure 6.5: CV F1-scores for meta-learning inspired RF methods showing influence of dimensionality reduction on performance (10-shot scenario). The corresponding `mean±std` are given on top of each box. The transparency is related to the dimensionality reduction technique.

As expected, dimensionality reduction seems to improve the classification performance of the RF models, namely from a mean CV performance of 0.705 to 0.732. In the meta-learning case, the same trend is observed, namely an increase of mean CV performance from 0.610 to 0.623. These improvements may be due to reduced noise when the number of features is lowered. In any case, achieving this while using fewer features is computationally attractive.

Overall, dimensionality reduction using PCA improves or maintains performance while significantly reducing feature dimensionality, suggesting that it can lower computational cost without hurting performance. From the analysis in Subsection B.2.2, the PCA-guided feature selection method appeared to provide good coverage only when retaining $90\%+$ of the taxa. Yet our experiments show that high accuracy can already be achieved with fewer than 1500 features (i.e., under 60% coverage, which was set as a hard upper bound during hyperparameter tuning). This indicates that PCA-guided feature selection is also a useful approach for reducing dimensionality.

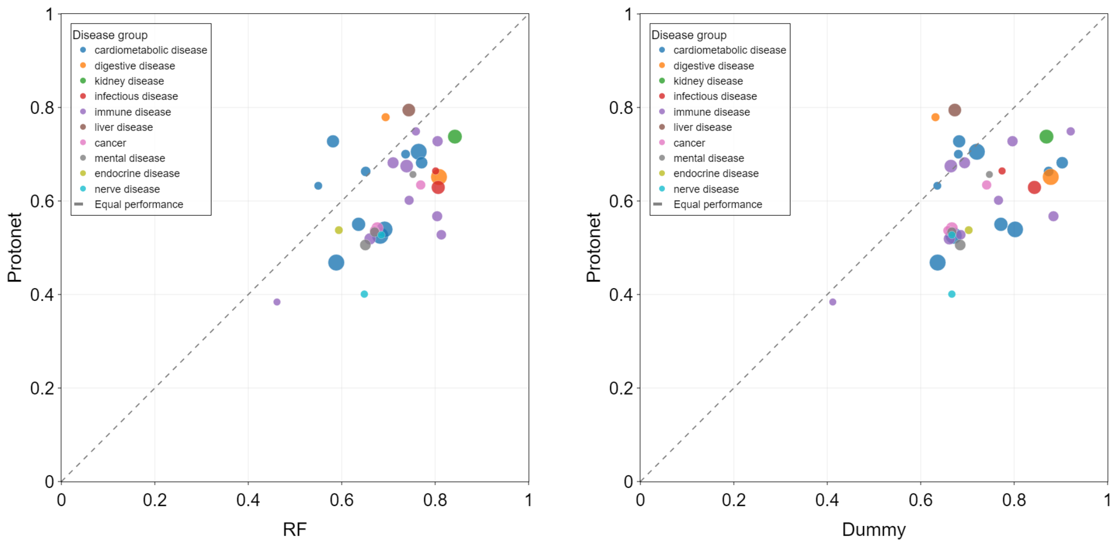
With PCA being a simple dimensionality method, we see the promise of dimensionality reduction methods in general. Based on this, the results from other studies, e.g. [50, 49, 80, 71, 38], and the existence of functional redundancies[6, 70] we see great potential for stronger dimensionality reduction techniques like autoencoders.

These findings highlight practical considerations, such as dimensionality reduction and increasing support set size, that can inform future microbiome modeling efforts.

6.4. Meta-Learning Falls Short of Random Forests

The main research question of this study is whether compared to classical machine learning models, meta-learning can contribute to better learning in classification tasks with microbiome data. The results in the previous section, visualized in Figure 6.4 and Figure 6.5, show a multifaceted comparison between these two methods. More detailed results are given in Appendix B for easier comparison.

We can also look at the study-wise performance of Protonets in comparison to the meta-learning inspired RF and `Dummy` classifier. Figure 6.6 shows that with the exception of some test studies, RF drastically outperforms Protonets by a considerable margin, which looking at the 10-shot setting shows a mean CV performance of 0.705 for RF vs 0.610 for Protonets.



(a) RF vs Protonet in 10-shot case

(b) Dummy vs Protonet in 10-shot case

Figure 6.6: Study-wise performance comparison between Protonet and the meta-learning inspired baselines with RF and `Dummy`. Each dot is the average score of the cross-validation performed. The size of the dots indicate the study size with colors representing different diseases.

Thus, a well-tuned RF still offers the best trade-off between simplicity and performance for microbiome classification.

6.4.1. The Performance Gap Is Statistically Significant

To address the research question whether compared to conventional baseline methods, meta-learning can contribute to better learning in classification tasks with microbiome data, we tested whether Protonet's $F1$ scores are systematically different from those of RF when both are evaluated on the same unseen microbiome studies.

We formalized this as:

- Null hypothesis (H_0): the typical paired difference in $F1$ equals 0.
- Alternative hypothesis (H_1): $\Delta \neq 0$ (two-sided).

A two-sided test was chosen because Protonet could be either better (by exploiting cross-study structure) or worse (e.g. due to overfitting or generalization issues).

The 34 paired $\Delta F1$-values, one for each leave-one-study-out evaluation in 10-shot case, are not guaranteed to be normally distributed, so we used the Wilcoxon signed-rank test[1]. To accompany the p-value

[1]SciPy implementation: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html`

with an interpretable effect size, we computed a bias-corrected bootstrap 95% confidence interval (CI) for the mean. See Section C.1 for details on how this is done.

Results

- Mean $\Delta F1$ (Protonet - RF): 0.0943
- 95% CI: $[-0.1245, -0.0622]$
- p-value: $p = 7 \times 10^{-6}$

The confidence interval lies entirely below zero, and the Wilcoxon p-value is well below the conventional $\alpha = 0.05$ threshold. Hence we reject the null hypothesis that the median paired difference is zero and conclude that Protonet's typical $F1$ score is significantly lower than that of the RF.

The average deficit of 0.09 $F1$ points is large on the $0 - 1$ scale and remains ~ 0.06 even in the most optimistic bootstrap resampling. Combined with the significant Wilcoxon test, this indicates that the observed shortfall is neither a sampling artifact nor driven by a small subset of studies. In practical terms, RF not only outperforms Protonet on average but does so with a margin that is likely to affect downstream biological interpretation

6.5. Why Did Protonets Underperform?

The previous section established that Protonets lag behind the meta-learning inspired RF baseline by 0.09 $F1$ on average, a gap that is statistically significant across all the 34 leave-one-study-out evaluations. The inability of Protonet to match (let alone surpass) the baseline indicates that cross-study information is not being captured effectively with the present architecture, training protocol, and/or dataset.

In this diagnostic chapter, we therefore probe Protonets from multiple complementary angles for better understanding of this underperformance. First, we ablate the encoder altogether, comparing the standard model with a ”no-embedding” variant that uses raw support means as prototypes. We then visualize both input- and embedding-space neighbourhoods with UMAPs, looking at cluster separation with addition of some metrics, and move from there on to looking at how performance may depend on feature space similarity of different studies and disease groups they belong to (see Section D.1 for some info on the disease groups).

By triangulating these perspectives, we obtain a clearer picture of why meta-learning under-delivers here and what ingredients are most likely to close the performance gap in future work.

6.5.1. Embedding Network Adds Little Benefit

Here, we look at how much added value the embeddings from Protonets have. Basically, in one set of experiments we skip the embedding neural network inside the Protonet algorithm and directly get the class representations (prototypes) of the support set from the raw original input data by calculating the mean of the support set per class. The classes of the query samples are predicted by using the raw query samples and finding to which prototype they are closest to. This analysis and the one in the next subsection are meant to show the usefulness of learning an embedding space with Protonets.

Figure 6.7 shows the comparison of Protonets with and without the use of the embedding space. What is surprising is that Protonet without using any embeddings is outperforming the standard Protonet algorithm.

These results suggest that, for the present dataset, prototypes computed directly in the raw input space perform just as well as those derived from the support set. In other words, the additional capacity of the embedding network does not translate into better generalization and may even introduce harmful variance here.

This could be due to data from different studies not sharing enough latent information that is transferable between them, the network overfitting the small support sets during adaptation, the support set being too small and not representative enough for the test set, or the network distorting naturally well-clustered features. The analysis from the next subsection will shed some light on these.

The results here underline two practical points: (i) always include a ”no-embedding” baseline when benchmarking Protonet models, and (ii) when the raw features are low-dimensional or already discriminative, the cost of learning an embedding can outweigh its benefits, unless, perhaps, additional training data or larger support sets ultimately shift the balance.

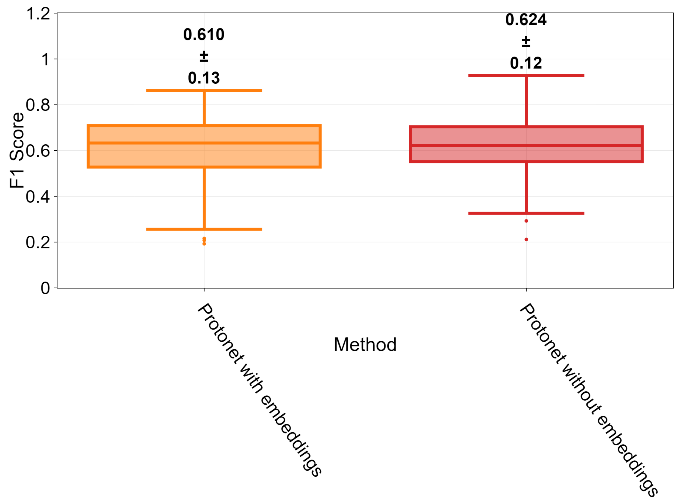


Figure 6.7: Performance comparison of Protonet with and without the embedding network (10-shot case). The corresponding `mean±std` values are given on top of each box. Protonet without embeddings is basically an algorithm that uses the mean of the raw support set as the prototypes for each class and predicts the classes of the query samples with these prototypes based on the raw query samples.

6.5.2. UMAP Projections Reveal Limited Class Separation in the Embedding Space

Seeing the results in the previous subsection, one hypothesis is that the embedding may be distorting naturally well-clustered features for some studies. In here, we would like to take a step into visualizing the data in a 2-dimensional space to get an idea of the change the embedding brings to the clustering of the data. If the encoder adds useful structure, same-class points should contract and different-class clusters should separate more cleanly.

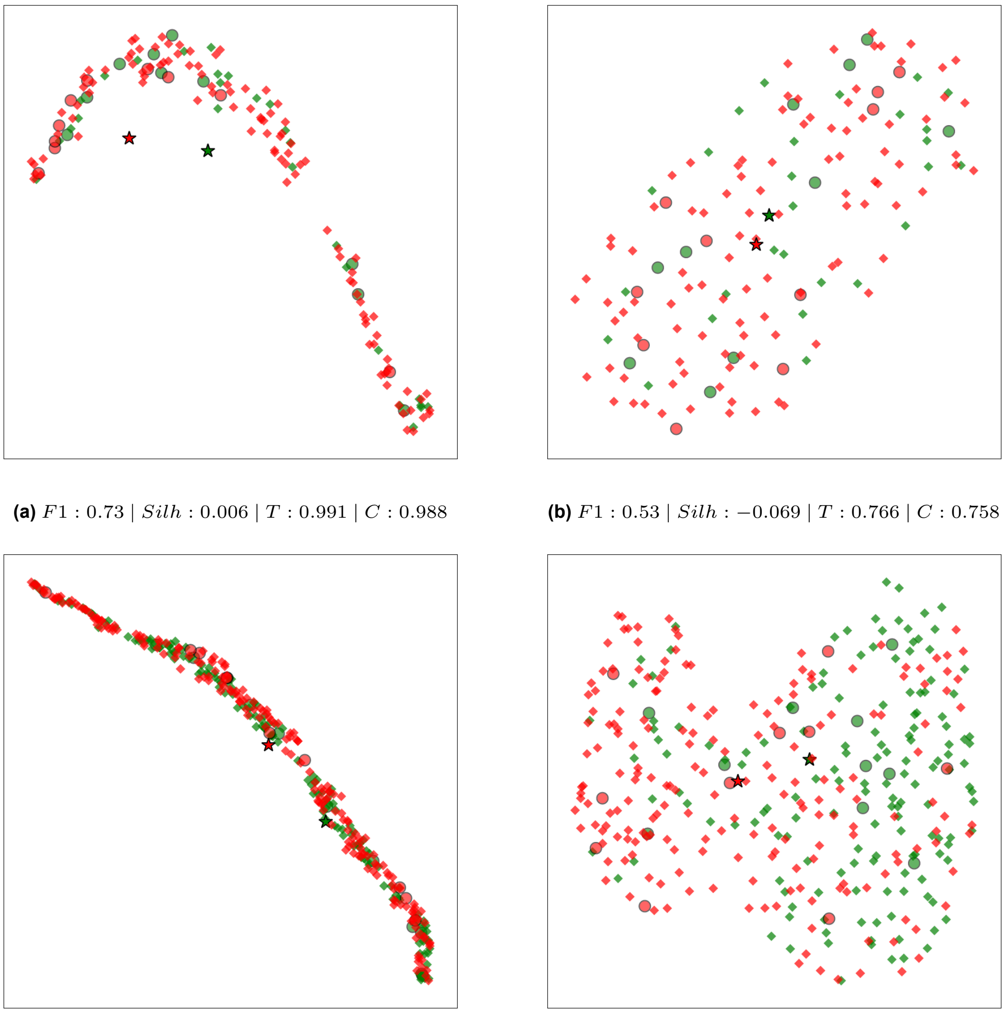
Here, we use Uniform Manifold Approximation and Projection (UMAP), a non-linear manifold learning algorithm that preserves local neighborhoods while providing faithful 2-D layouts for high-dimensional data [45].

For clarity, only a selection of studies is shown here; full results for all 34 studies appear in Appendix B.3.3.

Results of selected studies

The figures here juxtapose input- and embedding-space 2-D UMAPs for the one representative fold for the selected studies. Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars). The reason for separating the support and query set in the visualizations is to get an idea of why the prototypes are where they are and the performance is as it is. In the caption of the sub-figures, the $F1$-score for the corresponding study and analysis metrics are given in addition to the Silhouette coefficient, Trustworthiness , and Continuity.

From the UMAPs, it seems that in most cases the samples in the embedding space are easily pulled apart by the UMAP but that the classification is not invariably improving: the clustering of samples is still a challenge in the embedding space. The Silhouette coefficient indicates this too as on average it shows a decrease when going when original to embedding space (~ -0.003 which is negligible small); but for both the original space and embedding space the Silhouette coefficient is small on average (0.013235 and 0.004 respectively), meaning that overlap in the high dimensional space is substantial.



(a) $F1 : 0.73 \mid Silh : 0.006 \mid T : 0.991 \mid C : 0.988$

(b) $F1 : 0.53 \mid Silh : -0.069 \mid T : 0.766 \mid C : 0.758$

(c) $F1 : 0.64 \mid Silh : 0.012 \mid T : 0.983 \mid C : 0.981$

(d) $F1 : 0.71 \mid Silh : 0.047 \mid T : 0.842 \mid C : 0.837$

Figure 6.8: 2-D UMAP projection of support and query samples (a) `HuangR_2020` (studying immune disease) and (b) `JieZ_2017` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

In addition, the trustworthiness and continuity scores decrease for the embedded samples (on average with ~ -0.3, which means that the embedding space is harder for UMAP to reduce the dimensions of. This can be a sign of the embedding network not creating a cleaner, more ”manifold-like” representations. This by itself doesn’t prove the embedding is useless as it might encode task-relevant, non-local signals that boost a downstream classifier, but it does signal that neighbourhood-level information deteriorated.

We also had the expectation that disease groups with greater representation in the dataset, i.e. those associated with multiple studies (see Section D.1 for overview of disease groups), would benefit from better embeddings compared to underrepresented disease groups with only one or two studies. However, based on all the UMAP results together this was not observed, reinforcing the idea that even similar studies may require distinct embedding spaces. This makes us question how similar feature space similarity affects performance and whether studies from larger disease groups will perform better.

6.5.3. High Feature-Space Similarity $\neq$ High Performance

In Chapter 3, we have already stated that feature space heterogeneity of feature spaces between studies is prevalent for microbiome studies and in the previous subsection seen that even when a disease group with greater representation is being studied, the learned embedding is not always strong. Here, we take a look at whether the feature space similarity between studies can be related to the classification performance.

From Figure 6.9, we can see that the performance of the different models does increase monotonically with higher feature space similarity.

In addition, one expectation was to see disease groups that occur more often in the dataset to have a better summed Jaccard similarity, but from Figure 6.10 we can see that `YanQ_2017` (kidney disease) has a higher Jaccard similarity compared to `LiuP_2021` (cardiometabolic disease). This could be related to our functional redundancy discussion in Subsection 6.3.2, as taxonomic composition can indeed differ a lot but functional effects can be the same.

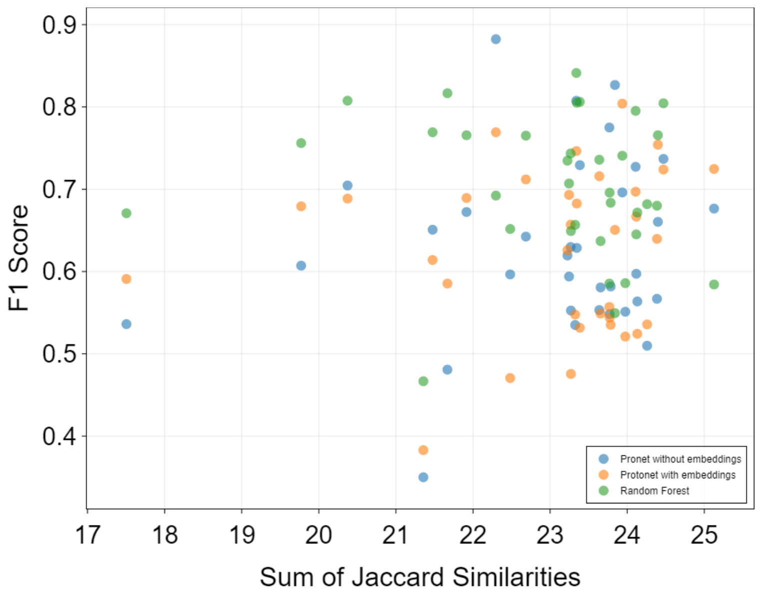
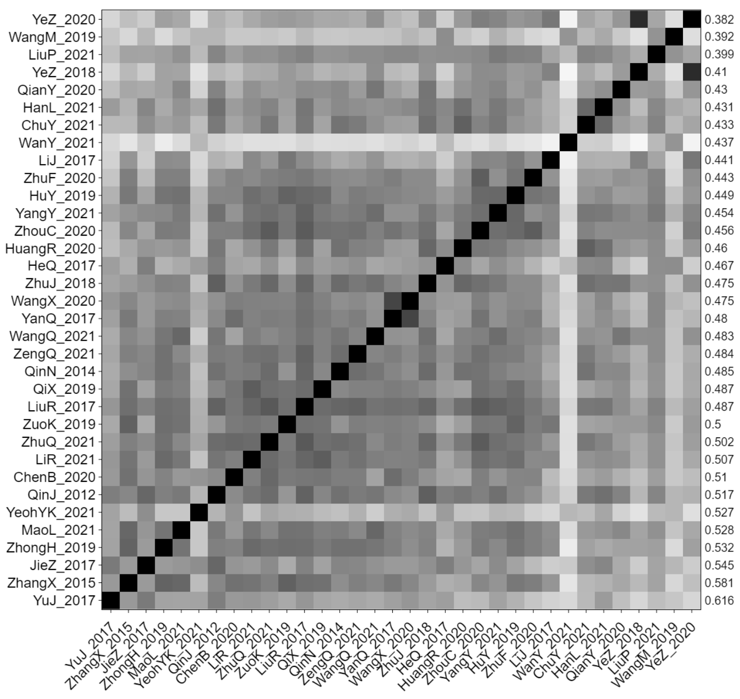
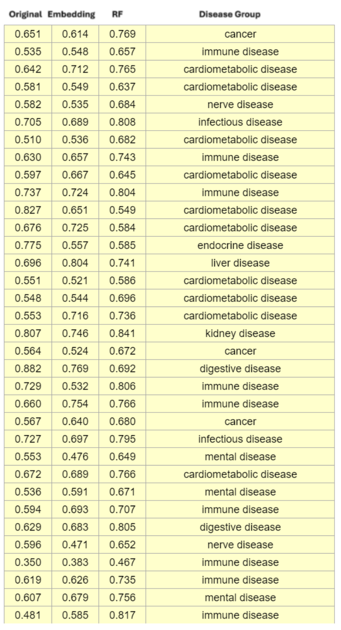


Figure 6.9: $F1$ score of different models for different sum of the Jaccard similarity.

Another lesson from this is that when using data from many different studies, simply taking the union of the feature set of the different studies may be counterproductive as the feature space similarity is low in general and sparsity and dimensionality will increase without adding new information. It may be better to make use of methods able to handle heterogeneous feature spaces, e.g. heterogeneous transfer learning methods [35].

In Subsection B.3.1, a different perspective is given on the above Jaccard similarity analysis.





Original	Embedding	RF	Disease Group
0.651	0.614	0.769	cancer
0.535	0.548	0.657	immune disease
0.642	0.712	0.765	cardiometabolic disease
0.581	0.549	0.637	cardiometabolic disease
0.582	0.535	0.684	nerve disease
0.705	0.689	0.808	infectious disease
0.510	0.536	0.682	cardiometabolic disease
0.630	0.657	0.743	immune disease
0.597	0.667	0.645	cardiometabolic disease
0.737	0.724	0.804	immune disease
0.827	0.651	0.549	cardiometabolic disease
0.676	0.725	0.584	cardiometabolic disease
0.775	0.557	0.585	endocrine disease
0.696	0.804	0.741	liver disease
0.551	0.521	0.586	cardiometabolic disease
0.548	0.544	0.696	cardiometabolic disease
0.553	0.716	0.736	cardiometabolic disease
0.807	0.746	0.841	kidney disease
0.564	0.524	0.672	cancer
0.882	0.769	0.692	digestive disease
0.729	0.532	0.806	immune disease
0.660	0.754	0.766	immune disease
0.567	0.640	0.680	cancer
0.727	0.697	0.795	infectious disease
0.553	0.476	0.649	mental disease
0.672	0.689	0.766	cardiometabolic disease
0.536	0.591	0.671	mental disease
0.594	0.693	0.707	immune disease
0.629	0.683	0.805	digestive disease
0.596	0.471	0.652	nerve disease
0.350	0.383	0.467	immune disease
0.619	0.626	0.735	immune disease
0.607	0.679	0.756	mental disease
0.481	0.585	0.817	immune disease

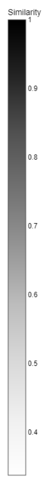


Figure 6.10: Jaccard similarity of the feature space between studies. The Jaccard similarity of a study (on the row) between the union of the feature space of all the other studies is given on the right y-axis. The table gives the $F1$-scores closest to the median $F1$-score per study for the Protonet without (original), Protonet with embedding, and meta-learning inspired RF. In addition, the disease group the study belongs to is given.

7 Conclusion and Future Work

This research aimed to benchmark the classification performance of Prototypical networks (a meta-learning method) against classical machine learning models (RFs, XGBoost, and MLPs) on microbiome taxonomic abundance data. Motivated by the claims that meta-learning enables better performance when there is data scarcity and when the environment is changing, we began with the expectation that Protonets would outperform classical machine learning models in microbiome classification tasks.

Along the way towards this goal, we have tried to find out how some other factors influence the classification performance of the models, to create an experimental setup useful for the benchmarking considered, and find practical steps that can be helpful to researchers in the field.

Based on initial experiments with classical machine learning models, we proposed a meta-learning inspired baseline strategy to enable fair benchmarking against meta-learning (and related transfer learning) methods. With this, we have implemented data sampling utilities useful for meta-learning within the context of multi-study microbiome data, see Section 5.2.

We have also confirmed the strong performance of the widely used RF models and emphasized the necessity for the use of an appropriate dummy classifier as a baseline to compare any model against.

Before explicit comparison of the meta-learning inspired baseline with Protonets, our analysis of the influence of support set size (shot number) showed no gain for the baseline model in the low shot settings considered (namely 2, 5, 10, and 25 shot scenarios); but for Protonets both accuracy and stability improved as expected. Our believe still is that higher shot settings can improve both the performance of the baseline and meta-learning methods.

From our analysis on dimensionality reduction, the conclusion is that simple PCA already reduces dimensionality with minimal performance loss (and sometimes even gains), easing computational burden. We are also able to keep interpretability by using PCA-guided feature selection. Based on this, the results from other studies (e.g. [50, 49, 80, 71, 38]) and the existence of functional redundancies[6, 70], our recommendation is to look further into strong dimensionality reduction techniques like autoencoders.

To return to our central benchmarking question of this research, where we asked ”Does meta-learning with Prototypical networks outperform classical machine learning models in few-shot microbiome classification?”, our experiments offer a clear, data-driven answer. Across all leave-one-study-out evaluations, shot regimes (2, 5, 10, 25) and representation settings (full, PCA projection, PCA-guided feature subset), Prototypical Networks consistently fell short of a well-tuned RF baseline. We have analyzed this further and found some points of consideration:

- **Encoder utility of Protonets is marginal at best:** Ablation showed that removing the Protonet embedding network altogether did not change the performance significantly. This implies that, given current data scale and heterogeneity, the learned representation does not add discriminative power and sometimes distorts naturally separable structures.

- **Size of the support set is a hard bottleneck:** Protonet accuracy and variance improved steadily from 2- to 25-shot settings, yet still fell short of RF. RFs already performed well at low shot settings and did not improve drastically, giving them a practical edge in truly low-shot settings.
- **Cross-study taxonomic overlap is not a free lunch:** Jaccard similarity between a test study and the training pool did not correlate with Protonet gains. Even studies sharing many taxa with the training set, saw no uplift, indicating that naively pooling heterogeneous feature spaces is insufficient; domain alignment or techniques able to handle heterogeneous feature spaces may be required.

We also hope that the benchmarking pipeline proposed here can serve as the foundation for a community-wide standard: one that can be iteratively improved, extended to new data modalities, and adopted as a reproducible reference framework for future microbiome ML studies.

7.1. Limitations and Future Work

While this thesis has provided a systematic benchmark of meta-learning versus classical machine learning on microbiome data, several factors constrain the generalizability and interpretability of the results:

1. **Gut-only data modality**

   This study focused exclusively on taxonomic abundance profiles derived from gut microbiome samples. However, knowing that co-occurence patterns and functional redundancies exist, and that with enough data shared fundamental patterns of microbiome can be found, extending data to include other biomes, and with increase in size, is an important step to take.

2. **Low sample sizes**

   Despite the aggregated 36 studies, the number of labeled samples per disease cohort were small. The small per study sample sizes exacerbate class imbalance, which can skew model training and evaluation (especially in few-shot regimes). We have tried to deal with this problem during evaluations by cross-validation and use of a dummy classifier, but this does not help improve training. In addition, due to the heterogeneity in taxa representation, our models may not fully capture the shared information if data is limited.

3. **Limited computational budget**

   Resource constraints restricted the scope of our hyperparameter optimization. With only a modest Optuna trial budget, we may not have fully explored the parameter spaces of classical models and more importantly Protonets, which are especially sensitive to tuning and have a larger hyperparameter space. This ”undertuning” could have narrowed the observed performance gap between baselines and meta-learners, as Protonets had a more complex hyperparameter space to tune.

4. **Restriction to few-shot scenarios**

   Our experiments concentrated on low-shot conditions (2, 5, 10, and 25 support examples) under the assumption that meta-learning excels compared to other methods when data are scarce. However, the performance trajectories in higher-shot regimes remain unexplored here; it is possible that both classical and meta-learning methods behave differently as sample sizes increase.

Building on the limitations, several concrete research avenues emerge that could strengthen both the generalizability and the biological insight of few-shot microbiome classifiers. First, the present benchmark is constrained to taxonomic tables from the human gut. Broadening the corpus to encompass other body sites (oral, skin, vaginal) and even non-human or environmental biomes, paired with more data, would introduce different community structures, helping a meta-learner disentangle study-specific noise from transferable biological patterns.

Second, our experiments showed that simple dimensionality reduction techniques are already promising. Future work should therefore explore more complex dimensionality reduction techniques, such as variational or denoising autoencoders, that respect the compositional constraints of abundance data and maybe use some prior information. Such representations could supply both classical and meta-learning models with more meaningful features without sacrificing interpretability and treat high dimensionality problems.

A final avenue worth exploring is the use of self-supervised learning to leverage the large volume of unlabeled microbiome data. Recent methods such as STUNT[48], which introduces self-supervision specifically for tabular data, and microbial representation learning approaches trained on millions of unlabeled profiles[56], have shown that it is possible to extract useful structure and transferable features without requiring manual labels. Applying such methods to microbiome data could help overcome the current limitations posed by small labeled datasets and improve downstream classification performance in low-data regimes.

References

[1] Laith Alzubaidi et al. “A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications”. In: *Journal of Big Data* 10.1 (2023), p. 46.

[2] Steven Benowitz. *Human microbiome meeting highlights research progress in a field already beginning to matter*. url: `https://www.genome.gov/27554771/human-microbiome-meeting-highlights-research-progress-in-a-field-thats-already-beginning-to-matter#:~:text=trillions%20of%20microorganisms%20that%20inhabit,microbes%20cause%20sickness%20and%20disease`. (accessed: 13.01.2025).

[3] Sebastiano Busato et al. “Compositionality, sparsity, spurious heterogeneity, and other data-driven challenges for machine learning algorithms within plant microbiome studies”. In: *Current opinion in plant biology* 71 (2023), p. 102326.

[4] Samuel Chaffron et al. “A global network of coexisting microbes from environmental and whole-genome sequence data”. In: *Genome research* 20.7 (2010), pp. 947–959.

[5] Nitesh V Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[6] Huaihai Chen et al. “Functional redundancy in soil microbial community based on metagenomics across the globe”. In: *Frontiers in microbiology* 13 (2022), p. 878978.

[7] Hui Chong et al. “EXPERT: transfer learning-enabled context-aware microbial community classification”. In: *Briefings in Bioinformatics* 23.6 (2022). issn: 1467-5463 1477-4054. doi: `10.1093/bib/bbac396`.

[8] U. Gülfem Elgün Çiftcioğlu and O. Ufuk Nalbanoglu. “DeepGum: Deep feature transfer for gut microbiome analysis using bottleneck models”. In: *Biomedical Signal Processing and Control* 91 (2024). issn: 17468094. doi: `10.1016/j.bspc.2024.105984`.

[9] John F Cryan and Timothy G Dinan. “Mind-altering microorganisms: the impact of the gut microbiota on brain and behaviour”. In: *Nature reviews neuroscience* 13.10 (2012), pp. 701–712.

[10] John F Cryan et al. “The gut microbiome in neurological disorders”. In: *The Lancet Neurology* 19.2 (2020), pp. 179–194. issn: 1474-4422. doi: `https://doi.org/10.1016/S1474-4422(19)30356-4`. url: `https://www.sciencedirect.com/science/article/pii/S1474442219303564`.

[11] John F. Cryan et al. “The Microbiota-Gut-Brain Axis”. In: *Physiological Reviews* 99.4 (2019), pp. 1877–2013. issn: 0031-9333. doi: `10.1152/physrev.00018.2018`.

[12] M. M. David et al. “Revealing General Patterns of Microbiomes That Transcend Systems: Potential and Challenges of Deep Transfer Learning”. In: *Msystems* 7.1 (2022). Zh2le Times Cited:3 Cited References Count:47. issn: 2379-5077. doi: `ARTNe01058-2110.1128/msystems.01058-21`. url: `HYPERLINK%20%22http://gateway.isiknowledge.com/gateway/Gateway.cgi?GWVersion=2&SrcAuth=ResearchSoft&SrcApp=EndNote&DestLinkType=FullRecord&DestApp=WOS&KeyUT=000760774900006%22%3CGo%20to%20ISI%3E://WOS:000760774900006`.

[13] Maude M. David et al. “Revealing General Patterns of Microbiomes That Transcend Systems: Potential and Challenges of Deep Transfer Learning”. In: *mSystems* 7.1 (2022), e01058–21. doi: `10.1128/msystems.01058-21`. eprint: `https://journals.asm.org/doi/pdf/10.1128/msystems.01058-21`. url: `https://journals.asm.org/doi/abs/10.1128/msystems.01058-21`.

[14] Francesca De Filippis et al. “Specific gut microbiome signatures and the associated pro-inflamatory functions are linked to pediatric allergy and acquisition of immune tolerance”. In: *Nature communications* 12.1 (2021), p. 5958.

[15] Natasha Katherine Dudek et al. “Supervised machine learning for microbiomics: bridging the gap between current and best practices”. In: *Machine Learning with Applications* (2024), p. 100607.

[16] Claire Duvallet et al. “Meta-analysis of gut microbiome studies identifies disease-specific and shared responses”. In: *Nature communications* 8.1 (2017), p. 1784.

[17] Sergey Feranchuk et al. “Evaluating the use of diversity indices to distinguish between microbial communities with different traits”. In: *Research in Microbiology* 169.4-5 (2018), pp. 254–261.

[18] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[19] Chelsea Finn and Sergey Levine. “Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm”. In: *CoRR* abs/1710.11622 (2017). arXiv: `1710.11622`. url: `http://arxiv.org/abs/1710.11622`.

[20] Jane A Foster and Karen-Anne McVey Neufeld. “Gut–brain axis: how the microbiome influences anxiety and depression”. In: *Trends in neurosciences* 36.5 (2013), pp. 305–312.

[21] Jessica Galloway-Peña and Blake Hanson. “Tools for Analysis of the Microbiome”. In: *Digestive diseases and sciences* 65 (2020), pp. 674–685.

[22] Victor Garcia and Joan Bruna. *Few-Shot Learning with Graph Neural Networks*. 2018. arXiv: `1711.04043 [stat.ML]`. url: `https://arxiv.org/abs/1711.04043`.

[23] Gregory B Gloor et al. “Microbiome datasets are compositional: and this is not optional”. In: *Frontiers in microbiology* 8 (2017), p. 2224.

[24] W. W. B. Goh, W. Wang, and L. Wong. “Why Batch Effects Matter in Omics Data, and How to Avoid Them”. In: *Trends Biotechnol* 35.6 (2017), pp. 498–507. issn: 1879-3096 (Electronic) 0167-7799 (Linking). doi: `10.1016/j.tibtech.2017.02.012`. url: `https://www.ncbi.nlm.nih.gov/pubmed/28351613`.

[25] Wilson Wen Bin Goh, Chern Han Yong, and Limsoon Wong. “Are batch effects still relevant in the age of big data?” In: *Trends in Biotechnology* 40.9 (2022), pp. 1029–1040. issn: 0167-7799. doi: `https://doi.org/10.1016/j.tibtech.2022.02.005`. url: `https://www.sciencedirect.com/science/article/pii/S0167779922000361`.

[26] Jacopo Grilli, Christine A. Tataru, and Maude M. David. “Decoding the language of microbiomes using word-embedding techniques, and applications in inflammatory bowel disease”. In: *PLOS Computational Biology* 16.5 (2020). issn: 1553-7358. doi: `10.1371/journal.pcbi.1007859`.

[27] H Soon Gweon et al. “The impact of sequencing depth on the inferred taxonomic composition and AMR gene content of metagenomic samples”. In: *Environmental Microbiome* 14.1 (2019), pp. 1–15.

[28] National Institutes of Health. *Science Highlights*. url: `https://commonfund.nih.gov/human-microbiome-hmp/science-highlights`. (accessed: 08.01.2025).

[29] Ricardo Hernández Medina et al. “Machine learning and deep learning applications in microbiome research”. In: *ISME Communications* 2.1 (2022). issn: 2730-6151. doi: `10.1038/s43705-022-00182-9`.

[30] Asmaul Hosna et al. “Transfer learning: a friendly introduction”. In: *Journal of Big Data* 9.1 (2022). issn: 2196-1115. doi: `10.1186/s40537-022-00652-w`.

[31] Timothy M Hospedales et al. “Meta-Learning in Neural Networks: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. issn: 0162-8828. doi: `10.1109/tpami.2021.3079209`.

[32] Kaijian Hou et al. “Microbiota in health and diseases”. In: *Signal transduction and targeted therapy* 7.1 (2022), p. 135.

[33] Mike Huisman, Jan N. Van Rijn, and Aske Plaat. “A survey of deep meta-learning”. In: *Artificial Intelligence Review* 54.6 (2021), pp. 4483–4541. issn: 0269-2821. doi: `10.1007/s10462-021-10004-4`.

[34] Yanrong Ji et al. “DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome”. In: *Bioinformatics* 37.15 (2021), pp. 2112–2120.

[35] Siraj Khan et al. “Heterogeneous transfer learning: Recent developments, applications, and challenges”. In: *Multimedia Tools and Applications* 83.27 (2024), pp. 69759–69795.

[36] Rob Knight et al. “Best practices for analysing microbiomes”. In: *Nature Reviews Microbiology* 16.7 (2018), pp. 410–422. issn: 1740-1526. doi: 10.1038/s41579-018-0029-9. url: https://www.nature.com/articles/s41579-018-0029-9.pdf.

[37] Bablu Kumar et al. “A comprehensive overview of microbiome data in the light of machine learning applications: categorization, accessibility, and future directions”. In: *Frontiers in microbiology* 15 (2024), p. 1343572.

[38] Vuong Le et al. “Deep in the bowel: highly interpretable neural encoder-decoder networks predict gut metabolites from gut microbiome”. In: *BMC genomics* 21 (2020), pp. 1–15.

[39] Karla A Lee et al. “Cross-cohort gut microbiome associations with immune checkpoint inhibitor response in advanced melanoma”. In: *Nature medicine* 28.3 (2022), pp. 535–544.

[40] Wodan Ling et al. “Batch effects removal for microbiome data via conditional quantile regression”. In: *Nature communications* 13.1 (2022), p. 5418.

[41] Catherine A Lozupone et al. “Diversity, stability and resilience of the human gut microbiota”. In: *Nature* 489.7415 (2012), pp. 220–230.

[42] Jingjie Luo et al. “Meta-learning with elastic prototypical network for fault transfer diagnosis of bearings under unstable speeds”. In: *Reliability Engineering System Safety* 245 (2024). issn: 09518320. doi: 10.1016/j.ress.2024.110001.

[43] Bin Ma et al. “Earth microbial co-occurrence network reveals interconnection pattern across microbiomes”. In: *Microbiome* 8 (2020), pp. 1–12.

[44] Laura Judith Marcos-Zambrano et al. “Applications of Machine Learning in Human Microbiome Studies: A Review on Feature Selection, Biomarker Identification, Disease Prediction and Treatment”. In: *Frontiers in Microbiology* 12 (2021). issn: 1664-302X. doi: 10.3389/fmicb.2021.634511.

[45] Leland McInnes et al. “UMAP: Uniform Manifold Approximation and Projection”. In: *The Journal of Open Source Software* 3.29 (2018), p. 861.

[46] Donald T. Mcknight et al. “Methods for normalizing microbiome data: An ecological perspective”. In: *Methods in Ecology and Evolution* 10.3 (2019), pp. 389–400. issn: 2041-210X. doi: 10.1111/2041-210x.13115.

[47] Alexander Moore and Max Bell. “XGBoost, a novel explainable AI technique, in the prediction of myocardial infarction: a UK Biobank cohort study”. In: *Clinical Medicine Insights: Cardiology* 16 (2022), p. 11795468221133611.

[48] Jaehyun Nam et al. *STUNT: Few-shot Tabular Learning with Self-generated Tasks from Unlabeled Tables*. 2023. arXiv: 2303.00918 [cs.LG]. url: https://arxiv.org/abs/2303.00918.

[49] Min Oh and Liqing Zhang. “DeepGeni: Deep generalized interpretable autoencoder elucidates gut microbiota for better cancer immunotherapy”. In: *Scientific Reports* 13.1 (2023), p. 4599.

[50] Min Oh and Liqing Zhang. “DeepMicro: deep representation learning for disease prediction based on microbiome data”. In: *Scientific reports* 10.1 (2020), p. 6026.

[51] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. “Tadam: Task dependent adaptive metric for improved few-shot learning”. In: *Advances in neural information processing systems* 31 (2018).

[52] Georgios Papoutsoglou et al. “Machine learning approaches in microbiome research: challenges and best practices”. In: *Frontiers in microbiology* 14 (2023), p. 1261889.

[53] Nikos Pechlivanis et al. “Microbial co-occurrence network demonstrates spatial and climatic trends for global soil diversity”. In: *Scientific Data* 11.1 (2024), p. 672.

[54] Huimin Peng. “A comprehensive overview and survey of recent advances in meta-learning”. In: *arXiv preprint arXiv:2004.11149* (2020).

[55] Quintin Pope et al. “Learning a deep language model for microbiomes: the power of large scale unlabeled microbiome data”. In: *bioRxiv* (2023). doi: 10.1101/2023.07.17.549267.

[56] Quintin Pope et al. “Learning a deep language model for microbiomes: the power of large scale unlabeled microbiome data”. In: *PLOS Computational Biology* 21.5 (2025), e1011353.

[57] Fernando Puente-Sánchez et al. “Cross-biome microbial networks reveal functional redundancy and suggest genome reduction through functional complementarity”. In: *Communications biology* 7.1 (2024), p. 1046.

[58] Timothy G Raben et al. “Biobank-scale methods and projections for sparse polygenic prediction from machine learning”. In: *Scientific Reports* 13.1 (2023), p. 11662.

[59] Ravi Ranjan et al. “Analysis of the microbiome: Advantages of whole genome shotgun versus 16S amplicon sequencing”. In: *Biochemical and biophysical research communications* 469.4 (2016), pp. 967–977.

[60] L. Richardson et al. “MGnify: the microbiome sequence data analysis resource in 2023”. In: *Nucleic Acids Res* 51.D1 (2023), pp. D753–D759. doi: 10.1093/nar/gkac1080. url: https://www.ncbi.nlm.nih.gov/pubmed/36477304.

[61] Lorna Richardson et al. “MGnify: the microbiome sequence data analysis resource in 2023”. In: *Nucleic acids research* 51.D1 (2023), pp. D753–D759.

[62] Stefano Romano et al. “Machine learning-based meta-analysis reveals gut microbiome alterations associated with Parkinson’s disease”. In: *Nature Communications* 16.1 (2025), pp. 1–17.

[63] Irwin Ross et al. “Chemformer: a pre-trained transformer for computational chemistry”. In: *Machine Learning : Science and Technology* 3.1 (Mar. 2022). Copyright - © 2022 The Author(s). Published by IOP Publishing Ltd. This work is published under http://creativecommons.org/licenses/by/4.0 (the “License”). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2024-10-16, p. 015022.

[64] Robert F Schwabe and Christian Jobin. “The microbiome and cancer”. In: *Nature Reviews Cancer* 13.11 (2013), pp. 800–812.

[65] Justin P Shaffer et al. “Standardized multi-omics of Earth’s microbiomes reveals microbial and metabolite diversity”. In: *Nature microbiology* 7.12 (2022), pp. 2128–2150.

[66] Gil Sharon et al. “The central nervous system and the gut microbiome”. In: *Cell* 167.4 (2016), pp. 915–932.

[67] Jake Snell, Kevin Swersky, and Richard Zemel. “Prototypical networks for few-shot learning”. In: *Advances in neural information processing systems* 30 (2017).

[68] Wen Sun et al. “A population-scale analysis of 36 gut microbiome studies reveals universal species signatures for common diseases”. In: *npj Biofilms and Microbiomes* 10.1 (2024). issn: 2055-5008. doi: 10.1038/s41522-024-00567-9.

[69] Marco Teixeira et al. “A review of machine learning methods for cancer characterization from microbiome data”. In: *NPJ Precision Oncology* 8.1 (2024), p. 123.

[70] Liang Tian et al. “Deciphering functional redundancy in the human microbiome”. In: *Nature communications* 11.1 (2020), p. 6217.

[71] Hanaa Torkey et al. “A novel deep autoencoder based survival analysis approach for microarray dataset”. In: *PeerJ Computer Science* 7 (2021), e492.

[72] Eleni Triantafillou et al. “Meta-dataset: A dataset of datasets for learning to learn from few examples”. In: *arXiv preprint arXiv:1903.03096* (2019).

[73] Peter J Turnbaugh et al. “A core gut microbiome in obese and lean twins”. In: *nature* 457.7228 (2009), pp. 480–484.

[74] Peter J Turnbaugh et al. “An obesity-associated gut microbiome with increased capacity for energy harvest”. In: *nature* 444.7122 (2006), pp. 1027–1031.

[75] Anna Vettoruzzo et al. “Advances and Challenges in Meta-Learning: A Technical Review”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.7 (2024), pp. 4763–4779. issn: 0162-8828. doi: 10.1109/tpami.2024.3357847.

[76] Benyou Wang et al. “Pre-trained language models in biomedical domain: A systematic survey”. In: *ACM Computing Surveys* 56.3 (2023), pp. 1–52.

[77] Haifeng Wang et al. “Pre-Trained Language Models and Their Applications”. In: *Engineering* 25 (2023), pp. 51–65. issn: 2095-8099. doi: https://doi.org/10.1016/j.eng.2022.04.024. url: https://www.sciencedirect.com/science/article/pii/S2095809922006324.

[78] Yiwen Wang and Kim-Anh Lêcao. “Managing batch effects in microbiome data”. In: *Briefings in Bioinformatics* 21.6 (2020), pp. 1954–1970. issn: 1477-4054. doi: 10.1093/bib/bbz105.

[79] Sophie Weiss et al. “Normalization and microbial differential abundance strategies depend upon data characteristics”. In: *Microbiome* 5 (2017), pp. 1–18.

[80] Dinithi Wickramaratne, Rupika Wijesinghe, and Ruvan Weerasinghe. “Human gut microbiome data analysis for disease likelihood prediction using autoencoders”. In: *2021 21st International Conference on Advances in ICT for Emerging Regions (ICter)*. IEEE. 2021, pp. 49–54.

[81] Yuanqi Wu et al. “Identification of microbial markers across populations in early detection of colorectal cancer”. In: *Nature communications* 12.1 (2021), p. 3063.

[82] Yijia Xu, Yuchi Ma, and Zhou Zhang. “Self-supervised pre-training for large-scale crop mapping using Sentinel-2 time series”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 207 (2024), pp. 312–325. issn: 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2023.12.005. url: https://www.sciencedirect.com/science/article/pii/S0924271623003386.

[83] Binghao Yan et al. *Recent advances in deep learning and language models for studying the microbiome*. 2024. arXiv: 2409.10579 [q-bio.QM]. url: https://arxiv.org/abs/2409.10579.

[84] Yongzhi Yang et al. “Dysbiosis of human gut microbiome in young-onset colorectal cancer”. In: *Nature communications* 12.1 (2021), p. 6757.

[85] Liang Zeng et al. “A Meta-learning Method for Few-shot Bearing Fault Diagnosis under Variable Working Conditions”. In: *Measurement Science and Technology* 35 (Feb. 2024). doi: 10.1088/1361-6501/ad28e7.

[86] Yiqian Zhang et al. “Review and revamp of compositional data transformation: A new framework combining proportion conversion and contrast transformation”. In: *Computational and Structural Biotechnology Journal* 23 (2024), pp. 4088–4107. issn: 2001-0370. doi: 10.1016/j.csbj.2024.11.003.

[87] Ruwen Zhou et al. “Data pre-processing for analyzing microbiome data – A mini review”. In: *Computational and Structural Biotechnology Journal* 21 (2023), pp. 4804–4815. issn: 2001-0370. doi: 10.1016/j.csbj.2023.10.001.

[88] Qiang Zhu et al. “Application of deep learning in microbiome”. In: *Journal of Artificial Intelligence for Medical Sciences* 1.1 (2020), pp. 23–29.

A
Training Details for the Models

In here, all the details deemed necessary with regards to the different baseline and meta-learning methods are explained.

A.1. Baseline Methods

To ensure a robust evaluation of the baseline classifiers, we employed a nested cross-validation setup with 10 outer folds and 3 inner folds, with $80/20$ shuffle splits. The inner loop was used for hyperparameter tuning, conducted using 50 trials using the default Optuna sampler (with the multivariate setting set to True).

Hyperparameters unrelated to the models included:

- number of components after feature reduction: $[0, 1500]$ with steps of 100 and 0 meaning no feature reduction.

The model specific hyperparameters included the following:

- Random forests:
  - n_estimators: $[10, 500]$ with steps of 10
  - max_depth: $[10, 200]$ with steps of 10
  - criterion: `gini` or `entropy`
  - bootstrap: `True` or `False`
  - oob_score: (False, ”F1”)
- XGBoost:
  - learning_rate: $[0.2, 0.1]$
  - gamma: $[0, 3]$
  - max_depth: $[3, 8]$
  - reg_lambda: $[0.0, 1.0]$
  - reg_alpha: $[0.0, 1.0]$
- Multi-layer perceptron:
  - batch_size: $[8, 64]$ with steps of 8
  - learning rate: continuous in $[10^{-5}, 10^{-2}]$
  - num_layers: $[1, 4]$
  - dropout_rate: $[0.0, 0.7]$ with steps of 0.1
  - base_size (initial layer size): $[16, 1008]$ with steps of 32

- reduction_factor (per layer): $[1.0, 3.0]$ with steps of 0.5
- layer_sizes: dynamically determined based on `base_size` and `reduction_factor`, decreasing with depth
- layer_norm: `True` or `False`
- batch_norm: `True` or `False`, but not used when `layer_norm` is `True`

The best-performing hyperparameter configuration from the inner loop was used to train the model on the full outer training set, and the performance was evaluated on the corresponding outer test fold.

The above belongs to all the different baseline methods, including the meta-learning inspired one. However, instead of nested CV, the CV loops are performed separately as shown in Figure 5.6.

A.2. Meta-Learning Methods

Again, for statistically significant experimentation, cross-validation has been used and as indicated in Section 5.2 two separate CV loops have been used instead of a nested one. The hyperparameter optimization CV loop repeated the experiments for each trial 5 times with at least 30 trials in total, and the test CV loop used 10 folds. For all these, reproducible data splits are used. Again, the default Optuna sampler (with the multivariate setting set to True) is used. The hyperparameters optimized for during meta-learning are the following:

- Training Hyperparameters
  - starting_lr: log-uniform in $[5 \times 10^{-7}, 5 \times 10^{-4}]$
  - scheduler_gamma: $[0.1, 1.0]$ with steps of 0.1
  - weight_decay: $[0.0, 0.5]$ with steps of 0.1
- Model Architecture Hyperparameters
  - num_layers: integer in $[1, 4]$
  - layer_sizes: for each layer i, chosen from $[60, 300]$ with steps of 60 *(determined dynamically based on num_layers)*
- Feature Reduction (if used)
  - feature_reduction_n_components: $[300, 1500]$ with steps of 300
- Early Stopping
  - early_stopping_patience: $[0, 50]$ with steps of 2
- Data Augmentation
  - jitter_fraction: $[0.0, 0.5]$ with steps of 0.1

For more optimal results, number of trials should be much larger, but due to limited computational resources, this was not possible.

B Elaborate Experimental Results

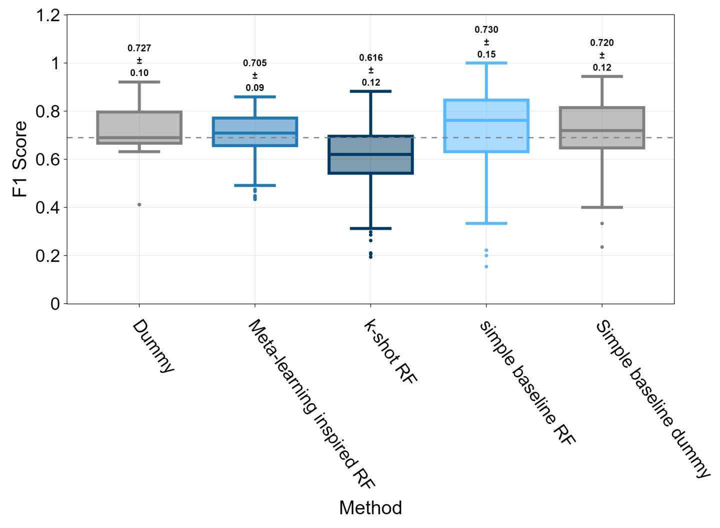
The goal here is to give a more comprehensive overview of the experimental results. Due to the fact that the the dataset consists out of many different studies and each study has been considered as a test set, the results can be overwhelming at first glance if not aggregated as in Chapter 6. The results here are meant to give a more detailed picture by showing study-wise performance comparisons or grouping of studies in various ways. The hope further is that these results will give some inspiration for researchers in the field to conduct useful experiments, collaborate on coming up with standards, and figure out what is missing in this research.

B.1. Baseline Methods

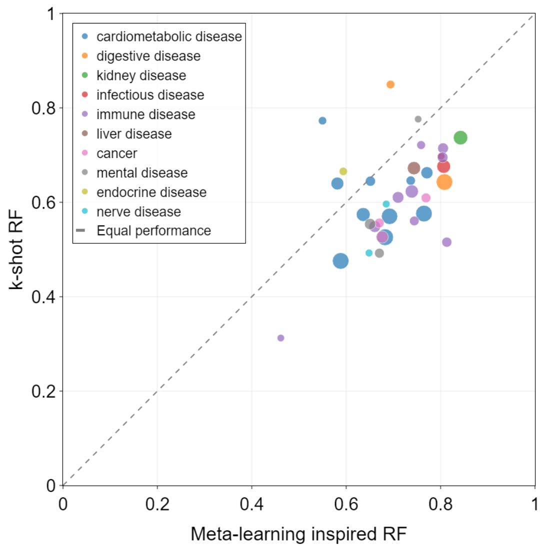
In Section 6.1, some metrics regarding the performance of baseline methods were given. This section will show the results of all these baseline model from a different perspective.

B.1.1. Different Baseline Strategies

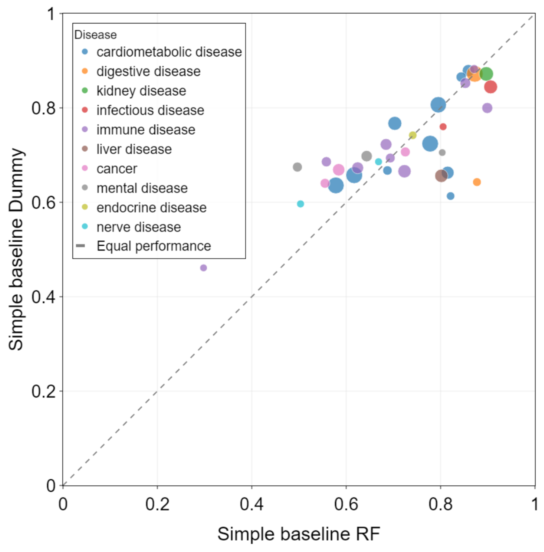
The first comparison we looked at was the one of the different strategies. Here, we show an aggregated comparison of the different strategies with the different classification models.



(a) Aggregated CV F1-score.

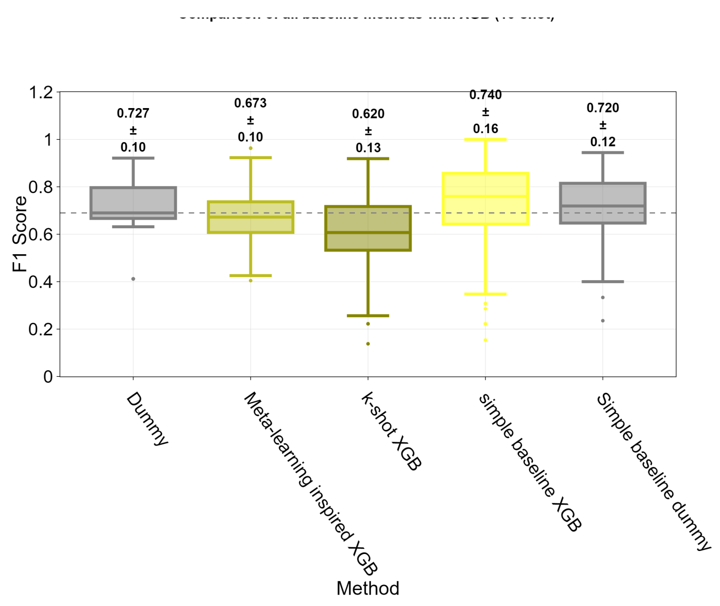


(b) Study-wise CV F1-score of meta-learning inspired RF versus k-shot RF.

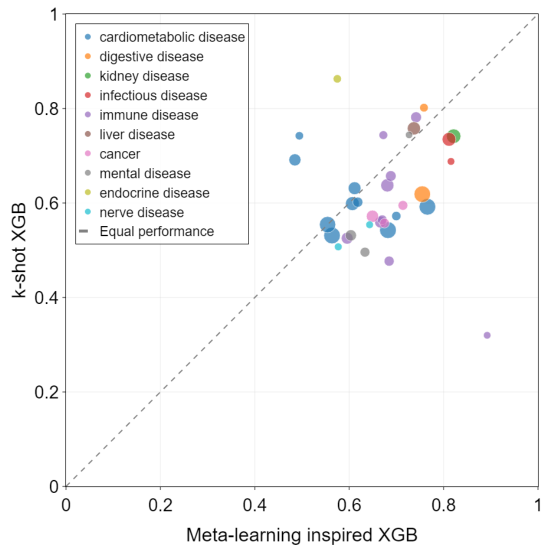


(c) Study-wise CV F1-score of simple baseline rf versus simple baseline dummy.

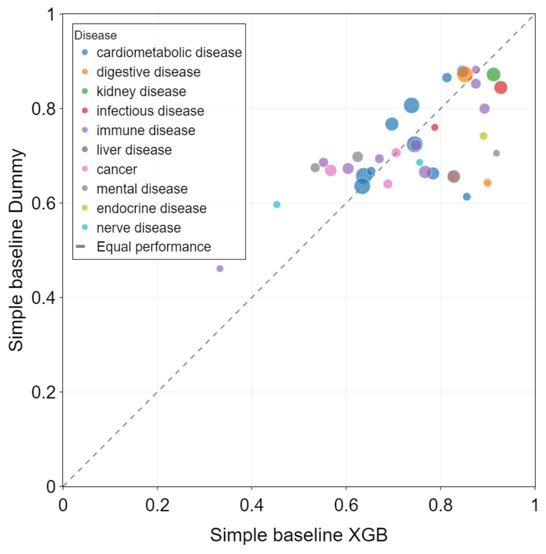
Figure B.1: Comparing the performance of different baseline methods with RF to each other (10-shot setting). The first `Dummy` result belongs to both meta-learning inspired RF and k-shot RF as the test set is the same in both cases. Further, the simple baseline RF should not be directly compared to the other RF models, as the test set is different. In addition, Study-wise results are shown.



(a) Aggregate CV F1-score.

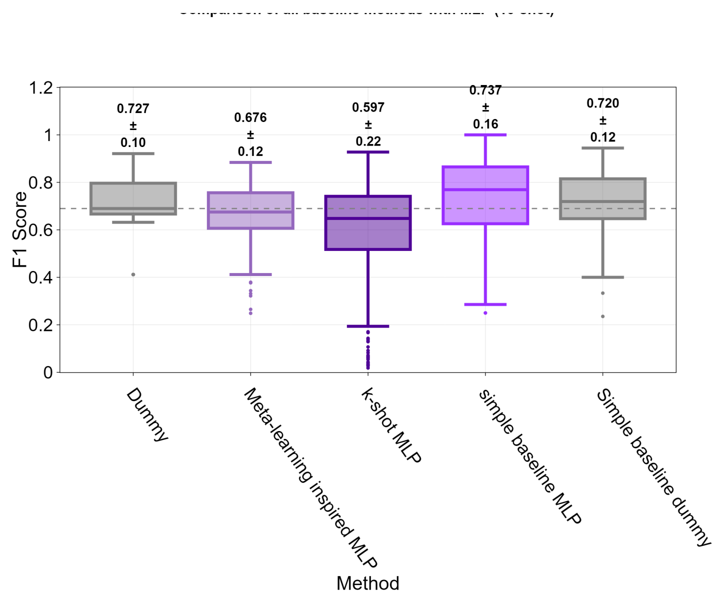


(b) Study-wise CV F1-score of meta-learning inspired XGBoost versus k-shot XGBoost.

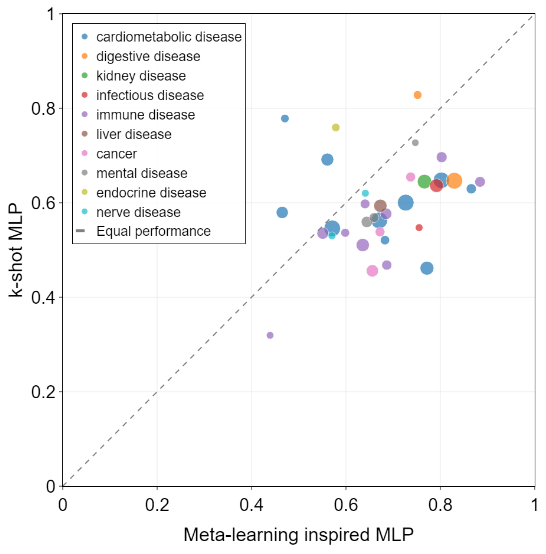


(c) Study-wise CV F1-score of simple baseline XGBoost versus simple baseline dummy.

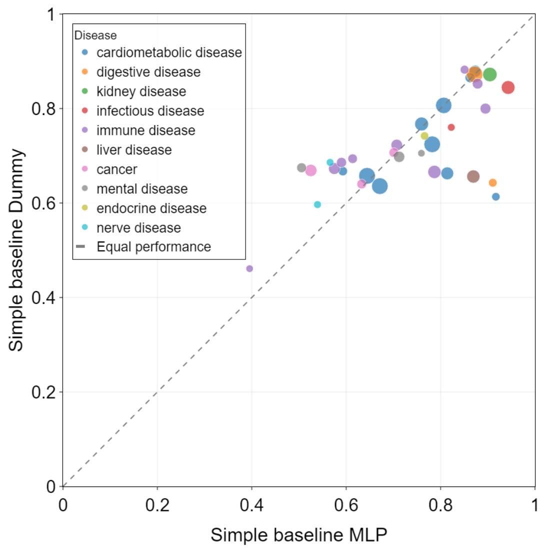
Figure B.2: Comparing the performance of different baseline methods with XGBoost to each other (10-shot setting). The first `Dummy` result belongs to both meta-learning inspired XGBoost and k-shot XGBoost as the test set is the same in both cases. Further, the simple baseline XGBoost should not be directly compared to the other XGBoost models, as the test set is different. In addition, Study-wise results are shown.



(a) Aggregate CV F1-score.



(b) Study-wise CV F1-score of meta-learning inspired MLP versus k-shot MLP.

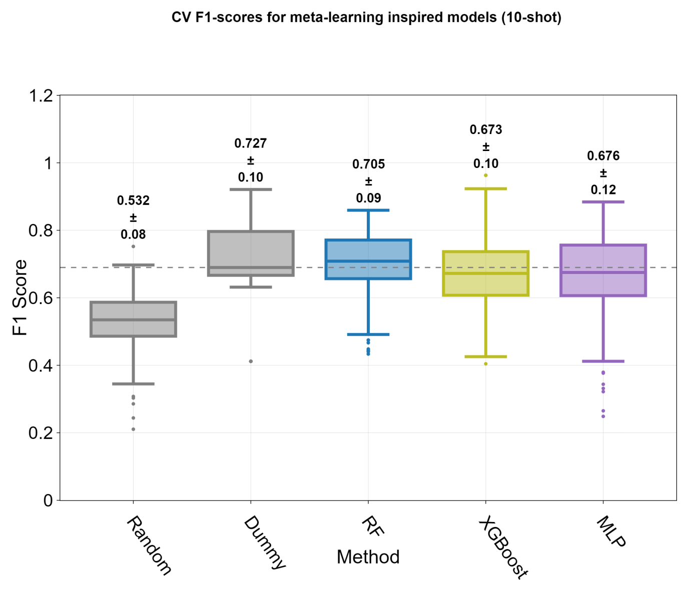


(c) Study-wise CV F1-score of simple baseline MLP versus simple baseline dummy.

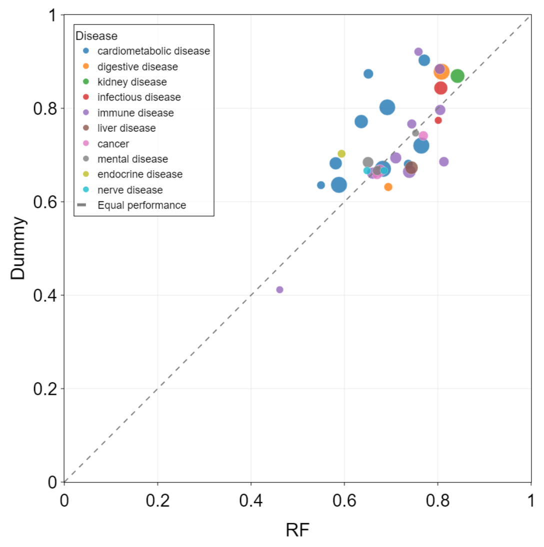
Figure B.3: Comparing the performance of different baseline methods with MLP to each other (10-shot setting). The first `Dummy` result belongs to both meta-learning inspired MLP and k-shot MLP as the test set is the same in both cases. Further, the simple baseline MLP should not be directly compared to the other MLP models, as the test set is different. In addition, Study-wise results are shown.

B.1.2. Different Models Within the Meta-Learning Inspired Baseline Method

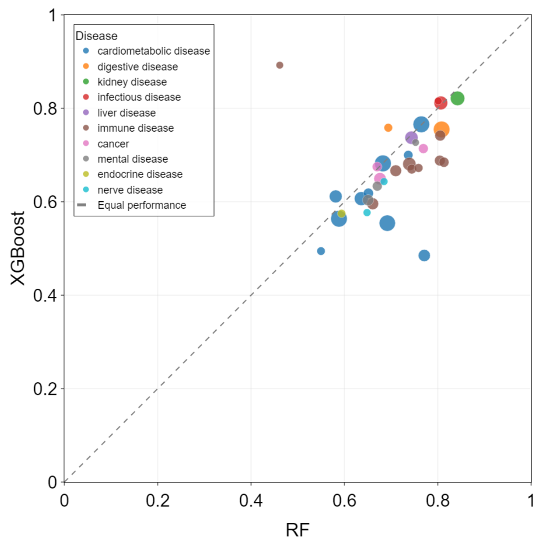
We have seen that using the meta-learning inspired baseline strategy with the RF model in general outperforms the other classification models in this strategy. Interestingly, the majority-class `Dummy` which classifies all samples as ”Diseased”, shows that the performance of all the models in general are disappointing.



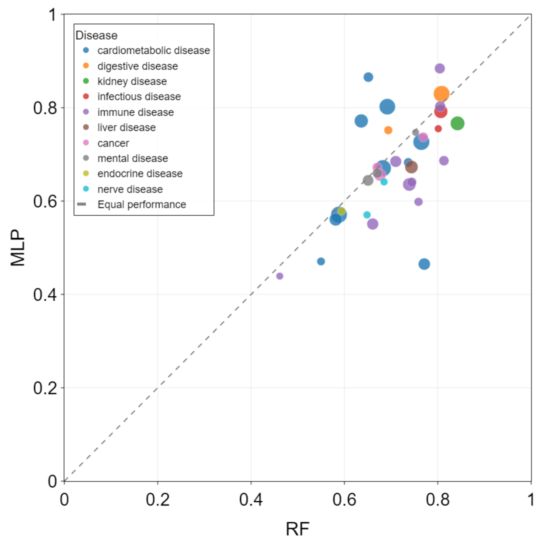
(a) Aggregated CV F1-scores.



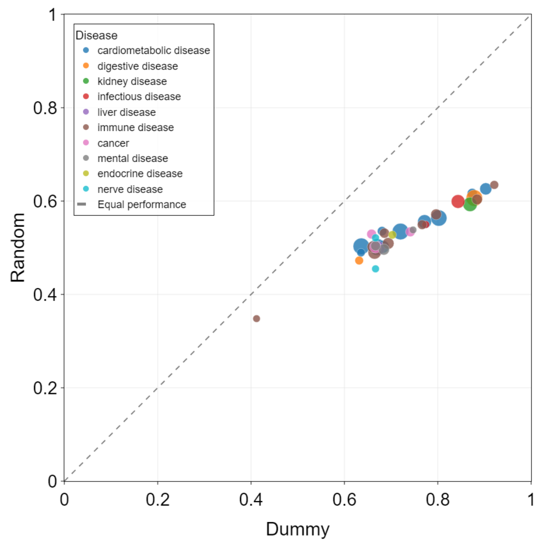
(b) Study-wise CV F1-score of RF versus Dummy.



(c) Study-wise CV F1-score of RF versus XGBoost.



(d) Study-wise CV F1-score of RF versus MLP.



(e) Study-wise CV F1-score of Dummy vs Random.

Figure B.4: Comparison of different models used within the meta-learning inspired baseline method (10-shot case). Figure B.4a shows the aggregated results, where RF is outperforming the other models. With this in mind, the other figures show the study-wise performance comparison of RF and the other models. Each dot is the average score of the cross-validation performed. The size of the dots indicate the study size with colors representing different diseases.

B.1.3. Balancing of Data

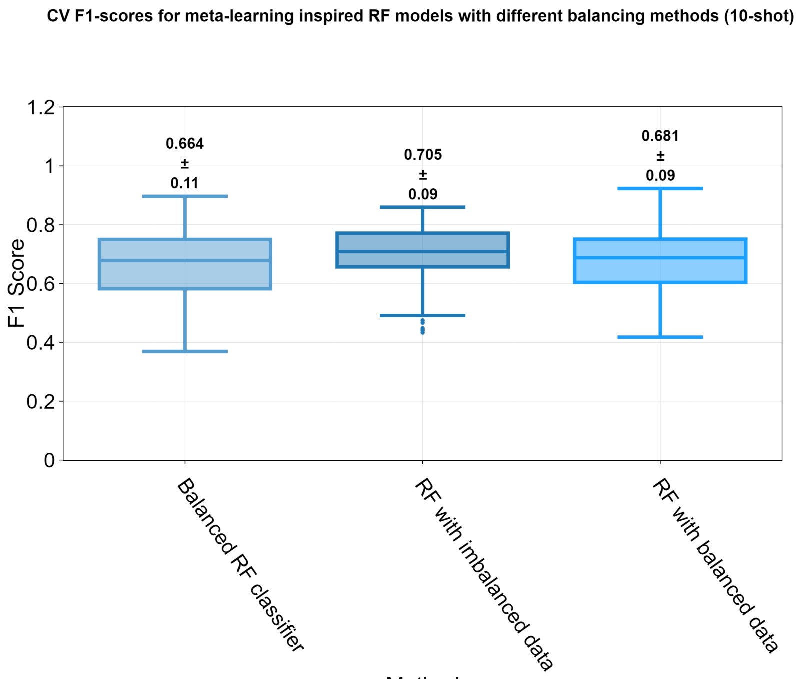
Due to the dataset used in this study being imbalanced, it is worthwhile to look at some results regarding random forests with focus on how balancing affects performance. Figure B.5 shows the results of running the random forest classifier on balanced and imbalanced data. In addition, the performance of the `BalancedRandomForestClassifier`[1]is compared to these, which inherently deals with imbalanced data. From this, it is clear that in these experiments the RF model on imbalanced data performs better in general. It is true that due to imbalanced datasets, the model may also learn the bias in class distribution, but the `Dummy` classifier is used as a baseline to show performance because of this bias and this way we can look at improvement of the models.

The corresponding `mean±std` values of the $F1$-scores are:

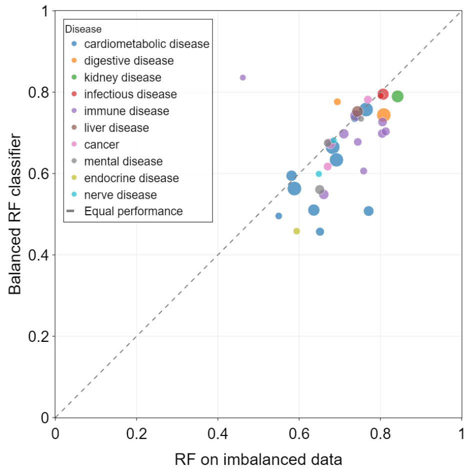
- Balanced RF classifier: 0.664 ± 0.11
- RF with unbalanced data: 0.705 ± 0.09
- RF with balanced data: 0.681 ± 0.09

showing the strong performance of the method in the imbalanced case. Based on these results, we have decided to keep the strongest method for comparison in the other experiments, although all of these results can be referenced for comparison to prevent the bias added by looking at these results.

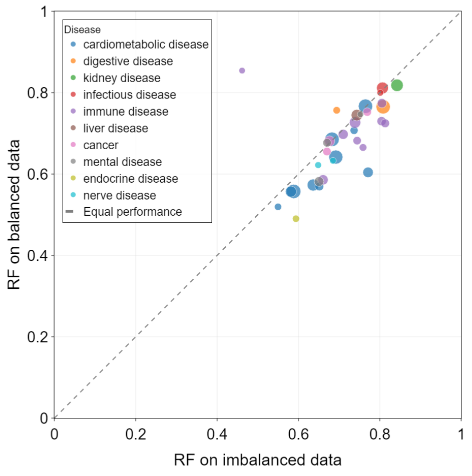
Also more generally, training on imbalanced data avoids problems regarding balancing which is a challenge in the microbiome field.



(a) Aggregated CV F1-scores.



(b) Study-wise CV F1-scores of BalancedRandomForestClassifier[1] versus RF on imbalanced data



(c) Study-wise CV F1-scores of RF on balanced data versus RF on imbalanced data

Figure B.5: Influence of balancing on performance of meta-learning inspired baseline method with RFs (10-shot setting). Figure B.5a shows that RF on imbalanced data outperforms the other two methods, the BalancedRandomForestClassifier and RF with balanced data (balancing done with SMOTE). Figure B.5b and Figure B.5c show study-wise performance comparison between RF on imbalanced data and the other two methods, showing how well it outperforms them. Each dot is the average score of the cross-validation performed. The size of the dots indicate the study size with colors representing different diseases.

B.2. Meta-Learning Inspired Baseline vs Protonets

B.2.1. Effect of Number of Shots

Figure B.6 visualizes the performance per study when meta-learning inspired baseline is used with different shot numbers. As can be seen, changing the number of shots shows varying performance changes.

[1]From the Imbalanced-learn library: `https://imbalanced-learn.org/stable/references/generated/imblearn.ensemble.BalancedRandomForestClassifier.html`

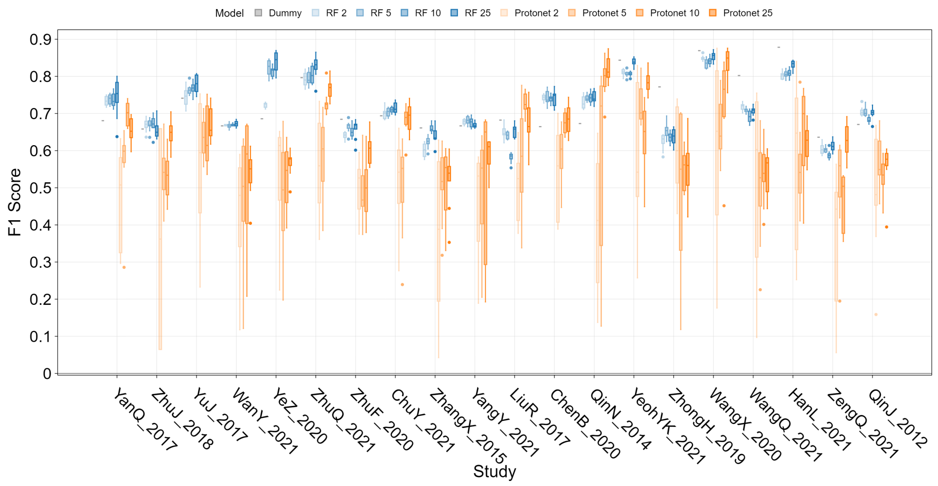


Figure B.6: CV F1-score for meta-learning inspired RF methods for different k-shots. Number of shots is indicated in the legend. Results are only shown for studies that are tested in all shot scenarios, which are only 20 studies.

B.2.2. Dimensionality Reduction

This appendix first presents some preliminary analyses to understand the structure of the microbiome feature space, specifically, how much variance can be explained by a reduced number of principal components. It then explores a PCA-guided feature selection method aimed at preserving taxonomic interpretability. These have been a motivation for doing the empirical analysis of the learning algorithms. Finally, we show more detailed, study-wise results here on how dimensionality reduction affects the performance of random forest models across different shot numbers.

Let us start by looking at the cumulative explained variance of the PCs, as shown in Figure B.7.

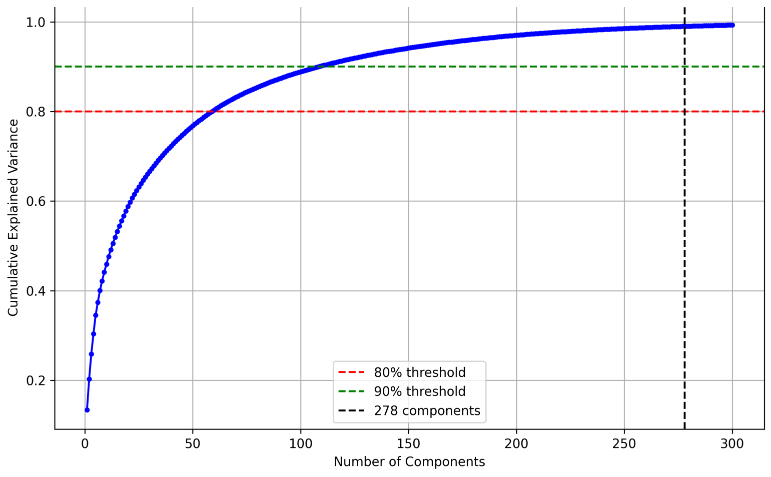


Figure B.7: This shows how much of the Sun et al.[68] dataset's variance is cumulatively explained by a certain number of principal components.

In the figure, some of the findings standing out are:

- Only 59 principal components are needed to explain 80% of the data. This is 2.3% of the original 2557 features in the data.
- 109 principal components are needed to explain 90% of the variance of the data, which is 4.3% of the original number of features.

- 278 principal components are enough to explain 99% of the variance of the data, which is just 10.9% of the original number of features.

These numbers are a strong signal that the data could be living in a much lower-dimensional sub-space than the raw feature set suggests. One reason for this is functional redundancy, which is prevalent in microbial communities.

Altogether, PCA seems to be able to easily help us reduce computational load necessary to learn from the data, and the reduction in number of features may help reduce noise and counteract problems caused by the curse of dimensionality; this way increase classification performance.

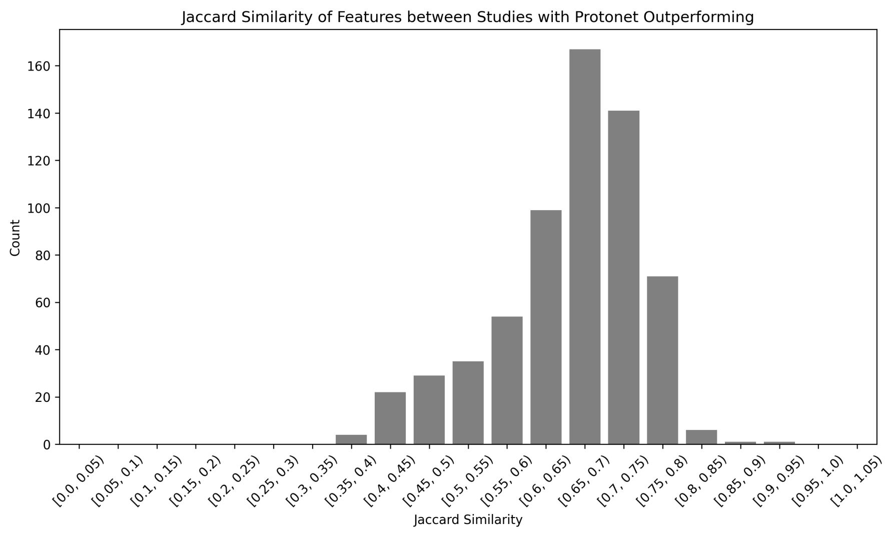


Figure B.8: Jaccard similarity of the feature space of the studies from gathered by sun et al.[68].

Taxon-Level Feature Selection Using PCA Loadings

Additionally, we can argue for keeping data in the original feature space and thus preserve taxonomic interpretability. With this in mind, we can look into a PCA-guided feature filtering, where features are ranked by their weighted contribution to the first c components and retained until a cumulative ”importance variance” threshold is reached, as explained in Section 5.1.

In Table B.1, we report the number of taxa selected for different important coverage threshold when 278 PCs are used with no weighting applied. The number of PCs does not have much of an influence here as can be seen in Figure B.9, as does not what weighting is used.

Table B.1: Number of taxa retained for different coverage thresholds when 278 PCs are used and no weighting is applied.

Feature-importance coverage τ	taxa retained
80%	279
85%	317
90%	391
95%	1133
99%	2271

Table B.1 and Figure B.9 reveal a key limitation: because microbiome loadings are diffuse, a high coverage threshold ($\tau \geq 0.90$) keeps most of the original taxa. In other words, almost every species contributes measurably to the loadings, so very little pruning occurs. Consequently, this filter delivers little computational relief. Only at much lower coverages ($\sim 90\%$ or less) does the retained set shrink to a few-hundred taxa, levels useful for computational load reduction.

From this, we hypothesize that for computational savings while keeping interpretability, we need to choose a lower coverage risking worse performance. When interpretability is not required, projecting into a few hundred PCs looks to be the better route, and when biological interpretation is paramount, feature selection can better be skipped it seems.

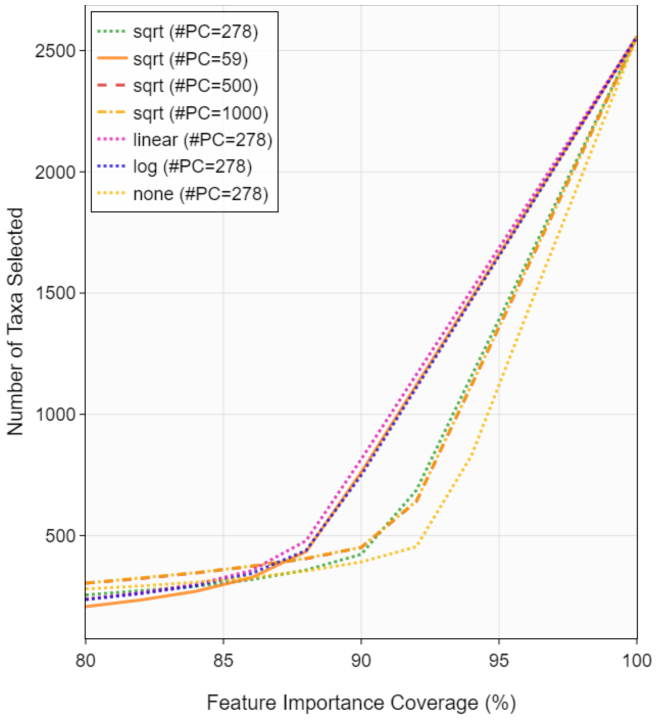
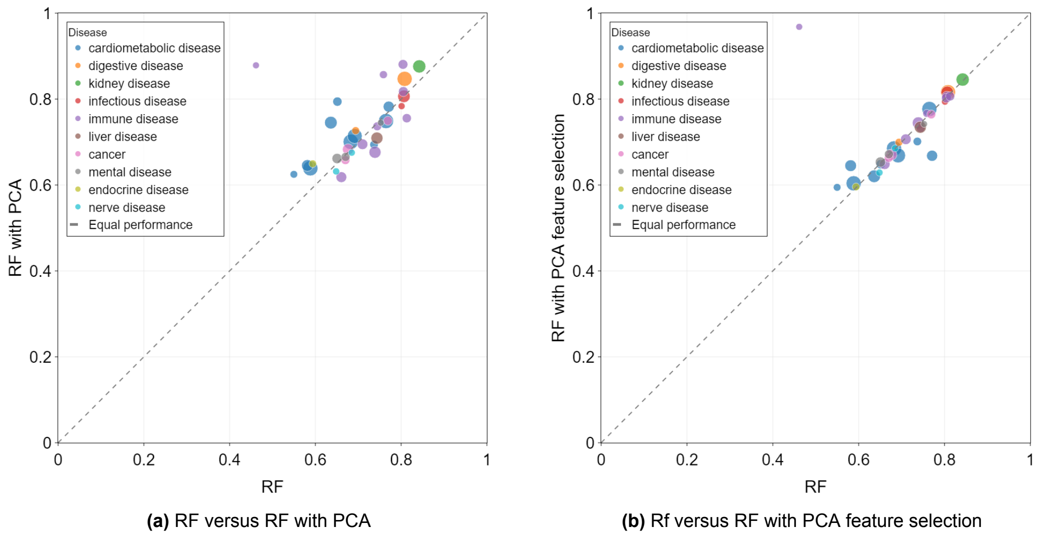


Figure B.9: Taxa retained vs. feature-importance coverage τ. Curves are nearly identical across different number of PCs and weighting schemes, confirming that the coverage criterion, not the PCA settings (, `linear`, `log`, or no weighting), drives the outcome.

Detailed results

Figure B.11 zooms into the results from Subsection 6.3.2 and shows study-wise how PCA and feature selection (with feature importance found by PCA) influences the performance of RF, for different shot numbers. As we can see, the performance change of RF with dimensionality reduction varies a lot between test studies and the conclusion from Subsection 6.3.2 is valid.



(a) RF versus RF with PCA

(b) Rf versus RF with PCA feature selection

Figure B.10: Study-wise comparison is shown between RF and RF with some dimensionality reduction technique (10-shot setting). Each dot is the average score of the cross-validation performed. The size of the dots indicate the study size with colors representing different diseases.

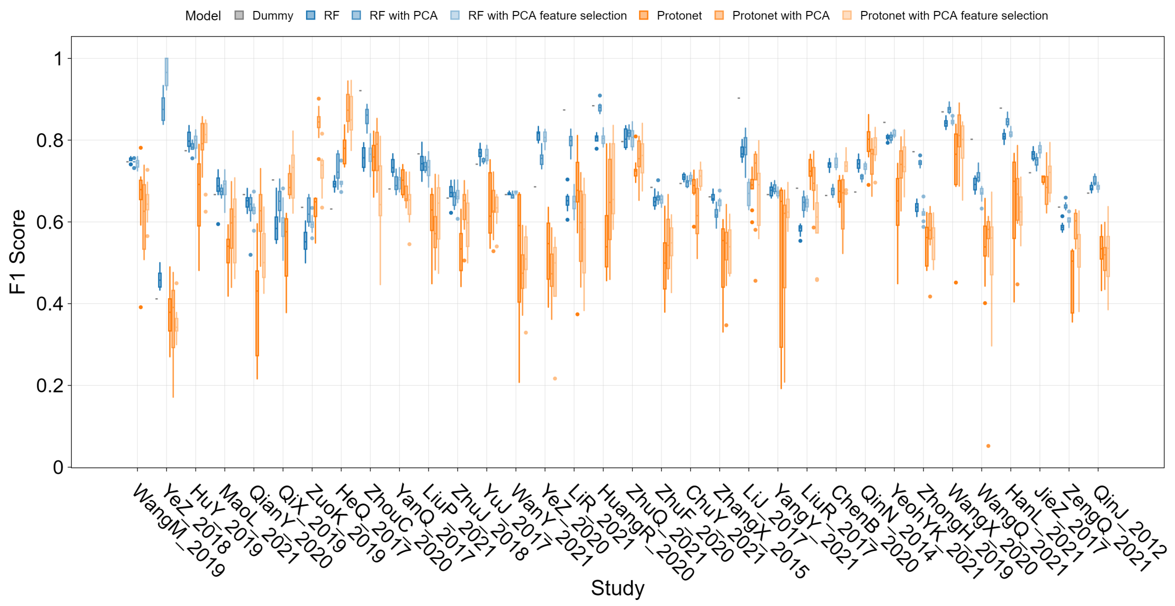
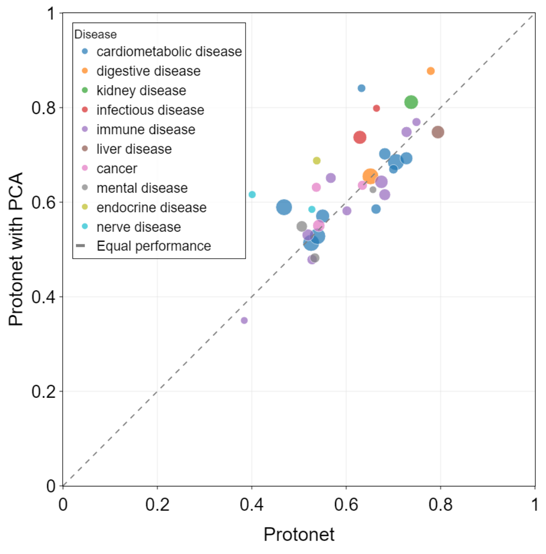
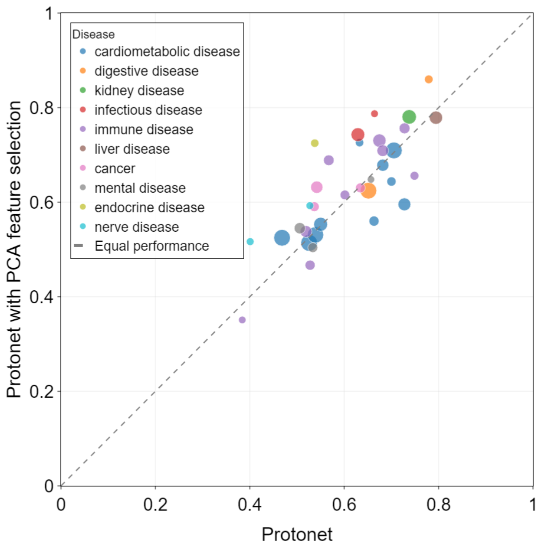


Figure B.11: Study-wise CV F1-scoring of meta-learning inspired RF models and Protonets with different dimensionality reduction techniques (10-shot setting).



(a) Protonet versus Protonet with PCA

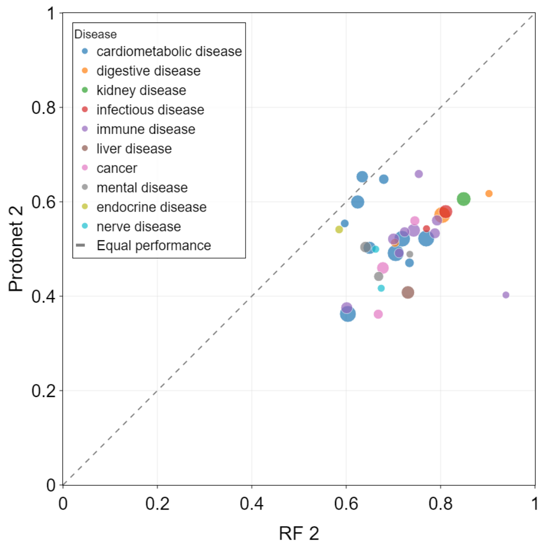


(b) Protonet versus Protonet with PCA feature selection

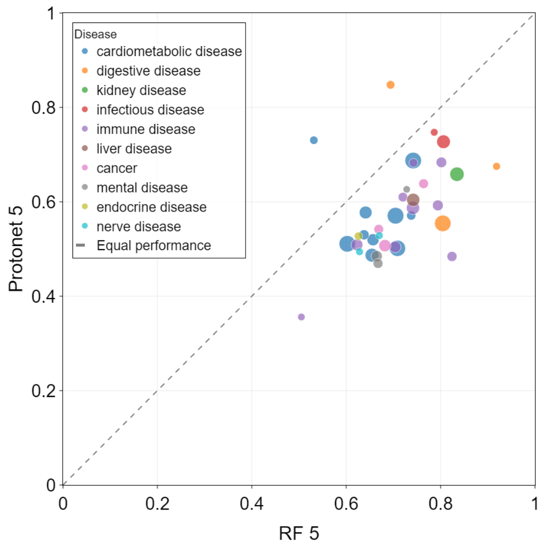
Figure B.12: Study-wise comparison is shown between Protonet and Protonet with the dimensionality reduction techniques used (10-shot setting). Each dot is the average score of the cross-validation performed. The size of the dots indicate the study size with colors representing different diseases.

B.3. Meta-Learning vs Baseline

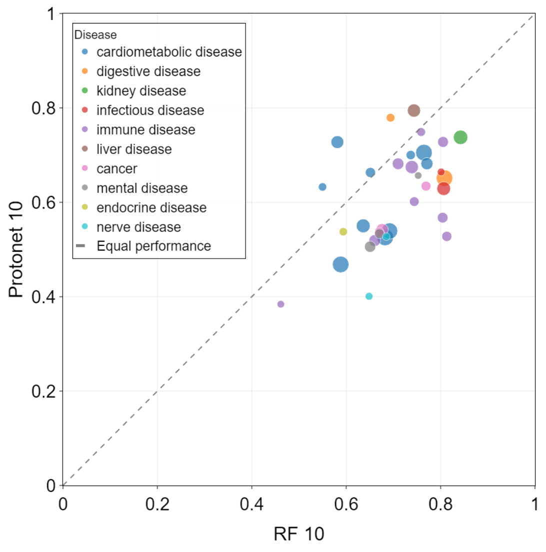
As we have seen in Section 6.4, the performance of Protonets is inadequate compared to the meta-learning inspired method. The following scatter plots shows the performance comparison in more detail.



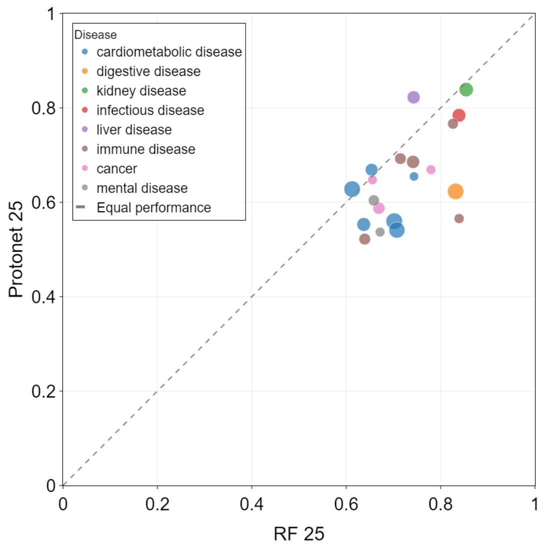
(a) 2-shot setting



(b) 5-shot setting

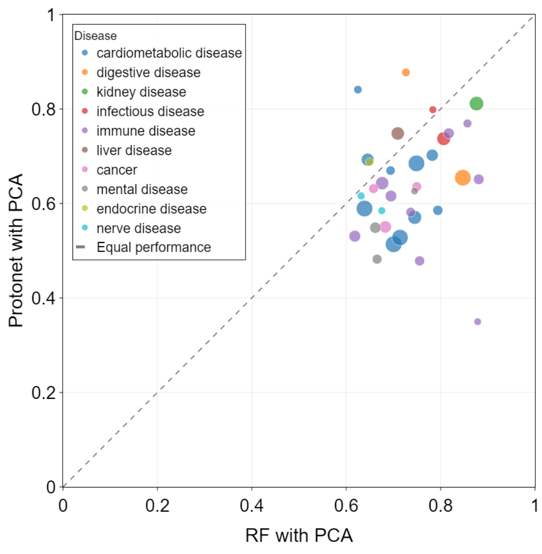


(c) 10-shot setting

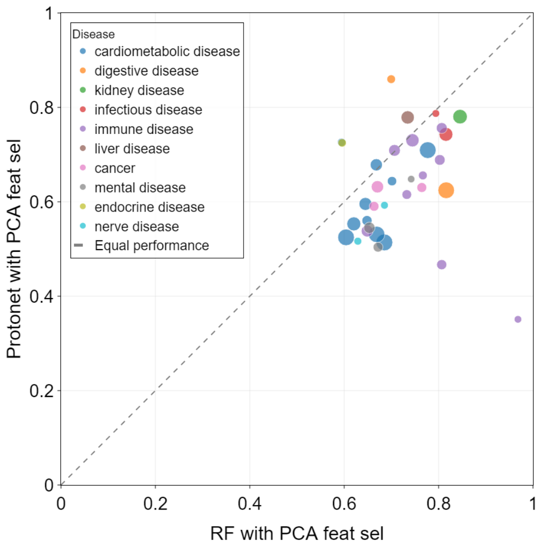


(d) 25-shot setting

Figure B.13: Study-wise comparison is shown between Protonet and RF for the same shot number. Each dot is the average score of the cross-validation performed. The size of the dots indicate the study size with colors representing different diseases.



(a) RF with PCA versus Protonet with PCA



(b) Protonet with RF with PCA feature selection versus PCA feature selection

Figure B.14: Study-wise comparison is shown between Protonet and RF for the same dimensionality reduction technique (10-shot setting). Each dot is the average score of the cross-validation performed. The size of the dots indicate the study size with colors representing different diseases.

B.3.1. Analysis of Protonets

Here, we depict some more detailed visualization regarding the analysis done on the performance of Protonets.

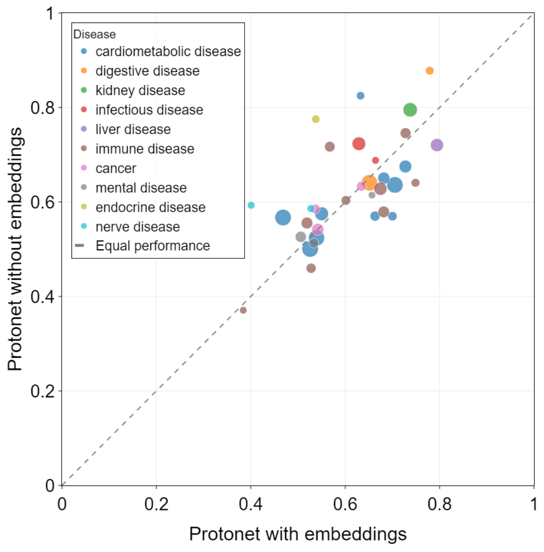
Embedding Network Performance

In Subsection 6.5.1 we have seen the aggregated performance of Protonets without embeddings. In Figure B.15, we show the study-wise performance for the purposes of being complete.

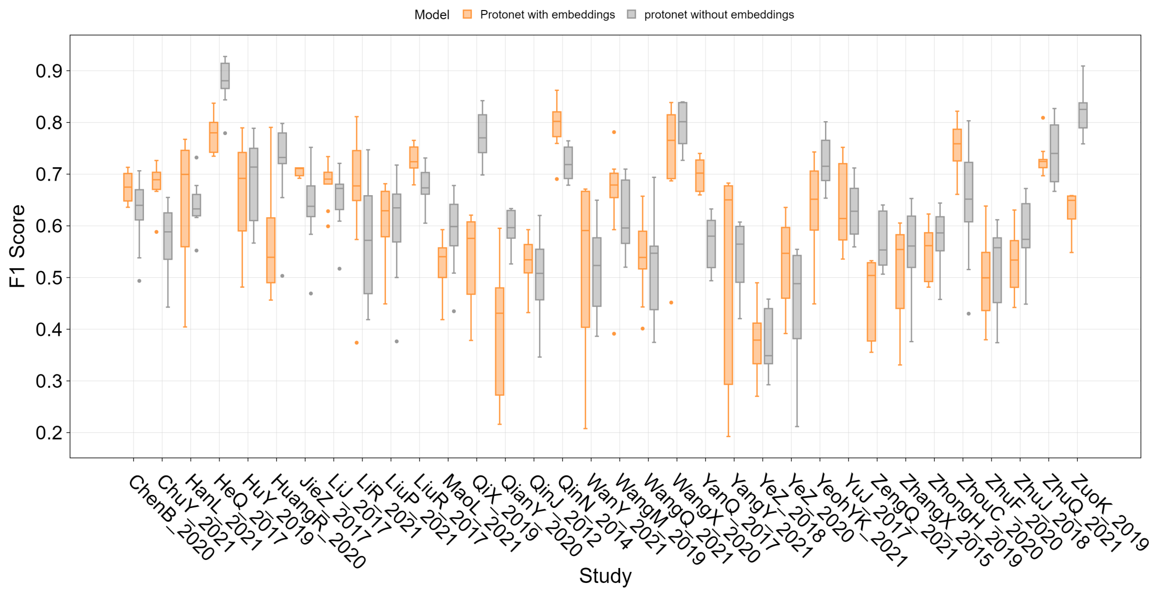
Feature Space Similarity Analysis

In Subsection 6.5.3, we saw the heatmap showing the Jaccard similarity of the feature space between studies. Here, we look at it from a different perspective.

Figure B.16 shows the pair-wise Jaccard similarity between the species-level feature sets of the gut-microbiome studies. Although several tight clusters emerge (e.g. 2 immune disease studies or 2 cardiometabolic disease studies next to each other), not all related diseases get clustered. The results here support the conclusion from Subsection 6.5.3 that there are no strong disease-associated signatures in microbiome profiles across different studies.



(a) RF with PCA versus Protonet with PCA



(b) Protonet with RF with PCA feature selection versus PCA feature selection

Figure B.15: Study-wise comparison between Protonets using an embedding network and Protonets skipping this network (10-shot setting).

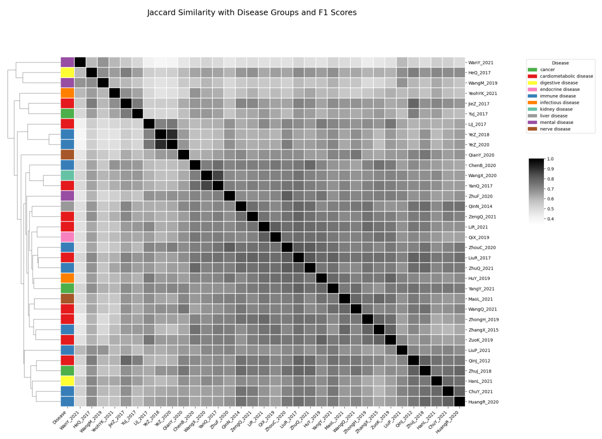


Figure B.16: Hierarchical clustering of microbiome studies based on Jaccard similarity of identified microbial features. The heatmap displays pairwise Jaccard similarity coefficients between studies, with darker shades indicating higher similarity. Studies are hierarchically clustered using average linkage based on the similarity matrix. The colored sidebar indicates the disease category associated with each study, indicated by the legend.

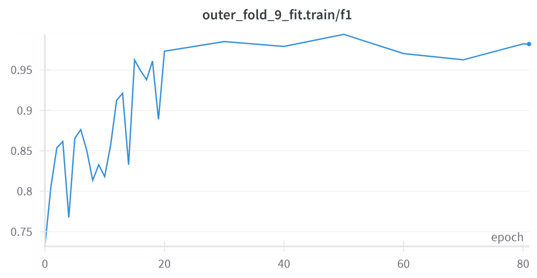
B.3.2. Learning Curves Reveal Overfitting and Generalization Issues

In addition to all that has been discussed in Chapter 6, there are some observations indicating more reasons for the inferior performance of Protonets. One of them is related to the fact that the model does not generalize well. Looking at Figure B.17, we see that although the model is fitting the training set well, the performance on the test set is not improving by much.

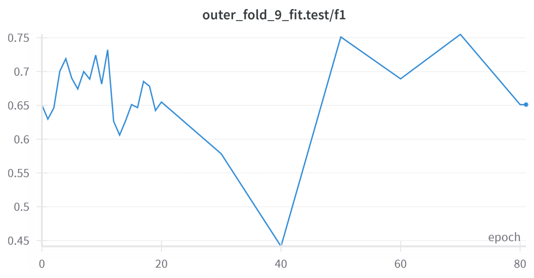
This can be due to the support set for this outer fold not being representative for the rest of the test study and the model not being able to adjust to the new task, or due to the fact that the data from the train studies is less generalizable to the data of the test study. The former is more probable, as some other folds show better performance while the train data is not changing.

The model itself of course can be a big limiting factor as well for how well generalization is done.

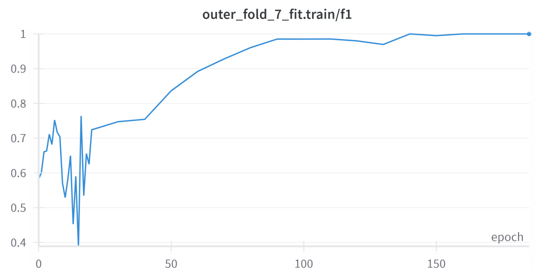
The sensitivity to support set is an important reason as we are focusing on a limited shot learning scenario. Due to the limited number of samples in the support set, the prototypes found differ a lot between different folds which have different support sets. This can also be seen in the UMAP manifolds shown Subsection B.3.3, where the data is quite intermingled and prototypes can easily shift drastically dependent on support set.



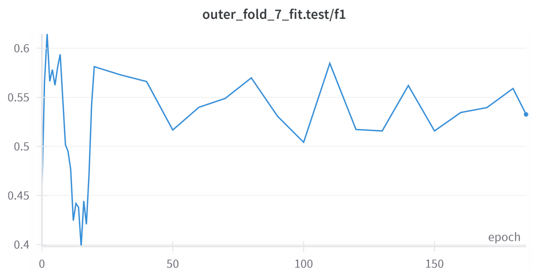
(a) `HuangR_2020`



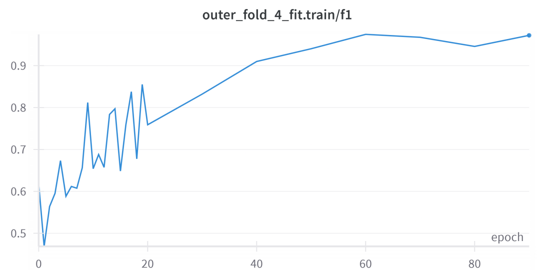
(b) `HuangR_2020`



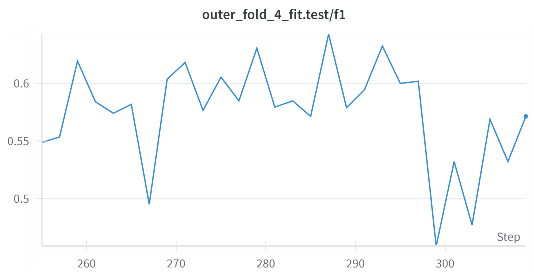
(c) `ZengQ_2021`



(d) `ZengQ_2021`



(e) `ZhuJ_2018`

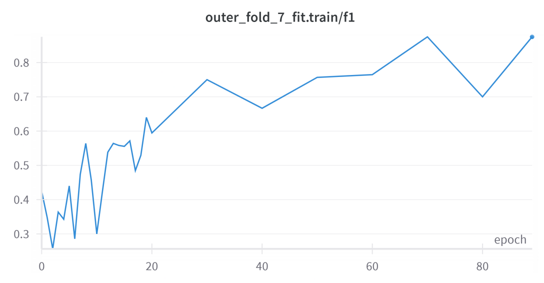


(f) `ZhuJ_2018`

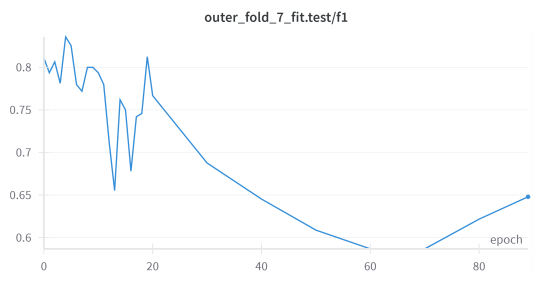
Figure B.17: A selection of learning curves from experiments with different test studies showing that the model has a hard time generalizing. F1-score is shown for one of the outer cross-validation loops for both the train and test set. The first 20 epochs are logged every epoch and after that logging is done every 10 epochs. This is the reason why the plot is changing more in the beginning. The caption of each image indicates which study has been the test study with training set being the data from other studies.

However, another reason for the lack of performance can be related to overfitting. Patterns of overfitting have also been observed where the performance on test data increases but decreases soon afterwards.

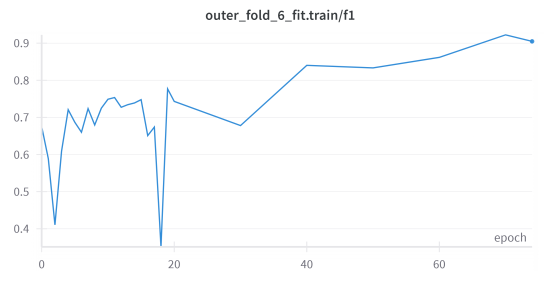
Some examples are given in Figure B.18. Here, early stopping has been a challenge to solve this problem, mainly due to limited data as validation set. In addition, the performance on train data is not even high when the performance on test data starts to decrease, showing the model having a hard time to learn all the patterns.



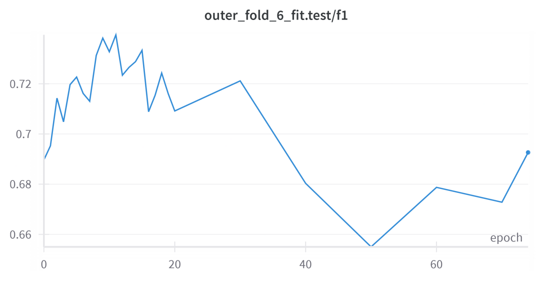
(a) `ZuoK_2019`



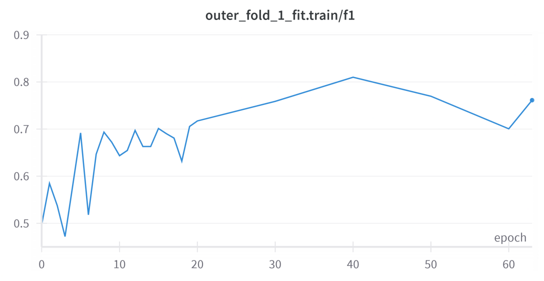
(b) `ZuoK_2019`



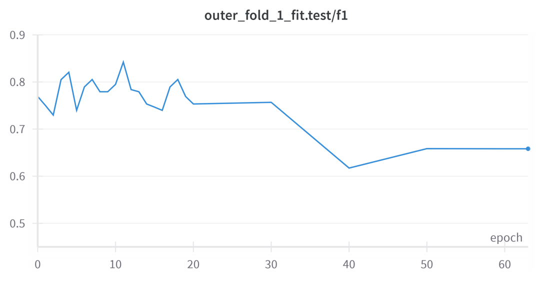
(c) `ChenB_2020`



(d) `ChenB_2020`



(e) `ZuoK_2019`

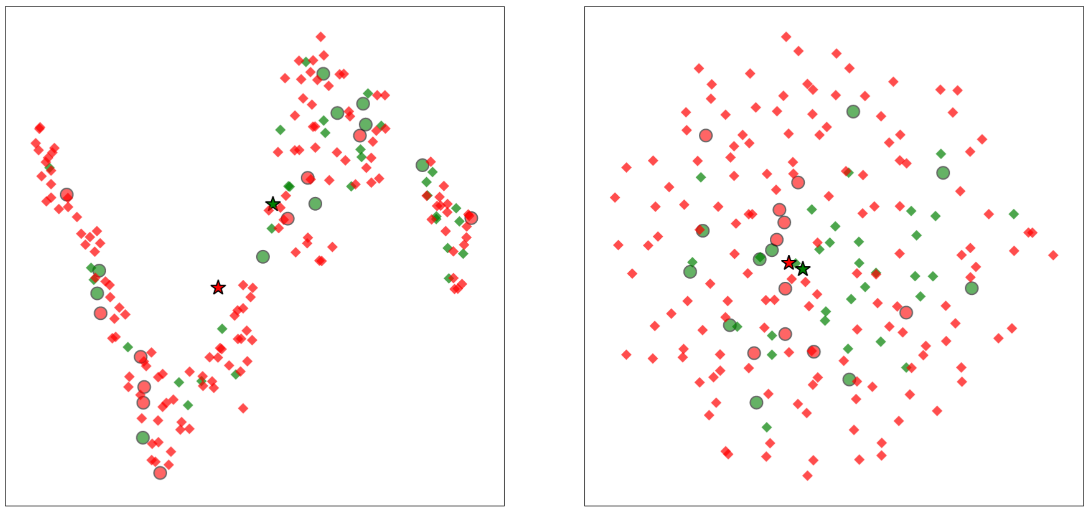


(f) `ZuoK_2019`

Figure B.18: A selection of learning curves from experiments with different test studies showing that the model overfitting fast. F1-score is shown for one of the outer cross-validation loops for both the train and test set. The caption of each image indicates which study has been the test study with training set being the data from other studies.

B.3.3. UMAPs

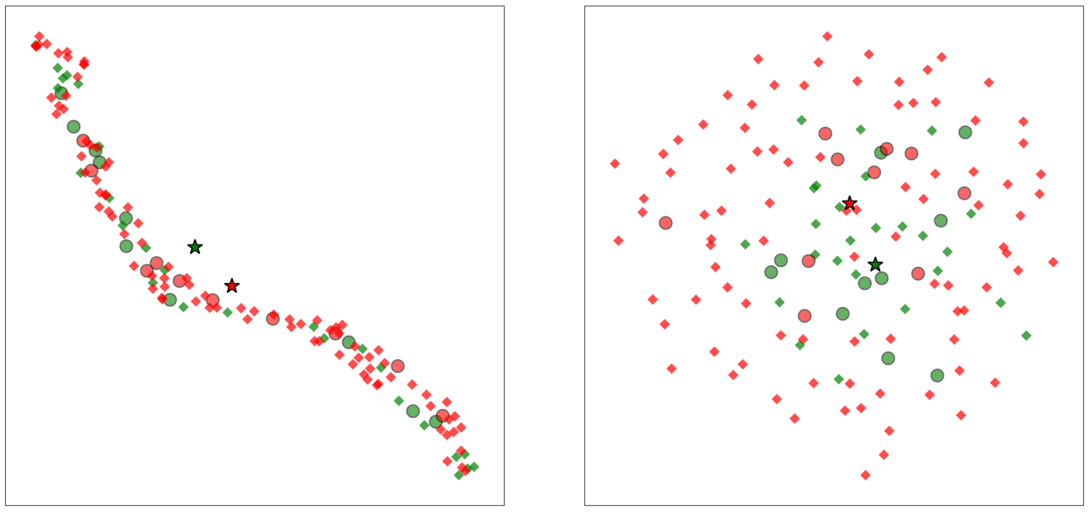
In here, we give the UMAPs for all the studies in the sun et al. [68] dataset analyzed in this research and with that elaborate on the discussion in Subsection 6.5.2. Also here, we show input- and embedding-space UMAPs for the one representative fold per study, which is the fold that gives the $F1$ closest to the median of the experiment.



(a) $F1 : 0.67 \mid Silh : 0.001 \mid T : 0.987 \mid C : 0.985$

(b) $F1 : 0.69 \mid Silh : 0.007 \mid T : 0.563 \mid C : 0.555$

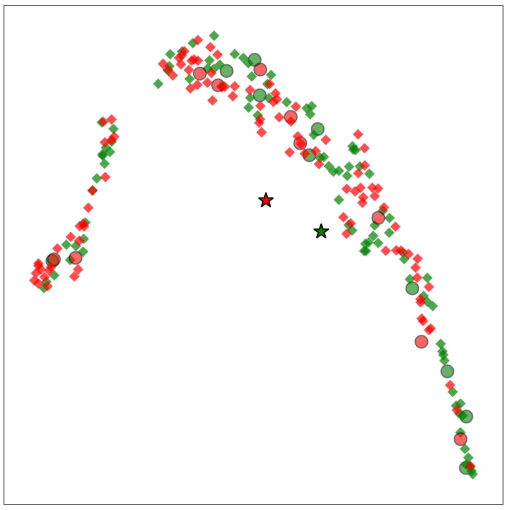
Figure B.19: 2-D UMAP projection of support and query samples from `LiJ_2017` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



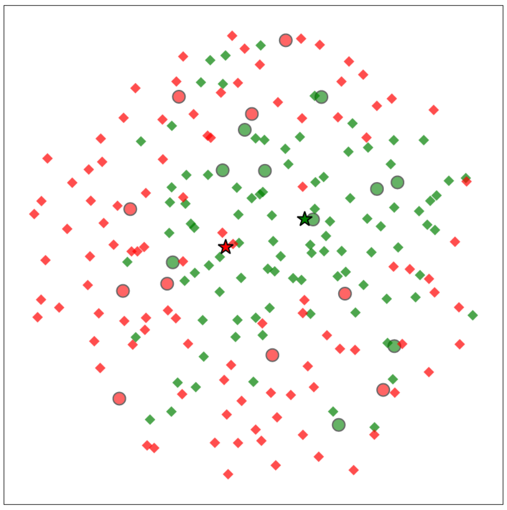
(a) $F1 : 0.60 \mid Silh : 0.005 \mid T : 0.984 \mid C : 0.980$

(b) $F1 : 0.67 \mid Silh : 0.010 \mid T : 0.569 \mid C : 0.558$

Figure B.20: 2-D UMAP projection of support and query samples from `LiR_2021` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

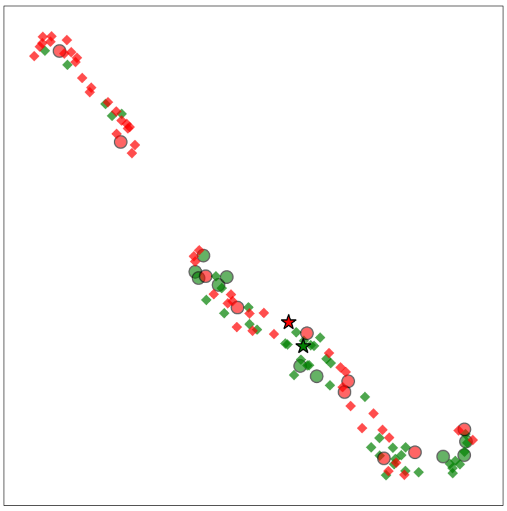


(a) $F1 : 0.68 \mid Silh : 0.019 \mid T : 0.984 \mid C : 0.982$

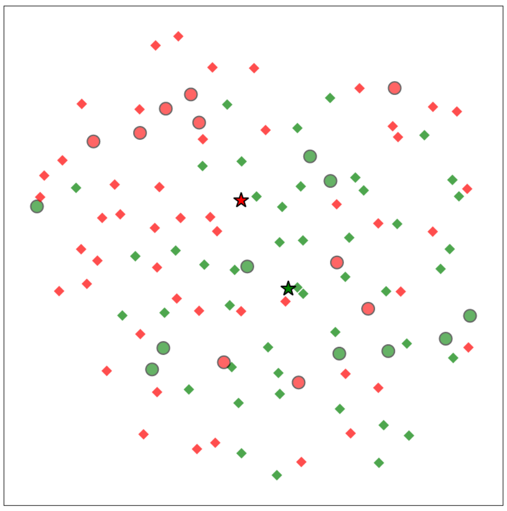


(b) $F1 : 0.72 \mid Silh : 0.007 \mid T : 0.580 \mid C : 0.573$

Figure B.21: 2-D UMAP projection of support and query samples from `LiuR_2017` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

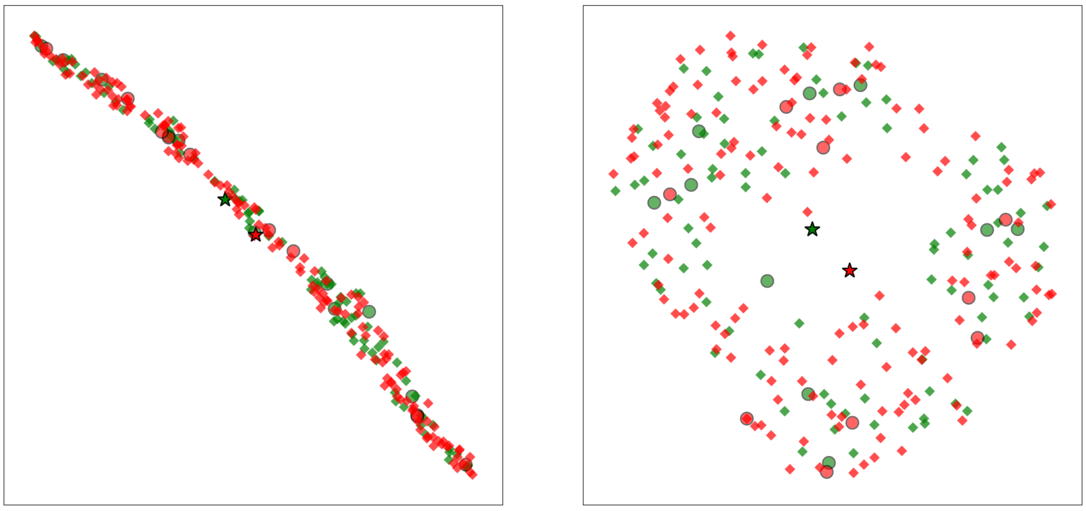


(a) $F1 : 0.55 \mid Silh : 0.009 \mid T : 0.983 \mid C : 0.977$



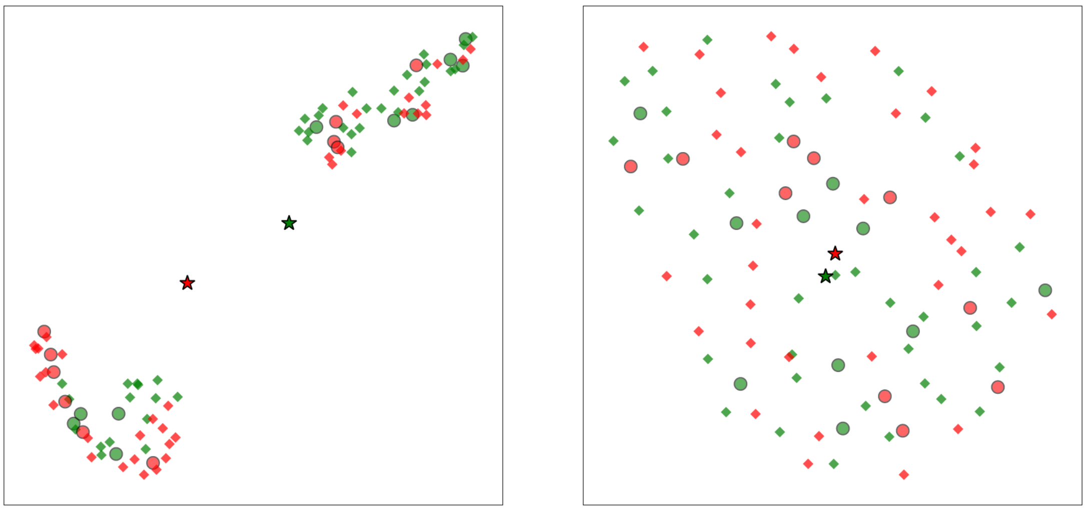
(b) $F1 : 0.72 \mid Silh : 0.005 \mid T : 0.632 \mid C : 0.617$

Figure B.22: 2-D UMAP projection of support and query samples from `YanQ_2017` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



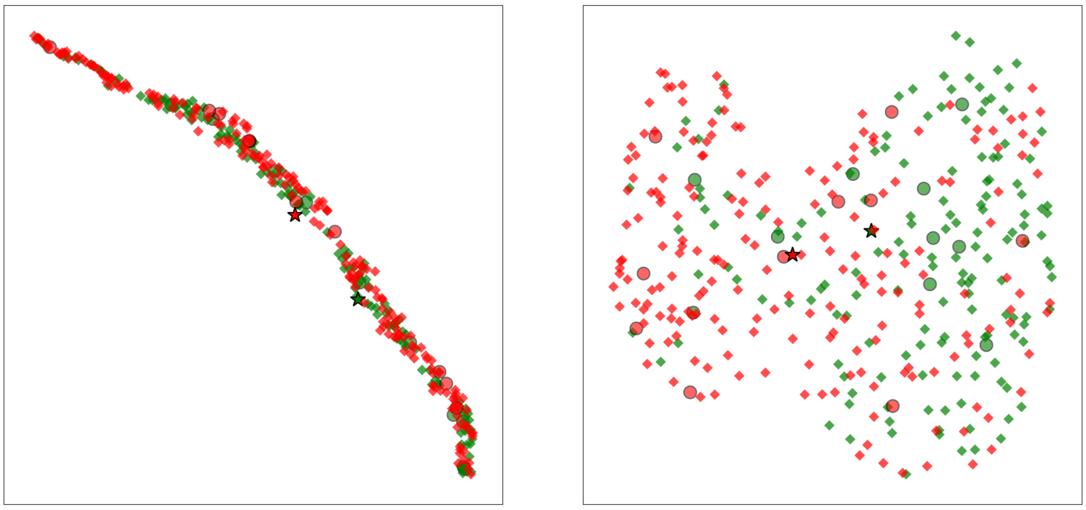
(a) $F1 : 0.58 \mid Silh : -0.001 \mid T : 0.988 \mid C : 0.986$ **(b)** $F1 : 0.55 \mid Silh : 0.000 \mid T : 0.868 \mid C : 0.863$

Figure B.23: 2-D UMAP projection of support and query samples from `ZhongH_2019` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



(a) $F1 : 0.83 \mid Silh : 0.053 \mid T : 0.990 \mid C : 0.983$ **(b)** $F1 : 0.65 \mid Silh : 0.006 \mid T : 0.739 \mid C : 0.723$

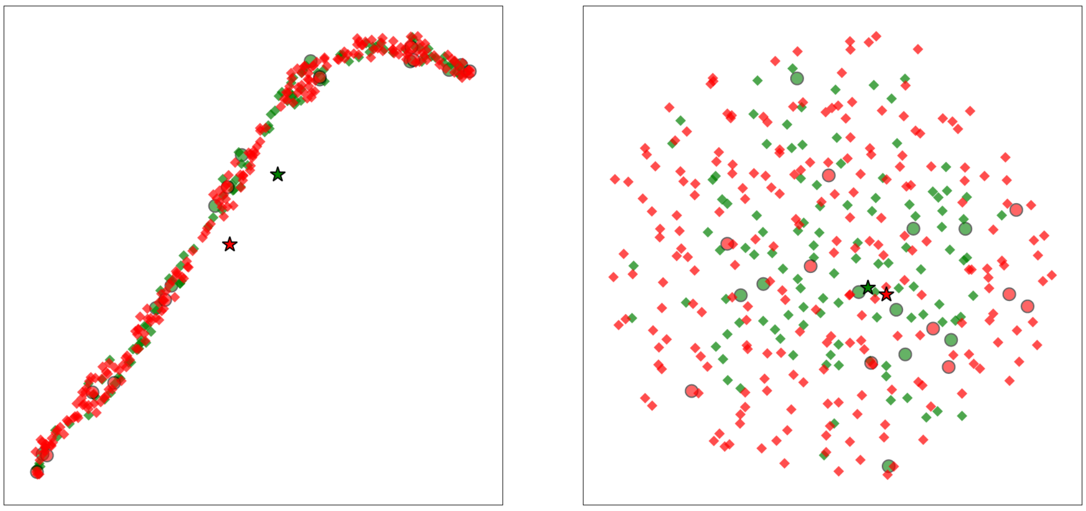
Figure B.24: 2-D UMAP projection of support and query samples from `ZuoK_2019` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



(a) $F1 : 0.64 \mid Silh : 0.012 \mid T : 0.983 \mid C : 0.981$

(b) $F1 : 0.71 \mid Silh : 0.047 \mid T : 0.842 \mid C : 0.837$

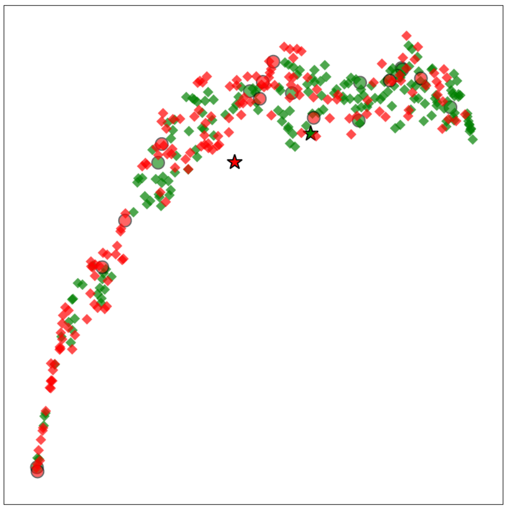
Figure B.25: 2-D UMAP projection of support and query samples from `JieZ_2017` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



(a) $F1 : 0.55 \mid Silh : -0.001 \mid T : 0.983 \mid C : 0.982$

(b) $F1 : 0.54 \mid Silh : -0.004 \mid T : 0.552 \mid C : 0.548$

Figure B.26: 2-D UMAP projection of support and query samples from `WangQ_2021` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).


(a) $F1 : 0.55 \mid Silh : 0.011 \mid T : 0.981 \mid C : 0.980$

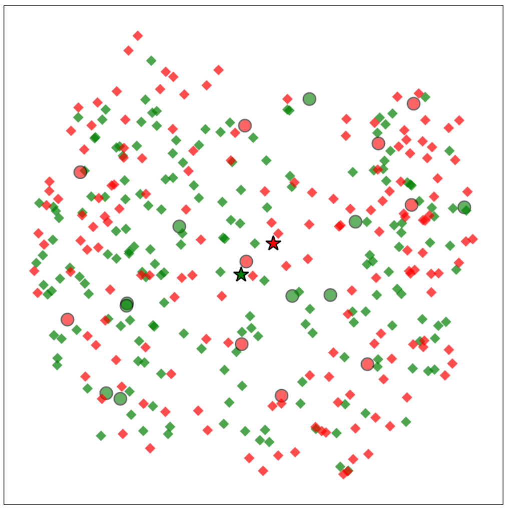
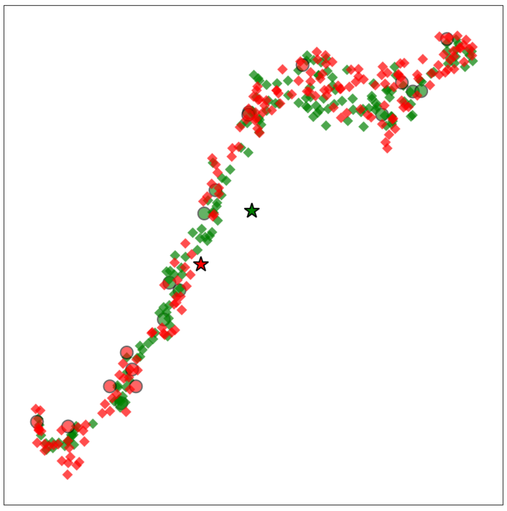

(b) $F1 : 0.52 \mid Silh : 0.010 \mid T : 0.729 \mid C : 0.726$

Figure B.27: 2-D UMAP projection of support and query samples from `ZengQ_2021` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).


(a) $F1 : 0.51 \mid Silh : 0.006 \mid T : 0.982 \mid C : 0.981$

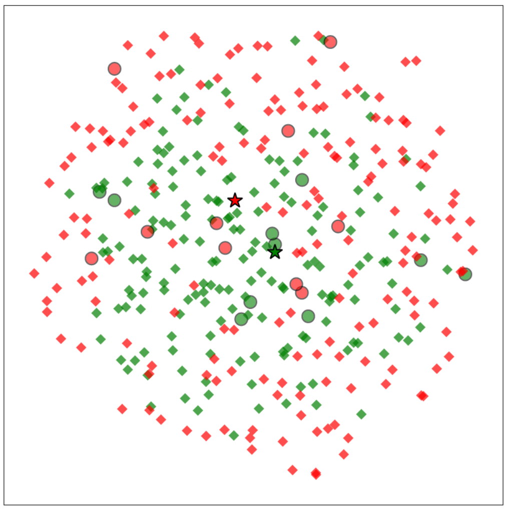
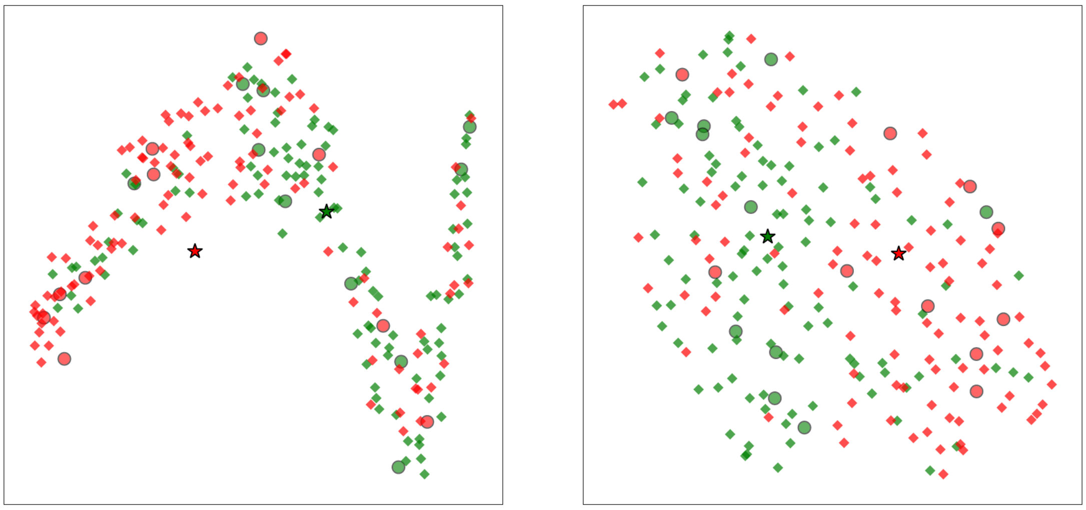
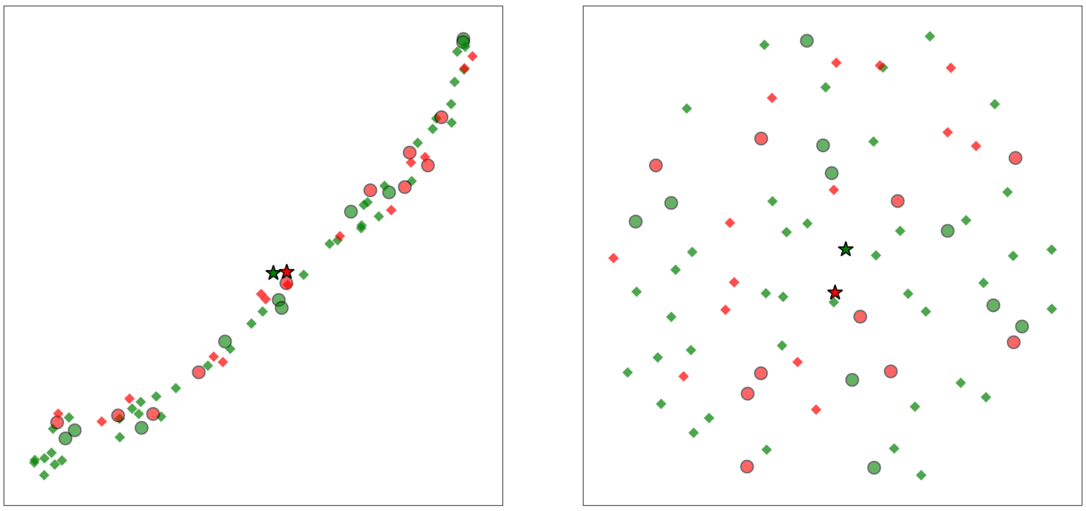

(b) $F1 : 0.54 \mid Silh : 0.003 \mid T : 0.578 \mid C : 0.574$

Figure B.28: 2-D UMAP projection of support and query samples from `QinJ_2012` (studying `cardiometabolic disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



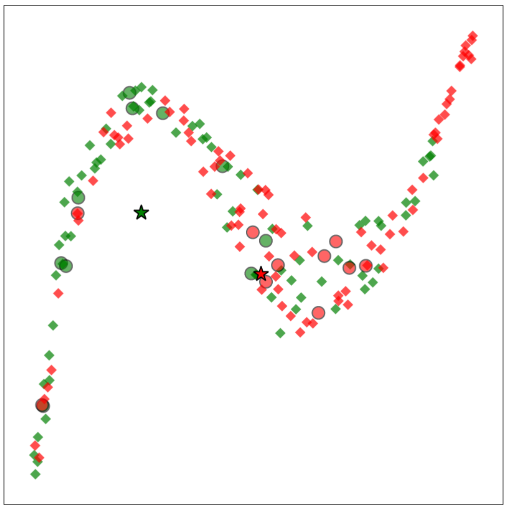
(a) $F1 : 0.63 \mid Silh : 0.023 \mid T : 0.986 \mid C : 0.984$ **(b)** $F1 : 0.66 \mid Silh : 0.025 \mid T : 0.700 \mid C : 0.693$

Figure B.29: 2-D UMAP projection of support and query samples from `ChenB_2020` (studying `immune disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

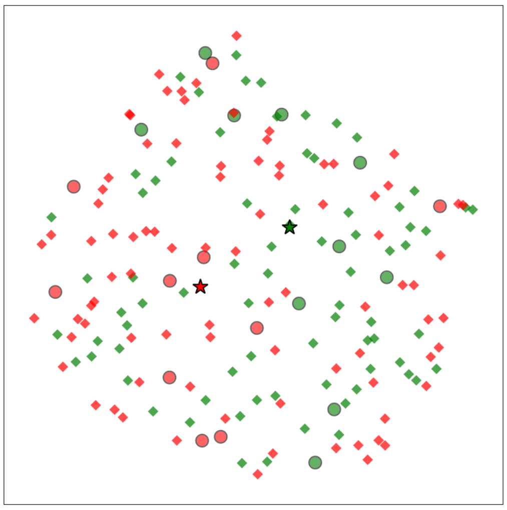


(a) $F1 : 0.35 \mid Silh : 0.005 \mid T : 0.986 \mid C : 0.980$ **(b)** $F1 : 0.38 \mid Silh : 0.017 \mid T : 0.683 \mid C : 0.657$

Figure B.30: 2-D UMAP projection of support and query samples from `YeZ_2018` (studying `immune disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

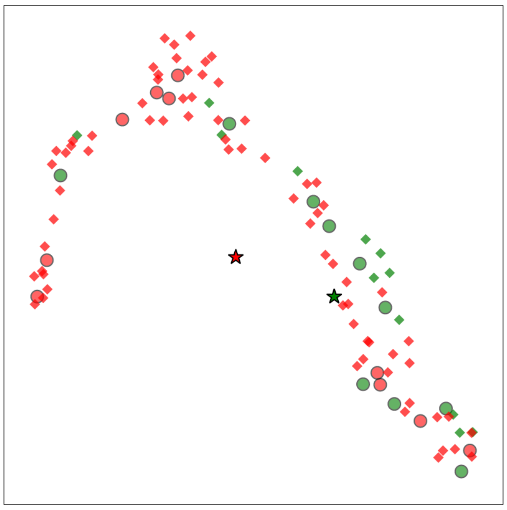


(a) $F1 : 0.59 \mid Silh : 0.015 \mid T : 0.987 \mid C : 0.985$

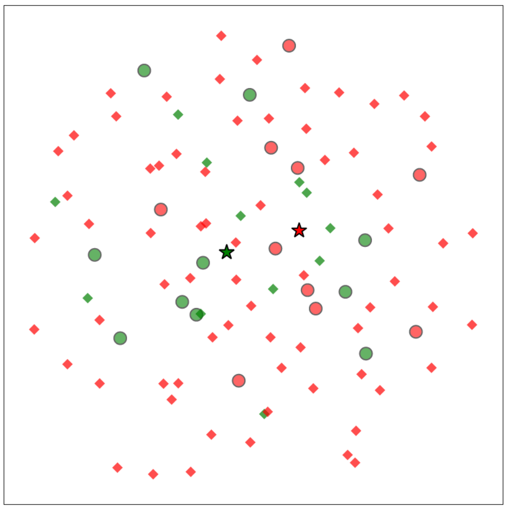


(b) $F1 : 0.69 \mid Silh : 0.008 \mid T : 0.820 \mid C : 0.812$

Figure B.31: 2-D UMAP projection of support and query samples from `ChuY_2021` (studying `immune disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

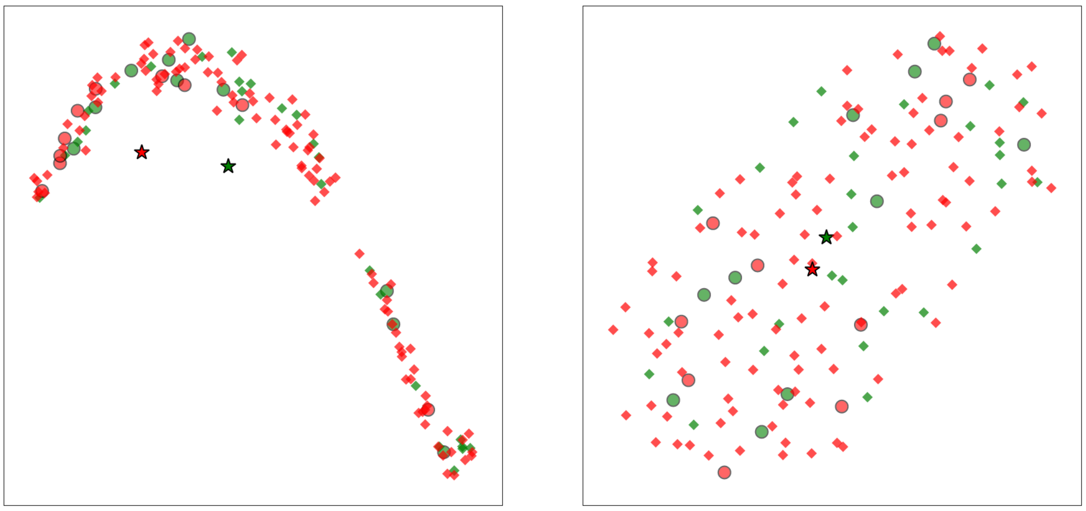


(a) $F1 : 0.66 \mid Silh : 0.010 \mid T : 0.987 \mid C : 0.983$



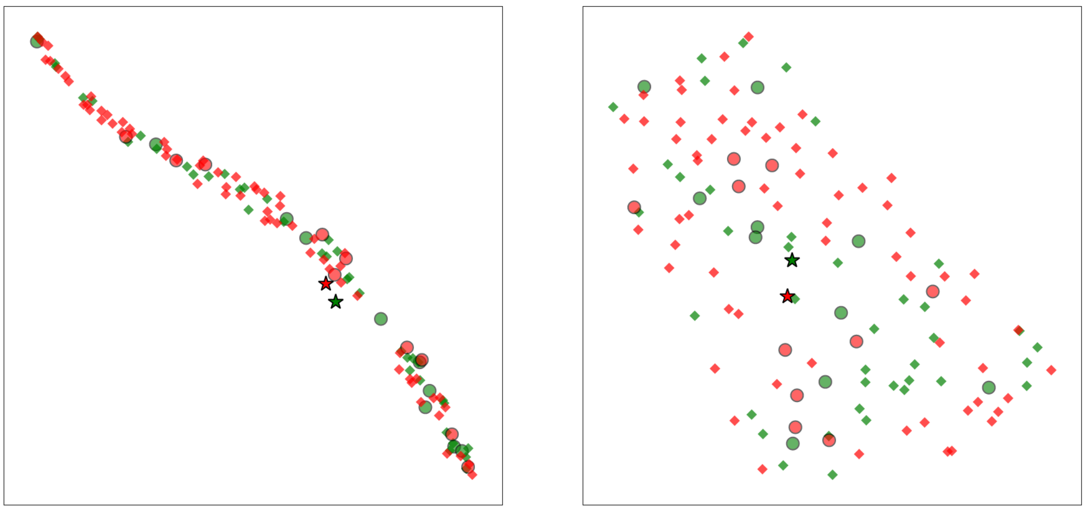
(b) $F1 : 0.75 \mid Silh : -0.014 \mid T : 0.599 \mid C : 0.584$

Figure B.32: 2-D UMAP projection of support and query samples from `ZhouC_2020` (studying `immune disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



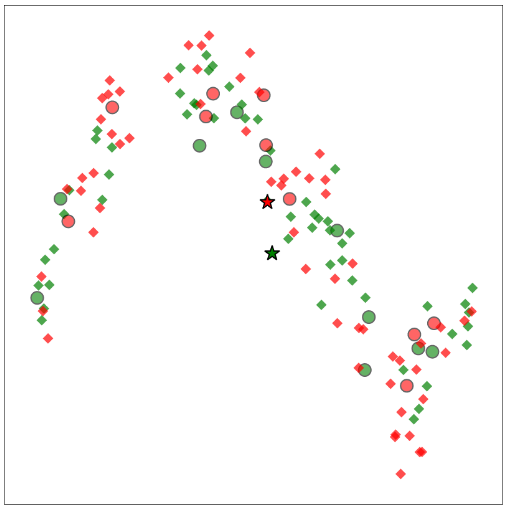
(a) $F1 : 0.73 \mid Silh : 0.006 \mid T : 0.991 \mid C : 0.988$ **(b)** $F1 : 0.53 \mid Silh : -0.069 \mid T : 0.766 \mid C : 0.758$

Figure B.33: 2-D UMAP projection of support and query samples from `HuangR_2020` (studying `immune disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



(a) $F1 : 0.62 \mid Silh : 0.008 \mid T : 0.994 \mid C : 0.990$ **(b)** $F1 : 0.63 \mid Silh : -0.018 \mid T : 0.765 \mid C : 0.752$

Figure B.34: 2-D UMAP projection of support and query samples from `LiuP_2021` (studying `immune disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).


(a) $F1 : 0.48 \mid Silh : -0.001 \mid T : 0.993 \mid C : 0.989$

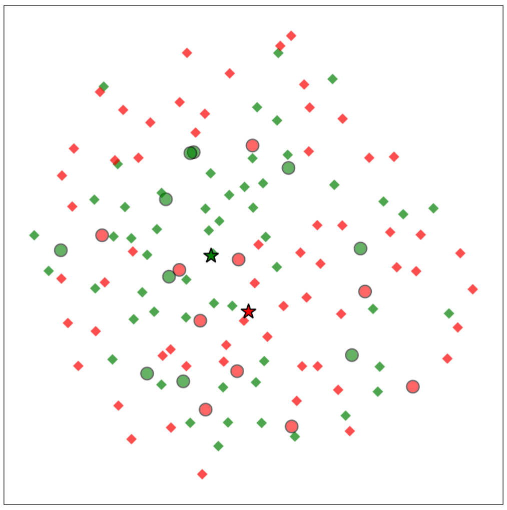
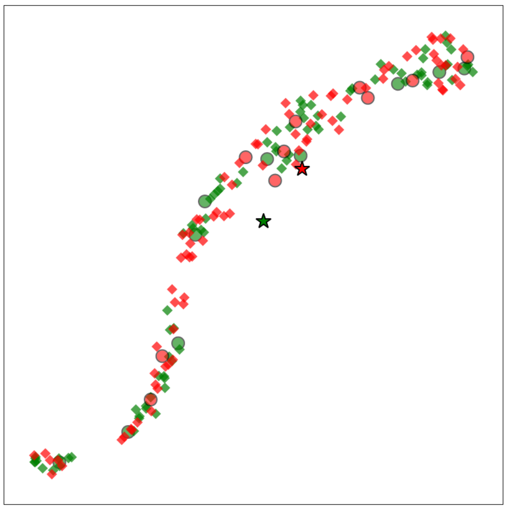

(b) $F1 : 0.59 \mid Silh : -0.001 \mid T : 0.577 \mid C : 0.562$

Figure B.35: 2-D UMAP projection of support and query samples from `YeZ_2020` (studying `immune disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).


(a) $F1 : 0.54 \mid Silh : 0.003 \mid T : 0.987 \mid C : 0.985$

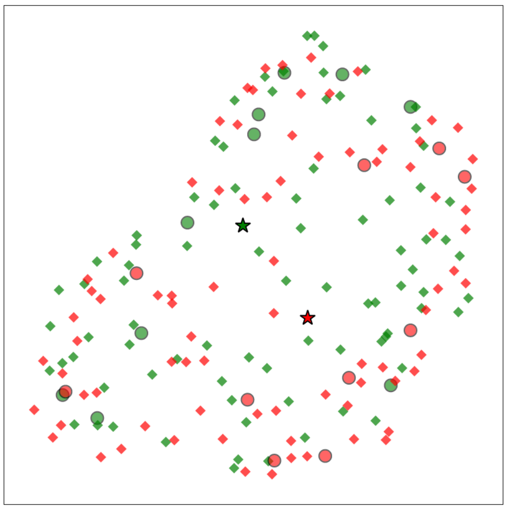
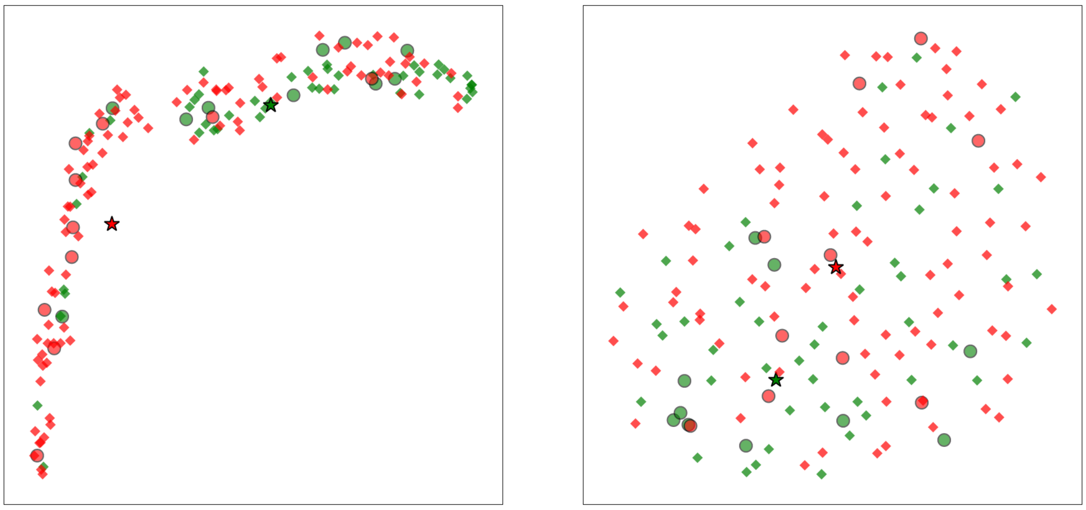
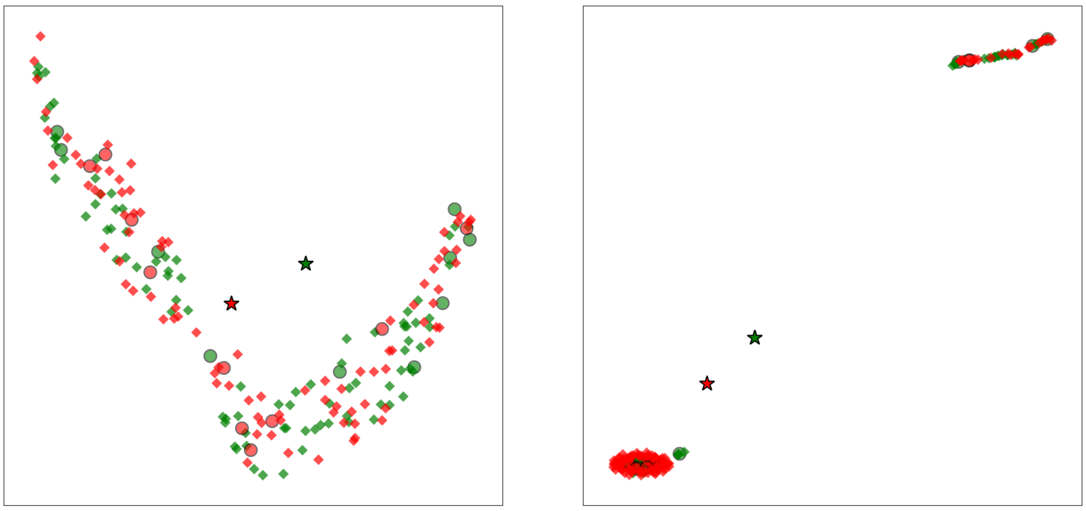

(b) $F1 : 0.55 \mid Silh : 0.000 \mid T : 0.864 \mid C : 0.857$

Figure B.36: 2-D UMAP projection of support and query samples from `ZhangX_2015` (studying `immune disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



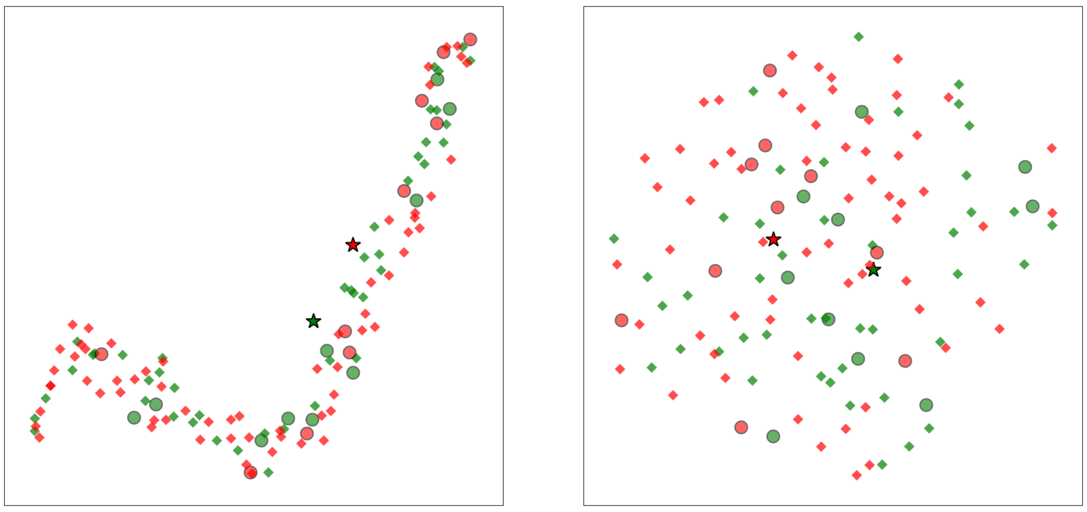
(a) $F1 : 0.74 \mid Silh : 0.015 \mid T : 0.992 \mid C : 0.989$ **(b)** $F1 : 0.72 \mid Silh : -0.042 \mid T : 0.688 \mid C : 0.676$

Figure B.37: 2-D UMAP projection of support and query samples from `ZhuQ_2021` (studying `immune disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



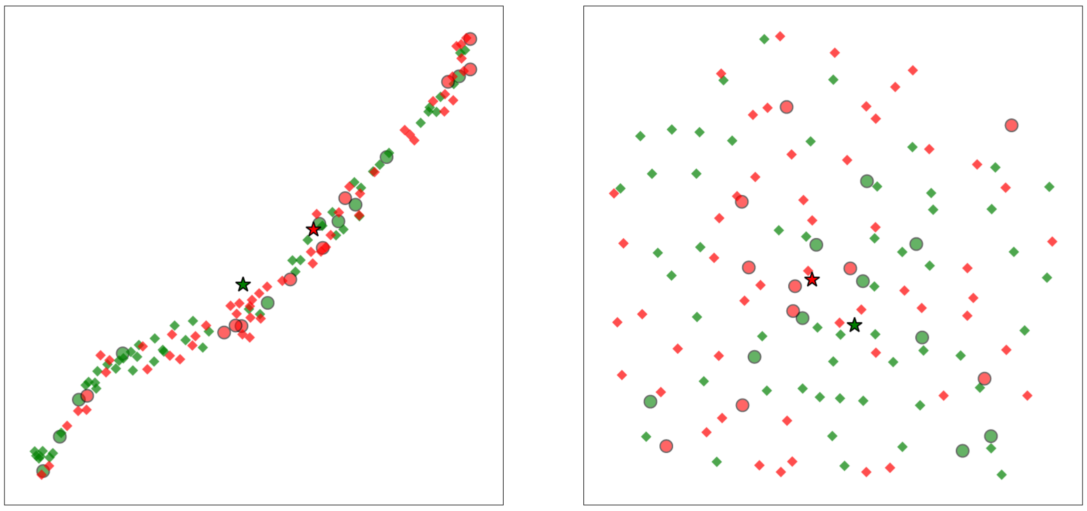
(a) $F1 : 0.57 \mid Silh : 0.007 \mid T : 0.988 \mid C : 0.986$ **(b)** $F1 : 0.64 \mid Silh : 0.006 \mid T : 0.832 \mid C : 0.829$

Figure B.38: 2-D UMAP projection of support and query samples from `YangY_2021` (studying `cancer`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



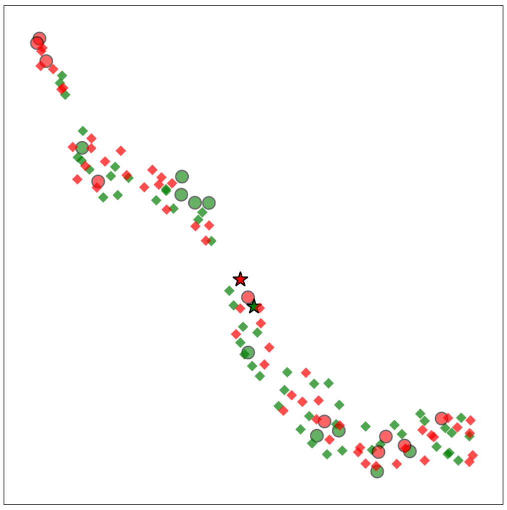
(a) $F1 : 0.65 \mid Silh : 0.007 \mid T : 0.981 \mid C : 0.975$ **(b)** $F1 : 0.61 \mid Silh : 0.004 \mid T : 0.657 \mid C : 0.641$

Figure B.39: 2-D UMAP projection of support and query samples from `YuJ_2017` (studying `cancer`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

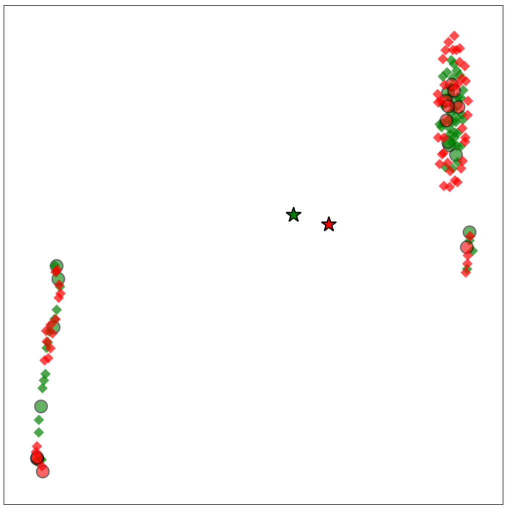


(a) $F1 : 0.56 \mid Silh : 0.008 \mid T : 0.990 \mid C : 0.987$ **(b)** $F1 : 0.52 \mid Silh : 0.004 \mid T : 0.581 \mid C : 0.569$

Figure B.40: 2-D UMAP projection of support and query samples from `ZhuJ_2018` (studying `cancer`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

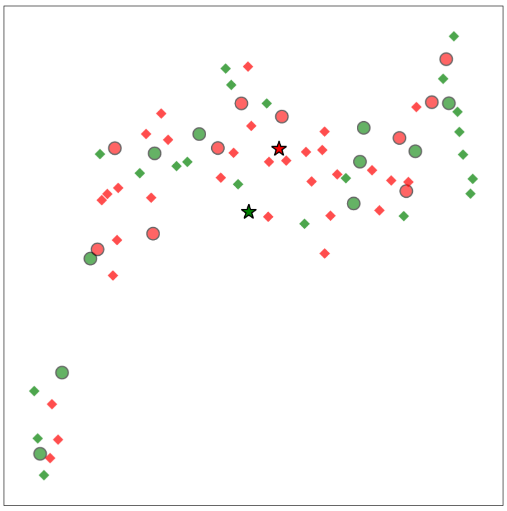


(a) $F1 : 0.54 \mid Silh : 0.006 \mid T : 0.992 \mid C : 0.989$

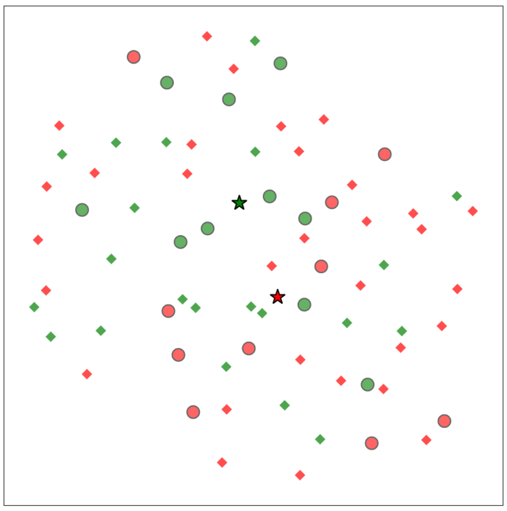


(b) $F1 : 0.59 \mid Silh : -0.013 \mid T : 0.868 \mid C : 0.863$

Figure B.41: 2-D UMAP projection of support and query samples from `WanY_2021` (studying `mental disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

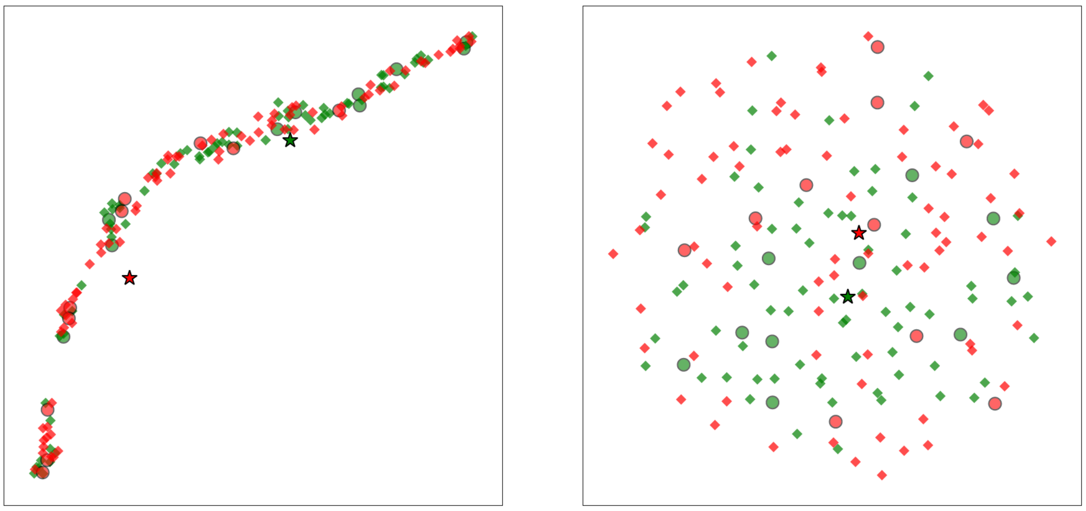


(a) $F1 : 0.61 \mid Silh : 0.010 \mid T : 0.987 \mid C : 0.978$



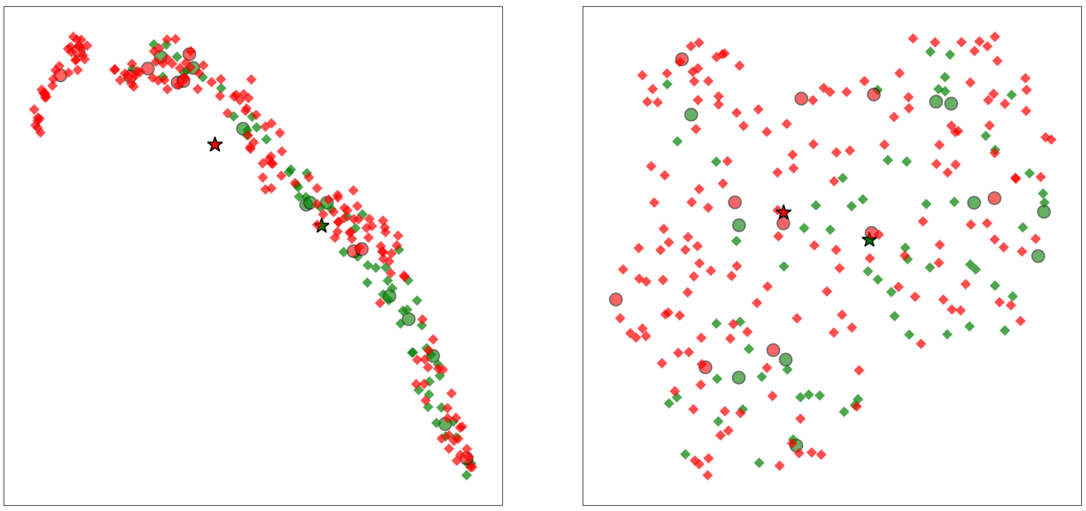
(b) $F1 : 0.68 \mid Silh : -0.002 \mid T : 0.631 \mid C : 0.602$

Figure B.42: 2-D UMAP projection of support and query samples from `WangM_2019` (studying `mental disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



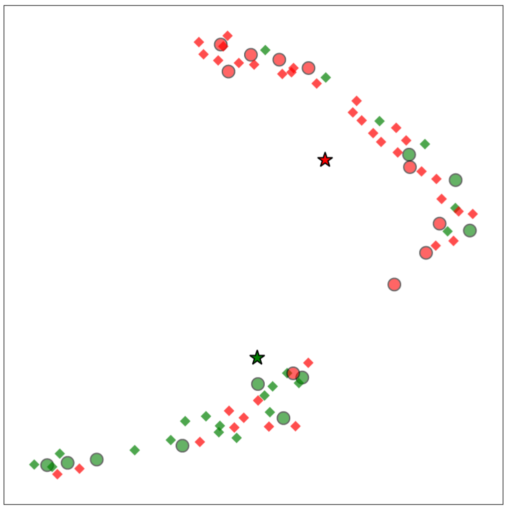
(a) $F1 : 0.55 \mid Silh : 0.003 \mid T : 0.990 \mid C : 0.987$ **(b)** $F1 : 0.48 \mid Silh : 0.001 \mid T : 0.570 \mid C : 0.561$

Figure B.43: 2-D UMAP projection of support and query samples from `ZhuF_2020` (studying `mental disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



(a) $F1 : 0.70 \mid Silh : 0.010 \mid T : 0.982 \mid C : 0.980$ **(b)** $F1 : 0.69 \mid Silh : -0.026 \mid T : 0.781 \mid C : 0.776$

Figure B.44: 2-D UMAP projection of support and query samples from `YeohYK_2021` (studying `infectious disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).


(a) $F1 : 0.73 \mid Silh : 0.016 \mid T : 0.994 \mid C : 0.989$

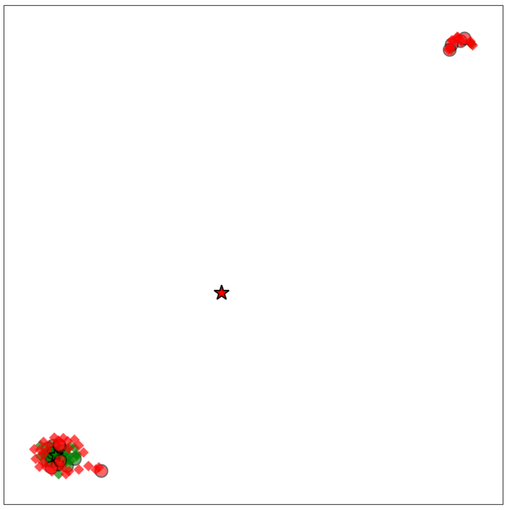
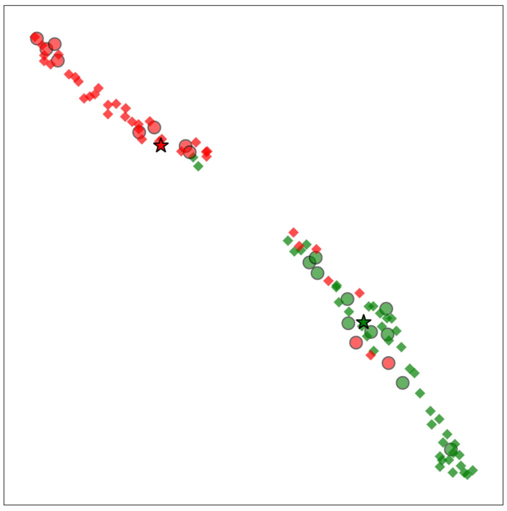

(b) $F1 : 0.70 \mid Silh : -0.008 \mid T : 0.790 \mid C : 0.773$

Figure B.45: 2-D UMAP projection of support and query samples from `HuY_2019` (studying `infectious disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).


(a) $F1 : 0.88 \mid Silh : 0.075 \mid T : 0.991 \mid C : 0.985$

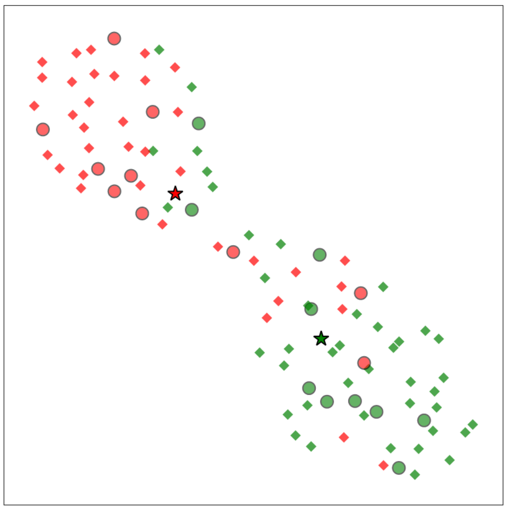
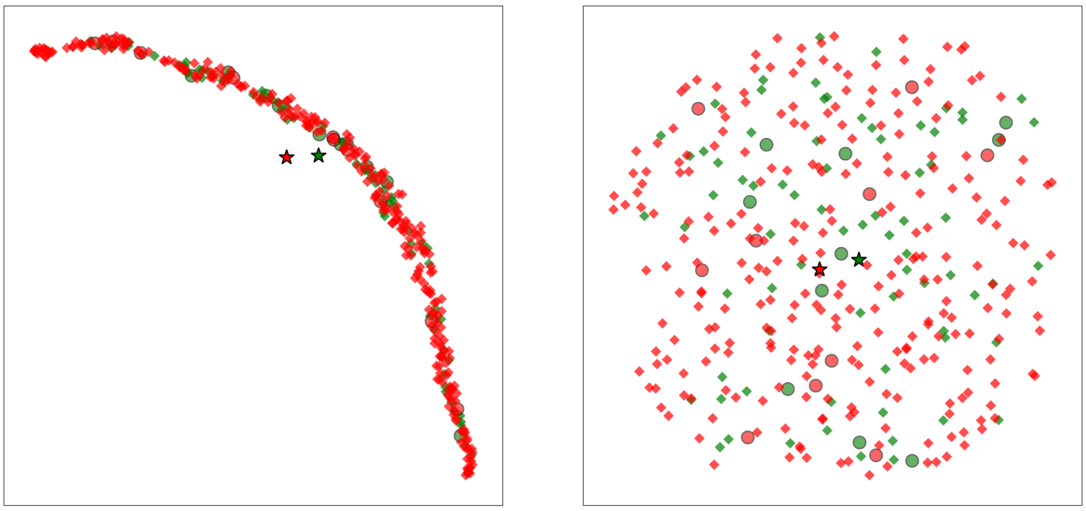
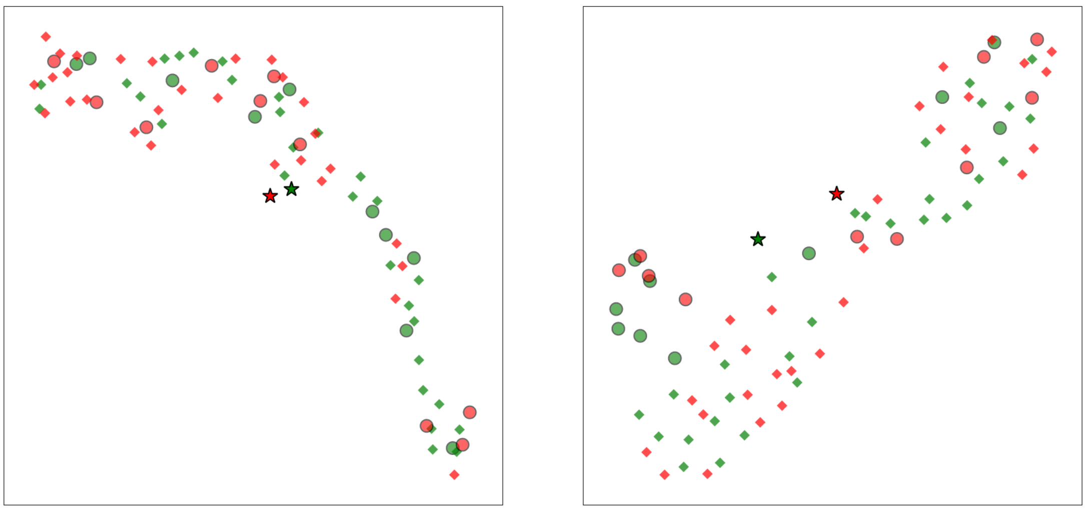

(b) $F1 : 0.77 \mid Silh : 0.113 \mid T : 0.833 \mid C : 0.825$

Figure B.46: 2-D UMAP projection of support and query samples from `HeQ_2017` (studying `digestive disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



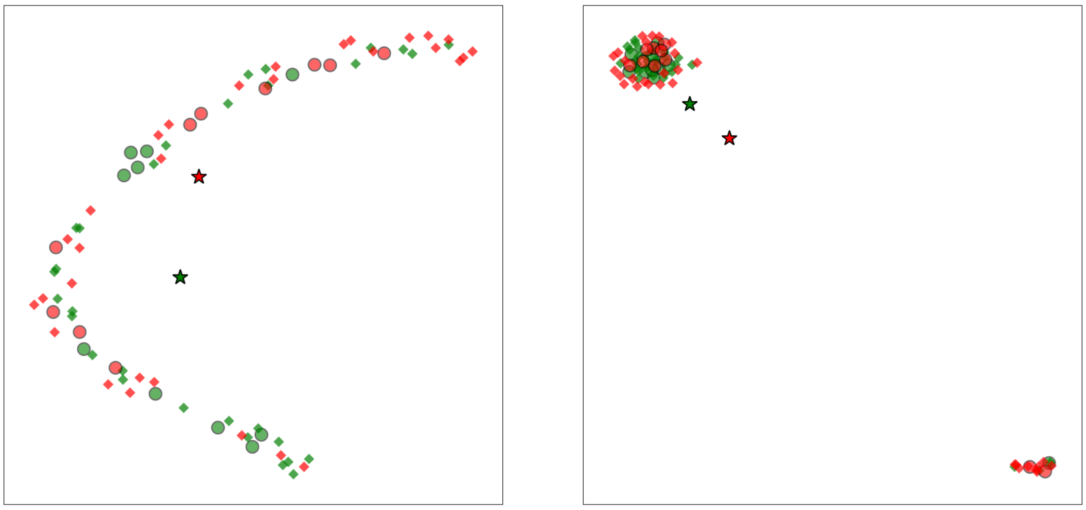
(a) $F1 : 0.63 \mid Silh : 0.000 \mid T : 0.984 \mid C : 0.982$ **(b)** $F1 : 0.68 \mid Silh : 0.025 \mid T : 0.627 \mid C : 0.624$

Figure B.47: 2-D UMAP projection of support and query samples from `HanL_2021` (studying `digestive disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



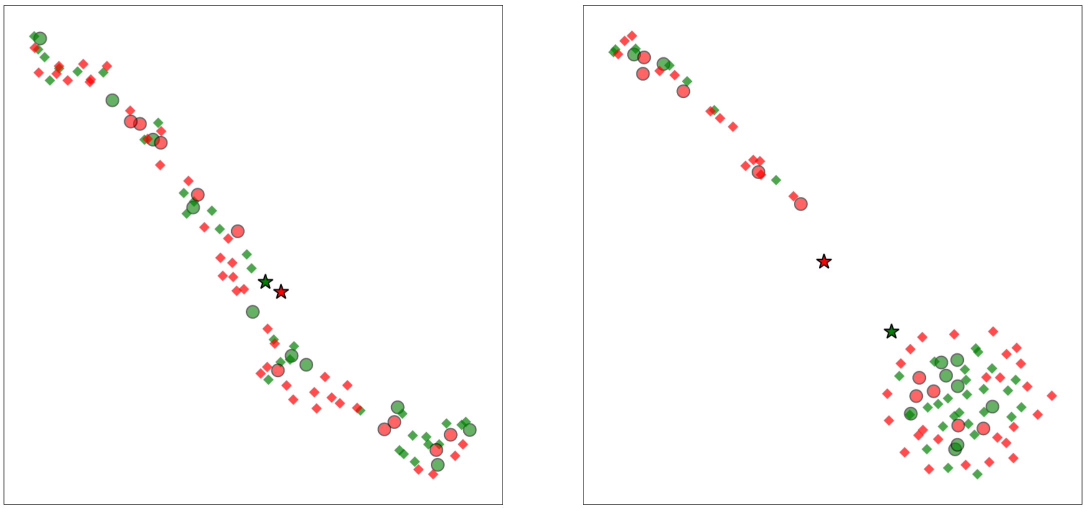
(a) $F1 : 0.58 \mid Silh : 0.010 \mid T : 0.989 \mid C : 0.983$ **(b)** $F1 : 0.54 \mid Silh : 0.001 \mid T : 0.891 \mid C : 0.877$

Figure B.48: 2-D UMAP projection of support and query samples from `MaoL_2021` (studying `nerve disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



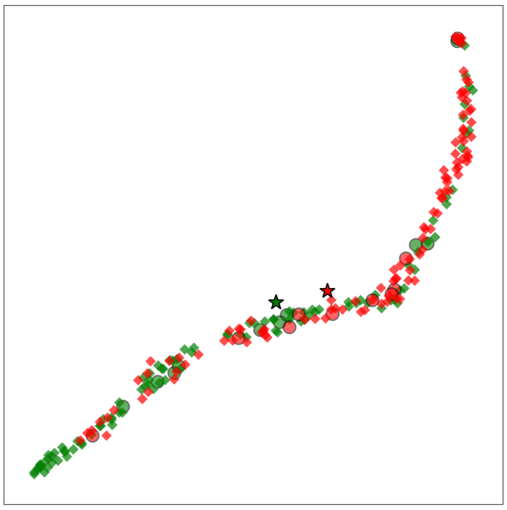
(a) $F1 : 0.60 \mid Silh : 0.012 \mid T : 0.988 \mid C : 0.982$ **(b)** $F1 : 0.47 \mid Silh : 0.047 \mid T : 0.760 \mid C : 0.748$

Figure B.49: 2-D UMAP projection of support and query samples from `QianY_2020` (studying `nerve disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

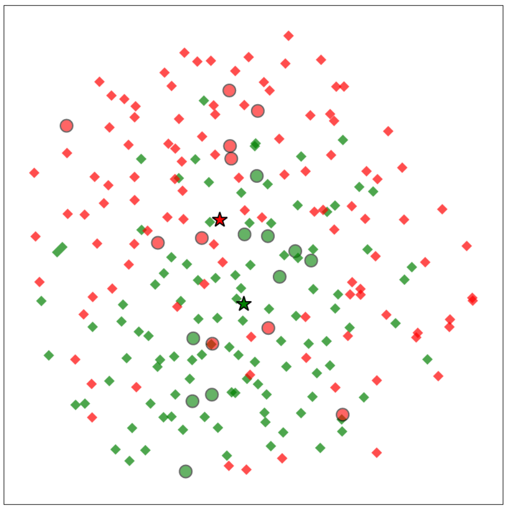


(a) $F1 : 0.78 \mid Silh : 0.044 \mid T : 0.981 \mid C : 0.974$ **(b)** $F1 : 0.56 \mid Silh : -0.004 \mid T : 0.805 \mid C : 0.794$

Figure B.50: 2-D UMAP projection of support and query samples from `QiX_2019` (studying `endocrine disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

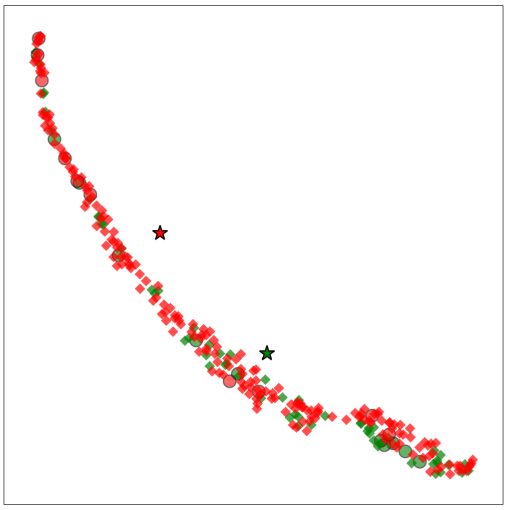


(a) $F1 : 0.70 \mid Silh : 0.033 \mid T : 0.990 \mid C : 0.988$

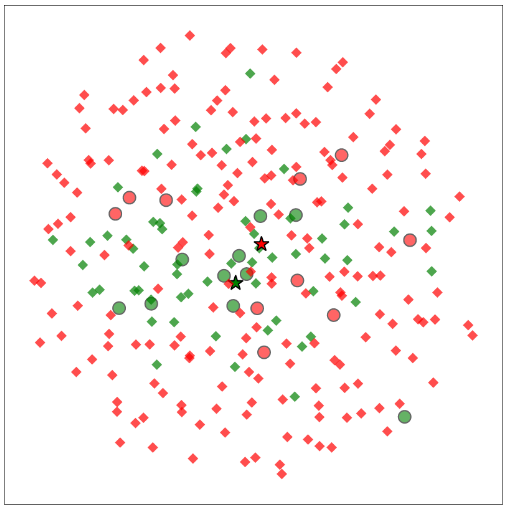


(b) $F1 : 0.80 \mid Silh : 0.007 \mid T : 0.584 \mid C : 0.577$

Figure B.51: 2-D UMAP projection of support and query samples from `QinN_2014` (studying `liver disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).



(a) $F1 : 0.81 \mid Silh : 0.011 \mid T : 0.988 \mid C : 0.986$



(b) $F1 : 0.75 \mid Silh : -0.003 \mid T : 0.568 \mid C : 0.563$

Figure B.52: 2-D UMAP projection of support and query samples from `WangX_2020` (studying `kidney disease`). Left plot is the projection of the original data and the right that of the embedded data. Each point represents a sample from the study, colored by class (green = Healthy, red = Disease), and shaped by type: support (circles), query (diamonds), and prototypes (stars).

C
Additional Methodologies Used

C.1. Statistical Significance Analysis Method

The 34 paired $\Delta F1$ values, one for each leave-one-study-out evaluation in the 10-shot case, are not guaranteed to be normally distributed, so we used the Wilcoxon signed rank test[1], a nonparametric counterpart to the paired *t*-test. To accompany the p-value with an interpretable effect size we computed a bias corrected bootstrap 95 % confidence interval (CI) for the mean Δ $(10,000$ resamples, seed = 42).

Procedure

1. For each of the 34 leave-one-study-out evaluations we computed first the mean classification performance over the folds and afterwards the difference:

$$\Delta_{study_i} = F1_{avg,protonet,study_i} - F1_{avg,RF,study_i} \tag{C.1}$$

2. The Wilcoxon signed-rank test was run on the vector Δ (two-sided, `zero_method="wilcox"`).
3. To obtain the CI, we drew $10,000$ bootstrap samples of length 34 with replacement, recorded the mean Δ for each, and took the 2.5th and 97.5th percentiles.

[1]SciPy implementation is used: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html`

D Data Characteristics

D.1. Disease Groups in the Data

Transferability of latent information between studies is important for generalizability. One hypothesis is that generalization to a test study is better when it comes from a disease group that appears more frequently in the training data, due to the richer representation learned during training.

Although each study in the sun et al. data[68] focuses on a different disease group, there is an imbalance here. Table D.1 gives an overview of the disease groups and their corresponding studies according to the original paper for the studies kept after the preprocessing pipeline.

Table D.1: Disease groups found in the dataset from Sun et al.[68] with the number of diseases belonging to that group and the corresponding studies per group.

Disease group	**Number of studies**	**Studies**
Cardiometabolic disease	10	LiJ_2017, LiR_2021, LiuR_2017, YanQ_2017, ZhongH_2019, ZuoK_2019, JieZ_2017, WangQ_2021, ZengQ_2021, QinJ_2012
Immune disease	9	ChenB_2020, YeZ_2018, ChuY_2021, ZhouC_2020, HuangR_2020, LiuP_2021, YeZ_2020, ZhangX_2015, ZhuQ_2021
Cancer	3	YangY_2021, YuJ_2017, ZhuJ_2018
Mental disease	3	WanY_2021, WangM_2019, ZhuF_2020
Infectious disease	2	YeohYK_2021, HuY_2019
Digestive disease	2	HeQ_2017, HanL_2021
Nerve disease	2	MaoL_2021, QianY_2020
Endocrine disease	1	QiX_2019
Liver disease	1	QinN_2014
Kidney disease	1	WangX_2020