

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Meta-learning for Cross-Domain Fraud Detection

Diogo Luís Henriques Costa



Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Dr. Carlos Soares

Co-supervisor: Dr. Jacopo Bono

October 4, 2024

Meta-learning for Cross-Domain Fraud Detection

Diogo Luís Henriques Costa

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Dr. Rui Camacho

Referee: Prof. Dr. João Gama

October 4, 2024

Abstract

In the financial industry, fraud detection is vital in reducing fraud rates and promoting customer trust in financial institutions. This detection can be formalized as a machine learning binary classification problem, where the goal is to detect fraudulent transactions. However, significant challenges arise because, depending on the location of the institution and the institution itself, the fraud patterns can vary significantly from one application to another.

For these reasons, given a set of historical fraud detection datasets from several different financial institutions or geographies, it is desirable to develop machine learning models that generalize to a new financial institution and adapt to its specifics. This model should transfer positive knowledge from the training domains while adapting to the specifics of the target domain. This process ensures an initial model for new financial institutions that uses knowledge harnessed in several domains.

Several methods have been proposed for that purpose in the area of Computer Vision. In this project, we explore one of those methods, the Meta-Distillation from Mixture-of-Experts (Meta-DMoE) method. Meta-DMoE is a state-of-the-art multi-source unsupervised domain adaptation meta-learning method. We deconstructed the method into several building blocks, adapted them to the fraud detection domain, and benchmarked it against various baselines.

The results show that the Meta-DMoE method did not perform as well as some simpler baselines, most notably the gated ensemble. These results suggest that the test-time adaptation step proposed in Meta-DMoE unexpectedly does not increase performance in our settings, possibly due to the specifics of the fraud detection problem, such as the extremely low number of positive samples. Other components, such as the instance-dependent weighting of experts by the gated ensemble, seem beneficial for the baselines. This and other insights uncovered by this work provide promising future directions to improve performance in the fraud detection domain.

Keywords: machine learning, meta-learning, domain generalization, domain adaptation, fraud detection

Acknowledgments

I want to thank my supervisor, Professor Carlos Soares, for his guidance and support, even before the official start of this work. I would also like to thank my manager at Feedzai, Jacopo Bono, for his relentless support and fruitful discussions throughout my internship. A warm thank you to Ricardo and Sofia for their continuous support whenever I needed it.

I also want to thank Ahmad, Bravo, Inês, João, Sérgio, and Vladimir for making me feel welcome since day 1, and Pedro Bizarro for the opportunity to conduct my master's dissertation at Feedzai.

Finally, I want to thank my family and friends for their unwavering support and encouragement throughout this journey, not only during this dissertation but also during my whole academic journey.

Diogo Luís Henriques Costa

Contents

Abstract	i
1 Introduction	1
1.1 Thesis statement and Objectives	1
1.2 Document Structure	2
2 Literature Review	3
2.1 Fraud detection	3
2.1.1 Machine learning for fraud detection	4
2.1.2 Evaluation metrics	4
2.1.3 Issues and challenges	6
2.2 Meta-learning	6
2.2.1 Model evaluations based	6
2.2.2 Task properties based	7
2.2.3 Prior models based	7
2.3 Meta-learning for Domain Adaptation	8
2.3.1 Methodology	8
2.3.2 Single vs. Multi-source	8
2.3.3 Labeled data usage	8
2.3.4 Data types, domain of application, and datasets	9
2.3.5 Tasks and evaluation measures	9
2.3.6 Model-Agnostic Meta-Learning	10
2.3.7 Meta-Distillation from Mixture-of-Experts	10
2.4 Summary	11
3 Meta-DMoE	12
3.1 Preliminaries	12
3.2 Training algorithm	13
3.3 Test-time adaptation	14
4 Cross-Domain Fraud Detection with Meta-DMoE	16
4.1 Experimental setup	16
4.1.1 Datasets	16
4.1.2 Meta-DMoE models	20
4.1.3 Baselines	23
4.2 Results	25
4.2.1 Expert models cross-domain evaluations	26
4.2.2 Meta-DMoE and baselines	27

4.2.3 Summary	28
5 Conclusions	30
5.1 Future Work	31
References	33
A Hyperparameter tuning	39
A.1 Experts	39
A.2 Aggregator	39
A.3 Student	40
A.4 Meta-DMoE	40

List of Figures

2.1	Model-Agnostic Meta-Learning diagram.	10
3.1	Expert models training diagram.	13
3.2	Meta-DMoE training diagram.	15
4.1	Time windows of the used datasets.	17
4.2	Data splits applied to the datasets.	19
4.3	Data splits used for the aggregator pre-training, student pre-training, and Meta-DMoE training.	20
4.4	Data splits used for the multi-domain validation setup training.	21
4.5	Blocked neural network diagram.	21
4.6	Aggregator composition diagram.	22
4.7	Mean ensemble baseline diagram.	23
4.8	Training of the gated ensemble model.	24
4.9	Pre-training of the aggregator model.	25
A.1	Aggregator pre-training optimization history.	41
A.2	Student pre-training optimization history.	41
A.3	Meta-DMoE optimization history.	42

List of Tables

2.1	Confusion Matrix for fraud detection as a binary classification problem.	5
2.2	Number of analyzed papers per year of publication.	8
4.1	Approximated # Transactions, # Frauds and fraud rates per dataset.	17
4.2	Expert models and mean ensemble performance across datasets.	27
4.3	Expert models and mean ensemble performance across datasets after removing domain-dependent features.	28
4.4	Evaluation results of the Meta-DMoE method and baselines on the target dataset.	29
A.1	Expert models hyperparameter values.	39
A.2	Best expert models hyperparameter values.	40
A.3	Aggregator model hyperparameter values.	40
A.4	Meta-DMoE hyperparameter values.	41

Abbreviations

ASD	Average Symmetric Surface Distance
AUC	Area Under the ROC Curve
BLEU	Bilingual Evaluation Understudy
DA	Domain Adaptation
DL	Deep Learning
DSC	Dice Similarity Coefficient
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
HPT	Hyperparameter Tuning
kNN	k-Nearest Neighbors
LSTM	Long Short-Term Memory
mAP	Mean Average Precision
MAML	Model-Agnostic Meta-Learning
MANN	Memory-Augmented Neural Networks
Meta-DMoE	Meta-Distillation from Mixture-of-Experts
ML	Machine Learning
MMD	Maximum Mean Discrepancy
MoE	Mixture-of-Experts
OSDA	Open Set Domain Adaptation
PDA	Partial Domain Adaptation
PCA	Principal Component Analysis
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks
SLR	Saved Loss Rate
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate

Chapter 1

Introduction

Standard machine learning models encounter difficulties in new environments, grappling with issues related to domain shift, data heterogeneity, and scarcity of labeled samples in the target domain. In response to these challenges, domain adaptation techniques have been developed to enable models trained on a source domain to generalize effectively to different yet related target domains. At Feedzai¹, the market leader in fighting financial crime with machine learning, when a new financial institution is contracted, no historical and labeled data is often readily available. A usual approach is to apply some out-of-the-box rule system, tweak that system to the client's needs, and collect data until a custom machine learning model can be trained. This approach does not leverage other clients' knowledge, potentially losing out on better detection.

Domain adaptation is a well-established field in machine learning, with several surveys and reviews available in the literature [53, 20, 15, 56, 12]. However, leveraging meta-learning for domain adaptation is a relatively new and promising approach that has been gaining traction in recent years [67, 35, 73, 36], and has shown promising results. Despite that, most of the work is focused on image and natural language processing tasks, which poses a challenge to this dissertation.

In this work developed at Feedzai, we wanted to investigate how we can use the available clients' domains to improve the fraud detection solution in the first stages of a new client while there is little data and still no labels are available.

1.1 Thesis statement and Objectives

Given a set of fraud detection domains, with both features and labels, and a new unseen related domain with the same schema, the ideal solution would maximize fraud detection performance on the new domain by improving over time the model used on it. This model improvement would start with a generalized model based on the set of known domains and would evolve taking advantage of the arrival of data from the new domain along time. In our particular setting, unlabeled data is available first and only after a label delay, labels for the previous unlabeled points start to arrive.

¹<https://feedzai.com/>

In this dissertation, we aim to tackle the previous problem from the beginning, even though not solving it completely. We want to build a 0-day solution for new domains improving the current approach to this problem but we also want to study the incorporation of meta-learning techniques when unlabeled data starts to become available on the new domain.

1.2 Document Structure

In Chapter 2, we present a background on Fraud Detection and Meta-learning and the state-of-the-art on Meta-learning for Domain Adaptation. In Chapter 3 we present the main method of this work. In Chapter 4 we present both the experimental setup and the results of the empirical study, and in Chapter 5 we present some future work directions and conclusions.

Chapter 2

Literature Review

This chapter presents the literature review for the dissertation. We start by presenting an overview of Fraud Detection in Section 2.1. Then, we present an overview of Meta-learning in Section 2.2. Finally, we present the state-of-the-art Meta-learning for Domain Adaptation in Section 2.3.

2.1 Fraud detection

According to the European Anti-Fraud Office, “fraud is a deliberate act of deception intended for personal gain or to cause a loss to another party”.¹ In 2023 alone, according to the United States Federal Trade Commission, American consumers lost over \$10 Billion to fraud, a figure that represents an increase of 14% relatively to the previous year.²

In order to systematize the concepts of fraud detection and prevention, several surveys can be found in the literature [47, 6, 1]. Fraud can be classified into four main areas: bank fraud, insurance fraud, telecommunication fraud, and internet marketing fraud (also known as e-commerce fraud) [1]. We will now focus on credit card fraud, a subset of fraud aligned with the context of this work, which can be seen as both a type of bank and e-commerce fraud and for which there are several categorizations found in the literature, such as:

- **Offline vs. online:** This distinction pertains to whether the fraud occurs through physical theft of the card or online exploitation of card information [30].
- **Application vs. behavioral:** This classification is based on the fraudster’s strategy. Application fraud involves entering false information to open a new credit card. In contrast, behavioral fraud occurs when cardholder details are obtained for subsequent sales, including telephone and e-commerce transactions [14].

¹https://anti-fraud.ec.europa.eu/olaf-and-you/report-fraud_en

²<https://www.ftc.gov/news-events/news/press-releases/2024/02/nationwide-fraud-losses-top-10-billion-2023-ftc-steps-efforts-protect-public>

- **Traditional vs. merchant-related vs. internet-based:** Another classification considers traditional card-related frauds, merchant-related frauds where the fraudster poses as a legitimate business, and internet-related frauds, each presenting distinct challenges and characteristics [40].

2.1.1 Machine learning for fraud detection

Fraud detection is a critical task in the financial sector that was traditionally performed by rule-based systems. However, the increasing complexity of fraud strategies has motivated the use of machine learning techniques.

Given a transaction X_{ij} made using a card number i with transaction number j , we want to predict whether the transaction is fraudulent. The transaction is represented by a set of features $X_{ij} = \{x_{ij1}, x_{ij2}, \dots, x_{ijn}\}$, where n is the number of features. The label y_{ij} is binary, with $y_{ij} = 1$ if the transaction is fraudulent and $y_{ij} = 0$ otherwise. A function f , the true function we want to model, maps the transaction features to the label, i.e., $f : X_{ij} \rightarrow y_{ij}$. To approximate f , $\hat{f} : X_{ij} \rightarrow \hat{y}_{ij}$ is modeled such that $\hat{y}_{ij} \approx y_{ij}$ according to a given loss function L . This formalization is straightforward, but as f is defined, taking as input only the information of one transaction at a time, it has no context to the usual behavior of the corresponding card holder, information that is absent from the transaction itself.

A more complete approach is to consider historical transactions made by the same card to adapt $\hat{f}$ to the historical context of the respective card [13]. Given a transaction $X_{i\lambda}$, let $dt(X_{i\lambda})$ be the date-time of the transaction. For a given time-frame T , e.g., one year, we can define the set of historical transactions occurring before $X_{i\lambda}$ for a given card i as $X_{i\lambda}^H = \{X_{ij}\}$ such that $0 < dt(X_{i\lambda}) - dt(X_{ij}) \leq T$. Using $X_{i\lambda}^H$, we can use feature engineering by adding aggregated features (e.g., count, max) of the historical transaction features to the original transaction $X_{i\lambda}$, resulting in a new instance $X'_{i\lambda}$ which contains information about the usual behavior of the cardholder. Note, however, that these historical aggregations do not necessarily need to be made around the cardholder entity. Entities such as merchants where the purchase was made, as well as IP addresses, can be used to aggregate the historical context around them, which makes it easier to detect unusual behavior given the context.

2.1.2 Evaluation metrics

Several performance metrics are used in the context of a binary classification problem. However, most can be defined in terms of the confusion matrix, as shown in Table 2.1. The four combinations of the actual and predicted labels are:

- **True Positive (TP):** The transaction is fraudulent, and the model correctly predicts it as fraudulent, $y = \hat{y} = 1$.
- **False Positive (FP):** The transaction is genuine, but the model predicts it as fraudulent, $y = 0, \hat{y} = 1$.

Table 2.1: Confusion Matrix for fraud detection as a binary classification problem.

	True Fraud	True Genuine
Predicted Fraud	TP	FP
Predicted Genuine	FN	TN

- **False Negative (FN):** The transaction is fraudulent, but the model predicts it as genuine, $y = 1 \hat{y} = 0$.
- **True Negative (TN):** The transaction is genuine, and the model correctly predicts it as genuine, $y = \hat{y} = 0$.

These in turn can be combined to calculate important metrics used later on:

- **True Positive Rate (TPR):** Also known as recall or sensitivity, it is the proportion of actual positives that are correctly predicted as positive. Mathematically, it is defined as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **False Positive Rate (FPR):** This is the proportion of actual negatives (genuine transactions) that are incorrectly predicted as positive (fraudulent). Mathematically, it is defined as:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

It measures how often the model falsely flags genuine transactions as fraudulent.

It is important to note that due to the unbalanced nature of the data, as discussed in Sec. 2.1.3, metrics such as accuracy are not suitable for evaluating the performance of a fraud detection system. AUC (Area Under the ROC Curve [7]) and Recall (TPR) at fixed FPR are more appropriate for these settings and used in the literature [42, 2, 5]. Furthermore, the latter one, Recall at fixed FPR, is essential in fraud detection scenarios where the FPR is fixed by business requirements to avoid customer dissatisfaction, which can lead to customer churn.

Other metrics take into account the cost structure of the misclassification, such as the cost-sensitive metric Saved Loss Rate (SLR) [49], which is defined in Eq. 2.1, where k = number of frauds detected, n_+ = total number of frauds, and $(C_{FN})_i$ = misclassification cost of FN_i .

$$\text{SLR} = \frac{\sum_{j=1}^k (C_{FN})_j}{\sum_{i=1}^{n_+} (C_{FN})_i} \quad (2.1)$$

This metric pertains to the percentage of the total amount of saved available usable limits (i.e., the amount of money lost if the transaction was not detected as fraudulent) by the model.

2.1.3 Issues and challenges

Several issues and challenges make credit card fraud detection a difficult task. The main issues are [1]:

- **Concept drift:** [38] It occurs when the data distribution changes over time, both the cardholder behavior and the fraudster strategy. For example, a cardholder may go through large time-frames without making any transactions and then make several transactions to buy Christmas gifts by the end of the year. On the other hand, a fraudster may also act adversarially to the defense system in place by changing the fraud strategy accordingly so that it goes unnoticed.
- **Unbalanced data:** The number of fraudulent transactions is usually much smaller than that of non-fraudulent transactions, some times in the order of 1 fraud per 1000 transactions as we will see later on Sec. 4.1.1 of the empirical study.
- **Curse of dimensionality:** By being a relevant process in this context, feature engineering can add many features to the original ones, which can motivate the use of dimensionality reduction and feature selection techniques.
- **Real time detection:** Most fraud detection systems must be able to detect fraud in real-time, typically with latency constraints on the order of milliseconds. This makes the use of computationally expensive techniques at classification time unfeasible.

2.2 Meta-learning

Meta-learning is a sub-field of ML that tries to automate machine learning [63]. If we establish an informal analogy between ML and human learning, we can say that humans learn new tasks using knowledge acquired from previously learned tasks. In this section, based mostly on *Automated Machine Learning* [58], we present the main approaches to Meta-learning distinguished by the meta-data used: model evaluations on Sec. 2.2.1, meta-features on Sec. 2.2.2, and prior models on Sec. 2.2.3.

2.2.1 Model evaluations based

Given a set of known tasks $t_j \in T$, and a set of learning algorithms with their respective configurations $\theta_i \in \Theta$, and the performances of the algorithms on the tasks $P_{i,j} = P(\theta_i, t_j) \in P$, the goal of this approach is to train a meta-learner L that can recommend the best algorithm and configuration θ_{best} for a new task t_{new} given some known evaluations $P_{i,new}$ of it, i.e., $\theta_{best} = \arg \max_{\theta_i \in \Theta} P(\theta_i, t_{new})$.

This approach leverages extensive collections of tasks, algorithms, configurations, and performance measures obtained, such as in OpenML [59]. Without much information on the new task t_{new} , this approach starts by recommending the best overall algorithm and configuration, which does not distinguish between new tasks but is enough to cold-start the meta-learner.

After having performance measures of some configurations on t_{new} , we can use the similarity between those measures and the measures of the tasks in T to approximate the similarity between t_{new} and the tasks in T . This can be done by using a distance function d such as Euclidean distance, and then using the k nearest neighbors (kNN) of t_{new} to recommend the best algorithm and configuration for t_{new} .

2.2.2 Task properties based

The main idea of this approach is to represent each task t_j as a vector of meta-features $m(t_j) = (m_1(t_j), \dots, m_n(t_j))$, where n is the number of meta-features. Then, the similarity between tasks can be measured by the distance between the vectors, i.e., $d(t_i, t_j) = d(m(t_i), m(t_j))$, according to some distance function d . They can be used to train a meta-learner that recommends the best algorithm and configuration for a new task t_{new} , i.e., $L : t_{new} \rightarrow \theta_{best}$.

The meta-features are computed on single features or combinations of features and aggregated using summary statistics such as mean and standard deviation and can be divided into six categories [46]: Simple, Statistical, Information-theoretic, Model-based, Landmarking, and Others.

Other than predicting the best configuration for a new task, meta-features can also be used to predict the performance of a given algorithm and configuration on a new task [32].

2.2.3 Prior models based

This approach is based on the idea that the knowledge acquired by a model on previous tasks can be used to learn a new task. Given a new task t_{new} , similar tasks $t_j \in T$, and the optimized models $l_j \in L$ for each task t_j , the goal is to train a learner l_{new} that learns t_{new} . This approach can be divided into transfer learning and few-shot learning.

The idea of transfer-learning is to use already trained models l_j as initialization for the learner l_{new} , which is then trained on t_{new} . l_{new} leverages the knowledge acquired by the models l_j , namely the model parameters and configuration, to learn t_{new} faster and with less data than if trained from scratch. Neural networks are particularly suitable for this approach.

Few-shot learning is the problem of training an accurate deep-learning (DL) model with limited training examples. The formal way of defining the training examples' limits is by using the *K-shot N-way* nomenclature, where K is the number of training examples per class, and N is the number of new classes [64, 27, 10].

Meta-learning in neural networks has recently been gaining attention in the literature. Various approaches have been proposed in the literature, such as replacing optimizers with Long Short-Term Memory (LSTM) networks [24] and using Memory-Augmented Neural Networks (MANN) [50] to store information about previous tasks.

Table 2.2: Number of analyzed papers per year of publication.

Year	2018	2019	2020	2021	2022	2023
# papers	1	2	4	10	4	3

2.3 Meta-learning for Domain Adaptation

Before going any further, it is important to clarify the relationship between the concepts of task and domain previously introduced. In meta-learning, task often refers to learning a model that solves a specific problem, such as mapping an input space to an output space. Similarly, in DA, a domain is associated with a feature space, a label space, and a joint probability distribution that defines the data from which the model learns. Therefore, learning in a specific domain can be viewed as learning a different task in the context of DA.

2.3.1 Methodology

To identify the state-of-the-art in meta-learning for domain adaptation, we searched both *Google Scholar*¹ and *IEEE Xplore*² using the following search query: ("*meta learning*" OR *meta-learning*) AND "*domain adaptation*". We got 146 results in *IEEE Xplore*, which we filtered by relevance based on the titles, abstracts, and keywords. In *Google Scholar*, we got around 8000 ordered results, which we filtered the same way as in *IEEE Xplore* until we had at least 20 relevant results. It is worth noting that after a few result pages, the relevance of the results decreased significantly, which is why we stopped at around 20 results. This way, we analyzed 24 papers, with a median of 28.5 citations per paper, published between 2018 and 2023, as shown in Table 2.2.

2.3.2 Single vs. Multi-source

The most popular setting in DA involves transferring knowledge from a source domain D_S to a target domain D_T , where $D_S \neq D_T$ [68, 73, 36, 66]. Domains can be defined by three components: feature space X , label space Y , and an associated joint probability distribution $p(x, y)$: $D = \{X, Y, p(x, y)\}$ [15].

On the other hand, in multi-source DA, the goal is to transfer knowledge from multiple source domains $D_{S_1}, D_{S_2}, \dots, D_{S_n}$ to a target domain D_T , where $D_{S_i} \neq D_T$ and $D_{S_i} \neq D_{S_j}$ for $i \neq j$ [33, 44, 74]. This setting is instrumental because, since it uses multiple source domains, it can also extract better domain-invariant features that provide a better starting point for an improvement of the model's performance on the target domain [41].

2.3.3 Labeled data usage

Depending on the availability of labeled data and the label space of the source and target domains, DA can be divided into three categories [62]:

¹<https://scholar.google.com/>

²<https://ieeexplore.ieee.org/>

- **Supervised or Semi-Supervised:** The source domain consists of labeled samples $D_s = \{x_i^s, y_i^s\}_{i=1}^{n_s}$ and the target domain consists of both labeled and unlabeled samples $D_t = \{x_i^t, y_i^t\}_{i=1}^{n_t} \cup \{x_i^t\}_{i=n_t+1}^{n_t+n_u}$, where n_s is the number of labeled samples in the source domain, n_t is the number of labeled samples in the target domain, and n_u is the number of unlabeled samples in the target domain, with $n_t \ll n_u$ and $n_t \ll n_s$. Due to the low number of labeled samples in the target domain, the joint probability distribution $p_t(x^t, y^t)$ is hard to model solely based on the available labeled samples n_t without risking overfitting [33].
- **Unsupervised:** In this setting, the source domain consists of labeled samples $D_s = \{x_i^s, y_i^s\}_{i=1}^{n_s}$ and the target domain consists of unlabeled samples $D_t = \{x_i^t\}_{i=1}^{n_t}$, where generally $n_s \approx n_t$. [9, 35, 70, 68, 52, 36, 66, 74].
- **Unconstrained:** While the previous two settings assume that the source and target domains have the same label space Y , this setting does not have this restriction. The two main variants of this setting are: *Partial Domain Adaptation* (PDA), where the target domain has a subset of the source domain's label space, and *Open Set Domain Adaptation* (OSDA), where there is an intersection between the label spaces of the source and target domains, but none is a subset of the other.

2.3.4 Data types, domain of application, and datasets

Most of the analyzed papers used image data [8, 28, 33, 9, 72, 44, 35, 68, 52, 57, 66, 36, 37, 71, 74], some used text [69, 21, 51, 43, 34, 36], only 3 used tabular data [16, 67, 73], and 1 used video [70]. Most papers had no specific application domain apart from the domains inherent to the datasets used for evaluation. However, some had a specific domain, such as mechanical fault diagnosis [16, 67, 73], medical imaging [72, 35] and fashion [52].

The papers that proposed methods for tabular data were all within the same domain of application, mechanical fault diagnosis [16, 67, 73]. All these works used several datasets for evaluation, but there was a common one among them, the Case Western Reserve University bearing dataset [54].

Among the other papers, a great variety of datasets were used for evaluation. However, some datasets were used by several, such as Office-Home [61] used by 5 papers [33, 9, 68, 36, 66], Office-31 [48] used by 4 papers [28, 9, 36, 66], and MNIST [31] used by 3 papers [28, 37, 71].

2.3.5 Tasks and evaluation measures

The majority of papers focused on classification tasks [16, 33, 9, 36, 66, 8, 67, 28, 21, 68, 73, 57, 37, 71, 74], 7 of which in the K-shot N-way setting [8, 67, 28, 21, 68, 73, 57]. Several other tasks were also tackled, such as image segmentation [72, 35], object detection [66], language translation [69, 51], text recognition [44], background subtraction [70], sentiment analysis [36], sequence labeling [34], dialog generation [43], and consumer-to-shop fashion retrieval [52].

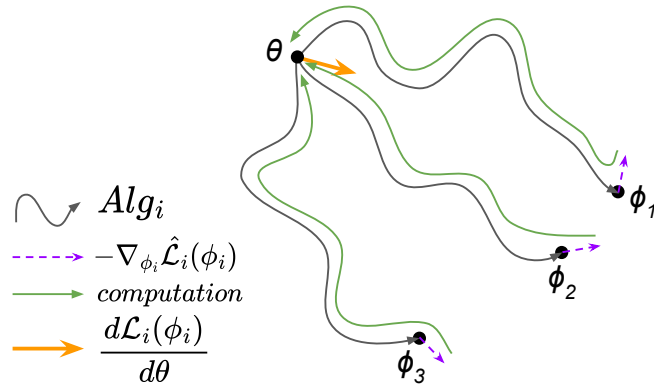


Figure 2.1: Model-Agnostic Meta-Learning optimizes for a representation θ that can quickly adapt to new tasks. [45]

As for the evaluation measures, they were also quite diverse so that they could appropriately evaluate the different tasks: 16 papers used accuracy [8, 16, 67, 28, 33, 9, 21, 44, 68, 73, 36, 57, 66, 37, 71, 74], 5 used the F1-Measure [69, 70, 43, 34, 74], 3 used bilingual evaluation understudy (BLEU) [69, 51, 43], and 2 used the dice similarity coefficient (DSC) [72, 35]. Other metrics were also used, such as maximum mean discrepancy (MMD) [8], mean average precision (mAP) [52], top-k accuracy [52], root mean square error (RMSE) [35], average symmetric surface distance (ASD) [35], false positive rate (FPR) [70], false negative rate (FNR) [70], recall [70], specificity [70], precision [70] and execution time [43].

2.3.6 Model-Agnostic Meta-Learning

After analyzing several dimensions of approaches, we will restrict the discussion to a generic approach widely used in the literature of meta-learning for gradient-descent-based methods: Model-Agnostic Meta-Learning (MAML) [17], which serves as background for the method summarized in the next section, Sec. 2.3.7 and described more thoroughly in Chapter 3. MAML is model-agnostic as long as the model can be trained using gradient descent. The main idea is to learn an initialization θ of the model parameters across tasks, such that the model can then be quickly adapted to a new task using a few examples as shown in Fig. 2.1. The general algorithm is shown in Algorithm 1, but more specific versions, adapted for specific settings, such as Few-Shot Learning and Reinforcement Learning, can be found in the original paper [17]. From the analyzed papers, several were either based on or used MAML [67, 72, 51, 43, 74] as the general framework of which they are instances of, although some present shortcomings of it and solutions to overcome them [28].

2.3.7 Meta-Distillation from Mixture-of-Experts

Meta-Distillation from Mixture-of-Experts (Meta-DMoE) [74] is an unsupervised method that performs multi-source domain adaptation at test-time with a single batch of unlabeled data from

Algorithm 1 Model-Agnostic Meta-Learning [17]

Require: $p(T)$: distribution over tasks**Require:** α, β : step size hyper-parametersInitialize θ randomly**while** not done **do** Sample batch of tasks $T_i \sim p(T)$ **for** each T_i **do** Evaluate $\nabla_{\theta} L_{T_i}(f_{\theta})$ with respect to K examples Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_{\theta} L_{T_i}(f_{\theta})$ **end for** Update $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta'_i})$ **end while**

the target domain. This adaptation is modeled as a knowledge distillation process [19], a technique in which a “student” model learns from the predictions of a larger or more complex “teacher” model, transferring knowledge in a compressed form. In the Meta-DMoE approach, the teacher is a Mixture-of-Experts, first introduced by Jacobs et al. [26], comprising several model experts, each specialized in a source domain. The outputs of these experts are combined by an aggregator model, which supervises the student model during distillation. The main goal is to leverage the knowledge acquired by the experts in the source domains to improve the student model’s performance in the target domain. Chapter 3 will describe this method in more detail.

2.4 Summary

In this literature review, we explored the concepts of Fraud Detection, Meta-learning, and DA. Fraud Detection involves identifying deliberate acts of deception for personal gain or causing a loss to another party, with credit card fraud being a specific focus.

Fraud Detection involves predicting the fraudulent nature of transactions, considering historical data on the card and cardholder behavior. Credit card fraud detection challenges include concept drift, unbalanced class distribution, many features, and the need for real-time detection.

Meta-learning is a sub-field of ML that studies how learning can be learned, often leveraging knowledge acquired from previously learned tasks. Applying meta-learning to domain adaptation involves adapting the model to new domains and addressing challenges such as the availability of labeled data, the number of source domains available, and the different label space of the domains.

The analyzed literature used diverse data types, including images, text, tabular data, and video. Tasks ranged from classification and image segmentation to sentiment analysis and sequence labeling. The evaluation measures were also diverse, including accuracy, F1-Measure, and BLEU. However, only three papers focused on tabular data and all of them focused on the same problem: mechanical fault diagnosis. We see this as an opportunity to try to adapt a method, namely Meta-DMoE [74] from the Computer Vision field, to tabular data in particular to a fraud detection scenario.

Chapter 3

Meta-DMoE

In this chapter, we describe the Meta-DMoE method [74], already summarized in Sec. 2.3.7. This method presents itself as a meta-learning approach to unsupervised multi-source domain adaptation that can be deconstructed into several building blocks. These, in turn, can be combined in some other ways to form different baselines to which we can compare the primary method and draw conclusions about the necessity of each component, as we will see in Chapter 4.

Given the nature of this work as described in Sec. 1.1, and after reviewing the literature on meta-learning for domain adaptation in Sec. 2.3, we can identify several reasons to choose the Meta-DMoE [74] method as the primary method to be studied in this dissertation. This method leverages the knowledge from multiple source domains, which can be beneficial in a real-world scenario where the model needs to adapt to a new domain. Interestingly, the method does not require access to the raw data of the source domains, only the models trained on them. This presents advantages regarding privacy concerns and deployment costs, which are relevant in a business scenario. The method is also flexible regarding the architecture of the expert and student models as long as they can be decomposed into a feature extractor and a classifier. Finally, the method matches the data availability setting, starting with domain generalization and then moving to the unsupervised setting, where the labels are unavailable initially. Furthermore, the method is based on an established generic framework, MAML [17], which is effective in several meta-learning tasks.

3.1 Preliminaries

Let $X = X_c \circ X_e$ be a generic machine learning model that can be decomposed into a feature extractor X_e followed by a classifier model X_c . Let X^D be the model X trained on domain D . Let $\mathcal{D}_S = \{D_S^i\}_{i=1}^N$ be a set of N source domains, each with its own dataset. Let $\mathcal{D}_T = \{D_S^j\}_{j=1}^K$ be a set of K target domains, each with its own dataset.

Let $M = M_c \circ M_e$ be a model trained on a specific subset of $\mathcal{D}_S$, either a single domain, a combination of domains, or a subset of a domain, which, for these reasons, it is called *expert model*, as in domain expert. Let $A = A_c \circ A_e$ be an *aggregator model* that takes as input the output

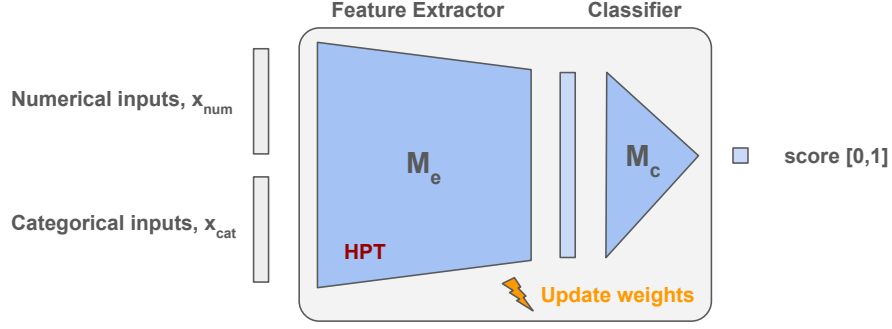


Figure 3.1: Expert models training diagram. Each expert model is trained on a different source domain. They are composed of a feature extractor followed by a classifier, but only the feature extractor has its architecture hyperparameter tuned.

of several M_e models and outputs a score value. Let $S = S_c \circ S_e$ be a meta-trained *student model* to adapt to a domain D_T using the knowledge distilled from all the M_e models passed through A_e . Further details on how these models interact will be presented in Sec. 3.2.

3.2 Training algorithm

The training algorithm is summarized in Algorithm 2, and the training diagram is shown in Fig. 3.2 and we will now go over the main steps of the training process.

The authors of the Meta-DMoE method define $\text{MoE} = \{M^i\}_{i=1}^N$ as a set of N expert models, each trained on a different source domain such that each M^i is a model $M^i = M_c^i \circ M_e^i$ trained on a domain D_{S^i} . The expert models are trained independently and in parallel, as shown in Fig. 3.1.

After the expert models are trained, they no longer need to be updated and remain fixed. After that, the method follows the MAML [17] framework by treating the learning on a given domain as a task. In the outer loop, a batch of B source domains (meta-batch) is sampled $\{D_S^b\}^B$ so that in the inner loop, by iterating over each domain on the meta-batch, a query set $(\mathbf{x}_Q, \mathbf{y}_Q)$ and support set $(\mathbf{x}_{SU})$ are sampled from the current domain b : $(\mathbf{x}_{SU}), (\mathbf{x}_Q, \mathbf{y}_Q) \sim D_S^b$.

The support set $(\mathbf{x}_{SU})$, which has no labels, is used to simulate during training the knowledge distillation process from the expert models M_e to the student model S_e , which will happen at test time. This is important as it aligns the training and evaluation objectives, improving the adaptation step. The query set $(\mathbf{x}_Q, \mathbf{y}_Q)$, which has labels $\mathbf{y}_Q$, is used to evaluate how good the previous knowledge distillation process was.

The knowledge distillation process is done by passing the support set through the expert models M_e , whose outputs are then passed through the aggregator model A_e , which outputs a single representation used to compute an L_2 distance with the representation output by the student model S_e , as defined by Eq. 3.1 where $\mathbf{y}$ is the output of the teacher model A_e and $\hat{\mathbf{y}}$ is the output of the student model S_e .

$$L_2(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^n (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2 \quad (3.1)$$

This distance function used as loss function is back-propagated to a copy of the student model S_e , which is then used to predict the query set. The outputs of the student model $S_e(\mathbf{x}_Q; \theta_e')$ are passed to a classifier head that outputs the final predictions: $S(\mathbf{x}_Q; \theta_e', \theta_c) = S_c(S_e(\mathbf{x}_Q; \theta_e'); \theta_c)$. The predictions are compared to the labels of the query set, and a cross-entropy loss is computed, $L_{CE}(\mathbf{y}_Q, S(\mathbf{x}_Q; \theta_e', \theta_c))$, and back-propagated to the classifier head, the student model, and the aggregator model. Note that to enforce that the aggregator model learns to combine the expert models without relying on the expert that was trained on the same domain as the current domain from which the support set was sampled, this expert is not considered at each iteration.

Algorithm 2 Meta-DMoE [74] training algorithm

Require: $\{D_S^i\}_{i=1}^N$: data of source domains; α, β : learning rates; B : meta batch size
for $i = 1, \dots, N$ **do** ▷ Pretrain domain-specific MoE models
 Train the domain-specific model M_i using D_S^i .
end for
Initialize: ϕ, θ_e, θ_c
while not converged **do** ▷ Meta-train aggregator $A_e(\cdot; \phi)$ and student model $S(\cdot; \theta_e, \theta_c)$
 Sample a batch of B source domains $\{D_S^b\}^B$, reset batch loss $L_B = 0$
 for each D_S^b **do**
 Sample support and query set: $(\mathbf{x}_{SU}), (\mathbf{x}_Q, \mathbf{y}_Q) \sim D_S^b$
 $M_e'(\mathbf{x}_{SU}) = \{M_e^i(\mathbf{x}_{SU})\}_{i=1, i \neq b}^N$
 Perform adaptation via knowledge distillation from MoE:
 $\theta_e' = \theta_e - \alpha \nabla_{\theta_e} \|A_e(M_e'(\mathbf{x}_{SU}); \phi) - S_e(\mathbf{x}_{SU}; \theta_e)\|_2$
 Evaluate the adapted θ_e' using the query set and accumulate the loss:
 $L_B = L_B + L_{CE}(\mathbf{y}_Q, S(\mathbf{x}_Q; \theta_e', \theta_c))$
 end for
 Update ϕ, θ_e, θ_c for the current meta batch:
 $(\phi, \theta_e, \theta_c) \leftarrow (\phi, \theta_e, \theta_c) - \beta \nabla_{(\phi, \theta_e, \theta_c)} L_B$
end while

3.3 Test-time adaptation

A single batch $\mathbf{x}_{\text{target}}$ from the target domain adapts the model S_e via the knowledge distillation process at test time using the L_2 distance loss mentioned earlier: $\theta_e' = \theta_e - \alpha \nabla_{\theta_e} \|A_e(\{M_e^i(\mathbf{x}_{\text{target}})\}_{i=1}^N; \phi) - S_e(\mathbf{x}_{\text{target}}; \theta_e)\|_2$. After that, only the adapted copy of the student network and the classifier head are used to classify the rest of the target domain's data: $S(\mathbf{x}_Q; \theta_e', \theta_c)$.

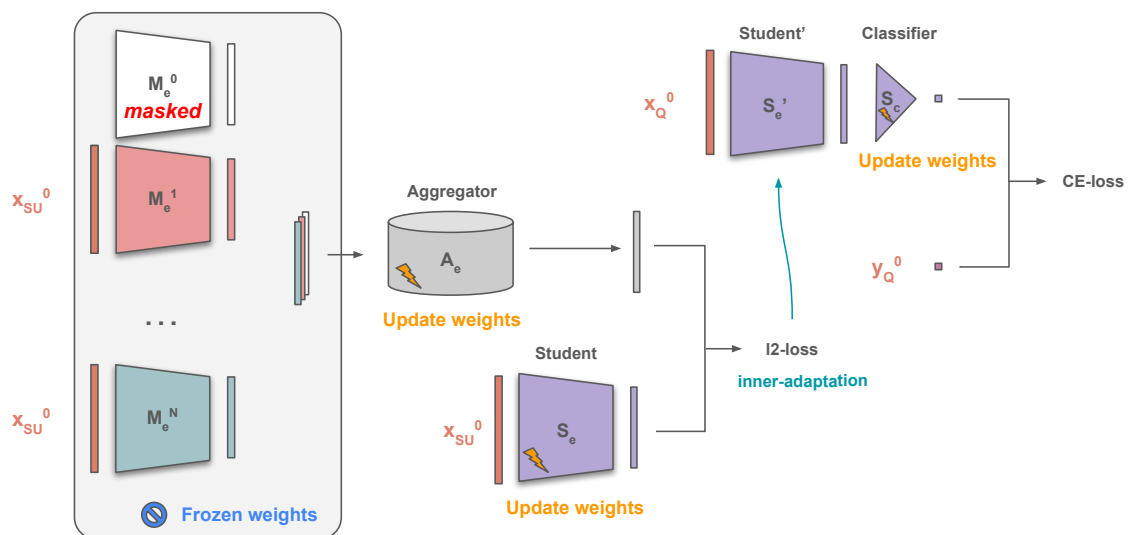


Figure 3.2: Meta-DMoE training diagram. The expert models are frozen after being trained on the source domains. The student model is trained to adapt to a target domain using the knowledge distilled from the expert models through the aggregator model.

Chapter 4

Cross-Domain Fraud Detection with Meta-DMoE

This chapter presents the empirical study of adapting Meta-DMoE [74], a Computer Vision meta-learning method, to cross-domain fraud detection with real-world datasets. We start with the experimental setup in Section 4.1 and discuss the results in Section 4.2.

4.1 Experimental setup

This section describes the experimental setup used to evaluate both the Meta-DMoE method and deconstructions of it on fraud detection real-world datasets. All the models and experiments were implemented using the PyTorch library [39] and we used MLflow¹ for model governance by storing experiments meta-data and results.

We start by describing the datasets used in Sec. 4.1.1, followed by the expert models in Sec. 4.1.2.1, the aggregator model in Sec. 4.1.2.2, the student model in Sec. 4.1.2.3 and the baselines used in Sec. 4.1.3.

4.1.1 Datasets

To develop and test the framework, we used eight different datasets from payment service provider clients of Feedzai², referred to as A, B, ..., H. Each represents a different domain, and these datasets are from different clients or geographies. We will avoid disclosing too many details to preserve the confidentiality of the respective clients.

These datasets vary in the total number of transactions, frauds, and even the resultant fraud rate. As seen in Table 4.1, the most extensive dataset is C, with around 700M transactions, followed by D, with around 100M. The smallest dataset is A, with only 10M total transactions. As for the number of confirmed fraudulent transactions, despite only having around 60M total transactions, dataset B leads with around 300k frauds, and A is the one with both the lesser number of

¹<https://mlflow.org/>

²<https://feedzai.com/>

Table 4.1: Approximated # Transactions, # Frauds and fraud rates per dataset. Values are approximated due to confidentiality of the data.

	A	B	C	D	E	F	G	H
# Transactions	10 M	60 M	700 M	100 M	30 M	70 M	40 M	20 M
# Frauds	1 k	300 k	100 k	40 k	10 k	100 k	50 k	10 k
Fraud rate (%)	0.01	0.5	0.02	0.03	0.02	0.2	0.1	0.07

fraudulent transactions with around 1k, but also in terms of observed fraud rate, which is around 0.01%.

The time windows in which the datasets were collected can be seen in Fig. 4.1. Datasets A, C, D, and E have the largest time windows going from January 2022 until March 2023, totaling 15 months, while datasets B and G have only four months long time windows.

4.1.1.1 Pre-processing

In order to use the datasets, we had to pre-process them by removing null entries, uniformizing some categorical features across datasets, and normalizing numerical features using z-score, as defined by Eq. 4.1 where μ is the mean and σ is the standard deviation. This last step is crucial given that all models used are neural networks, which are sensitive to the input features' scale, and take advantage of having zero-centered data. From a practical point of view, and in light of the dimensions of the datasets used as previously seen in Table 4.1, we needed to use a distributed computing library, namely Apache Spark³, to pre-process all the datasets efficiently.

$$z = \frac{x - \mu}{\sigma} \quad (4.1)$$

³<https://spark.apache.org/>

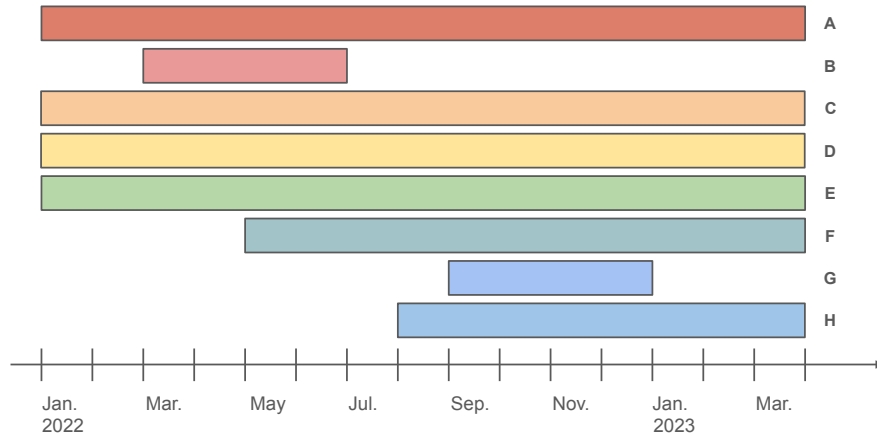


Figure 4.1: Time windows of the used datasets.

In total, we used 340 features, with 14 being categorical and 326 numerical features. Most numerical features were historical aggregations similar to what was mentioned in Sec. 2.1.1. For example, for a given transaction, we had features for the average amount of transactions made with the same card in the last hour, day, week, and month.

We split all datasets into training, validation, and test sets to correctly evaluate the models, as shown in Fig. 4.2. The training set was used to train the models, the performance on the validation set was used both to tune the hyperparameters of the models (Appendix A) and for early stopping, and the test set was used to evaluate the final performance of the models. Given the temporal nature of the data, for each dataset, the data for training had to be before the data for validation, which in turn had to be before the data for testing. This was done to avoid data leakage and ensure the models were not trained on future data. Furthermore, since we are handling a cross-domain scenario, we also ensured that the test sets contained transactions that happened after all the training and validation data, even from other domains, to ensure that even when evaluating in a cross-domain scenario, the models were not trained on future data. Unfortunately, given the datasets we had available, this was not possible for dataset B, for which we only had data until the end of June 2022. Therefore, we defined the general test set to start from December 1st, 2022, and used the November 2022 data as a validation set. Furthermore, given the deficient number of frauds in dataset A we also made an exception to the general rule by using a 2-month validation set starting in October 2022 for this dataset. This way, we ensured the validation set had enough frauds to evaluate the models correctly.

Since neural networks cannot handle raw categorical features, we used an embedding layer in all models to handle the categorical input features. This layer was used to map the categorical features into a continuous space, which is then concatenated with the numerical features and fed into the rest of the model. The size of the embedding layer for a categorical feature with cardinality c was chosen to be $2 \times \text{ceil}(\sqrt[4]{c})$ so that it was not as large as one-hot encoding but also not too small as to be able to capture the semantics of the feature.⁴

4.1.1.2 Balanced batches loading

Since we used neural network models, which we trained with Adam [29], a variant of stochastic gradient descent, we had to load the data in batches to the models. Given that the problem is a binary classification problem with highly imbalanced datasets referring to the fraud rates in table 4.1, we had to over-sample the positive class at the batch level to ensure that the models were not biased towards the negative class. However, we refrained from setting the fraud rate per batch to 50% as it would be in a perfectly balanced dataset since the metric we are interested in is the recall at a fixed false positive rate (FPR) of 1%, which is a very low FPR. Therefore, we set the fraud rate per batch to a value between 1% and 30% in all settings, which was tuned during the hyperparameter tuning (HPT) phase. This way, we ensured that the models were trained with a higher fraud rate than the natural one, but not too high, to avoid the models learning to predict

⁴<https://developers.googleblog.com/en/introducing-tensorflow-feature-columns>

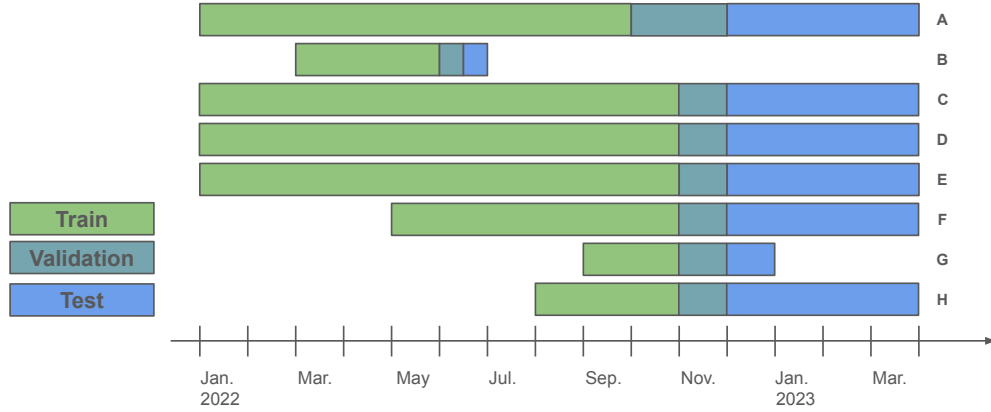


Figure 4.2: Data splits applied to the datasets. On each dataset, the training split is followed by the validation split and the test split.

fraud in a way that would not be useful at low FPR values. Given that we are using several large-scale datasets and they do not fit in the main memory, we had to implement custom loaders that read from disk for which we used the Webdataset⁵ format and the respective PyTorch [39] loaders.

4.1.1.3 Removing domain-dependent features

We experimented with all the available features and then experimented after removing two categorical features that were semantically domain-dependent, namely geographic-dependent. The two features were the currency used in the transaction and the origin country of the payment card. We thought the models could be using these features to over-fit the source domains and not generalize well to other domains, given that, for example, a particular currency in one domain might be the regular one. However, it might be an anomaly in other domains.

4.1.1.4 Data setups

As a general setup for the aggregator pre-training, student pre-training, and Meta-DMoE training, we used the data splits as shown in Fig. 4.3. We used the train splits of datasets A-F as train data, the validation split of dataset G as validation data, and dataset H’s test split as test data. This way, we ensured that the models were trained on data from different domains than the ones they were evaluated on, and there was no data leakage given the temporal nature of the data.

However, we knew that, ideally, we would also want several validation domains to have more robust model selection in the hyperparameter tuning and more robust results on the final evaluation. Therefore, given that the available datasets limited us, we decided to try a multi-domain validation setup using as validation the validation splits from the datasets whose train splits were used to train the models (Fig. 4.4). While using this setup, we also had to mask the corresponding expert M_e during validation. This way, we aimed to have a more robust validation process that would be more representative of the final evaluation.

⁵<https://github.com/webdataset/webdataset>

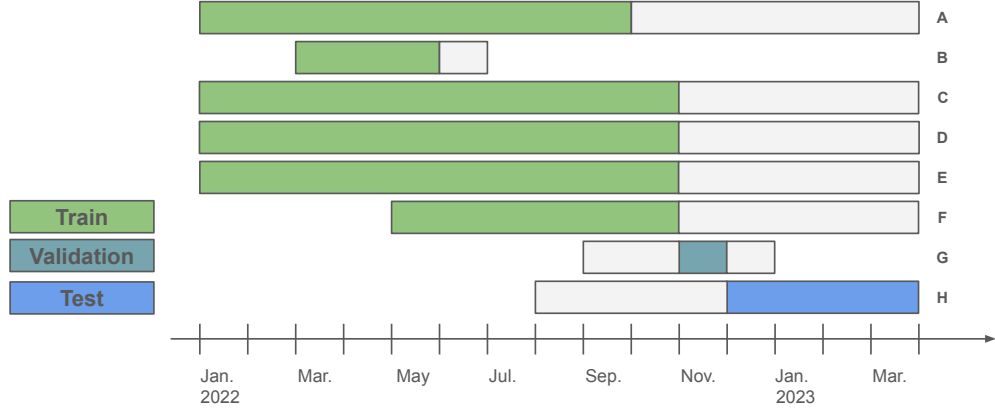


Figure 4.3: Data splits used for the aggregator pre-training, student pre-training, and Meta-DMoE training. The train splits of datasets A-F were used as train data, the validation split of dataset G was used as validation data and the test split of dataset H was used as test data.

The hypothesis behind trying this setup was that by using a single validation domain, we were selecting the model that would perform better on the specific validation domain in HPT. However, this does not mean it would be the most performant in the test domain. In fact, we could be aligning the model selection to a specific validation domain that is not necessarily aligned with the target domain. Therefore, we used multiple validation domains to aim for a more generalizable model that would perform better on the test domain. This way, we were trying to avoid overfitting to the validation domain and to have a more robust model selection process.

4.1.2 Meta-DMoE models

4.1.2.1 Experts

Since the original Meta-DMoE [74] method was designed for image data, which in turn allowed the usage of a pre-trained large state-of-the-art model, namely ResNet [23], to act as expert models, we had to adapt the expert models to the tabular data we were working with. To do so, we designed an architecture called a blocked neural network inspired by a ResNet-like architecture for tabular data from the literature [18]. This architecture comprises B residual blocks, each composed of a residual connection alongside a feed forward neural network composed of L layers. These layers comprise a linear transformation, an activation function, which introduces the non-linearity, a normalization layer, and dropout [55]. The described architecture can be seen in Fig. 4.5.

We assembled the expert models, M , using the blocked neural network introduced above. These expert models are a composition of two blocked neural networks: $M = M_c \circ M_e$. M_e receives the input features outputting an intermediate representation, and M_c is the classifier that receives the intermediate representation and outputs the final score of the model. The rationale behind this composition is to generalize the feature extractor and classifier head concept.

Given that the expert models training aims to train domain-specific M_e models, we defined the architecture of the classifier M_c to be the same across domains. We wanted to find a suitable

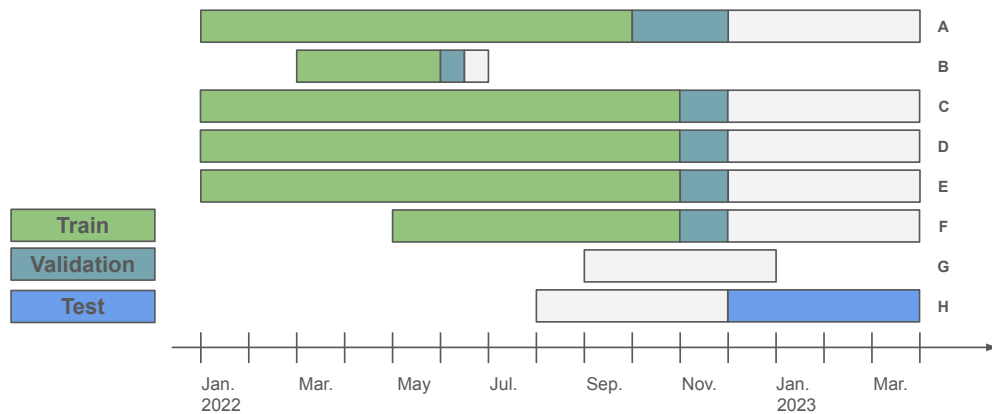


Figure 4.4: Data splits used for the multi-domain validation setup training. The train splits of datasets A-F were used as train data, the validation splits of datasets A-F were used as validation data, and the test split of dataset H was used as test data.

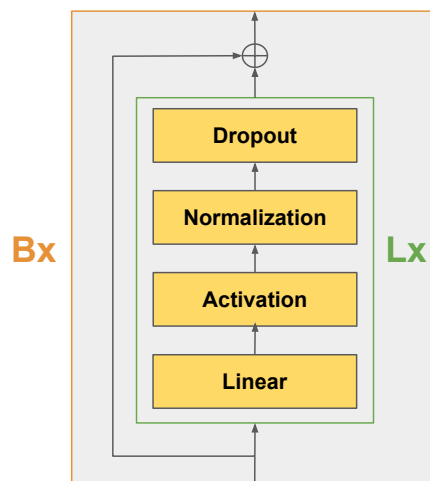


Figure 4.5: Blocked neural network diagram. The blocked neural network comprises B residual blocks, each composed of a residual connection alongside a feed forward neural network composed of L layers.

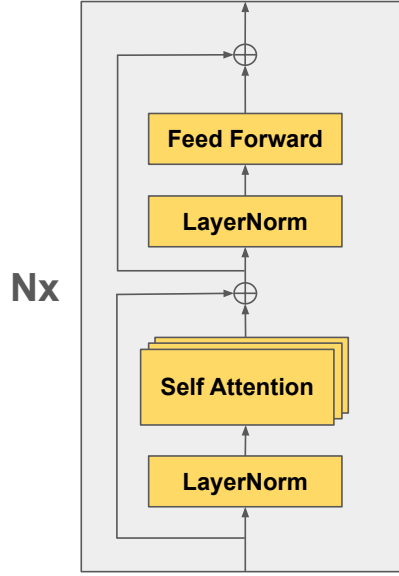


Figure 4.6: Aggregator composition diagram. The aggregator model A_e comprises N transformer blocks, each composed of a multi-head self-attention sub-block and multi-layer perceptron sub-block.

feature embedding that is informative for the task on a particular domain and generalizes as well as possible to other domains. It is unclear which layer is most generalizable a priori in a deep neural network [22]. Due to time limitations, we fixed the classification head M_c for all setups, hence the depth counting from the end at which we extracted the features, and only tuned the feature extraction network M_e during pre-training, fixing its output size to 64. Therefore, after some initial experiments on several domains, we defined the architecture of M_c to be used across domains as a blocked neural network with $B = 1$, $L = 2$, no dropout [55], a width of 16 and batch normalization [25].

4.1.2.2 Aggregator

Following the original Meta-DMoE application [74], we used a transformer encoder [60] as the aggregator model A_e so that the interconnections between different domain knowledge can be taken advantage of by the self-attention [60] mechanism. The transformer encoder comprises N transformer blocks, each composed of a multi-head self-attention sub-block and a multi-layer perceptron sub-block. Each sub-block is preceded by a layer normalization [4] and succeeded by a residual connection. The composition of the transformer encoder can be seen in Fig. 4.6.

4.1.2.3 Student

For the architecture of the student model $S = S_c \circ S_e$, we used the same blocked neural network architecture as we did for the expert models M . The student model S comprises two blocked neural

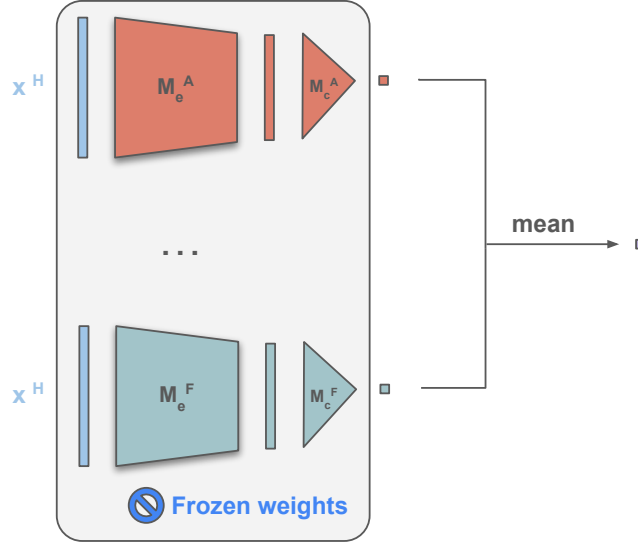


Figure 4.7: Mean ensemble baseline diagram. The mean ensemble model uses the output of all expert models M except the one trained on the domain being used as a test to calculate the mean output of the ensemble.

networks, S_e and S_c , where S_e is the feature extractor and S_c is the classifier. We also fixed the architecture of the classifier S_c and only tuned the architecture of the feature extractor S_e and made sure that the output dimensionality of A_e was the same as the output dimensionality of the student model S_e as to be able to compute an L2 loss between the outputs of the two models during the Meta-DMoE training.

4.1.3 Baselines

4.1.3.1 Mean ensemble

After evaluating each model M across all domains directly, we created a mean ensemble baseline model. For a given test set from domain D_T , we used the output of all expert models M except the one trained on D_T to calculate the mean output of the ensemble. This way, we evaluated the performance of the ensemble model on the test set of D_T without having to train anything since we just used the trained expert models M . The diagram of the mean ensemble baseline can be seen in Fig. 4.7.

4.1.3.2 Gated ensemble

Following the mean ensemble in Sec. 4.1.3.1, we decided to implement and test a gating mechanism [26], to see if we could improve the performance of the ensemble model by taking into account the transaction features in the weighting process of the expert models M . The gated ensemble has a gate model composed of a single linear layer followed by a softmax activation function and a set of expert models M . The input features are fed to the expert models M and the gate. The gate outputs a set of weights that are used to weight the expert models M outputs to

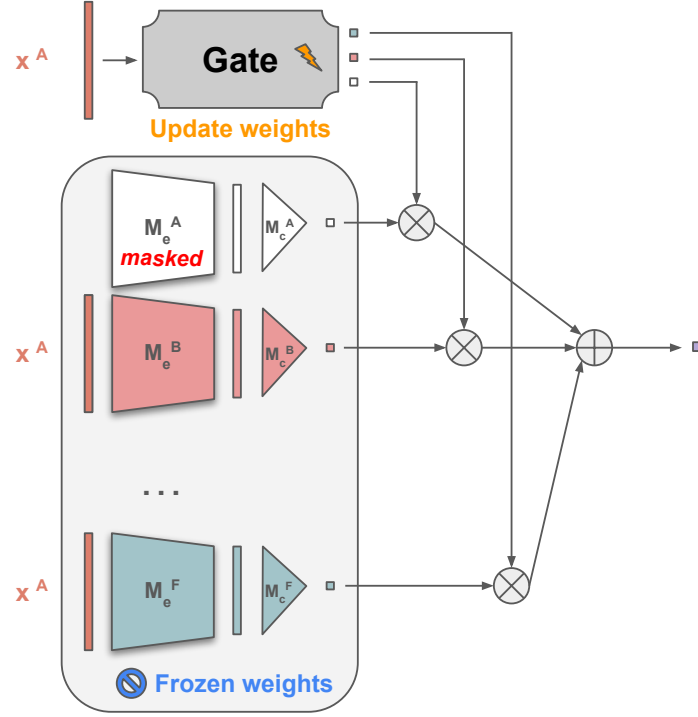


Figure 4.8: Training of the gated ensemble model. The gate model comprises a single linear layer and a softmax activation function. The input features are fed to the expert models M and the gate. The gate outputs a set of weights that are used to weight the expert models M outputs to produce the gated ensemble’s final output.

produce the gated ensemble’s final output. This way, we can see the gate as training the weights of a weighted average of the individual experts M .

In Fig. 4.8, we can see the training process emphasizing that we only update the gate weights during training, using the pre-trained expert models M . During training, we mask the expert model M output on the same domain as the current batch being used to train the gate model. Also, we had to force the softmax to output 0 as the value to multiply the masked expert output with, so that all the used gate outputs sum up to 1. This was done by setting the corresponding value to a large negative value before passing through the softmax function. During testing, we do not need to mask any expert model output, and we can use all outputs for the gating model.

4.1.3.3 Aggregator pre-training

Since the output of the aggregator model A_e is not a final single score value, but instead an aggregated embedding, we added a classifier head A_c that receives the output of A_e and outputs a final score in order to pre-train it. This classifier follows the same architecture as the expert model’s classifier M_c as described in Sec. 4.1.2.1.

The pre-training diagram of the aggregator can be seen in Fig. 4.9. During training, we randomly sample the training domains A-F. At a given iteration, we take a batch from the current domain and pass it to all M_e models. We proceeded by masking the output of the M_e that was

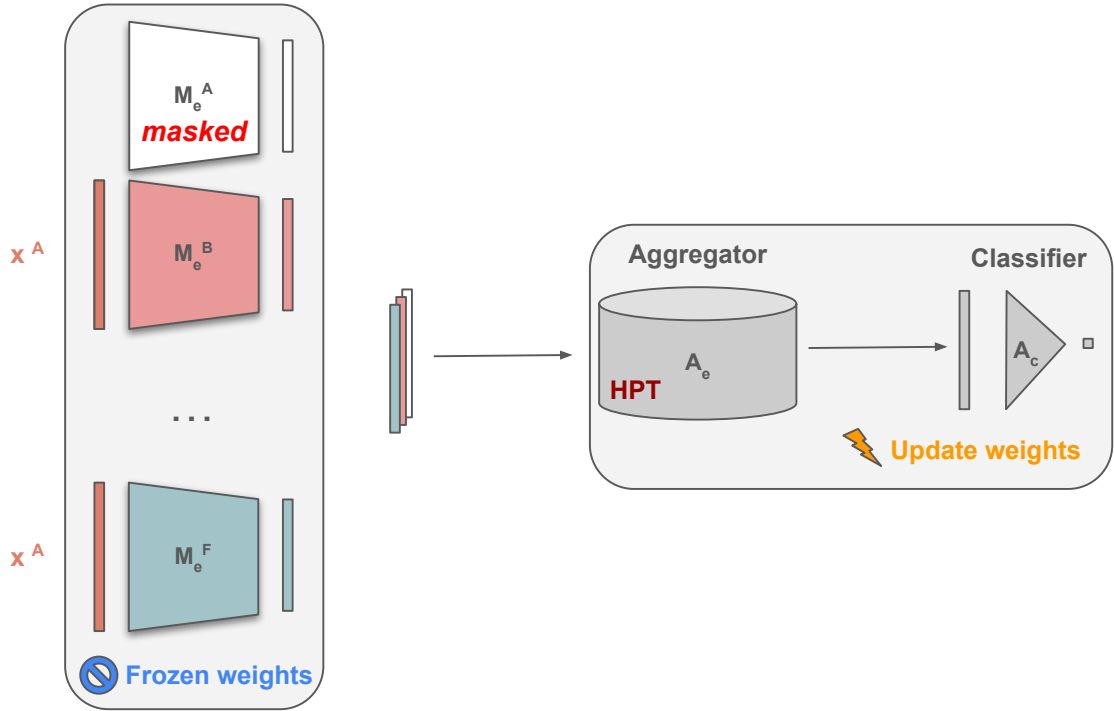


Figure 4.9: Pre-training of the aggregator model. The aggregator model is pre-trained using the expert models M_e outputs. During training, we cycle through the training domains A-F. At a given iteration, we take a batch from the current domain and pass it to all M_e models. We proceed by masking the output of the M_e that was trained on the domain where the batch came from, and after that, we stack all the M_e outputs and feed them to the aggregator.

trained on the domain where the batch came from, and after that, we stacked all the M_e outputs and fed them to the aggregator. Note that we are only performing hyperparameter tuning on the aggregator architecture. We do not have to mask any M_e output during validation and testing because we are not using source domains.

4.1.3.4 Student pre-training

We pre-trained the student model across all source domains by randomly sampling batches from all source train splits (A-F), while the expert models were trained on a single domain each. This way, we aimed to have a pre-trained student model that could generalize better to unseen domains than a random one or an individual expert model. Furthermore, this pre-trained model was also evaluated and used as a baseline.

4.2 Results

This section contains the main results and a discussion of them. The cross-domain evaluations of individual expert models can be seen in Sec. 4.2.1, and the results of the main method and baselines can be seen in Sec. 4.2.2. It is worth mentioning again that the model selection process

during hyperparameter tuning was based on the recall at a fixed false positive rate (FPR) of 1% metric. The results are presented in terms of this metric. However, we also present the area under the receiver operating characteristic curve (AUC) metric to provide a more comprehensive view of the models' performance.

4.2.1 Expert models cross-domain evaluations

With the eight individual expert models M trained each on the train split of different datasets (A-H), we evaluated them on the test splits of all datasets except for dataset C for a practical reason. As seen in Table 4.1, dataset C is larger than the 2nd largest by an order of magnitude, which would considerably delay our experiments if we evaluated it, so we decided to use the computation resources for the subsequent experiments instead.

As we can see in Table 4.2 the expert models perform well on the domain they were trained on but drop significantly on the other domains. This can be justified by fraud patterns specific to each domain that take time to change within each domain by concept drift. On the primary metric used, recall at a fixed FPR of 1%, for all except dataset E, the expert model that performed the best was the one trained on the same domain as the test domain. This behavior is expected given that the expert models are trained on a single domain and are not expected to generalize well to other domains. Even on dataset E, where the model trained on dataset C was the most performant, the difference was less than 1pp.

Despite the significant drop in the primary metric, by analyzing the AUC metric, we can see that the expert models are learning something useful even across domains. Except for 3 out of 56 evaluations, the AUC of the expert models was above 50%, which is the expected performance of a random classifier. This shows that there are common fraud patterns across domains, but each domain has specificities that make a naive transfer approach not a viable option.

On the last line of Table 4.2, we can see the performance of the mean ensemble model. This model uses all expert models except the one trained on the dataset being used as a test dataset. It does not consistently outperform the best out-of-domain expert model, such as in datasets D, E, F, and H. However, in the other datasets, the mean ensemble model outperforms the best out-of-domain expert model in some cases by a large margin, such as in dataset A, where the mean ensemble model outperforms the best out-of-domain expert model by 22.47pp on the primary metric.

As explained in Sec. 4.1.1.3, we removed two categorical features that were semantically domain-dependent, namely geographic-dependent, from the datasets and re-did the hyperparameter tuning and training of the expert models. The cross-domain evaluations of this second iteration can be seen in Table 4.3.

What we were expecting with this was that the in-domain performance would drop, but the out-of-domain performance would increase. This is because the models would be unable to rely on the geographic-dependent features to over-fit the source domains and would have to learn more generalizable features. However, as we can see in Table 4.3, the in-domain performance did drop apart from dataset B, where it slightly increased, but the out-of-domain performance also mostly

Table 4.2: Expert models and mean ensemble performance across datasets. Each row represents a model trained on a single domain, and each column represents a dataset. The mean ensemble uses all expert models except the one trained on the dataset being used as a test. In-domain performances are greyed out, the best performances on a dataset are in bold, and the second best is underlined.

Model	A		B		D		E		F		G		H	
	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC
A	51.69	83.02	0.45	54.23	4.2	69.49	0	58.91	2.51	63.35	4.97	58.89	4.64	67.19
B	2.25	73.75	49.13	94.42	4.88	76.91	9.17	65.44	<u>6.13</u>	<u>78.41</u>	5.43	72.6	5.79	74.28
C	<u>6.74</u>	<u>84.13</u>	3.36	79.24	<u>39.23</u>	<u>93.36</u>	48.62	88.36	4.71	71.27	5.54	73.62	<u>15</u>	<u>79.98</u>
D	3.37	85.57	<u>4.11</u>	74.89	65.52	95.41	14.68	75.43	2.49	61.79	<u>6.29</u>	72.08	8.25	64.6
E	2.25	71.87	0.09	63.19	2.99	65.67	<u>47.71</u>	<u>85.51</u>	1.13	52.97	0.77	57.02	1.46	66.63
F	<u>6.74</u>	60.32	1.28	52.98	1.26	60.07	0	44.47	42.66	89.59	3.48	76.88	0.37	59.72
G	3.37	26.89	0.62	51.8	1.54	59.64	1.83	78.95	0.21	71.08	37.47	90.76	2.33	69.1
H	0	76.33	2.7	76.72	33.29	84.8	0.92	46.48	4.46	68.84	1.01	62.89	44.41	90.89
Ensemble	29.21	88.08	6.44	81.29	17.64	91.04	6.42	88.39	5.29	77.24	10.03	78.02	11.27	78.67

dropped. Looking at the mean ensemble baseline in all datasets apart from E, where there was a 26.61pp increase on the primary metric, the performance degraded compared to the first iteration. This suggests that the removed features were not as geographic-dependent as we thought, that the models could not learn more generalizable features without them, or that the algorithm was already able to avoid overfitting to start with.

4.2.2 Meta-DMoE and baselines

The main results of the Meta-DMoE method, along with the baselines we defined in Sec. 4.1.3, namely mean ensemble, gated ensemble, aggregator and student pre-trained, can be seen in Table 4.4. All models on the table, except *Mean ensemble*, used as source domains datasets A-F, and as test domain the dataset H. The *Mean ensemble* model used all expert models except the one trained on dataset H, meaning it used information from the domain G, which other models that required validation did not use.

When comparing Meta-DMoE with the pre-trained student model, which we can see as its starting point, we can see that both AUC and recall at fixed FPR of 1% Meta-DMoE performed slightly better, but very little. We thought it would come closer to the aggregator pre-trained model, which acted as the teacher during the meta-training of Meta-DMoE, but it did not being one possible explanation for this the student model not having the learning capacity to detect patterns in such a variety of domains.

Given that we are dealing with datasets with very low fraud rates, and we are only using a single batch at test-time to perform the adaptation, we think that this might be hurting the adaptation process of Meta-DMoE, that in the original paper [74] was being applied to a multi-class classification problem with more balanced class distributions than the ones we used. Since at test-time, we cannot control the fraud rate as we do at training time, by using a batch size of 2048 and for a fraud rate of 0.1%, we will have, on average, two frauds per batch, which seems too small a number to be able to adapt the model well. Another possibility is that the general architecture

Table 4.3: Expert models and mean ensemble performance across datasets after removing domain-dependent features. Each row represents a model trained on a single domain, and each column represents a dataset. In-domain performances are greyed out, the best performances on a dataset are in bold, and the second best is underlined.

Model	A		B		E		F		G		H	
	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC	TPR@ 1%FPR	AUC
A	47.19	<u>84.01</u>	0.88	43.24	<u>24.77</u>	71.29	5.43	74.63	0.99	52.74	3.24	48.39
B	0	75.17	50.06	94.04	0.92	64.98	4.68	69.77	3.98	72.07	5.29	<u>75.72</u>
C	4.49	82.33	<u>2.59</u>	<u>74.79</u>	9.17	88	1.82	71.16	<u>5.54</u>	67.57	4.33	74.78
D	<u>6.74</u>	87.77	0.8	71.38	5.5	87.22	2.35	67.76	0	71.24	<u>7.87</u>	73.03
E	0	73.36	1.01	68.08	39.45	<u>87.49</u>	1.17	63.73	0	64.98	3.17	60.39
F	3.37	71.18	0.25	45.07	0	44.98	42.99	91.96	1.01	<u>73.88</u>	0	47.28
G	2.25	73.64	0.78	48.85	0	38.92	8.1	<u>78.62</u>	32.26	89.44	3.77	71.41
H	1.12	65.47	0.11	67.73	1.83	58.87	1.24	63.9	0.6	56.78	31.4	86.86
Ensemble	15.73	86.83	2.87	72.81	33.03	81.47	4.91	78.82	5.26	71.63	10.02	79.69

we defined for the student model is not appropriate for dealing with multiple domains, despite seeming appropriate for single domains, as we saw in the in-domain evaluations of Table 4.2.

By comparing the baselines, we can see that on the primary metric, recall at fixed FPR of 1%, the gated ensemble model performed the best at 14.97%, almost as good as the best out-of-domain expert model on the same test set which was 15%. In terms of AUC, the mean ensemble model without both the expert from domain G and H, *Mean ensemble₂*, performed the best, closely followed by the aggregator pre-trained model, both with AUCs above 79%.

An interesting point that partially motivated trying the multi-domain validation setup is that on the validation set, the recall at a fixed FPR of 1% was higher for the pre-trained student at 19.5%, than the pre-trained aggregator at 11.4%. Furthermore, the increase in performance from the student to the Meta-DMoE is more noticeable, reaching 21.97%, than on the test set.

When comparing the pre-trained aggregator with a single domain validation versus the multi-domain validation, i.e., *Aggregator₂*, we saw that the performance dropped on both AUC and recall at fixed FPR of 1% when using the multi-domain validation. However, note that as mentioned in Sec. 4.1.1.4, since the number of available datasets limited us, we used different splits from the same datasets used for training, which could be penalizing generalization. Furthermore, by calculating the metrics on the combination of datasets instead of separately while focusing on very low FPRs, we end up using a threshold purely based on the data points from one of the validation domains but apply it to all of them. This could be the reason for the drop in performance.

4.2.3 Summary

The individual expert models cross-evaluations show that each model performed well in the domain it was trained on, but its performance decreased significantly when applied directly to other domains, indicating that, while there are still common fraud patterns shared across domains, each domain has specific ones.

Table 4.4: Evaluation results of the Meta-DMoE method, mean and gated ensembles, aggregator, and student pre-trained models on the target dataset. The best results for each metric are in bold, and the second best is underlined.

	TPR@0.5%FPR	TPR@1%FPR	AUC
Mean ensemble	<u>7</u>	11.27	78.67
Mean ensemble ₂	6.04	11.95	79.3
Gated ensemble	9.55	14.97	76.23
Aggregator	6.22	<u>12.85</u>	<u>79.04</u>
Aggregator ₂	6.22	11.73	71.19
Student	5.7	9.18	72.06
Meta-DMoE	5.54	9.77	72.29

We also tested by removing some geographic-dependent features, given that we thought the models were overfitting to them, but the results did not confirm this hypothesis.

As for the Meta-DMoE [74] method and baselines, the gated ensemble was the most performant on the primary metric, recall at fixed FPR of 1%, while the mean ensemble had the best AUC. We also tried a multi-domain validation set for the aggregator HPT, Aggregator₂, hypothesizing that the model selection was incorrectly aligned with a single domain that could not be aligned with the target domain. This test did not confirm the hypothesis, but in this case, we identified one possible problem with how the validation metrics were calculated on the multi-domain validation set.

The Meta-DMoE negligibly improved the performance relative to the student model, and this also makes us wonder what would be the performance of the pre-trained aggregator on the original paper setup[74]. Nevertheless, changing the nature of the data from images to tabular and having such an unbalanced class distribution can be the reasons for the different outcomes.

Chapter 5

Conclusions

In this work, we adapted Meta-DMoE, a Computer Vision state-of-the-art multi-source unsupervised domain adaptation meta-learning method, to the problem of cross-domain fraud detection using real-world datasets. We also compared the Meta-DMoE method to several baselines, including deconstructions like the pre-trained aggregator, whose evaluation was lacking in the original paper, and others, such as the gated ensemble, which provides a learnable weighted ensemble.

In our problem setting, the Meta-DMoE method did not perform as well as some of the baselines, namely the gated ensemble, which outperformed the pre-trained transformer-encoder aggregator. This result was surprising, given that Meta-DMoE is considered state-of-the-art in computer vision tasks. Whether this discrepancy is due to the fundamental differences between the nature of Computer Vision data and data streams or if domain-specific characteristics influence it, such as class distribution, remains an open question. It is also interesting to note that the adaptation step on the new target data did not yield improvements in our case. This was unexpected, as one of the main strengths of Meta-DMoE, according to the authors, is the meta-learning of this adaptation step. Further investigation is needed to understand the factors behind these results.

The fact that the gated ensemble outperformed other methods is an interesting result by itself, and could inform future directions. For example, using the input features in the aggregation process of Meta-DMoE could potentially lead to better results. Given that the Meta-DMoE method does not use the input features in the original implementation, it would be interesting to explore how this would affect the results on our setup. Furthermore, the weighting of individual classifiers of the gated ensemble method can be seen as a dynamic classifier selection method, whose literature could be further explored [11].

On a more practical note, training the various baselines and model variations on eight large-scale datasets took quite some implementation (e.g., pre-processing using distributed computing libraries and implementing custom data loading from disk) and compute time. This experience was very valuable compared to the work on small benchmark datasets but also limited the number of experiments and methods we could execute during this work.

5.1 Future Work

Explore state of the art deep learning for tabular data

We used a simple feed-forward neural network as the expert and student models. However, several other recent architectures could be explored, such as the TabNet [3] and FT-Transformer [18] architectures. These architectures have shown promising results on tabular data, and it would be interesting to see how they perform in the context of multi-source domain adaptation as a whole and the context of the Meta-DMoE method in particular.

Recurrent expert models

The expert models are feed-forward neural networks in both the original Meta-DMoE implementation and the one presented in this work. However, it would be interesting to explore RNN-based expert models, as well as student models, to try to capture temporal dependencies in the data firstly and secondly, to see if we could remove the necessity of calculating manual historical aggregation features while maintaining or even increasing the performance of the framework.

Incorporate more datasets

We used a small set of datasets to evaluate the Meta-DMoE method. However, it would be interesting to incorporate more datasets to have a more standardized setup with multiple datasets for training, validation, and testing. This would allow for a more comprehensive evaluation of the method and a better understanding of its generalization capabilities and the designed baseline methods.

Tune the size of the intermediate feature embedding

After some preliminary experiments, we used an intermediate feature embedding size of 64 for the M_e , A_e , and S_e model outputs. However, empirically tuning this hyperparameter would be interesting to see if a different size would lead to better results. Furthermore, tuning the depth of the complete model from which this embedding would be extracted would also be interesting.

Explore label availability

We did unsupervised multi-source domain adaptation in the experiments presented, where the target domains had no labels. However, it would be interesting to explore how well each method would perform if we later fine-tuned them after the label delay period when labels are available. This would allow us to see which methods can generalize better and if the meta-learning process helped.

Incorporate input features into the aggregator

As we saw in the gated ensemble results, incorporating the input features into the aggregator model can lead to better results. It would be interesting to explore how this would affect the Meta-DMoE method and if it would also lead to better results for the aggregator pre-trained baseline. A simple way would be to concatenate the input features to the expert model outputs before passing them to the aggregator model.

Multi-domain batch

In the methods that iterated through several source domains, we used a data-loader that sampled a batch of data from a single domain. However, exploring a multi-domain data-loader that samples data from all source domains in a single batch would be interesting while incorporating the masking inside the batch. This would allow for every update to be based on data from all source domains, which could lead to better generalization.

Aggregator and student pre-training with MAML

We pre-trained both the aggregator and student models by randomly sampling batches from the train splits of the source domains. However, it would be interesting to explore pre-training the models using the MAML [17] framework, where the learning objective would be more aligned with the multi-domain adaptation task.

References

- [1] Aisha Abdallah, Mohd Aizaini Maarof, and Anazida Zainal. Fraud detection system: A survey. *Journal of Network and Computer Applications*, 68:90–113, 2016.
- [2] Jean V. Alves, Diogo Leitão, Sérgio Jesus, Marco O. P. Sampaio, Pedro Saleiro, Mário A. T. Figueiredo, and Pedro Bizarro. Fifar: A fraud detection dataset for learning to defer, 2023.
- [3] Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687, May 2021.
- [4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.
- [5] Ricardo Barata, Miguel Leite, Ricardo Pacheco, Marco O. P. Sampaio, João Tiago Ascensão, and Pedro Bizarro. Active learning for imbalanced data under cold start. In *Proceedings of the Second ACM International Conference on AI in Finance, ICAIF '21*, New York, NY, USA, 2022. Association for Computing Machinery.
- [6] Richard J. Bolton and David J. Hand. Statistical Fraud Detection: A Review. *Statistical Science*, 17(3):235 – 255, 2002.
- [7] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.
- [8] Zhengyu Chen and Donglin Wang. Multi-initialization meta-learning with domain adaptation. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1390–1394, 2021.
- [9] Ziliang Chen, Jingyu Zhuang, Xiaodan Liang, and Liang Lin. Blending-target domain adaptation by adversarial meta-adaptation networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2243–2252, 2019.
- [10] Yu Cheng, Mo Yu, Xiaoxiao Guo, and Bowen Zhou. Few-shot learning with meta metric learners, 2019.
- [11] Rafael M.O. Cruz, Robert Sabourin, and George D.C. Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41:195–216, 2018.
- [12] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey, 2017.
- [13] Andrea Dal Pozzolo, Olivier Caelen, Yann-Aël Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, 41(10):4915–4928, 2014.
- [14] Linda Delamaire, Hussein Abdou, and John Pointon. Credit card fraud and detection techniques: a review. *Banks and Bank systems*, 4(2):57–68, 2009.

- [15] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A brief review of domain adaptation. In Robert Stahlbock, Gary M. Weiss, Mahmoud Abou-Nasr, Cheng-Ying Yang, Hamid R. Arabnia, and Leonidas Deligiannidis, editors, *Advances in Data Science and Information Engineering*, pages 877–894, Cham, 2021. Springer International Publishing.
- [16] Yong Feng, Jinglong Chen, Zhuozheng Yang, Xiaogang Song, Yuanhong Chang, Shuiling He, Enyong Xu, and Zitong Zhou. Similarity-based meta-learning network with adversarial domain adaptation for cross-domain fault identification. *Knowledge-Based Systems*, 217:106829, 2021.
- [17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.
- [18] Yury Gorishniy, Ivan Rubachev, Valentin Khurlov, and Artem Babenko. Revisiting deep learning models for tabular data, 2023.
- [19] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, Jun 2021.
- [20] Hao Guan and Mingxia Liu. Domain adaptation for medical image analysis: A survey. *IEEE Transactions on Biomedical Engineering*, 69(3):1173–1185, 2022.
- [21] ChengCheng Han, Zeqiu Fan, Dongxiang Zhang, Minghui Qiu, Ming Gao, and Aoying Zhou. Meta-learning adversarial domain adaptation network for few-shot text classification, 2021.
- [22] Zhi Han, Siquan Yu, Shao-Bo Lin, and Ding-Xuan Zhou. Depth selection for deep relu nets in feature extraction and generalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4):1853–1868, 2022.
- [23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.
- [25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.
- [26] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.
- [27] Taewon Jeong and Heeyoung Kim. Ood-maml: Meta-learning for few-shot out-of-distribution detection and classification. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3907–3916. Curran Associates, Inc., 2020.
- [28] Bingyi Kang and Jiashi Feng. Transferable meta learning across domains. In *Conference on Uncertainty in Artificial Intelligence*, 2018.
- [29] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

- [30] Naeimeh Laleh and Mohammad Abdollahi Azgomi. A taxonomy of frauds and fraud detection techniques. In Sushil K. Prasad, Susmi Routray, Reema Khurana, and Sartaj Sahni, editors, *Information Systems, Technology and Management*, pages 256–267, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [32] Jun won Lee and Christophe Giraud-Carrier. Predicting algorithm accuracy with a small set of effective meta-features. In *2008 Seventh International Conference on Machine Learning and Applications*, pages 808–812, 2008.
- [33] Da Li and Timothy Hospedales. Online meta-learning for multi-source and semi-supervised domain adaptation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 382–403, Cham, 2020. Springer International Publishing.
- [34] Jing Li, Peng Han, Xiangnan Ren, Jilin Hu, Lisi Chen, and Shuo Shang. Sequence labeling with meta-learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):3072–3086, 2023.
- [35] Jun Li, Chaolu Feng, Xiaozhu Lin, and Xiaohua Qian. Utilizing gcnn and meta-learning strategy in unsupervised domain adaptation for pancreatic cancer segmentation. *IEEE Journal of Biomedical and Health Informatics*, 26(1):79–89, 2022.
- [36] Shuang Li, Wenxuan Ma, Jinming Zhang, Chi Harold Liu, Jian Liang, and Guoren Wang. Meta-reweighted regularization for unsupervised domain adaptation. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):2781–2795, 2023.
- [37] Hong Liu, Mingsheng Long, Jianmin Wang, and Yu Wang. Learning to adapt to evolving domains. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22338–22348. Curran Associates, Inc., 2020.
- [38] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019.
- [39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [40] Raghavendra Patidar, Lokesh Sharma, et al. Credit card fraud detection using neural network. *International Journal of Soft Computing and Engineering (IJSCE)*, 1(32-38), 2011.
- [41] Trung Phung, Trung Le, Tung-Long Vuong, Toan Tran, Anh Tran, Hung Bui, and Dinh Phung. On learning domain-invariant representations for transfer learning with multiple sources. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan,

- editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27720–27733. Curran Associates, Inc., 2021.
- [42] José Pombal, André F. Cruz, João Bravo, Pedro Saleiro, Mário A. T. Figueiredo, and Pedro Bizarro. Understanding unfairness in fraud detection through model and data bias interactions, 2022.
 - [43] Kun Qian and Zhou Yu. Domain adaptive dialog generation via meta learning. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2639–2649, Florence, Italy, July 2019. Association for Computational Linguistics.
 - [44] Shuhao Qiu, Chuang Zhu, and Wenli Zhou. Meta self-learning for multi-source domain adaptation: A benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1592–1601, 2021.
 - [45] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients, 2019.
 - [46] Adriano Rivolli, Luís P.F. Garcia, Carlos Soares, Joaquin Vanschoren, and André C.P.L.F. de Carvalho. Meta-features for meta-learning. *Knowledge-Based Systems*, 240:108101, 2022.
 - [47] Vinicius Facco Rodrigues, Lucas Micol Policarpo, Diógenes Eugênio da Silveira, Rodrigo da Rosa Righi, Cristiano André da Costa, Jorge Luis Victória Barbosa, Rodolfo Stoffel Antunes, Rodrigo Scorsatto, and Tanuj Arcot. Fraud detection and prevention in e-commerce: A systematic literature review. *Electronic Commerce Research and Applications*, 56:101207, 2022.
 - [48] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 213–226, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
 - [49] Yusuf Sahin, Serol Bulkan, and Ekrem Duman. A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40(15):5916–5923, 2013.
 - [50] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA, 20–22 Jun 2016. PMLR.
 - [51] Amr Sharaf, Hany Hassan, and Hal Daumé III. Meta-learning for few-shot NMT adaptation. In Alexandra Birch, Andrew Finch, Hiroaki Hayashi, Kenneth Heafield, Marcin Junczys-Dowmunt, Ioannis Konstas, Xian Li, Graham Neubig, and Yusuke Oda, editors, *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 43–53, Online, July 2020. Association for Computational Linguistics.
 - [52] Vivek Sharma, Naila Murray, Diane Larlus, M. Saquib Sarfraz, Rainer Stiefelhagen, and Gabriela Csurka. Unsupervised meta-domain adaptation for fashion retrieval. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1347–1356, 2021.

- [53] Peeyush Singhal, Rahee Walambe, Sheela Ramanna, and Ketan Kotecha. Domain adaptation: Challenges, methods, datasets, and applications. *IEEE Access*, 11:6973–7020, 2023.
- [54] Wade A. Smith and Robert B. Randall. Rolling element bearing diagnostics using the case western reserve university data: A benchmark study. *Mechanical Systems and Signal Processing*, 64-65:100–131, 2015.
- [55] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.
- [56] Shiliang Sun, Honglei Shi, and Yuanbin Wu. A survey of multi-source domain adaptation. *Information Fusion*, 24:84–92, 2015.
- [57] Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation, 2020.
- [58] Joaquin Vanschoren. Meta-learning. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Automatic Machine Learning: Methods, Systems, Challenges*, pages 39–68. Springer, 2019.
- [59] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.
- [60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS’17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.
- [61] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5385–5394, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.
- [62] Hemanth Venkateswara and Sethuraman Panchanathan. *Introduction to Domain Adaptation*, pages 3–21. Springer International Publishing, Cham, 2020.
- [63] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, Jun 2002.
- [64] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [65] Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance, 2023.
- [66] G. Wei, C. Lan, W. Zeng, and Z. Chen. Metaalign: Coordinating domain alignment and classification for unsupervised domain adaptation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16638–16648, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society.

- [67] Tianyuan Yang, Tang Tang, Jingwei Wang, Chuanhang Qiu, and Ming Chen. A novel cross-domain fault diagnosis method based on model agnostic meta-learning. *Measurement*, 199:111564, 2022.
- [68] Wanqi Yang, Chengmei Yang, Shengqi Huang, Lei Wang, and Ming Yang. Few-shot unsupervised domain adaptation via meta learning. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2022.
- [69] Runzhe Zhan, Xuebo Liu, Derek F. Wong, and Lidia S. Chao. Meta-curriculum learning for domain adaptation in neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14310–14318, May 2021.
- [70] Jin Zhang, Xi Zhang, Yanyan Zhang, Yexin Duan, Yang Li, and Zhisong Pan. Meta-knowledge learning and domain adaptation for unseen background subtraction. *IEEE Transactions on Image Processing*, 30:9058–9068, 2021.
- [71] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: Learning to adapt to domain shift. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 23664–23678. Curran Associates, Inc., 2021.
- [72] Penghao Zhang, Jiayue Li, Yining Wang, and Judong Pan. Domain adaptation for medical image segmentation: A meta-learning method. *Journal of Imaging*, 7(2), 2021.
- [73] Yu Zhang, Dongying Han, Jinghui Tian, and Peiming Shi. Domain adaptation meta-learning network with discard-supplement module for few-shot cross-domain rotating machinery fault diagnosis. *Knowledge-Based Systems*, 268:110484, 2023.
- [74] Tao Zhong, Zhixiang Chi, Li Gu, Yang Wang, Yuanhao Yu, and Jin Tang. Meta-dmoe: Adapting to domain shift by meta-distillation from mixture-of-experts. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22243–22257. Curran Associates, Inc., 2022.

Appendix A

Hyperparameter tuning

In all individual hyperparameter tuning (HPT) processes we performed 60 total trials being the first 50 random search and the last 10 using the TPE sampler [65].

A.1 Experts

We performed 8 separate HPT processes, one for each domain. The hyperparameters that were tuned and the respective options can be seen in Table A.1. The best hyperparameters found for each domain can be seen in Table A.2 as well as the performance of each model M in terms of the recall at a fixed FPR of 1% (TPR@1%FPR) on the validation set.

A.2 Aggregator

We followed the same approach as we did for the expert models regarding not performing tuning on the classifier model and only tuning the aggregator model A_e . The hyperparameters that were tuned and the respective options can be seen in Table A.3. Also, in the same table, the hyperparameters that yielded the model with best performance in terms of the recall at a fixed FPR of 1%

Hyperparameter	Options	Best student
Dropout	{0, 0.05, 0.1, 0.15, 0.20, 0.25, 0.3, 0.35, 0.4}	0.35
#Blocks	{1, 2, 3}	3
Block width	{64, 128, 192, 256}	64
#Layers per block	{1, 2, 3, 4}	4
Normalization	{Batch, Layer}	Layer
Batch size	{512, 1024, 2048}	2048
Fraud rate	[0.01, 0.3]	0.099
Learning rate	[1e-4, 1e-2]	4.06e-4

Table A.1: Hyperparameters tuned for the M_e models. The best hyperparameters found for each domain are shown in Table A.2, while the best for the student model are also shown.

Hyperparameter	A	B	C	D	E	F	G	H
Dropout	0.35	0.15	0.25	0.05	0.3	0.0	0.25	0.3
#Blocks	3	2	1	2	3	2	3	3
Block width	256	192	192	256	64	128	128	64
#Layers per block	4	4	3	4	3	4	3	3
Normalization	Batch	Layer	Layer	Batch	Layer	Layer	Layer	Layer
Batch size	512	2048	512	2048	512	2048	2048	512
Fraud rate	0.012	0.099	0.212	0.029	0.015	0.093	0.016	0.018
Learning rate	1.38e-4	2.96e-3	1.32e-4	3.69e-3	7.23e-3	3.62e-3	4.55e-3	6.02e-3
TPR@1%FPR	50.00	43.00	75.36	69.88	67.57	55.04	51.80	36.45

Table A.2: Best hyperparameters found for the M_e models. The performance of each model in terms of the recall at a fixed FPR of 1% (TPR@1%FPR) on the validation set is also shown.

(TPR@1%FPR) on the validation set can be seen. The validation performance history of the HPT process can be seen in Figure A.1.

A.3 Student

The hyperparameters that were tuned and the respective options can be seen in Table A.1. Also, in the same table, the hyperparameters that yielded the model with best performance in terms of the recall at a fixed FPR of 1% (TPR@1%FPR) on the validation set can be seen. The validation performance history of the HPT process can be seen in Figure A.2.

A.4 Meta-DMoE

The hyperparameters that were tuned and the respective options can be seen in Table A.4. Also, in the same table, the hyperparameters that yielded the model with best performance in terms of the recall at a fixed FPR of 1% (TPR@1%FPR) on the validation set can be seen. The validation performance history of the HPT process can be seen in Figure A.3.

Hyperparameter	Options	Best
Depth (N)	{1, 2, 3}	2
#Heads	{1, 4, 8, 16}	1
MLP dimension	{64, 128, 192, 256}	192
Dropout	{0, 0.05, 0.1, 0.15, 0.20, 0.25, 0.3, 0.35, 0.4}	0.15
Fraud rate	[0.01, 0.3]	0.269
Learning rate	[1e-4, 1e-2]	8.46e-3

Table A.3: Hyperparameters tuned for the A_e model. The best hyperparameters found are also shown.

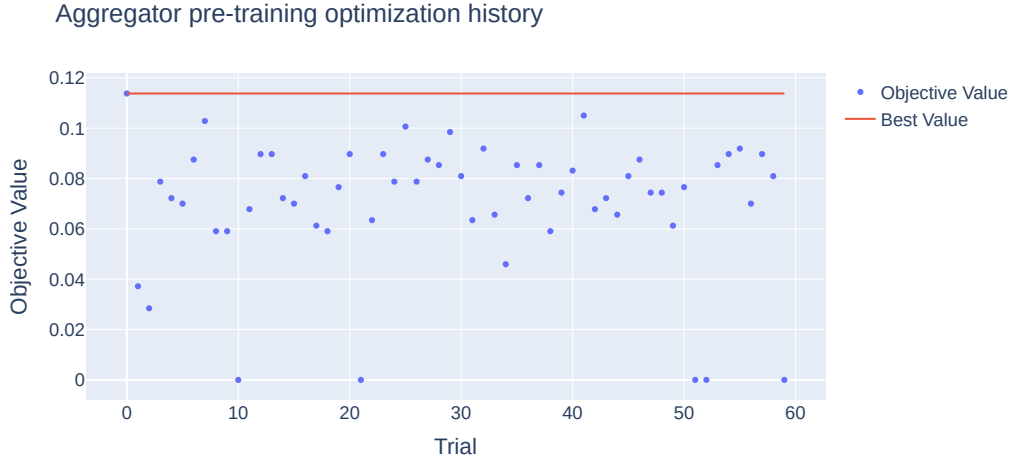


Figure A.1: Aggregator pre-training optimization history. Objective value was defined as the TPR@1%FPR on the validation set.

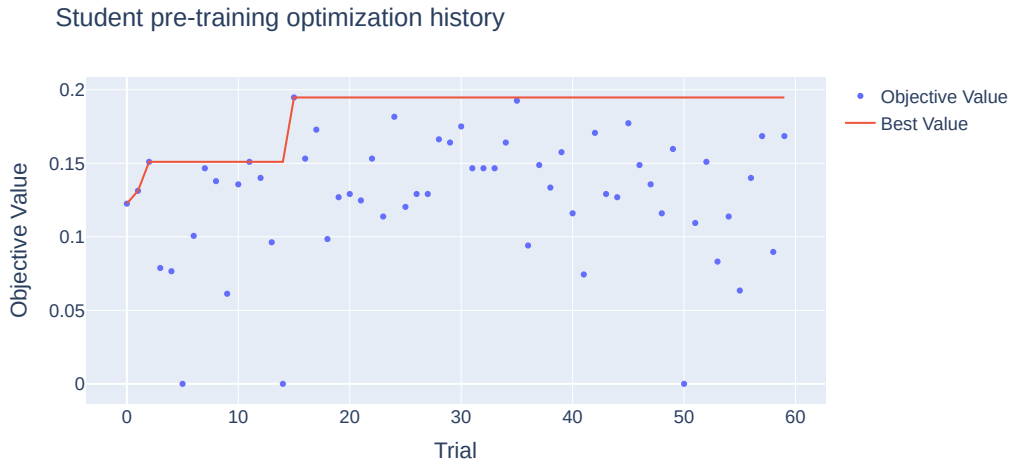


Figure A.2: Student pre-training optimization history. Objective value was defined as the TPR@1%FPR on the validation set.

Hyperparameter	Options	Best
Inner-adaptation learning rate	[1e-10, 1e-6]	5.14e-7
Support size	{128, 256, 512, 1024}	256
Fraud rate	[0.01, 0.3]	0.13
Teacher learning rate	[1e-10, 1e-3]	4.06e-5
Student learning rate	[1e-10, 1e-3]	1.17e-6

Table A.4: Hyperparameters tuned for the training of Meta-DMoE. The best hyperparameters found are also shown.

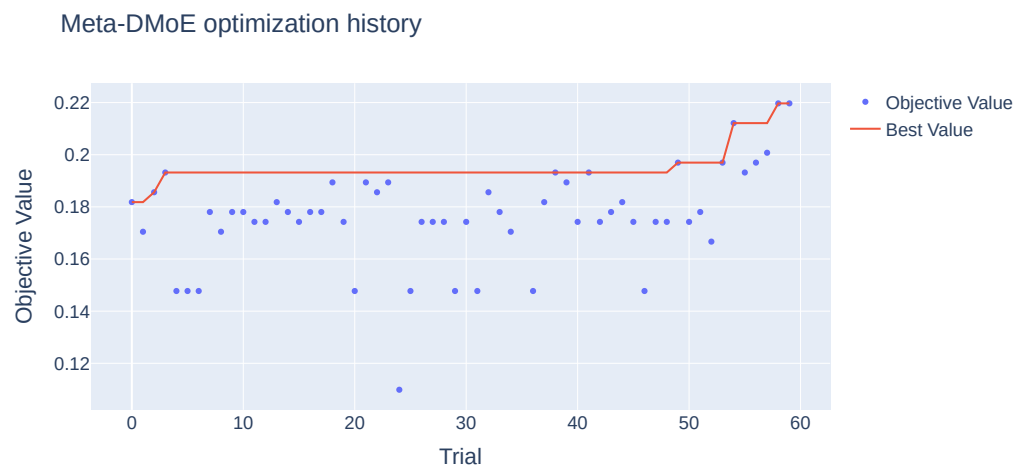


Figure A.3: Meta-DMoE optimization history. Objective value was defined as the TPR@1%FPR on the validation set.