FSL-LFMG: Few-shot Learning with Augmented Latent Features and Multitasking Generation for Enhancing Multiclass Classification on Tabular Data: Existing and New Concepts

Aviv A. Nur

A Thesis
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Master of Science
in
Data Science

August 2024

APPROVED:

Professor Chun-Kit Ngan, Thesis Advisor
Worcester Polytechnic Institute

Dr. Rolf Bardeli, Thesis Co-Advisor
thyssenkrupp Materials Services GmbH, Essen, Germany

Professor Randy Paffenroth, Thesis Reader
Worcester Polytechnic Institute

Professor Elke A. Rundensteiner, Head of Department
Worcester Polytechnic Institute

Abstract

In my Master's thesis, I propose advancing Prototypical Networks that employs augmented latent features (LF) by an autoencoder and multitasking generation (MG) by STUNT in the few-shot learning (FSL) mechanism. Specifically, the achieved contributions to this work are sixfold. First, I propose the FSL-LFMG framework for few-shot multiclass classification on tabular data. This framework incorporates sample-level data augmentation using autoencoders, task-level data augmentation via an enhanced STUNT framework, and Prototypical Networks to capture generalized knowledge. Second, I design the latent features learning and augmentation process that employs autoencoders to extract significant features, which are then used to enhance the quality and diversity of the training data. Third, I employ the enhanced STUNT Multitasking Generation framework that uses K-medoids instead of K-means to generate more accurate tasks. Fourth, I implement an advanced Prototypical Networks with Manhattan distance as a classifier effectively address the multiclass classification problem. Fifth, I conduct an extensive experimental study on four diverse domain datasets—Net Promoter Score segmentation, Dry Bean type, Wine type, and Forest Cover type—to prove that my FSL-LFMG approach on the multiclass classification outperforms the Tree Ensemble models and the One-vs-the-rest classifiers by 7.8% in 1-shot and 2.5% in 5-shot learning. Finally, I demonstrate the adaptation of the new concept task in the model obtained from the FSL-LFMG framework — from the NPS segmentation (the existing concept) and obtain a level of customer's loyalty (the new concept) — to assess the power of generalization of this framework by significant results of the mean test accuracy in both 1-shot setting (83.95%) and 5-shot setting (103.52%).

Keywords— Few-shot Learning, Machine Learning, Deep Learning, Multiclass Classification, New Concept Learning, Autoencoders, Random Forest, CatBoost, One Vs Rest Classifier, STUNT, Prototypical Networks, Tabular Data

Acknowledgments

First and foremost, I express my deepest gratitude to Allah for providing me with the strength, wisdom, and perseverance to complete this thesis. I am profoundly grateful to my wife, whose unwavering support and encouragement have been invaluable throughout this journey. To my son, thank you for being a source of joy and motivation. I also extend my heartfelt thanks to my parents and parents-in-law, whose sacrifices and guidance have been fundamental to my achievements.

I would like to extend my sincere appreciation to my thesis advisor, Professor Chun-Kit Ngan, for his insightful guidance, continuous support, and invaluable feedback. His expertise and dedication have been instrumental in shaping this research.

Special thanks go to my thesis co-advisor, Dr. Rolf Bardeli, for his support and invaluable insights, which have greatly contributed to the success of this work.

I am also grateful to my thesis reader, Professor Randy Paffenroth, for his thoughtful comments and constructive suggestions that have significantly improved the quality of this thesis.

I would like to acknowledge my academic advisor, Professor Xiangnan Kong, and the Head of Data Science Program, Professor Elke Rundensteiner, for their advice and support throughout my academic journey.

Finally, I express my gratitude to the Fulbright Program and Alex F. Backlin Fund Scholarship for providing me with the opportunity and resources to pursue my studies and research. This thesis would not have been possible without the generous support of the

Fulbright Scholarship and Alex F. Backlin Fund Scholarship.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Background

The increasing volume of data across various sectors, such as telecommunications, agriculture, and finance, has led to a pressing demand for effective multiclass classification [1]. For instance, the telecom industry has used many machine learning (ML) models, such as random forests, decision trees, and discriminant feature analysis, to forecast customer attrition and enhance investment optimization. These techniques seek to predict customer behavior and improve investment choices [2]–[6]. In the field of agriculture, ML and deep learning (DL) improve crop monitoring, yield estimation, and productivity, showcasing their essential impact on improving farm management and productivity [7]–[11]. In the finance industry, ML and DL address tasks like risk assessment, pricing, and the development of optimal insurance packages through various methodologies such as artificial neural networks and clustering algorithms. These approaches underscore the crucial role of data and the necessity to adapt to changing financial patterns for improved decision making and efficiency [12]–[14].

The widespread use of data in multiple industries typically involves the utilization of tabular data that has been demonstrated by a 2023 Kaggle survey of 14,000 data scientists. The poll indicated that a substantial proportion of professionals within those industries,

ranging from 50% to 90%, relied on tabular data in their work environments [15], [16]. The inclination towards tabular data presents distinct challenges such as high dimensionality, heterogeneity, and critical interdependencies among features, which are not found in images or other data modalities [17]. Despite these challenges, the adoption of innovative multiclass classification methods is still growing that demonstrates the importance of these methods in enhancing decision-making and operational efficiency across industries.

1.2 Research Challenges

Presently, the existing approaches for multiclass classification on tabular data can be broadly divided into two categories: DL Models and Tree Ensemble (TE) Models [18]. Recent advances in DL models, such as TabNet, Neural Oblivious Decision Ensembles (NODE), and Disjunctive Normal Formulas (DNF-Net), have demonstrated exceptional outcomes across diverse domain datasets [19]–[21]. These models possess the ability to delve into intricate connections among features, resulting in heightened efficiency and performance for tasks involving high-dimensional, structured data. Each model utilizes distinct mechanisms for processing feature selection, which further improves their overall effectiveness. However, these models present challenges in terms of complexity and computation, as well as interpretability. On the other hand, the TE models, including Random Forest and Gradient Boosting, offer enhanced interpretability and reduced computational complexity. In particular, Gradient Boosting, such as XGBoost, exhibits significantly better performance in tabular data compared to DL models [17], [18]. However, the remarkable performance of these models is highly reliant on the utilization of copious amounts of training data, which are inadequate in some domains and require substantial storage space [22], [23]. Additionally, if the amount of training data is insufficient, it results in an overfitted model that lacks generalizability.

Few-shot learning (FSL) is an ML technique that trains on a small number of labeled samples, typically one to five samples per class, providing a potential solution to the afore-

mentioned issues [24], [25]. This technique enables efficient learning of multiclass classification tasks with only a limited amount of data [26]. Although FSL has achieved noteworthy success in the domain of image classification, research on these techniques on tabular data has been widely underexplored [27]. Furthermore, the application of FSL in conjunction with TE models on tabular data is very challenging because of the models' limitations in generalizing on a few data samples per class.

An effort to address the limitations of TE models led to the implementation of the One-vs-the-rest (OvR) multiclass technique [28]. This method is specifically designed for multiclass classification and involves dividing the tasks into a series of binary tasks. The OvR classifier strategy can be integrated into various existing ML conventional models, including TE models, as the base estimators. This technique is expected to enhance the classification capabilities of tree-based models by splitting tasks into binary tasks. However, there is an opportunity that this technique provides suboptimal results due to the potential loss of significant data characteristics, such as complex interclass correlation and interaction, which could result in unsatisfactory performance in classification tasks.

To improve and generalize the ability of models is to augment the data, thereby increasing data variability. Data augmentation can be conducted either at the sample or task level [29]. At the sample level, typical methods for image data involve modifying pixel properties through actions, such as rotation, scaling, cropping, and other similar approaches. These actions are performed to increase the variety of data. On the contrary, when it comes to tabular data, there is currently no recognized approach that can complete this data augmentation task. In order to tackle this issue, we investigate the use of autoencoders to extract significant latent features. Two methods have been experimented in this context: one is to directly apply the extracted latent features to a classifier, and the other is to concatenate these encoded latent features with the original data in order to enhance the number of features. The main contributions in this work are to utilize the knowledge found in large datasets to improve the accuracy of multiclass classification that leads to higher levels of accuracy. Despite

its potential merits, this methodology is not reliable for comprehending new tasks, as it primarily concentrates on a single task, i.e., the process of learning to predict a single outcome, including binary, multiclass, or continuous values, respectively, from a labeled dataset. Hence, in addition to latent features, it is essential to employ task-level data augmentation methods that can improve the precision of classification, while also facilitating the model to efficiently learn new tasks. The task-level augmentation entails generating new tasks to offer the model a boarder range of learning experiences.

The incorporation of Self-generated Tasks from UNlabeled Tables (STUNT) is recognized as a prominent task-level data augmentation strategy [27]. Through the treatment of data as unlabeled, this technique has the potential to generate various tasks for a single dataset. This outcome is attained by applying the K-means clustering method to create new labels. It is anticipated that the generation of self-tasks leads to effective generalization, as the model acquires knowledge from multiple tasks, i.e., the process of jointly learning to predict multiple outcomes on inputs of the same dataset, simultaneously. In order to capture the generalized knowledge, a meta-learning scheme called Prototypical Networks (ProtoNet) is utilized as a classifier [30]. In contrast to ProtoNet, tree-based models lack capabilities to perform generalization on small datasets because their complicated structures tend to overfit specific training samples instead of capturing broader patterns. A lack of data also makes methods, such as bagging, less useful, resulting in trees that are not varied and less-than-ideal decisions at splits [31]. ProtoNet has proven to be highly accurate and effective across various types of data [27], [30], [32]. This approach successfully generates representative prototypes or mean embeddings for each class by utilizing Euclidean distance to determine the proximity of a target task to its prototype.

To incorporate the advantages of the above methods into my approach, I propose advancing ProtoNet that employs augmented latent features (LF) by an autoencoder and multitasking generation (MG) by STUNT in the few-shot learning mechanism. Specifically, the achieved contributions achieved to this work are fourfold. First, I propose an FSL-LFMG framework

to develop an end-to-end few-shot multiclass classification workflow on tabular data. This framework is composed of three main stages that include (i) data augmentation at the sample level utilizing autoencoders to generate augmented LF, (ii) data augmentation at the task level involving self-generating multitasks using the STUNT approach, and (iii) the learning process taking place on ProtoNet, followed by various model evaluations in my FSL mechanism. Second, due to the outlier and noise sensitivity of K-means clustering [33] and the curse of dimensionality of the Euclidean distance [32], I enhance and customize the STUNT approach by using K-medoids clustering that is less sensitive to noisy outliers and Manhattan distance that is the most preferable for high-dimensional data. Third, I conduct an extensive experimental study on four diverse domain datasets—Net Promoter Score (NPS) segmentation, Dry Bean type, Wine type, and Forest Cover type—to prove that my FSL-LFMG approach on the multiclass classification outperforms the TE models and the OvR classifiers by 7.8% in 1-shot and 2.5% in 5-shot learning. Finally, I demonstrate the adaptation of new concept task on the model obtained from the FSL-LFMG framework — from the NPS segmentation (the existing concept) and obtain a level of customer's loyalty (the new concept) — to assess the power of generalization of this framework by significant results of the mean test accuracy in both 1-shot setting (83.95%) and 5-shot setting (103.52%).

1.3 Problem Statement

After identifying the key challenges in this research, I can define the problem statement that this study aims to address as follows:

1. How to create a framework that can integrate few-shot learning for multiclass classification problems on tabular data and overcome the challenges of current methods?

2. How to extract the most significant features to support data augmentation on few-shot examples?

3. How to generate diverse tasks to offer a model a broader range of learning experience on few-shot examples?

4. How to capture the generalized knowledge learned from above to support the development of a multiclass classifier?

5. How to design and conduct experimental studies to verify and evaluate the superiority of my proposed approach over traditional tree ensemble models and OvR classifiers in few-shot learning scenarios?

6. How to adapt my developed framework to the new concepts in few-shot learning settings?

1.4 Thesis Statement and Summary of Research Contributions

According to the problem statement provided, the thesis statement can be defined as follows:

> It is possible to create a framework that integrates few-shot learning for multiclass classification on tabular data by enhancing data augmentation, generating diverse learning tasks, and capturing generalized knowledge for robust classifier development. This framework is experimentally validated to demonstrate its superiority over traditional tree ensemble models and OvR classifiers in few-shot learning scenarios and adapts to new concepts in these settings.

In this thesis, I focus on a framework, models, algorithms, and experimental case studies to bridge the gaps to solve the above problems. Specifically, I propose an advanced Prototypical Networks-based framework incorporating augmented latent features through autoencoders and multitask generation using the STUNT approach to address the limitations of current methods. The key technical contributions of this thesis are as follows:

1. Develop the FSL-LFMG framework for few-shot multiclass classification on tabular data, involving sample-level data augmentation using **autoencoders**, task-level data augmentation using the **enhanced STUNT**, and **Prototypical Networks** to capture generalized knowledge.

2. Utilize autoencoders to extract significant features in the Latent Features Learning and Augmentation process, enhancing the quality and diversity of the training data through sample-level data augmentation.

3. Employ an enhanced STUNT Multitasking Generation process, replacing K-means with K-medoids to generate tasks more accurately, leveraging the benefits of K-medoids over K-means.

4. Implement an advanced Prototypical Networks, a common neural network in few-shot learning, with Manhattan distance as a classifier to capture generalized knowledge and address the multiclass classification problem effectively.

5. Conduct the extensive experiments on four diverse datasets to demonstrate that my FSL-LFMG approach outperforms both TE models and OvR classifiers by significant margins in both 1-shot and 5-shot learning scenarios.

6. Demonstrate the adaptation of new concept tasks on the models obtained from the FSL-LFMG framework and provide its experimental results.

The remainder of this thesis is organized as follows. Chapter 2 discusses related works on FSL. Chapter 3 introduces my proposed FSL-LFMG framework. Chapter 4 describes the process that learns the LF using autoencoders. Chapter 5 explains the MG approach using STUNT. Chapter 6 explains the meta learning process using Prototypical Networks. Chapter 7 details the experimental results, analyses, and discussion. Chapter 8 demonstrates the adaptation of new concept task on the model obtained from the FSL-LFMG framework.

Finally, in Chapter 9, I provide a conclusion and outline my future work for this Master's thesis.

Chapter 2

Related Work

2.1 Traditional Multiclass Classification Algorithms

Two widely recognized algorithms that have attracted significant attention in the field of multiclass classification are Random Forest and CatBoost.

2.1.1 Random Forest

Random Forest, introduced by Breiman [34], is an ensemble learning method that combines multiple decision trees to improve the overall accuracy and robustness of the classification task. The algorithm constructs a collection of decision trees, each trained on a random subset of the features and a random subset of the training data. The final prediction is made by aggregating the predictions of the individual trees through majority voting.

2.1.2 CatBoost

On the other hand, CatBoost, developed by Yandex in 2017, is a gradient boosting framework particularly adept at handling categorical variables, a common occurrence in tabular data. One of CatBoost's key advantages is its ability to automatically handle categorical variables without the need for explicit feature engineering, a process that can be time-consuming and

often requires domain expertise [35].

2.2 Challenges with Limited Data

Both Random Forest and CatBoost have demonstrated impressive performance in multiclass classification problems in tabular data, especially when ample training data is available. For example, studies by [36], [37] and [38] show that Random Forest performs well in big data environments with abundant training data. Similarly, studies by [39], [40] indicate that CatBoost performs competitively in fields such as human resource analytics, finance, and marketing. However, the implementation of these algorithms in limited sample settings or few-shot scenarios is still underexplored. Some studies such as those by [41] and [42] address the problem of multiclass classification but focus on imbalanced data where limited data is available only for minority classes. These studies do not tackle scenarios where all classes have a limited number of examples.

In a few-shot setting, CatBoost is used as a baseline by [27] which results in suboptimal performance, but the gap is not significant compared to the approach they propose. Given the strong performance of Random Forest and CatBoost in large-data scenarios, it is important to include these two models as the baselines in my FSL study. I also design an OvR model using both Random Forest and CatBoost as baseline models to examine the effect of data augmentation at the task level in FSL.

2.3 Few-shot Learning

2.3.1 Common Approach

The idea of FSL comes from the ability of human to learn from a limited example. Some FSL methods [43]–[47] have shown success in various domains, especially in image classification. In general, there are three approaches for image classification using FSL, including metric-based

learning, transfer learning, and meta-learning models. A metric-based learning model is trained to learn a similarity metric or space in which samples from the same class are closer together and samples from different classes are farther apart, often used in the classification of hyperspectral images [48]. A transfer learning model is a pre-trained model that possesses knowledge (i.e., features and weights) leveraged for an FSL task, especially useful when the few-shot task has limited data but is related to the larger dataset, like decomposing images into objects for object-level representation learning [49]. A meta-learning model is trained on various learning tasks to quickly adapt to new tasks using few training examples, such as combining meta-learning with transfer learning and metric learning for medical image classification [44]. However, the application of FSL in other domains, especially in tabular data, has not received much attention and its research is still not widely explored by researchers even though its application can be a solution to the high demand on the need for a large amount of training data.

2.3.2 Few-shot Learning on Tabular Data

Some researchers have tried to propose some specialized FSL frameworks on tabular data: (1) TabLLM that is a method to use large language models for zero-shot and few-shot classification of tabular data by serializing tabular data into natural-language strings and fine-tuning with labeled examples, competing with traditional models like gradient-boosted trees in very-few-shot settings [50]; (2) FLAT that is an approach for few-shot learning on tabular datasets to handle heterogeneous feature spaces by learning low-dimensional embeddings of datasets and columns using a Dataset2Vec encoder, and then applying a graph attention network to manage the heterogeneity [51], [52]; (3) STUNT that is a framework for few-shot semi-supervised tabular learning to generate diverse tasks from unlabeled data, using a meta-learning scheme to acquire generalizable knowledge and an unsupervised validation scheme for hyperparameter optimization [27]. It utilizes randomly chosen columns as target labels for these tasks, employing a meta-learning scheme to acquire generalizable knowledge.

2.3.3 New Concept Learning

Few-shot learning trains models to recognize new classes or new concepts with few labeled samples, essential for situations where large datasets are impractical or expensive [53]. Prototypical Networks and Relation Networks are frameworks designed to improve a few-shot learning by efficiently capturing the data structure and enabling the generalization of models from a few examples in image classification problems [30], [54]. In this context, a new concept is defined as an unseen class or category that has not been available during training, and with the power of generalization of the FSL framework, learning a new concept is possible using meta-learning [55]–[57].

Learning a new concept can also be related to open-world classification that starts with a known set of classes and incrementally learns about unknown items or hidden classes from a dynamic data stream [58]. In this approach, inputs are classified into known classes or recognize them as unknown, handle unknown or hidden instances efficiently, and continuously learn new classes while retaining previously acquired knowledge [58], [59]. The open-world framework usually addresses the challenges of dynamic environments and continuous learning.

Chapter 3

FSL-LFMG Framework

3.1 Introduction to FSL-LFMG Framework

In this section, I describe and explain my proposed FSL-LFMG framework that is an end-to-end pipeline consisting of four main modules shown in Figure 3.1. The modules include Data Preprocessing (DP), Latent Features Augmentation (LFA), Multitasking Generation (MG), and Prototypical Network (PN).

3.2 Data Preprocessing (DP)

First, raw tabular data is passed into the DP module that processes and cleans the data in three separate steps in sequence. In STEP 1, the data is divided into three parts, i.e., training set, validation set, and test set. The ratio among them is 64:16:20. For instance, in the NPS telecom dataset, which comprises 100,000 samples, the dataset is divided into 64,000 samples for the training set, 16,000 samples for the validation set, and 20,000 samples for the test set. In STEP 2, to deal with numerical features in the dataset, I apply the Min-Max scaler to ensure that all those features are on the same scale, for instance, between 0 and 1. The Min-Max scaling is defined by the following equation:

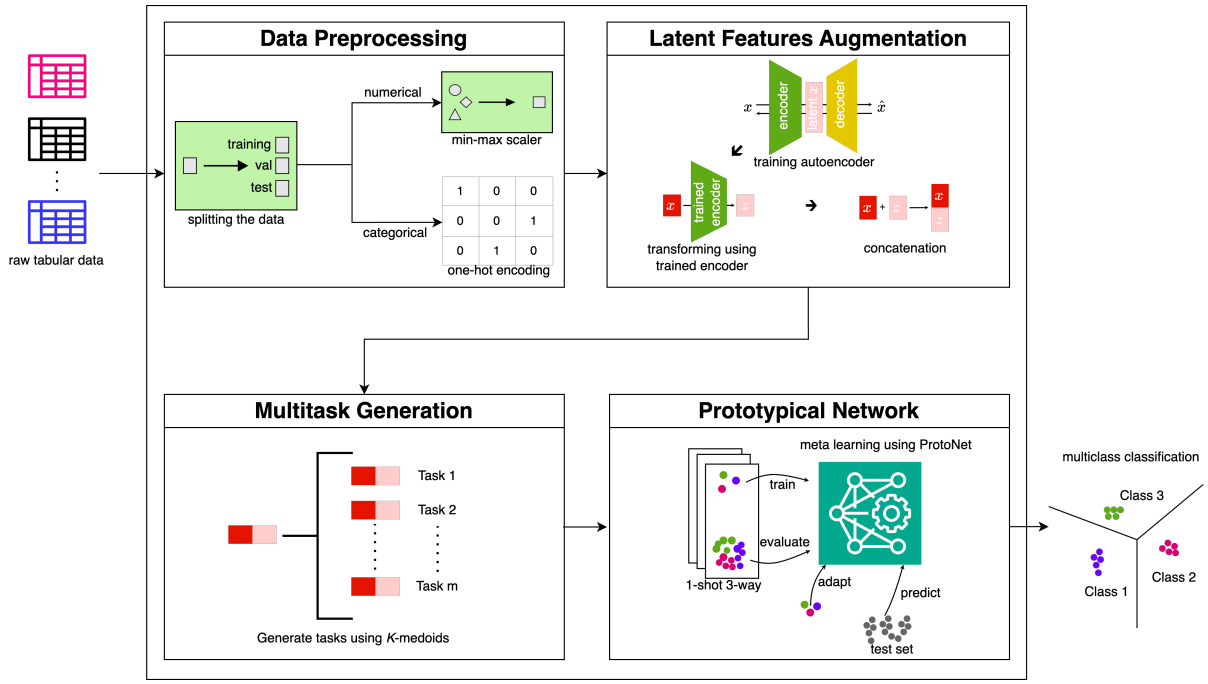


Figure 3.1: FSL-LFMG Framework.

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{3.1}$$

where x_{scaled} is the new scaled value, $x_{\min}$ is the minimum value of the feature, $x_{\max}$ is the maximum value of the feature, and x is the original value. This approach helps minimize the influence of varying scales and measurements among different numerical features. For example, the 'data usage' feature of my NPS dataset has a range of 100 to 90,000, while the 'upload' feature has a range of 1 to 1,250. By using the Min-Max scaler, I transform these features into the same range of 0 to 1. In STEP 3, to manage categorical features, I employ the one-hot encoding technique to encode categorical data into numerical ones that enables ML models to understand. For instance, the 'tariff' feature has three unique categorical values, i.e., Level 1, Level 2, and Level 3. By performing the one-hot encoding technique on this feature, I convert the values in the categorical variable into a numeric form (i.e., 1, 2, and 3, respectively) that can be read by the model while maintaining its ordinal nature.

3.3 Latent Features Augmentation (LFA)

After the data is cleaned, they are passed into the LFA module that augments the existing features with the latent features learned from the autoencoder. This module aims to increase the data variation at the sample level by increasing the number of input features. Specifically, the core of the LFA module involves training the autoencoder and then utilizing the trained encoder to extract the most important and compact latent features, which are then concatenated with the existing features to obtain a larger number of features. This process is further detailed in Chapter 4.

3.4 Multitasking Generation (MG)

After the features are augmented, they are fed into the MG module, where the STUNT methodology [27] is implemented to facilitate the multitasking generation process. This approach is designed to address the challenges of FSL and aims to enhance the diversity of data at the task level by generating a range of diverse tasks. In each task, a random selection process is conducted with a specified number of samples according to the designated support and query sets. The support set consists of examples that is used for training, while the query set contains examples that is used for testing. For instance, in Task i-th, where $i = 1, ..., m$, in the 1-shot setting with three classes, one sample is randomly selected from each class, so the number of support set is three. Then the number of queries is set at fifteen samples per class to evaluate the performance of the model in Task i-th. This process is replicated in the predetermined number of tasks. In my work, I employ K-medoids rather than K-means for the task generation process, as K-medoids is a more robust method for overcoming the influence of noisy outliers in the dataset. In contrast to the original method, which utilizes K-means for segmentation and produces pseudo labels that resemble existing labels, my work employs K-medoids to improve the accuracy and reliability of the results. This process is thoroughly explained in Chapter 5.

3.5 Prototypical Networks (PN)

Finally, the tasks corresponding to the selected data and features are passed into my meta-learning paradigm of the PN module, which effectively generalize from minimal examples by shaping a metric space conducive to distance-based classification which is explained in detail in Chapter 6. This holistic framework not only addresses the complexities inherent in FSL but also sets a new benchmark for processing tabular datasets more efficiently and accurately.

Chapter 4

Latent Features Learning and Augmentation

4.1 Introduction to Autoencoders

Autoencoders are employed in high-dimensional data for feature extraction to generate compact representations that accurately reflect the original data [60]. This technique is particularly advantageous for image and video data, as it minimizes storage requirements. For tabular data, autoencoders extract critical features that aim to replicate the original dataset's characteristics fully. The training of autoencoders shown in Figure 4.1 utilizes a substantial portion of data to capture prevalent attributes, yielding the encoder, depicted by green layers that consist of three hidden layers (i.e., h_1, h_2, and h_3) to map the input features x to a latent representation z that extracts the significant features. The decoder, shown in the blue layers that consist of three symmetrical hidden layers with the encoder's hidden layers (i.e., $h_1^{'}$, $h_2^{'}$, and $h_3^{'}$), reconstructs the input features $\hat{x}$ from z, aiming to minimize the difference between x and $\hat{x}$. The trained encoder can then transform the new input data into the latent representations z that is useful for the downstream tasks, including data augmentation and classification, respectively.

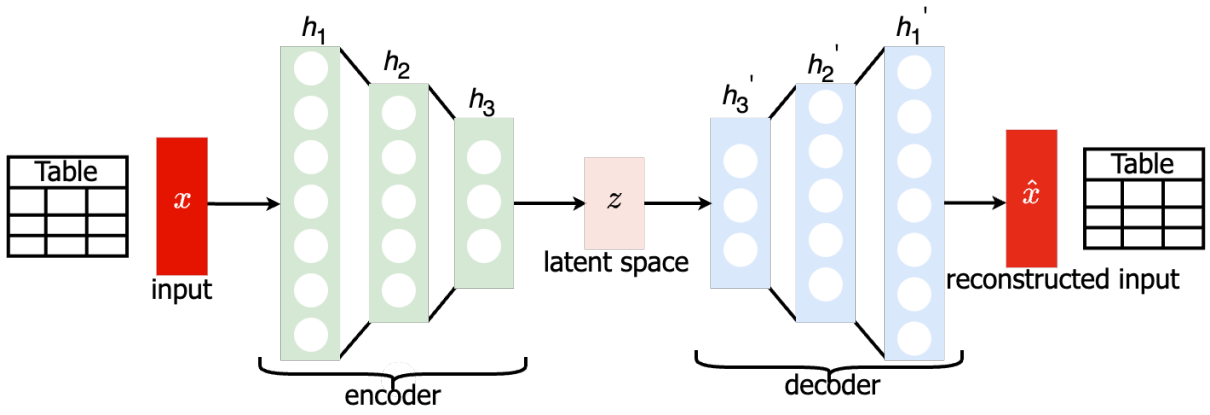


Figure 4.1: A high-level architecture of autoencoders adapted from [60].

4.2 Autoencoders Architecture

4.2.1 Encoder and Decoder

Mathematically, the encoder E and decoder D can be formulated in the following transformations:

$$\textbf{Encoder}: \begin{cases} h_1 = \sigma(W_1^{(E)} x + b_1^{(E)}) \\ h_2 = \sigma(W_2^{(E)} h_1 + b_2^{(E)}) \\ h_3 = \sigma(W_3^{(E)} h_2 + b_3^{(E)}) \\ z = \sigma(W_4^{(E)} h_3 + b_4^{(E)}) \end{cases} \tag{4.1}$$

$$\textbf{Decoder}: \begin{cases} h_3' = \sigma(W_4^{(D)} z + b_4^{(D)}) \\ h_2' = \sigma(W_3^{(D)} h_3' + b_3^{(D)}) \\ h_1' = \sigma(W_2^{(D)} h_2' + b_2^{(D)}) \\ \hat{x} = \sigma(W_1^{(D)} h_1' + b_1^{(D)}) \end{cases}, \tag{4.2}$$

where x is a set of input features, z is a set of latent features, $\hat{x}$ is a set of reconstructed input features, W_i is a weight matrix, b_i is a bias, h_j is a hidden layer, and σ is the ReLU activation function shown in Equation (4.3), for $i = 1, 2, 3, 4$ and $j = 1, 2, 3$.

$$\sigma = \text{ReLU}(x) = \max(0, x) \tag{4.3}$$

4.2.2 Optimization

In this work, I develop a three-layer symmetric autoencoder architecture with the ReLU activation functions to regularize the process. During the learning process, I utilize the mean squared error (MSE) loss criterion to optimize the architecture by using the Adam optimizer with a learning rate set at 10^{-3}. To prevent overfitting, the early stopping is implemented with a patience parameter of 5 that not only ensures the optimal model performance but also reduces the likelihood of training divergence. The optimization of autoencoder quantified using MSE between the original inputs x and the reconstructed outputs $\hat{x}$ can be defined as follows:

$$MSE = \mathcal{L}(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^{N} \|x_i - \hat{x}_i\|^2 \tag{4.4}$$

This loss function *MSE* guides the training process that encourages the model to find the most representative latent features, where N is the total number of data instances in the training set. The Adam optimizer, which adaptively adjusts the learning rate for each parameter based on the estimations of first and second moments of the gradients and the learning rate $\eta = 10^{-3}$, can be defined as follows:

$$\mathbf{W}^{(E)}, \mathbf{b}^{(E)}, \mathbf{W}^{(D)}, \mathbf{b}^{(D)} \leftarrow \text{Adam}(\nabla \mathcal{L}, \eta) \tag{4.5}$$

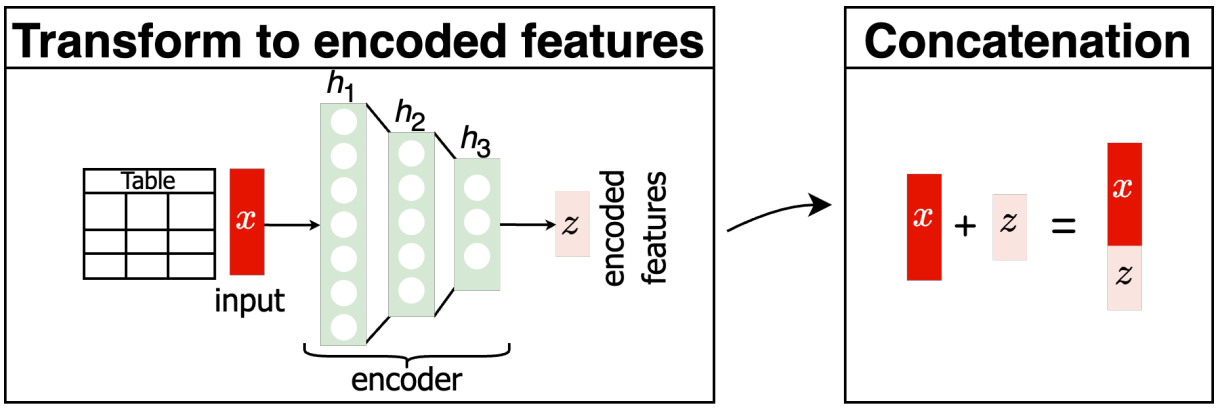


Figure 4.2: Process of augmentation latent features.

4.3 Feature Extraction and Augmentation

Once the encoder is trained, it transforms the original data x to the latent features z shown in Figure 4.2. After that, I perform the min-max normalization using Equation 3.1 to ensure the encoded features has the same scale with the original features. These scaled features are then integrated with the original dataset to generate the augmented data, i.e., $x_{\text{augmented}}$, shown in Equation (4.6), which enhances the overall feature set and increases the data variability. This augmentation is expected to improve the model's generalization capabilities.

$$x_{\text{augmented}} = [x : z] \tag{4.6}$$

4.4 Example of Augmentation Process

To illustrate this process, for example, I consider a dataset that has three different instances and 11 features shown as follows:

$$x = \begin{bmatrix} 0.50 & 0.20 & \dots & 0.20 \\ 0.40 & 0.10 & \dots & 0.10 \\ 0.30 & 0.60 & \dots & 0.30 \end{bmatrix} \in \mathbb{R}^{3\times 11}$$

Training the autoencoder on this dataset x reduces the number of features from 11 to two.

These two encoded features are then used in my augmentation process, which results in generating a total number of 13 features used in the MG process. The detailed procedure is described in the following steps:

Step 1: Encoding. The encoder network is defined in Equation (4.1), where

$$W_1^{(E)} \in \mathbb{R}^{7\times 11}, \quad b_1^{(E)} \in \mathbb{R}^7$$
$$W_2^{(E)} \in \mathbb{R}^{5\times 7}, \quad b_2^{(E)} \in \mathbb{R}^5$$
$$W_3^{(E)} \in \mathbb{R}^{3\times 5}, \quad b_3^{(E)} \in \mathbb{R}^3$$
$$W_4^{(E)} \in \mathbb{R}^{2\times 3}, \quad b_4^{(E)} \in \mathbb{R}^2$$

Using the learned weights and biases obtained from Equation (4.4) after 100 epochs, the computations proceed as follows:

$$h_1 = \sigma(W_1^{(E)} x + b_1^{(E)})$$
$$h_2 = \sigma(W_2^{(E)} h_1 + b_2^{(E)})$$
$$h_3 = \sigma(W_3^{(E)} h_2 + b_3^{(E)})$$
$$z = \sigma(W_4^{(E)} h_3 + b_4^{(E)})$$

For example, if the learned weights and biases are obtained as follows:

$$W_1^{(E)} = \begin{bmatrix} 0.1 & 0.2 & \dots & 0.1 \\ \vdots & \vdots & \ddots & \vdots \\ 0.1 & 0.2 & \dots & 0.1 \end{bmatrix}, \quad b_1^{(E)} = \begin{bmatrix} 0.1 \\ \vdots \\ 0.1 \end{bmatrix}$$

$$W_2^{(E)} = \begin{bmatrix} 0.2 & 0.1 & \dots & 0.1 \\ \vdots & \vdots & \ddots & \vdots \\ 0.2 & 0.1 & \dots & 0.1 \end{bmatrix}, \quad b_2^{(E)} = \begin{bmatrix} 0.1 \\ \vdots \\ 0.1 \end{bmatrix}$$

$$\vdots$$

$$W_4^{(E)} = \begin{bmatrix} 0.1 & 0.2 & \dots & 0.1 \\ 0.1 & 0.2 & \dots & 0.1 \end{bmatrix}, \quad b_4^{(E)} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

, I calculate:

$$\begin{aligned} h_1 &= \sigma(W_1^{(E)} x + b_1^{(E)}) \\ h_2 &= \sigma(W_2^{(E)} h_1 + b_2^{(E)}) \\ h_3 &= \sigma(W_3^{(E)} h_2 + b_3^{(E)}) \\ z &= \sigma(W_4^{(E)} h_3 + b_4^{(E)}) \end{aligned}$$

The encoded feature matrix is obtained as:

$$z = \sigma\left(\begin{bmatrix} 0.78 & 1.00 \\ 1.08 & 1.42 \\ 1.23 & 1.58 \end{bmatrix}\right)$$

The ReLU activation function from Equation (4.3) keeps all values the same, so

$$z = \begin{bmatrix} 0.78 & 1.00 \\ 1.08 & 1.42 \\ 1.23 & 1.58 \end{bmatrix}$$

Step 2: Min-Max Scaling of Latent Features. To ensure the latent features that are on the same scale as the original ones, I apply a Min-Max scaler from Equation (3.1) to the latent representation z before the concatenation and then compute the scaled vector z^{scaled} as follows:

$$z^{scaled} = \begin{bmatrix} 0.00 & 0.00 \\ 0.67 & 0.72 \\ 1.00 & 1.00 \end{bmatrix}$$

Step 3: Concatenation with Original Input Features. After scaling by Equation 4.6, the augmented dataset $x_{\text{augmented}}$ is formed by concatenating the original input x with the scaled latent features z^{scaled}:

$$x_{\substack{\text{augmented} \\ 3\times13}} = \begin{bmatrix} 0.50 & 0.20 & \dots & 0.20 & 0.00 & 0.00 \\ 0.40 & 0.10 & \dots & 0.10 & 0.67 & 0.72 \\ 0.30 & 0.60 & \dots & 0.30 & 1.00 & 1.00 \end{bmatrix} \tag{4.7}$$

Then $x_{\text{augmented}}$ here is used in the MG process in the next module.

Chapter 5

STUNT Multitasking Generation

5.1 Introduction to Task-Level Data Augmentation

Data augmentation at the task level is to build the common knowledge by performing multiple tasks from a dataset. STUNT is a specific framework that can generate those multiple diverse tasks using the K-means clustering as a pseudo-label generator [27]. The idea behind this approach is that each feature can serve as a label for the other features. For instance, Figure 5.1 is a original dataset with three input features (x) (i.e., complaints, data usage, and age) and a target binary variable (y) (i.e., cancellation (yes/no)). In this example, I assume that there is a positive correlation between complaints and cancellation, from which I can use complaints as a new target variable and use the other variables as the input features. By generalizing this concept, I can first consider the data that is unlabeled; and this STUNT method randomly selects some features and then utilizes the K-means algorithm to perform the clustering. In each cluster, the pseudo-label can be obtained by computing the center of the cluster aka the centroids. By iteratively the above task generation process using different combinations of features, I can generate the corresponding new datasets with their own labels from the original dataset.

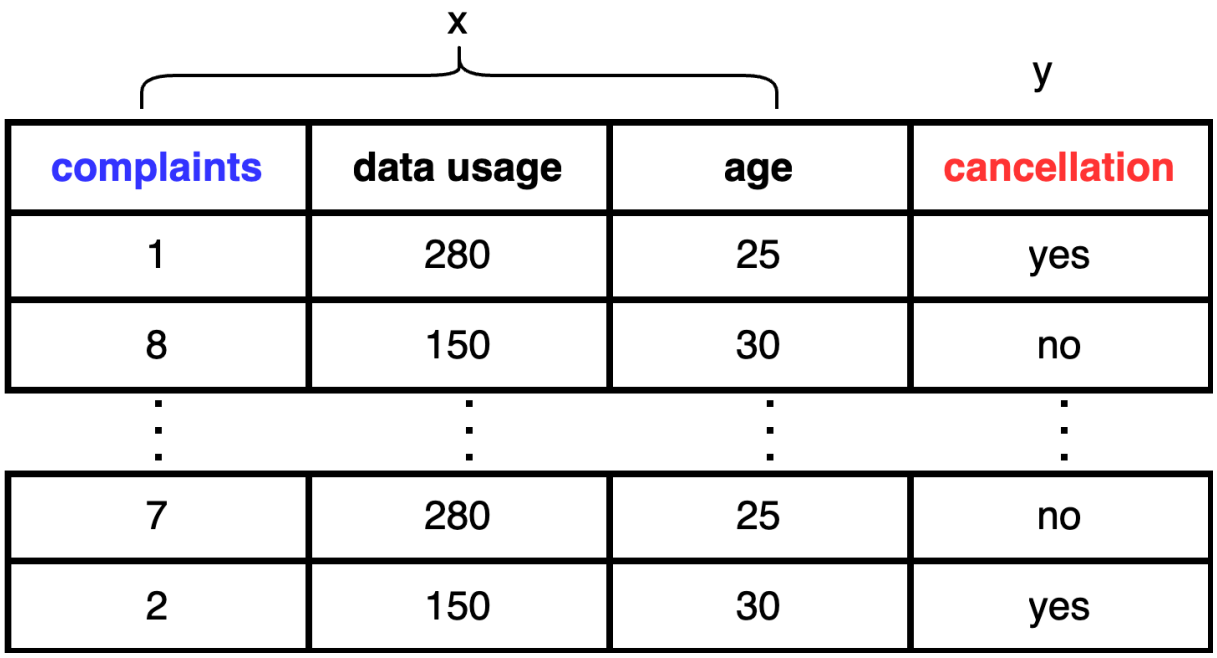


Figure 5.1: Example data from telecom dataset, adapted from [27].

5.2 STUNT Framework

More specifically, following [27], given a dataset $\mathbf{X}$ of unlabeled tabular data, I summarize their approach and formalize the process as follows:

Step 1: Masking Ratio Sampling. Sample a masking ratio p from a uniform distribution over a range of hyperparameter $[r_1, r_2]$, where $0 < r_1 < r_2 < 1$.

Step 2: Binary Mask Creation. Generate a random binary mask $\mathbf{m} \in \{0, 1\}^d$, where d is the number of features and the sum of elements in $\mathbf{m}$ is $\lfloor dp \rfloor$ where $\lfloor \cdot \rfloor$ is floor function applied to dp.

Step 3: Column Selection. Use the mask $\mathbf{m}$ to select columns from the unlabeled data $\mathbf{X}$. The selected data is denoted by $sq(\mathbf{x} \circ \mathbf{m})$, where $\circ$ indicates element-wise multiplication, and $sq(\cdot)$ represents a squeezing operation that removes elements corresponding to

zeros in $\mathbf{m}$.

Step 4: K-means Clustering. Apply K-means clustering on the selected columns to generate pseudo-labels $\tilde{\mathbf{y}}$. The objective function for the K-means is given by:

$$\min_{C \in \mathbb{R}^{\lfloor dp \rfloor \times k}} \frac{1}{N} \sum_{i=1}^{N} \min_{\tilde{\mathbf{y}}_i \in \{0,1\}^k} \|sq(\mathbf{x}_i \circ \mathbf{m}) - C\tilde{\mathbf{y}}_i\|_2^2 \tag{5.1}$$

$$\text{such that} \quad \tilde{\mathbf{y}}_i^T \mathbf{1}_k = 1$$

, where C is the centroid matrix, k is the number of centroids, $\mathbf{1}_k$ is a vector of ones and $\mathbf{x}_i$ represents the i-th sample in the dataset. $\|\cdot\|_2^2$ indicates squared Euclidean distance, used here to measure the distance between the transformed data points and the cluster centroids.

Step 5: Data Perturbation. To prevent trivial learning by the classifier, perturb the selected column features by:

$$\tilde{\mathbf{x}} := \mathbf{m} \circ \hat{\mathbf{x}} + (1 - \mathbf{m}) \circ \mathbf{x} \tag{5.2}$$

, where $\hat{\mathbf{x}}$ is sampled from the empirical marginal distribution of each column feature.

Step 6: Task Definition. The generated task T_{STUNT} from the process is defined as:

$$\mathcal{T}_{\text{STUNT}} := \{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)\}_{i=1}^{N} \tag{5.3}$$

5.3 Enhancements with *K*-Medoids

In this work, I enhance and customize the STUNT approach by using the K-medoids as an alternative clustering method to the K-means shown in Figure 5.2 that is a high-level overview of the modified STUNT approach. I propose this approach because the K-medoids clustering is more robust to outliers, as it uses the actual data points as the centers aka

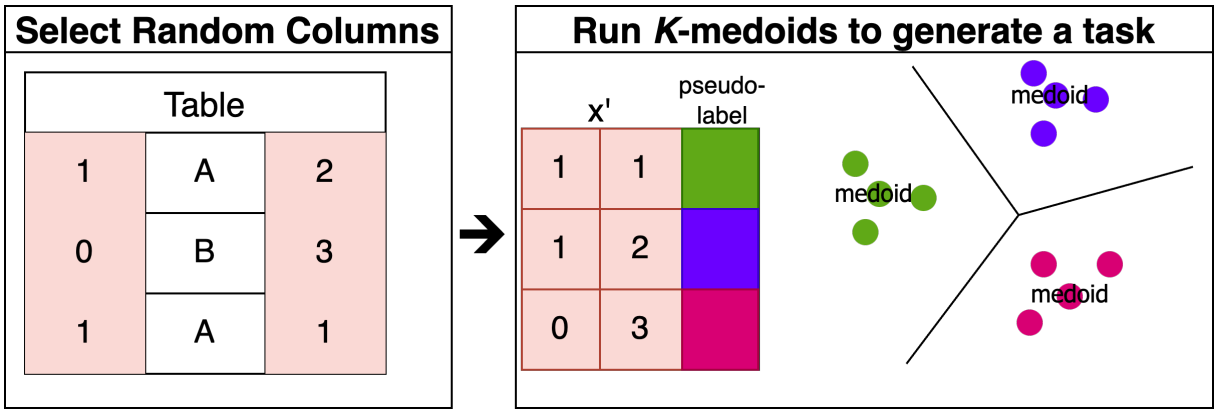


Figure 5.2: Multitasking generation using K-medoids, adapted from [27].

medoids, thereby avoiding the influence of extreme values, unlike K-means. Thus, in Equation (5.1) above, I change to use the K-medoids clustering instead of the K-means clustering.

More precisely, I apply the K-medoids clustering on the selected features to generate the pseudo-label $\tilde{\mathbf{y}}$. The objective function for the K-medoids clustering using the Manhattan distance is given by:

$$\min_{C \in \mathbb{R}^{\lfloor dp \rfloor \times k}} \frac{1}{N} \sum_{i=1}^{N} \min_{\tilde{\mathbf{y}}_i \in \{0,1\}^k} \|sq(\mathbf{x}_i \circ \mathbf{m}) - C\tilde{\mathbf{y}}_i\|_1 \tag{5.4}$$

$$\text{such that} \quad \tilde{\mathbf{y}}_i^T \mathbf{1}_k = 1,$$

, where C is the centroid matrix which is the medoids matrix, k is the number of medoids, and $\mathbf{x}_i$ represents the i-th sample in the dataset. $\|\cdot\|_1$ represents the Manhattan distance to measure the distance between the transformed data points and the cluster medoids. The medoids are selected from the dataset $\mathbf{X}$, and each data point $\mathbf{x}_i$ is assigned to the nearest medoid based on the Manhattan distance.

5.4 Example of Task Generation

To provide a clear implementation of this module, I take the matrix obtained by Equation (4.7), as an example, to generate a task $\mathcal{T}_{\text{STUNT}_1}$. Given a matrix $x_{\text{augmented}}$:

Step 1: Masking Ratio Sampling. For instance, if $r_1 = 0.2$ and $r_2 = 0.5$, then $p = 0.3$

Step 2: Binary Mask Creation. Given

$$d = 13, \quad \lfloor dp \rfloor = \lfloor 13 \times 0.3 \rfloor = 3$$

, the example mask is

$$m = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 3: Column Selection.

$$sq(\mathbf{x} \circ \mathbf{m}) = \begin{bmatrix} 0.50 & 0.20 & 0.00 \\ 0.40 & 0.10 & 0.72 \\ 0.30 & 0.60 & 1.00 \end{bmatrix}$$

Step 4: K-medoids clustering. The objective function for K-medoids clustering using Manhattan distance is given by Equation (5.4). Assume $k = 2$, I obtain:

$$\text{pseudo-label} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}; \text{ medoids} = \begin{bmatrix} 0.50 & 0.20 & 0.00 \\ 0.40 & 0.10 & 0.72 \end{bmatrix}$$

Step 5: Data Perturbation. Perturb the selected column features by Equation (5.2). The red color means the result of perturbation.

$$\tilde{\mathbf{x}} = \begin{bmatrix} 0.40 & 0.60 & \dots & 0.20 & 0.00 & 0.72 \\ 0.50 & 0.20 & \dots & 0.10 & 0.67 & 1.00 \\ 0.30 & 0.10 & \dots & 0.30 & 1.00 & 0.00 \end{bmatrix}$$

Step 6: Task Definition. The generated task T_{STUNT} from the process is defined as in Equation (5.3):

$$\mathcal{T}_{\text{STUNT}_1} = \left\{ \left(\tilde{\mathbf{x}}_i, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right) \right\}_{i=1}^{3} \tag{5.5}$$

Then the diverse tasks generation from this module is defined as follows:

$$\{\mathcal{T}_{\text{STUNT}_j}\}_{j=1}^{m} \tag{5.6}$$

, where m is a hyperparameter which is the number of tasks. After the completion of diverse tasks generation, the learning process is undertaken by ProtoNet, which aims to develop a model capable of generalizing based on diverse inputs from various tasks.

Chapter 6

Prototypical Networks

6.1 Introduction to Prototypical Networks

Prototypical Networks (ProtoNet) is a neural network that employs meta-learning to learn variety of tasks. Specifically, after the MG module generates the diverse tasks by Equation 5.6, data samples are taken from a collection of those tasks. For each task, the support ($\mathcal{S}$) set and the query ($\mathcal{Q}$) set are then selected. As shown in Figure 6.1, i.e., a high-level framework of few-shot learning concept using ProtoNet, the model is trained on the support set and evaluated on the query set, with the meta-learner being updated based on the query set's performance. Following this, the meta-learner is utilized for adaptation and prediction on a new test set using a fresh batch of labeled data, with a small portion serving as the support test set.

6.2 Advantages of Prototypical Networks

Several advantages have been identified by [27] regarding the use of ProtoNet as an embedding function or learner in few-shot settings, including flexible centroids, agnostic application, and optimal performance. **Flexible centroids** refer to the adaptability of ProtoNet to various cases by adjusting the number of k or centroids. **Agnostic application** allows for the direct

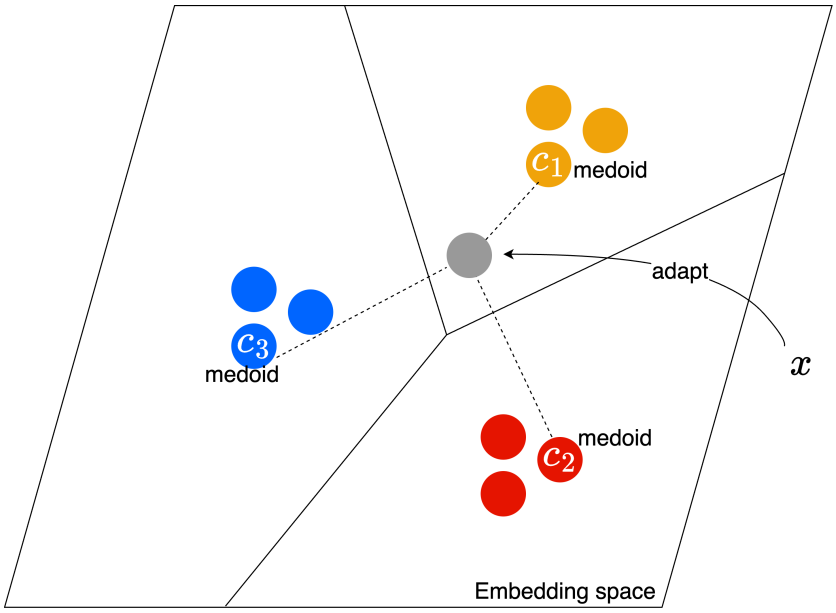


Figure 6.1: Few-shot learning concept using ProtoNet adapted from [30].

application of this architecture to tabular data without significant difficulty. Additionally, ProtoNet has demonstrated **strong performance** in various modalities, as reported in some studies [27], [30], [32]. This study also highlights the flexibility of ProtoNet as a benefit. The original ProtoNet employs the Euclidean distance for metric learning, but the research work conducted by [32] on image classification suggests that the Manhattan distance is a strong substitute for this metric, potentially improving performance. It would therefore be intriguing to apply this substitution to tabular data, as the Manhattan distance has advantages over the Euclidean one, especially in a high-dimensional data. The differences between the Euclidean and the Manhattan distance are visually shown in Figure 6.2. The Euclidean distance (i.e., the red line) measures the shortest straight-line distance between the two points that is calculated by using the Pythagorean theorem. The Manhattan distance (i.e., the blue path) measures the distance between the two points by summing the absolute differences of their coordinates.

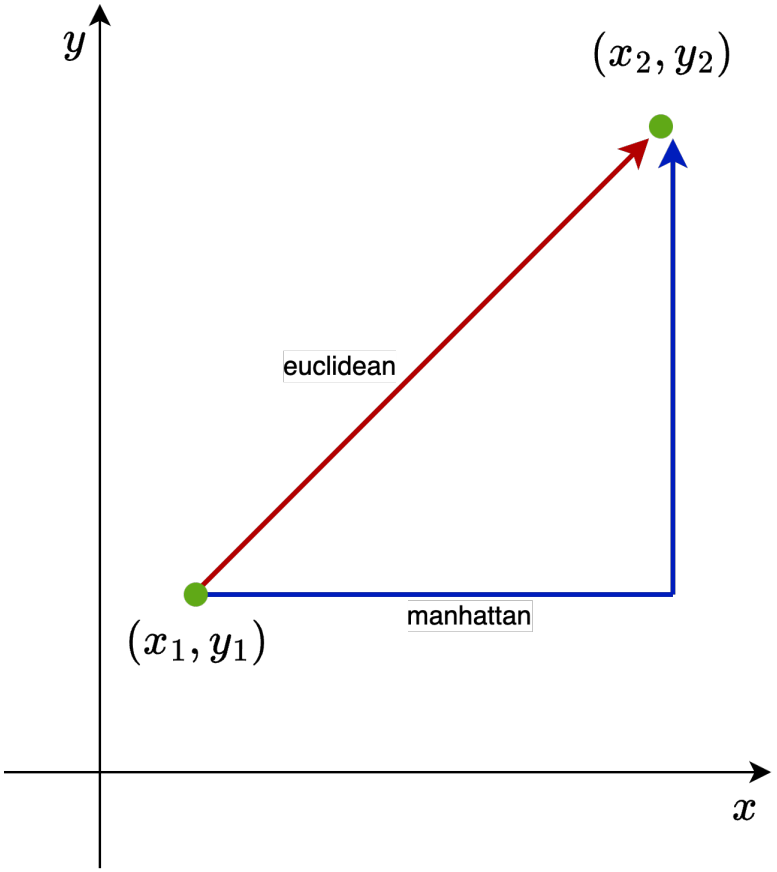


Figure 6.2: Comparison between Euclidean distance and Manhattan distance.

6.3 ProtoNet Configuration

In this work, the architecture of ProtoNet follows a multilayer perceptron (MLP) design that consists of a 2-layer fully connected neural network with a hidden dimension of 1,024, as recommended by [27]. Given a task selected by Equation 5.5, I construct the classifier using the episodic training way. Training episodes are created by selecting random subsets of classes and examples, with some examples acting as ($\mathcal{S}$) and ($\mathcal{Q}$) from each task. The Prototypical networks create a prototype or an average representation for each class using an embedding function $\mathcal{F}_\theta$ with a learnable parameter θ. Each prototype is the mean of the embedded points in its class calculated as follow:

$$c_k = \frac{1}{|\mathcal{S}_k|} \sum_{(\tilde{\mathbf{x}}_i, \tilde{y}_i) \in \mathcal{S}_k} \mathcal{F}_\theta(\tilde{\mathbf{x}}_i) \tag{6.1}$$

, where $\mathcal{S}_k$ is the support set associated with the prototype k. Using a distance function d, which is the Manhattan distance, the network calculates the probability of a class for a query point $\tilde{x_i}$ by taking a softmax over distances to the prototypes:

$$p_\theta(y = k \mid \tilde{\mathbf{x}}_i; \mathcal{S}) = \frac{\exp(-d(\mathcal{F}_\theta(\tilde{\mathbf{x}}_i), c_k))}{\sum_{k'} \exp(-d(\mathcal{F}_\theta(\tilde{\mathbf{x}}_i), c_{k'}))} \tag{6.2}$$

, where the Manhattan distance is

$$d(\mathcal{F}_\theta(\tilde{\mathbf{x}}_i), c_k) = \sum_{i=1}^{N_{\mathcal{S}_k}} \|\mathcal{F}_\theta(\tilde{\mathbf{x}}_i) - c_k)\|_1 \tag{6.3}$$

Next, I compute the cross-entropy loss on the classifier p_θ as follows:

$$\mathcal{L}_{CE}(p_\theta, \tilde{y_i}) = -\sum_{j=1} (\tilde{y_i})_j \log p_\theta \tag{6.4}$$

The ultimate objective is to minimize the meta-learning loss over diverse tasks generated by Equation (5.6) as follow:

$$\mathcal{L}_{meta}(\theta, \mathcal{Q}) := \sum_{(x_i, y_i) \in \mathcal{Q}} \mathcal{L}_{CE}(p_\theta, \tilde{y_i}) \tag{6.5}$$

6.4 Example of Prototypical Network Process

To provide a better understanding, I illustrate the whole process using the below example. Given the diverse sets of tasks generated by Equation (5.6), for each task, I sample two independent sets of ($\mathcal{S}$) and ($\mathcal{Q}$). For the support set ($\mathcal{S}$), I set $n_{shot} = 1$ per class for 1-shot and set $n_{shot} = 5$ per class for 5-shot. For the query set ($\mathcal{Q}$), I set $n_{query} = 15$ per class.

For instance in 1-shot setting, for simplicity, given $n_{shot} = 1$, $n_{query} = 2$, the number of classes $k = 3$, and $n_{features} = 4$, the dimension is shown as follow:

$$\mathcal{T}_{\text{STUNT}_1} = \begin{cases} \mathcal{S}_1 \in \mathbb{R}^{3\times 4} \\ \mathcal{Q}_1 \in \mathbb{R}^{6\times 4} \end{cases}$$

, where **Support Set** ($\mathcal{S}_1$):

$$\begin{aligned} \text{Class 1:} \quad \tilde{\mathbf{x}}_{11} &\rightarrow \begin{bmatrix} 1.0 & 2.0 & 1.5 & 2.5 \end{bmatrix} \\ \text{Class 2:} \quad \tilde{\mathbf{x}}_{21} &\rightarrow \begin{bmatrix} 2.0 & 1.5 & 2.5 & 1.0 \end{bmatrix} \\ \text{Class 3:} \quad \tilde{\mathbf{x}}_{31} &\rightarrow \begin{bmatrix} 1.5 & 1.0 & 2.0 & 2.0 \end{bmatrix} \end{aligned}$$

and **Query Set** ($\mathcal{Q}_1$):

$$\begin{aligned} \text{Class 1:} \quad \tilde{\mathbf{x}}_{12} &\rightarrow \begin{bmatrix} 1.2 & 2.1 & 1.4 & 2.6 \end{bmatrix} \\ \text{Class 1:} \quad \tilde{\mathbf{x}}_{13} &\rightarrow \begin{bmatrix} 1.1 & 2.0 & 1.6 & 2.4 \end{bmatrix} \\ \text{Class 2:} \quad \tilde{\mathbf{x}}_{22} &\rightarrow \begin{bmatrix} 2.1 & 1.4 & 2.6 & 1.1 \end{bmatrix} \\ \text{Class 2:} \quad \tilde{\mathbf{x}}_{23} &\rightarrow \begin{bmatrix} 2.2 & 1.6 & 2.4 & 1.2 \end{bmatrix} \\ \text{Class 3:} \quad \tilde{\mathbf{x}}_{32} &\rightarrow \begin{bmatrix} 1.6 & 1.1 & 2.1 & 1.9 \end{bmatrix} \\ \text{Class 3:} \quad \tilde{\mathbf{x}}_{33} &\rightarrow \begin{bmatrix} 1.4 & 1.2 & 1.9 & 2.1 \end{bmatrix} \end{aligned}$$

Then, I use Equation (6.1) to calculate each prototype c_k for $k = 1, 2, 3$:

$$c_1 = \mathcal{F}_\theta(\tilde{\mathbf{x}}_{11}) = [1.0, 2.0, 1.5, 2.5]$$

$$c_2 = \mathcal{F}_\theta(\tilde{\mathbf{x}}_{21}) = [2.0, 1.5, 2.5, 1.0]$$

$$c_3 = \mathcal{F}_\theta(\tilde{\mathbf{x}}_{31}) = [1.5, 1.0, 2.0, 2.0]$$

Since $n_{shot} = 1$, the prototype for each class is the support example itself. Next, I compute the distance of query examples to the prototypes using Equation (6.3):

For query example $\tilde{\mathbf{x}}_{12} = [1.2, 2.1, 1.4, 2.6]$:

$$\begin{aligned} d(\mathcal{F}_\theta(\tilde{\mathbf{x}}_{12}), c_1) &= |1.2 - 1.0| + |2.1 - 2.0| + \\ &\quad |1.4 - 1.5| + |2.6 - 2.5| \\ &= 0.2 + 0.1 + 0.1 + 0.1 \\ &= 0.5 \end{aligned}$$

$$\begin{aligned} d(\mathcal{F}_\theta(\tilde{\mathbf{x}}_{12}), c_2) &= |1.2 - 2.0| + |2.1 - 1.5| + \\ &\quad |1.4 - 2.5| + |2.6 - 1.0| \\ &= 0.8 + 0.6 + 1.1 + 1.6 \\ &= 4.1 \end{aligned}$$

$$\begin{aligned} d(\mathcal{F}_\theta(\tilde{\mathbf{x}}_{12}), c_3) &= |1.2 - 1.5| + |2.1 - 1.0| + \\ &\quad |1.4 - 2.0| + |2.6 - 2.0| \\ &= 0.3 + 1.1 + 0.6 + 0.6 \\ &= 2.6 \end{aligned}$$

The following process uses Equation (6.2) to calculate the softmax probability. For $\tilde{\mathbf{x}}_{12}$:

$$p_\theta(y = 1 \mid \tilde{\mathbf{x}}_{12}; \mathcal{S}_1) = \frac{e^{-0.5}}{e^{-0.5} + e^{-4.1} + e^{-2.6}}$$

$$p_\theta(y = 2 \mid \tilde{\mathbf{x}}_{12}; \mathcal{S}_1) = \frac{e^{-4.1}}{e^{-0.5} + e^{-4.1} + e^{-2.6}}$$

$$p_\theta(y = 3 \mid \tilde{\mathbf{x}}_{12}; \mathcal{S}_1) = \frac{e^{-2.6}}{e^{-0.5} + e^{-4.1} + e^{-2.6}}$$

by normalizing these:

$$p_\theta(y = 1 \mid \tilde{\mathbf{x}}_{12}; \mathcal{S}_1) = \frac{0.6065}{0.6065 + 0.0166 + 0.0743} \approx 0.86$$

$$p_\theta(y = 2 \mid \tilde{\mathbf{x}}_{12}; \mathcal{S}_1) = \frac{0.0166}{0.6065 + 0.0166 + 0.0743} \approx 0.02$$

$$p_\theta(y = 3 \mid \tilde{\mathbf{x}}_{12}; \mathcal{S}_1) = \frac{0.0743}{0.6065 + 0.0166 + 0.0743} \approx 0.12$$

Next, I compute the cross-entropy loss by using Equation (6.4). Assume the true label for $\tilde{\mathbf{x}}_{12}$ is class 1.

$$\begin{aligned}
\mathcal{L}_{CE}(p_\theta, \tilde{y}_{12}) &= -\log(p_\theta(y = 1 \mid \tilde{\mathbf{x}}_{12}; \mathcal{S})) \\
&\approx -\log(0.86) \\
&\approx 0.15
\end{aligned}$$

After that, I repeat the same steps for all query examples in $\mathcal{Q}_1$ and sum their cross-entropy losses to get $\mathcal{L}_{meta}$ using Equation (6.5).

After I finish the model training, I use the model obtained to adapt with the few-shot sample (x_i, y_i), where y_i is the existing label from 100 different seed. Finally, using an independent test set, I compute the mean test accuracy of this few-shot learning process.

Chapter 7

Experimental Results, Analyses, and Discussion

7.1 Datasets

In this experimental studies, I utilize four different domain datasets. Three of them are publicly available from the UCI Machine Learning Repository, including Wine [61], Dry Bean [62], and Forest Cover Type [63]. One of them is a proprietary dataset provided by a telecommunications corporation, specifically related to the NPS segmentation. The Wine dataset contains 178 instances and 13 attributes that are used for the classification of wine variants. The Dry Bean dataset includes 13,611 instances and 16 attributes that are aimed at classifying different types of beans. The Forest Cover Type dataset is composed of 581,012 instances and 54 attributes that are used for predicting forest cover types based on cartographic variables. The proprietary NPS segmentation dataset consists of customer demographic profile and feedback data, segmented into promoters, passives, and detractors based on their likelihood to recommend the company's services. Table 7.1 provides the detailed descriptions of these four datasets, including the number of instances and attributes, as well as the primary classification objective for each dataset.

Table 7.1: Summary of Datasets

No	Name	# N	# features	Description	Source
1	Net Promoter Score (NPS) segmentation	100,000	11	Predict a customer segmentation into three groups: promoters, passives, detractors, based on demographics and customer experiences.	Private
2	Drybean types	13,611	16	Predict seven various sorts of dry beans according to market conditions, including form, shape, type, and structure.	Public
3	Wine types	178	13	Predict three different types of wines using the findings of a chemical analysis of wines grown in the same region of Italy.	Public
4	Forest cover types	581,102	54	Predict seven forest cover classes based on variables such as elevation, aspect, slope, hill shade, soil type, and others.	Public

7.2 Baselines

During this experimental evaluations, I examine various TE models as the benchmark, including Random Forest, CatBoost, and One-vs-Rest (OvR) Classifier. I also combine these three baseline models with augmentation techniques utilizing autoencoders to enhance the feature representation and improve classification performance. Random Forest, known for its robustness and ease of implementation, provides a strong baseline through its ensemble of decision trees. CatBoost, a gradient boosting algorithm, is particularly effective in handling categorical features and improving accuracy. The OvR Classifier, a strategy for multiclass classification, breaks down the problem into multiple binary classification tasks. In addition to these models, I employ the standard STUNT framework as a comparison benchmark for my proposed method. The STUNT framework, known for its comprehensive approach to generate multiple tasks on tabular data setting, served as a rigorous benchmark to evaluate the efficacy of my proposed enhancements. Table 7.2 shows a more detailed and extensive explanation of these baseline methods.

Table 7.2: Baselines Details

No	Methods	Description
1	Random Forests (RF)	An ensemble of tree predictors, where each tree's predictions are based on the values of a random vector that is separately sampled and has the same distribution for all trees in the forest.
2	CatBoost (CB)	A gradient boosting method that utilizes binary decision trees as its base predictors.
3	Autoencoders (AE) + Classifier	Using only encoded features to be trained into classifier (RF or CB)
4	Concatenation Autoencoders (ConcatAE) + Classifier	Using original and encoded features (concatenation) to be trained into classifier (RF or CB)
5	One-vs-the-rest (OvR) multiclass strategy	The one-vs-the-rest (OvR) multiclass strategy, often referred to as one-vs-all, involves training a separate classifier for each class.
6	Self-generated Tasks from unlabeled Tables (STUNT)	A few-shot tabular learning system that utilizes meta-learning to train on self-generated problems derived from unlabeled tables.

7.3 1-shot Learning Result

In the 1-shot learning, I observe varying levels of performance among the baseline models in terms of mean test accuracy. The results in Table 7.3 illustrate several noteworthy trends in the performance of different classification methods across the datasets examined. First, RF classifier generally outperforms CB classifier in all cases by 0.74% in average. Second, OvR strategy specifically on CB, consistently outperforms the baseline models (RF and CB) on most datasets by 0.99% in average. Additionally, ConcatAE approach surpasses both the OvR strategy by 2.7% in average and the base models by 2.8% in average. These findings suggest that employing advanced techniques such as OvR and ConcatAE can significantly enhance classification accuracy in 1-shot learning scenarios. Compared to standard STUNT, my method, which employs ConcatAE in conjunction with K-medoids clustering and Manhattan ProtoNet, achieved the highest mean test accuracy across all datasets and tasks by 4.03% in average, showcasing the superiority of this approach in 1-shot learning classification.

Table 7.3: Mean test accuracy on 1-shot setting.

Methods	NPS	Dry Bean	Wine	Cover Type	Average
RF	32.74	68.41	81.39	24.19	51.68
CB	34.01	64.48	84.17	22.54	51.30
AE + RF	34.41	63.14	85.56	22.41	51.38
AE + CB	33.99	61.77	86.53	22.57	51.22
ConcatAE + RF	32.54	68.47	87.64	23.76	53.10
ConcatAE + CB	33.14	66.15	87.92	23.81	52.76
OvR RF	32.31	69.24	81.25	22.46	51.32
OvR CB	33.42	69.54	81.81	22.47	51.81
AE + OvR RF	33.99	60.49	86.94	21.38	50.70
AE + OvR CB	32.91	63.03	86.81	22.02	51.19
ConcatAE + OvR RF	31.77	68.28	87.36	22.39	52.45
ConcatAE + OvR CB	32.40	70.63	87.36	23.43	53.46
STUNT (k-Means + Euclidean ProtoNet)	35.69	67.44	85.75	24.66	53.39
ConcatAE + STUNT	34.47	70.43	87.64	23.76	54.07
ConcatAE + k-Medoid + Manhattan ProtoNet	**36.06**	**71.17**	**88.86**	**26.07**	**55.54**

7.4 5-shot Learning Result

In the 5-shot learning, the performance patterns observed in Table 7.4 are similar to those seen in 1-shot settings for various datasets in base models and when combined with augmentation techniques. For instances, RF classifier generally still outperforms CB classifier in all cases by 2.04% in average. Then, OvR strategy specifically on CB, consistently outperforms the baseline models (RF and CB) on most datasets by 2.13% in average. In addition, ConcatAE approach surpasses both the OvR strategy by 0.99% and the base models by 1.09%. These findings suggest that employing advanced techniques such as OvR and ConcatAE can significantly enhance classification accuracy in 5-shot learning scenarios. Compared to standard STUNT, my method, which employs ConcatAE in conjunction with K-medoids clustering and Manhattan ProtoNet, achieved the highest mean test accuracy in 3 out of 4 datasets — NPS, Dry Bean, and Wine — by 1.47%, showcasing the optimal performance of this approach in 5-shot learning classification.

I also observe some significant result on comparison between scenarios with augmentation

Table 7.4: Mean test accuracy on 5-shot setting.

Methods	NPS	Dry Bean	Wine	Cover Type	Average
RF	39.56	84.37	92.50	35.73	63.04
CB	38.51	82.68	88.47	37.44	61.78
AE + RF	39.47	80.62	89.58	31.49	60.29
AE + CB	39.96	79.84	90.97	31.78	60.64
ConcatAE + RF	40.52	84.68	92.92	34.91	63.26
ConcatAE + CB	40.06	83.71	89.86	**38.07**	62.93
OvR RF	39.55	84.54	92.92	33.54	62.64
OvR CB	39.88	85.25	91.81	35.44	63.10
AE + OvR RF	38.69	79.85	90.28	30.52	59.84
AE + OvR CB	39.47	81.69	92.64	31.69	61.37
ConcatAE + OvR RF	40.18	84.66	94.03	33.43	63.08
ConcatAE + OvR CB	40.53	85.53	93.15	35.82	63.91
STUNT (k-Means + Euclidean ProtoNet)	40.76	83.48	94.03	34.72	63.25
ConcatAE + STUNT	40.93	84.15	95.00	31.76	62.96
ConcatAE + k-Medoid + Manhattan ProtoNet	**41.25**	**85.62**	**95.28**	34.58	**64.18**

— ConcatAE + k-Medoid + Manhattan ProtoNet (my approach)— and no augmentation — RF, CB, OvR RF, and OvR CB. Figure 7.1 clearly shows that the method with augmentation give improvement both on 1-shot and 5-shot settings. Specifically, my approach outperforms the traditional ensemble (TE) models and the OvR classifiers by 7.8% in the 1-shot setting and 2.5% in the 5-shot setting. This enhancement underscores the effectiveness of my approach in multiclass classification, demonstrating optimal generalization capabilities compared to models without augmentation.

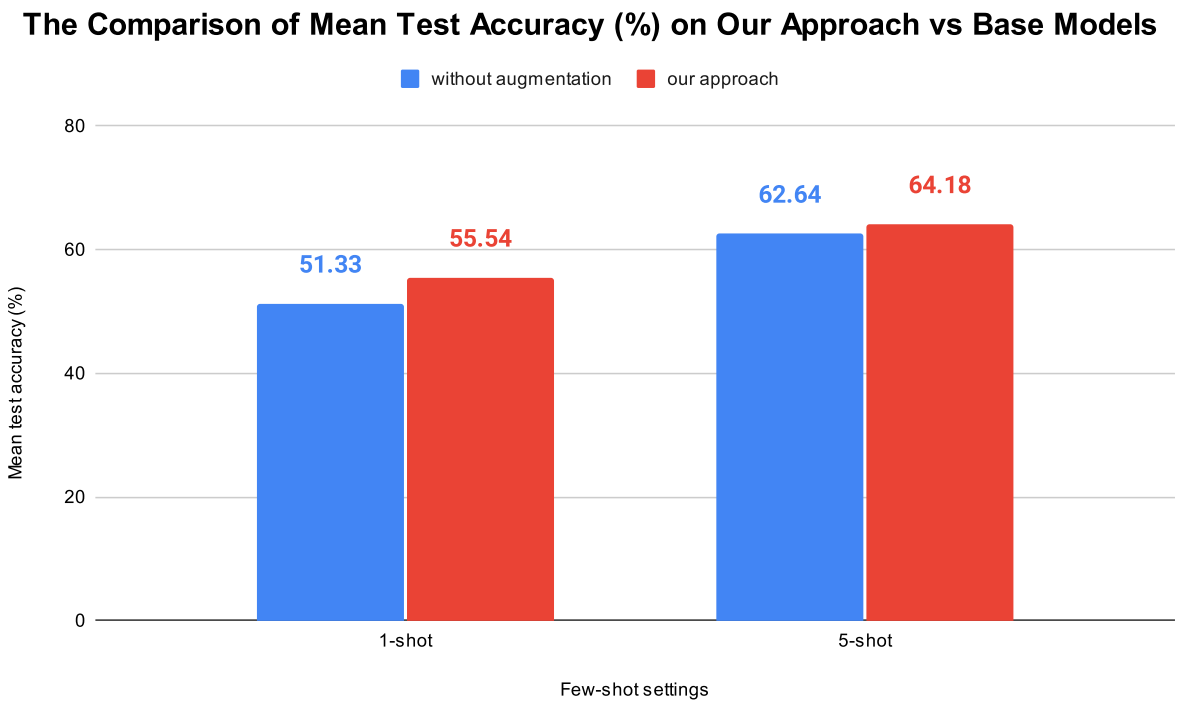


Figure 7.1: The effect of data augmentation techniques compared to base models.

Chapter 8

New Concept Adaptation

8.1 What is a New Concept?

We can think of a new concept as a new task or a new label in the machine learning term. For example, in the 'adult' income dataset, we have binary task to predict whether a person has income more than 50K or less based on his/her demographic and social-economic profile. In this case, the label is 'income level':

$$income = \begin{cases} 1, \texttt{if} \geq \texttt{50K} \\ 0, \texttt{else} \end{cases}$$

and we can say this is the concept that we want to focus on to learn. The concept of income can be learned using the available features, and it is possible to learn another label/new concept from these features. We can take the example from Figure 5.1, which the dataset consists of three features and one target. We can remove the target 'cancellation', then replace it with a new target or new concept that we want to learn. For example, the loyalty level of the customer, where this concept can be generated by manual annotation or from a survey to the existing customer. However, since we do not have this label yet, we generate the label by using K-medoids procedure, by using all features available, and then proceed with the simple

Decision Tree (DT) classifier to name the label based on how trees divide the features. To support this naming concept result, we obtain justification from a subject matter expert to ensure that the concept we obtain is correct. Then, after we obtain the new concept, we can continue the learning process by adapting a few-shot setting multiclass classification where we fed the new concept dataset into a model we obtained from the FSL-FMLG framework.

8.2 New Concept Generation

Since I do not have a new concept data set that usually comes from a heavily manual annotation or a new survey of existing customers, I use the new concept using K-medoids that can generate a new label by clustering on available features. In this demonstration, I want to focus on the telecom dataset which consist of 11 features and one target. The snapshot of this dataset can be seen in Table 8.1 below:

customerID	tariff	zip	hotline_calls	complaints	data_usage	age_group	upload	contract_age	demographics_inhabitants	network_coverage	cancellation	NPS
0	1	190	2	2	96081	20	563.33	26	71938	93	FALSE	passive
1	3	212	0	5	87044	35	419.27	15	29317	62	FALSE	passive
2	3	116	0	2	19847	35	209.95	34	107546	98	FALSE	passive
3	2	264	3	2	1349	30	15.58	8	24604	51	FALSE	promoter
4	2	261	3	8	8418	25	104.48	16	23685	59	TRUE	detractor

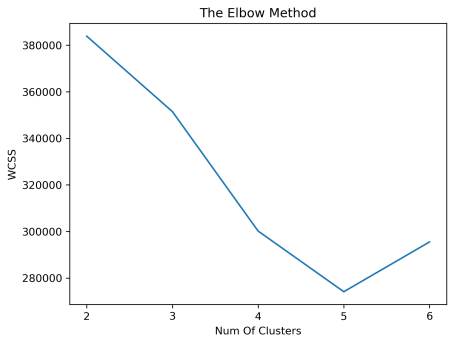
Table 8.1: Telecommunication's Customer Data

As we can see, this dataset has an existing concept called NPS that had already been learned by using the FSL-LFMG framework before. In the case of a new concept, I will remove the NPS column and replace it with the new concept that I obtain from the K-medoids procedure. The result of this procedure can be illustrated in Table 8.2 below:

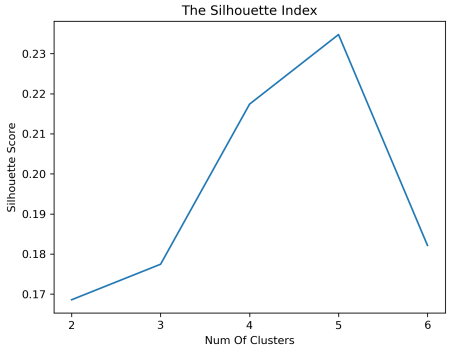
customerID	tariff	zip	hotline_calls	complaints	data_usage	age_group	upload	contract_age	demographics_inhabitants	network_coverage	cancellation	new_concept
0	1	190	2	2	96081	20	563.33	26	71938	93	FALSE	0
1	3	212	0	5	87044	35	419.27	15	29317	62	FALSE	1
2	3	116	0	2	19847	35	209.95	34	107546	98	FALSE	2
3	2	264	3	2	1349	30	15.58	8	24604	51	FALSE	3
4	2	261	3	8	8418	25	104.48	16	23685	59	TRUE	4

Table 8.2: Telecommunication's Customer Data with a New Concept

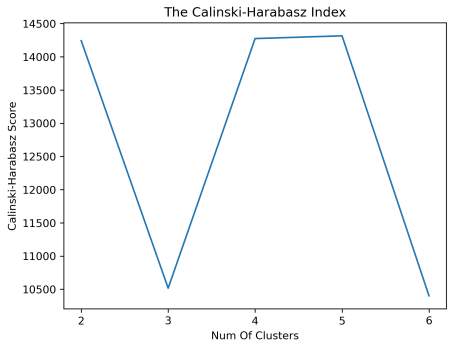
To obtain the number of K in K-medoids, I perform four evaluation plots using the elbow method, the silhouette index, the Calinski-Harabasz index (CHI) and the Davies-Bouldin index (DBI). The elbow method plot is plotting the within cluster sum of squares (WCSS) againts the number of clusters (K). Using the elbow method in Figure 8.1a, I obtain the elbow point when K=5, where the WCSS starts to decrease at a slower rate. This point indicates that five is the optimal number of clusters.



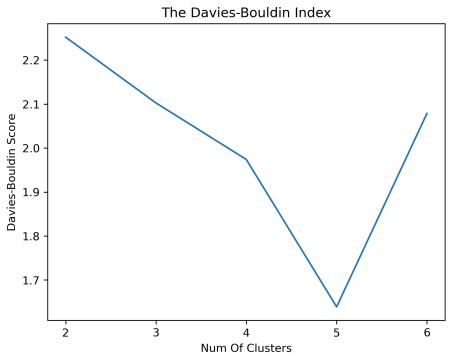
(a) Elbow Methods.



(b) Shilloute Index.



(c) Calinski-Harabasz Index (CHI).



(d) Davies-Bouldin Index (DBI)

Figure 8.1: The Evaluation Plot for K-medoids Procedures.

Using the Shilloute index, I plot the shilloute scores for the different number of clusters.The silhouette score ranges from -1 to 1. Scores close to 1 indicate well-defined clusters, while scores close to -1 indicate poorly defined clusters. From Figure 8.1b, I obtain the highest average silhouette score at K=5 which means that five is the optimal number of clusters based on Shilloute index.

Using CHI, I plot the CHI scores for different numbers of clusters. Higher CHI scores

indicate better-defined clusters. Based on Figure 8.1c, I select the number of clusters K=5 which is with the highest CHI score. Then, using DBI, I plot the DBI scores for different numbers of clusters. Lower DBI scores indicate better clustering. Based on Figure 8.1d, I choose the number of clusters K=5, which has the lowest DBI score.

After I obtained the best number of cluster K=5, I proceed to name this new concept. Using Decision Tree approach, I want to breakdown the features by tree to get better understanding how the cluster are formed. Here is the tree generated by Decision Tree Classifier in Figure 8.2. Then, I use this tree as a consideration how I name the new concept beside the justification from a subject matter expert on this dataset.

Table 8.3: Mean of Features from Each Class

Class	Mean of upload	Mean of data usage	Mean of complaints
Class 0	0.05	0.11	0.4
Class 1	0.17	0.42	0.43
Class 2	0.66	0.71	0.32
Class 3	0.07	0.15	0.43
Class 4	0.04	0.09	0.3

In addition, I use three significant features to get a better understanding of the distinction between class. I utilize aggregation metric (mean) as shown in Table from three features — upload, data usage, and complaints — that come up as the important features based on the tree split. Using this information, I get into details the main characteristics of each class as follows:

- Class 0: This class has very low mean uploads and data usage, and high mean complaints, indicating the lowest loyalty.
- Class 1: This class has moderate mean data usage and high mean complaints, but low mean uploads, suggesting moderate to high loyalty.
- Class 2: This class has the highest mean uploads and data usage, and relatively low mean complaints, indicating high engagement and loyalty.

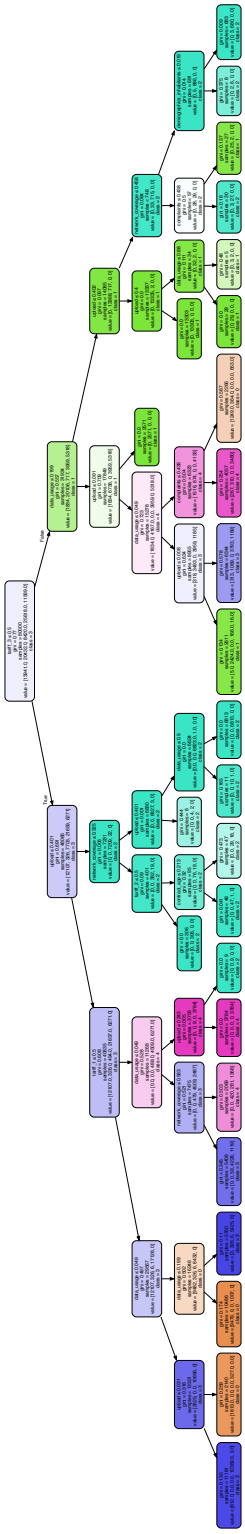


Figure 8.2: Tree from Decision Tree

- Class 3: This class has low mean uploads and data usage, and high mean complaints, indicating lower loyalty.
- Class 4: This class has very low mean uploads and data usage, but the lowest mean complaints, indicating lower engagement but fewer issues. (moderate)

In the end, I obtain "the level of customer's loyalty" as the new concept, then I use these new data to learn or adapt into my few-shot learning model.

8.3 New Concept Adaptation: Experimental Results and Discussion

The process of adapting the new concept into my framework is basically similar to adapting the existing concept. The existing concept of this dataset is NPS segmentation, while the new concept is "loyalty". Since the features for both concepts remain unchanged, no further procedure is applied. From the loyalty concept, I perform one-shot and five-shot setups and then compare the performance of the mean test accuracy with the existing concept.

Table 8.4: The Comparison of Mean Test Accuracy between The Existing Concept and The New Concept.

Methods	1-shot	5-shot
Existing Concept	36.06	41.25
New Concept	63.83	83.95

In 1-shot scenarios, Table 8.4 shows that the average test accuracy for the new concept is 63.83%, which represents a 77.01% improvement compared to the mean test accuracy for the existing concept. In 5-shot scenarios, the mean test accuracy of the new concept is 83.95%, which is a notable increase of 103. 52% compared to the mean test accuracy of the existing concept. The results demonstrate that my suggested model, implemented with the

FSL-LFMG framework, achieves optimal generalization. Furthermore, it has the ability to acquire new concepts that may be present in other datasets.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

In my Master's thesis, I propose advancing Prototypical Networks that employs augmented latent features (LF) by an autoencoder and multitasking generation (MG) by STUNT in the few-shot learning (FSL) mechanism. In conclusion, this study presents several key findings and contributions, as follows:

1. The achieved contributions to this work are sixfold:

    - I propose the FSL-LFMG framework for few-shot multiclass classification on tabular data. This framework incorporates sample-level data augmentation using autoencoders, task-level data augmentation via an enhanced STUNT framework, and Prototypical Networks to capture generalized knowledge.
    - I design the latent features learning and augmentation process that employs autoencoders to extract significant features, which are then used to enhance the quality and diversity of the training data.
    - I employ an enhanced STUNT Multitasking Generation framework that uses K-medoids instead of K-means to generate more accurate tasks.

- I implement an advanced Prototypical Networks with Manhattan distance as a classifier effectively address the multiclass classification problem.
- I conduct an extensive experimental study on four diverse domain datasets—Net Promoter Score segmentation, Dry Bean type, Wine type, and Forest Cover type—to prove that my FSL-LFMG approach on the multiclass classification outperforms the Tree Ensemble models and the One-vs-the-rest classifiers by 7.8% in 1-shot and 2.5% in 5-shot learning.
- I demonstrate the adaptation of new concept task on the model obtained from the FSL-LFMG framework — from the NPS segmentation (the existing concept) and obtain a level of customer's loyalty (the new concept) — to assess the power of generalization of this framework by significant results of the mean test accuracy in both 1-shot setting (83.95%) and 5-shot setting (103.52%).

2. Data augmentations (using autoencoders and the enhanced STUNT) play crucial role in improving classification performance.
3. Utilization of K-medoids over K-means gives improvement on classification performance.
4. Utilization of Manhattan distance over Euclidean distance also enhances classification accuracy.
5. The main work of this thesis has been accepted by The 16th International Conference on Neural Computation Theory and Applications (NCTA 2024) in Porto, Portugal.

9.2 Future Work

Moving forward, I plan to investigate more data augmentation techniques for tabular data including variational autoencoder and generative adversarial network. I also aim to explore more state-of-the-art few-shot learning techniques, such as meta-learning algorithms and

advanced metric learning approaches, which have shown promising results on tabular data in other real-world domains and areas.

Bibliography

[1] N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter, "Tabpfn: A transformer that solves small tabular classification problems in a second," *arXiv preprint arXiv:2207.01848*, 2022.

[2] A. Sikri, R. Jameel, S. M. Idrees, and H. Kaur, "Enhancing customer retention in telecom industry with machine learning driven churn prediction," *Scientific Reports*, vol. 14, no. 1, p. 13 097, 2024.

[3] C. Şahin, "Predicting base station return on investment in the telecommunications industry: Machine-learning approaches," *Intelligent Systems in Accounting, Finance and Management*, vol. 30, no. 1, pp. 29–40, 2023.

[4] S. O. Abdulsalam, M. O. Arowolo, Y. K. Saheed, and J. O. Afolayan, "Customer churn prediction in telecommunication industry using classification and regression trees and artificial neural network algorithms," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 10, no. 2, pp. 431–440, 2022.

[5] M. Loukili, F. Messaoudi, and M. El Ghazi, "Supervised learning algorithms for predicting customer churn with hyperparameter optimization.," *International Journal of Advances in Soft Computing & Its Applications*, vol. 14, no. 3, 2022.

[6] N. Saini, G. K. Monika, and K. Garg, "Churn prediction in telecommunication industry using decision tree," *International Journal of Engineering Research & Technology (IJERT)*, vol. 6, no. 4, pp. 439–433, 2017.

[7] I. Attri, L. K. Awasthi, and T. P. Sharma, "Machine learning in agriculture: A review of crop management applications," *Multimedia Tools and Applications*, vol. 83, no. 5, pp. 12 875–12 915, 2024.

[8] M. S. Khan, T. D. Nath, M. M. Hossain, *et al.*, "Comparison of multiclass classification techniques using dry bean dataset," *International Journal of Cognitive Computing in Engineering*, vol. 4, pp. 6–20, 2023.

[9] M. O. Adebiyi, R. O. Ogundokun, A. A. Abokhai, *et al.*, "Machine learning–based predictive farmland optimization and crop monitoring system," *Scientifica*, vol. 2020, 2020.

[10] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and electronics in agriculture*, vol. 147, pp. 70–90, 2018.

[11] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, p. 2674, 2018.

[12] I. Matloob, S. A. Khan, F. Hussain, W. H. Butt, R. Rukaiya, and F. Khalique, "Need-based and optimized health insurance package using clustering algorithm," *Applied Sciences*, vol. 11, no. 18, p. 8478, 2021.

[13] C. Blier-Wong, H. Cossette, L. Lamontagne, and E. Marceau, "Machine learning in p&c insurance: A review for pricing and reserving," *Risks*, vol. 9, no. 1, p. 4, 2020.

[14] H. Paruchuri, "The impact of machine learning on the future of insurance industry," *American Journal of Trade and Policy*, vol. 7, no. 3, pp. 85–90, 2020.

[15] B. Tunguz, Dieter, H. or Tails, *et al.*, *2023 kaggle ai report*, 2023. [Online]. Available: `https://kaggle.com/competitions/2023-kaggle-ai-report`.

[16] B. Sun, L. Yang, W. Zhang, *et al.*, "Supertml: Two-dimensional word embedding for the precognition on structured tabular data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2019.

[17] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[18] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022.

[19] S. O. Arik and T. Pfister, "TabNet: Attentive Interpretable Tabular Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, pp. 6679–6687, May 2021. DOI: `10.1609/aaai.v35i8.16826`. [Online]. Available: `https://ojs.aaai.org/index.php/AAAI/article/view/16826`.

[20] L. Katzir, G. Elidan, and R. El-Yaniv, "Net-dnf: Effective deep modeling of tabular data," in *International conference on learning representations*, 2020.

[21] S. Popov, S. Morozov, and A. Babenko, "Neural oblivious decision ensembles for deep learning on tabular data," *arXiv preprint arXiv:1909.06312*, 2019.

[22] R. Wang, M. Pontil, and C. Ciliberto, "The role of global labels in few-shot classification and how to infer them," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 160–27 170, 2021.

[23] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: A good embedding is all you need?" In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, Springer, 2020, pp. 266–282.

[24] W. Li, Z. Wang, X. Yang, *et al.*, "Libfewshot: A comprehensive library for few-shot learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[25] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[26] A. Parnami and M. Lee, "Learning from few examples: A summary of approaches to few-shot learning," *arXiv preprint arXiv:2203.04291*, 2022.

[27] J. Nam, J. Tack, K. Lee, H. Lee, and J. Shin, "Stunt: Few-shot tabular learning with self-generated tasks from unlabeled tables," *arXiv preprint arXiv:2303.00918*, 2023.

[28] sklearn, *Sklearn Documentation*, en, Documentation, Apr. 2024. [Online]. Available: `https://scikit-learn/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html` (visited on 04/04/2024).

[29] R. Zhang and Q. Liu, "Learning with few samples in deep learning for image classification, a mini-review," *Frontiers in Computational Neuroscience*, vol. 16, p. 1 075 294, 2023.

[30] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[31] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, pp. 197–227, 2016.

[32] Z. Yu, K. Wang, S. Xie, Y. Zhong, and Z. Lv, "Prototypical network based on manhattan distance," *Cmes-Comput. Model. Eng. Sci*, vol. 131, pp. 655–675, 2022.

[33] P. Arora, S. Varshney, *et al.*, "Analysis of k-means and k-medoids algorithm for big data," *Procedia Computer Science*, vol. 78, pp. 507–512, 2016.

[34] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[35] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[36] S. Lakshmanaprabu, K. Shankar, M. Ilayaraja, A. W. Nasir, V. Vijayakumar, and N. Chilamkurti, "Random forest for big data classification in the internet of things using optimal features," *International journal of machine learning and cybernetics*, vol. 10, no. 10, pp. 2609–2618, 2019.

[37] R. Genuer, J.-M. Poggi, C. Tuleau-Malot, and N. Villa-Vialaneix, "Random forests for big data," *Big Data Research*, vol. 9, pp. 28–46, 2017.

[38] Y. Liu, "Random forest algorithm in big data environment," *Computer modelling & new technologies*, vol. 18, no. 12A, pp. 147–151, 2014.

[39] J. T. Hancock and T. M. Khoshgoftaar, "Catboost for big data: An interdisciplinary review," *Journal of big data*, vol. 7, no. 1, p. 94, 2020.

[40] A. A. Ibrahim, R. L. Ridwan, M. M. Muhammed, R. O. Abdulaziz, and G. A. Saheed, "Comparison of the catboost classifier with other machine learning methods," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 11, 2020. DOI: `10.14569/IJACSA.2020.0111190`. [Online]. Available: `http://dx.doi.org/10.14569/IJACSA.2020.0111190`.

[41] W. Jitpakdeebodin and K. Sinapiromsaran, "Random forest algorithm using quartile-pattern bootstrapping for a class imbalanced problem," in *Proceedings of the 2023 5th International Conference on Image, Video and Signal Processing*, 2023, pp. 191–196.

[42] Q. Gu, J. Tian, X. Li, and S. Jiang, "A novel random forest integrated model for imbalanced data classification problem," *Knowledge-Based Systems*, vol. 250, p. 109 050, 2022.

[43] Y. Liu, H. Zhang, W. Zhang, G. Lu, Q. Tian, and N. Ling, "Few-shot image classification: Current status and research trends," *Electronics*, vol. 11, no. 11, p. 1752, 2022.

[44] H. Jiang, M. Gao, H. Li, R. Jin, H. Miao, and J. Liu, "Multi-learner based deep meta-learning for few-shot medical image classification," *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 1, pp. 17–28, 2022.

[45] D. Chen, Y. Chen, Y. Li, F. Mao, Y. He, and H. Xue, "Self-supervised learning for few-shot image classification," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 1745–1749.

[46] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto, "A baseline for few-shot image classification," 2020.

[47] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," *arXiv preprint arXiv:1904.04232*, 2019.

[48] B. Liu, X. Yu, A. Yu, P. Zhang, G. Wan, and R. Wang, "Deep few-shot learning for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 4, pp. 2290–2304, 2018.

[49] L. Long, W. Wang, J. Wen, M. Zhang, Q. Lin, and B. C. Ooi, "Object-level representation learning for few-shot image classification," *arXiv preprint arXiv:1805.10777*, 2018.

[50] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag, "Tabllm: Few-shot classification of tabular data with large language models," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2023, pp. 5549–5581.

[51] M. Zhu, K. Kobalczyk, A. Petrovic, *et al.*, "Tabular few-shot generalization across heterogeneous feature spaces," *arXiv preprint arXiv: 2311.10051*, 2023.

[52] H. S. Jomaa, L. Schmidt-Thieme, and J. Grabocka, "Dataset2vec: Learning dataset meta-features," *Data Mining and Knowledge Discovery*, vol. 35, no. 3, pp. 964–985, 2021.

[53] X. Wang, H. Wang, and D. Zhou, "Feature transformation network for few-shot learning," *IEEE Access*, vol. 9, pp. 41 913–41 924, 2021. DOI: 10.1109/ACCESS.2021.3065904.

[54] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1199–1208. DOI: 10.1109/CVPR.2018.00131.

[55] Y. Chen, Z. Liu, H. Xu, T. Darrell, and X. Wang, "Meta-baseline: Exploring simple meta-learning for few-shot learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 9062–9071.

[56] Y. Li and J. Yang, "Meta-learning baselines and database for few-shot classification in agriculture," *Computers and Electronics in Agriculture*, vol. 182, p. 106 055, 2021.

[57] M. Russwurm, S. Wang, M. Korner, and D. Lobell, "Meta-learning for few-shot land cover classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2020.

[58] M. Jafarzadeh, A. R. Dhamija, S. Cruz, C. Li, T. Ahmad, and T. E. Boult, "A review of open-world learning and steps toward open-world learning without labels," *arXiv preprint arXiv:2011.12906*, 2020.

[59] L. Shu, H. Xu, and B. Liu, "Unseen class discovery in open-world classification," *arXiv preprint arXiv:1801.05609*, 2018.

[60] A. Ye and Z. Wang, *Modern deep learning for tabular data: novel approaches to common modeling problems*. Springer, 2023.

[61] S. Aeberhard and M. Forina, *Wine*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C5PC7J, 1991.

[62] *Dry Bean*, UCI Machine Learning Repository, DOI: https://doi.org/ 10.24432/C50S4B, 2020.

[63] J. Blackard, *Covertype*, UCI Machine Learning Repository, DOI: https://doi.org/ 10.24432/C50K5N, 1998.