Incremental Tabular Learning on Heterogeneous Feature Space

HANMO LIU, Hong Kong University of Science and Technology (Guangzhou), China
SHIMIN DI*, Hong Kong University of Science and Technology, China
LEI CHEN, Hong Kong University of Science and Technology (Guangzhou), China

Recently, incremental learning has attracted a lot of interest in both research communities and industries. Generally, given a series of data sets sequentially, it tries to achieve good performance on the new data set while maintaining not bad performance on the old ones. Despite the recent success of incremental learning, existing works mainly assume that the coming data set is from the feature space of old ones, i.e., homogeneous feature space. And they adopt one feature extractor to forcibly project different feature spaces into one space. However, this assumption is hard to hold in real-world scenarios. Especially, the attributes of tables may sequentially increase in tabular learning. Thus, classic incremental learning models may hinder their effectiveness. In this paper, we propose a new method, incremental tabular learning on heterogeneous feature space (ILEAHE) to solve this issue. We first propose the ideas that feature extractors should be decomposed into shared and specific extractors to process the shared and specific features across different data sets respectively. Then, we propose a novel measurement named discriminative ability to measure specific extractors. Thus, two kinds of extractors can be discriminated and the specific extractor will more focus on those domain-specific features. We further demonstrate the effectiveness of ILEAHE through empirical studies.

CCS Concepts: • **Computing methodologies → Supervised learning by classification**; **Online learning settings**.

Additional Key Words and Phrases: Incremental Learning, Transformer, Tabular Data

ACM Reference Format:
Hanmo Liu, Shimin Di*, and Lei Chen. 2023. Incremental Tabular Learning on Heterogeneous Feature Space. *Proc. ACM Manag. Data* 1, 1, Article 18 (May 2023), 18 pages. https://doi.org/10.1145/3588698

1 INTRODUCTION

Incremental Learning [5, 35], which is also called Continual Learning or Lifelong Learning, targets the problem of designing a model that can sequentially adapt to new data sets, especially the model still achieves good performance on old ones.

For instance, given images of cats and dogs, the image classification model first learns how to classify them. Then the model needs to handle new coming data sets, like images of birds and horses, and then elephants and fish [51]. Different from multi-class or multi-task learning which learn all the data sets simultaneously, incremental learning assumes that the new data sets sequentially appear, especially old ones are not available when facing the new one. However, deep learning models would bias heavily towards new data sets and lose the memory of old ones, which may

*Corresponding Author.
Authors' addresses: Hanmo Liu, Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China, hliubm@connect.ust.hk; Shimin Di*, Hong Kong University of Science and Technology, Hong Kong SAR, China, sdiaa@connect.ust.hk; Lei Chen, Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China, leichen@ust.hk.

2836-6573/2023/5-ART18 $15.00
https://doi.org/10.1145/3588698

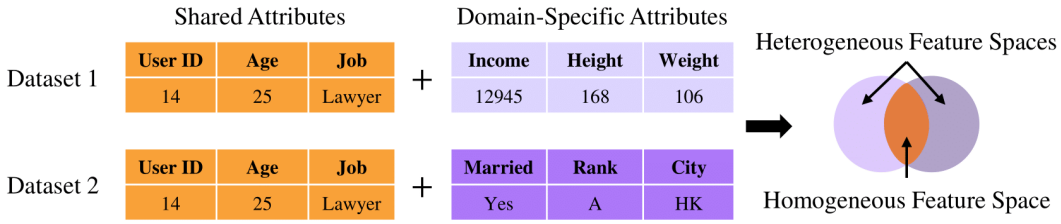


Fig. 1. The shared and specific attributes of different tabular data sets form heterogeneous feature spaces.

harm the effectiveness. This challenge is known as *Catastrophic Forgetting* [14]. Besides, as new data sets continuously come, the incremental learning models also need to alleviate the efficiency issue by carefully balancing the time cost for optimizing and model complexity for handling large number of data sets.

To solve the above challenges, there has been a substantial number of works in recent years, which can be categorized: model adaptation models [1, 12, 13, 31, 38, 42], memory replay models [4, 19, 20, 28, 29, 37, 43, 45] , and regularization-based models [6, 11, 16, 18, 21, 25]. Model adaptation models specialize the parameters or add supportive structures to accommodate new data sets. Memory replay models store representative data or parameters in old ones to persevere prior knowledge. When new data comes, the stored data or parameters will be utilized to train the model. Regularization-based models restrict the extent of forgetting by introducing the regularization term into the loss function. Some regularization terms keep the output of updated model on the old data sets consistent with the output of old model on the old data sets, while other kinds try to identify and maintain important parameters of old data. Despite the diverse angles of dealing with the forgetting problem, these models generally employ one feature extractor θ to extract features from all data sets [32]. They generally assume that all incremental data sets can be described by the same feature space.

However, it is natural that different data sets come from various feature spaces. Especially, in real-world tabular learning scenarios, it is often encountered that the attributes of the table sequentially increase [3]. For example, when granting loans to users based on their personal profiles, the profiles may only have the credit records in the beginning and contain purchasing behaviors later caused by the change of collected information. For the scenario of recommending services to users of a bank, besides the basic personal data, different related data could be included corresponding to various services. The formation of heterogeneous feature spaces for tabular data sets is illustrated in Fig. 1. Unfortunately, it is hard to apply existing incremental solutions with one feature extractor to such heterogeneous feature spaces. One feature extractor will force heterogeneous feature spaces to be projected into the same space, which leads to the specific knowledge of data sets lost.

To solve the issue on heterogeneous space, we propose a new incremental learning method for tabular scenarios, named *I*ncremental tabular *LEA*rning on *HE*terogeneous feature space (ILEAHE). Intuitively, although feature spaces are heterogeneous, various data sets still have some minor correlations under the incremental setting, while others are some domain-specific features. Thus, we propose to split the feature extractor θ into shared extractor θ_s and domain-specific extractors θ_{sp} to capture the shared and domain-specific features correspondingly. To distinguish the shared and specific features, we follow the intuition that the shared features should be able to support all data sets and specific features should be especially effective for their own data set. Considering that simply declaring the specific extractors can not fulfill the objective, we introduce the novel discriminability score to evaluate to what extent the domain-specific feature extractor θ_{sp} performs better on its corresponding data set than the others. By maximizing the discriminability score, we can ensure that each θ_{sp} extracts domain-specific features that can best represent the corresponding

data set. During training, θ_s will be trained for all the data sets, while a separate θ_{sp} will be built and trained for each data set. Afterwards, when generating predictions, the two branches of features will be classified in parallel with their own classifiers, and the outputs will be weighted averaged as the final prediction. For backward propagation, we use the cross entropy loss to learn the new data set. Especially, we apply discriminative loss, which incorporates the discriminability score, to enhance θ_{sp} of the new data set. Then for the old data sets, we maintain the performance of θ_s on them through regularization loss. Our contributions are summarized as follows:

- In this paper, to relax the homogeneous assumption in existing incremental learning models, we develop a new structure of incremental learning that implements the shared and specific feature extractors to handle data sets with heterogeneous feature spaces.
- To automatically identify domain-specific features, we propose a novel discriminability score to measure the capability of specific extractors. Then, the specific extractor will more focus on those features that can help the model to achieve good performance on the given data set.
- The empirical studies demonstrate the effectiveness and efficiency of our method on tabular data sets. Especially, compared with the joint learning method that uses all data sets for training, the proposed method still can achieve comparable results when old data sets are inaccessible.

2 RELATED WORKS

Incremental learning aims to learn new data sets continuously, and make the model's performance on all learnt data sets competitive. Such settings are similar to dynamic learning [22, 23] and transfer learning [7, 8, 34]. In general, dynamic learning and transfer learning can access past or existing data, but incremental learning cannot.

Formally, the model $f(\cdot)$ has learnt n data sets $\{(X^1, Y^1), (X^2, Y^2), \ldots, (X^n, Y^n)\}$ and is receiving the new data set (X^{n+1}, Y^{n+1}). In general, the deep learning model $f(X^i) = \varphi \circ \theta(X^i)$ can be decomposed into consecutively two parts, feature extractor θ and classifier φ, where φ gives prediction based on the features from θ. In this paper, we use $\tilde{f} = \tilde{\varphi} \circ \tilde{\theta}$ and $\hat{f} = \hat{\varphi} \circ \hat{\theta}$ to represent the model before $n+1$'s increment and the model after $n+1$'s increment, respectively. The loss function evaluating the model performance on old data sets are noted as L^o and the one for the new data set is L^e. Following the setting of incremental learning, when processing the increment on X^{n+1}, the model can only access (X^{n+1}, Y^{n+1}) and it should achieve the following goal:

$$\hat{\theta}, \hat{\varphi} = \arg\min_{\theta,\varphi} L^e \left(\varphi \circ \theta(X^{n+1}), Y^{n+1}\right) \tag{1}$$

$$s.t.\ L^o \left(\hat{\varphi} \circ \hat{\theta}(X^i), Y^i\right) \leq L^o \left(\tilde{\varphi} \circ \tilde{\theta}(X^i), Y^i\right)\ \forall\ i \leq n. \tag{2}$$

Under above formulation, we can optimize and update the model on the given new data set X^{n+1} by optimizing the upper level problem. Generally, the upper level objective follows the classic classification task to instantiate the loss function $L^e(\cdot)$, like cross entropy loss: $L^e(\hat{y}^{n+1}, y^{n+1}) = -\sum_k y_k^{n+1} \log(\hat{y}_k^{n+1})$, where y^{n+1} is the ground truth and $\hat{y}^{n+1}$ is the prediction by the model $\hat{f}$ after $n+1$ increment. The lower level objective maintains the effectiveness of updated model $\hat{\varphi} \circ \hat{\theta}(\cdot)$ on old data sets (X^i, Y^i) $(i \leq n)$ by constraining the loss of updated model no larger than that of model before updating $\tilde{\varphi} \circ \tilde{\theta}(\cdot)$. But it is hard to preserve old performance in Eq. (2) because old data sets are usually inaccessible during incremental setting, which leads to the *Catastrophic Forgetting* problem. Three types of methods have been proposed to solve this problem: model adaptation models, memory replay models and regularization-based models.

Model Adaptation This branch of methods focus on adjusting the model structure or modifying parameters to learn different data sets. PNN [39] and Expert Gate [1] train a separate model for each new data set. ACL [12] builds shared and domain-specific feature extractors, and uses adversarial

Table 1. Summary on Important Notations.

Symbols	Meanings
x, X	x is an entry of data set X
$f(X)$	the model predicting on X
X^i, Y^i	ith data set and label
$\tilde{f}, \tilde{\theta}, \tilde{\varphi}$	model with extractor and classifier before learning new data
$\hat{f}, \hat{\theta}, \hat{\varphi}$	model with extractor and classifier after learning new data
$\tilde{y}, \hat{y}$	prediction from $\tilde{f}$ and $\hat{f}$

learning method to make shared features invariant to different data sets. MUC [27] uses multiple auxiliary classifiers to make better prediction. For earlier works about parameter modification, important parameters of old data sets are frozen by layer [31] or predicting pathway [13], and the less important ones are reset to learn new ones. They are improved in later models [38, 42] by applying various kinds of masks to parameters to achieve more flexible and more thorough effects. But methods specializing parameters can hardly handle a large number of increments. Additional structures grow the model capacity, which is inevitable when learning numerous data sets. [12].

Memory Replay Memory Replay methods deal with the forgetting problem by replaying old knowledge when learning the new data set. They replace the old data of the constraint Eq. (2) by the stored memory to meet the incremental learning requirement. Let (M^n, Y_M^n) denote the memory of old data set, Eq. (2) is then transformed into:

$$L^o\left(\hat{\varphi} \circ \hat{\theta}(M^n), Y_M^n\right) \leq L^o\left(\tilde{\varphi} \circ \tilde{\theta}(M^n), Y_M^n\right). \tag{3}$$

iCaRL [37] keeps M^n by storing exemplars of old data sets and uses the nearest mean representations of data sets to make classification. GEM [29] believes minimizing L^o on M^n introduces overfitting problem, so that the gradients from M^n are only used to avoid negative optimization on old data sets. GEM is then improved by [4, 45] on efficiency and effectiveness. Besides, [19, 20, 28, 43] replay old knowledge without storing raw data. But storing data faces the data privacy issue or ineffective utilization [40], and reserving parameters adds more complexity to the model. In this paper, we avoid storing data for a more practical scenario.

Regularization-Based This kind of methods confront forgetting by including a regularization term. Learning without Forgetting (LwF) [25] first introduces the distillation loss [15], which encourages the prediction distribution on old data sets to be consistent after learning new ones, to preserve the old ones' knowledge. Instead of minimizing the difference between the model outputs and the ground truth at Eq. (2), LwF proposes the distillation loss to minimizing the difference between old output $\tilde{y} = \tilde{f}(x^{n+1})$ and new output $\hat{y} = \hat{f}(x^{n+1})$ as:

$$L_{LwF}^o(\hat{y}, \tilde{y}) = -\frac{1}{n}\sum\nolimits_{i=1}^{n}\sum\nolimits_{k=1}^{|C^i|} q(\tilde{y}_k)\log q(\hat{y}_k), \tag{4}$$

where C^i represents different classes of the old data set X^i. The normalization function $q(\cdot)$ is defined as: $q(a_k) = (a_k)^{1/T}/\sum_{k'=1}^{|C^i|}(a_{k'})^{1/T}$, where T is a manually specified hyper-parameter. The key idea of this distillation loss is to preserve the predicting behavior of the previous model for maintaining the old performance. In addition to the prediction, more distillation mechanisms are introduced in later works to better resist forgetting, like based on interpretability [6], feature [16] or parameter [11].

Besides distillation, there are methods updating parameters elastically to preserve old performance while learning new data sets. EWC [21] estimates the importance of parameters after learning

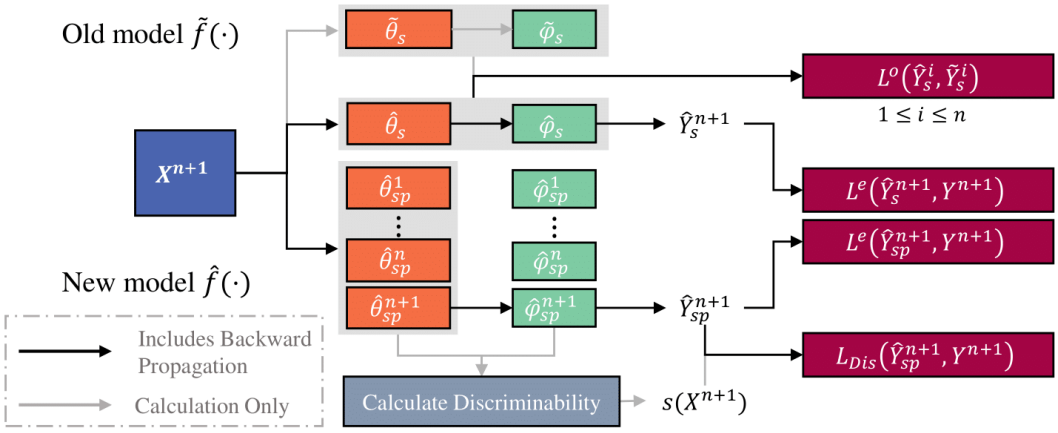


Fig. 2. The overview of our model. Cross entropy loss L^e is applied to shared and domain-specific outputs to learn new data sets. Regularization loss L^o is applied to shared extractors $\tilde{\theta}_s$ and $\hat{\theta}_s$ for preventing forgetting old knowledge. Discriminative loss L_{Dis} is utilized to help specific extractors $\hat{\theta}_{sp}^{n+1}$ based on discriminability score s.

the old data sets, and punish the changes made to the parameters with high importance when optimizing on a new one. The estimation is approximated by the diagonal of the old parameters' Fisher Information Matrix. $L^o(\cdot)$ can be expressed as:

$$L_{EWC}^o(\hat{f}, \tilde{f}) = \frac{1}{2} \sum_{w=1}^{|\tilde{f}|} \Omega_w (\tilde{f}_w - \hat{f}_w)^2, \tag{5}$$

where w represents the model parameter, the old model has a total of $|\tilde{f}|$ parameters, and Ω_w is for the corresponding importance information. After learning data set X^{n+1}, Ω_w is defined as: $\Omega_w = \mathbb{E}_{(X^{n+1}, Y^{n+1})} \left[(\partial L / \partial w)^2 \right]$, where L refers to the total loss of learning X^{n+1}. Following EWC, Liu et al. [26] improved the approximation by rotating the parameter space. SI [50] also proposes an online estimation method that updates the importance knowledge during training the new data set. However, regularization-based methods may not effectively maintain the performance when data sets vary too much and could prevent the model from learning new knowledge [33]. In this paper, we improve the regularization-based methods by introducing discriminative ability.

3 METHODOLOGY

In this section, we present the incremental tabular learning on heterogeneous feature space in detail. We formulate the problem definition in Sec. 3.1 and explain ILEAHE in Sec. 3.2 and Sec. 3.3.

The overview of ILEAHE has been shown in Fig. 2. ILEAHE uses shared and domain-specific feature extractors, θ_s and θ_{sp}, to handle the shared and domain-specific parts of heterogeneous feature spaces. Similar to classic machine learning methods, both parts are trained by minimizing cross entropy losses $L^e(\hat{y}_s^{n+1}, y^{n+1})$ and $L^e(\hat{y}_{sp}^{n+1}, y^{n+1})$. But θ_s faces the forgetting problem as it incrementally learns new data sets. Thus, we keep an old copy of shared extractor $\tilde{\theta}_s$ and use regularization loss $L^o(\hat{y}_s^i, \tilde{y}_s^i)$, $i \le n$, to prevent the newly learnt θ_s from forgetting. For θ_{sp}, we use $L_{Dis}(\hat{y}_{sp}^{n+1}, y^{n+1})$ to help extract domain-specific features more effectively.

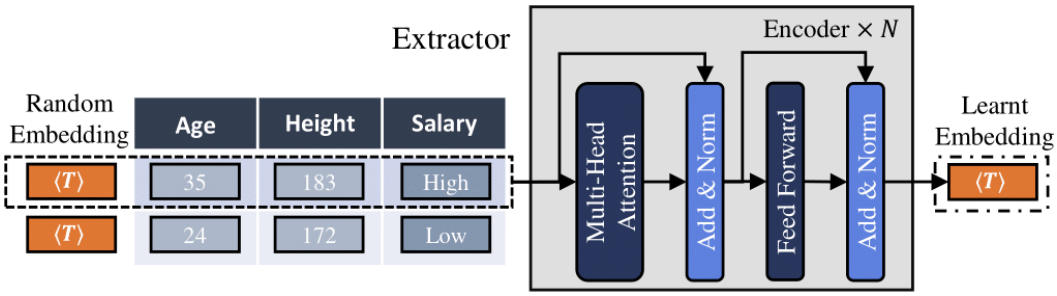


Fig. 3. Illustration on the feature extraction process. Each line of tabular data will be encoded by the Transformer. Then the embedding vector $\langle T \rangle$ will be regarded as the representation of input data.

3.1 Problem Formulation

As in the examples raised before, different data sets have heterogeneous feature spaces. If we treat them homogeneously and use a shared feature space to make prediction, the information from different data sets will mix with each other and disturb the inference [41]. Besides, the shared feature space can not represent all the data sets. Especially when the incoming data sets have small correlation with the existing ones, the shared feature space could fail to describe either data sets. Thus, to guarantee the model's effectiveness on all learnt data sets, it is necessary to consider different feature spaces heterogeneously. To achieve this purpose, an additional domain-specific feature space should be considered when learning each data set, along with the shared one. In this paper, we split the extractor θ into the shared extractor θ_s and domain-specific extractor θ_{sp}. Then the incremental learning problem is reformulated as:

$$\hat{\theta}_s, \hat{\theta}_{sp}^{n+1}, \hat{\varphi} = \underset{\theta_s, \theta_{sp}^{n+1}, \varphi}{\arg\min} L^e \left(\varphi \circ (\theta_s \circ \theta_{sp}^{n+1})(X^{n+1}), Y^{n+1} \right) \tag{6}$$
$$s.t.\ L^o \left((\hat{\varphi}_s \circ \hat{\theta}_s)(X^i), Y^i \right) \leq L^o \left((\tilde{\varphi}_s \circ \tilde{\theta}_s)(X^i), Y^i \right) \forall\ i \leq n,$$

where superscripts of θ_{sp} note the correspondence between θ_{sp} and its data set. After comparing the original problem (Eq. (1), (2)) with above formulation, we can observe that existing solutions use one feature extractor across all data sets, i.e., assuming that all features are in one shared space. In our definition, we try to optimize the shared extractor θ_s and specific extractor θ_{sp} for new coming data (X^{n+1}, Y^{n+1}) in upper level, and maintain the model performance on old data sets that built on the shared feature space. It is more reasonable to clearly distinguish the shared and domain-specific parts in the incremental scenario.

3.2 Feature Extractor for Tabular Data and Discriminative Ability

3.2.1 Feature Extractors. Following the problem formulation, we use shared extractor θ_s and domain-specific extractors θ_{sp} to extract features from any given transaction (line) of tabular data $x^i \in X^i$. We first append a token $\langle T \rangle$, a randomized embedding vector, to each transaction data x^i. We hope the token $\langle T \rangle$ can be optimized and learnt during the training phase, thus $\langle T \rangle$ can represent the input data x^i. But the number of attributes may vary in different tabular data sets, i.e., the length of x^i may be different for $i \in \{1, \cdots, n+1\}$. Inspired by SAINT [44], we regard each data transaction x^i as a sequence and employ a promising method Transformer[47] as feature extractor θ_s and θ_{sp} to encode x^i, noted as $\theta_s(x^i)$ and $\theta_{sp}(x^i)$. We omit positional embedding as the spatial correlation of attributes provides little help for prediction. The configuration of feature extractor is illustrated in Fig. 3. Overall, we input the randomized token together with data transaction x^i to the

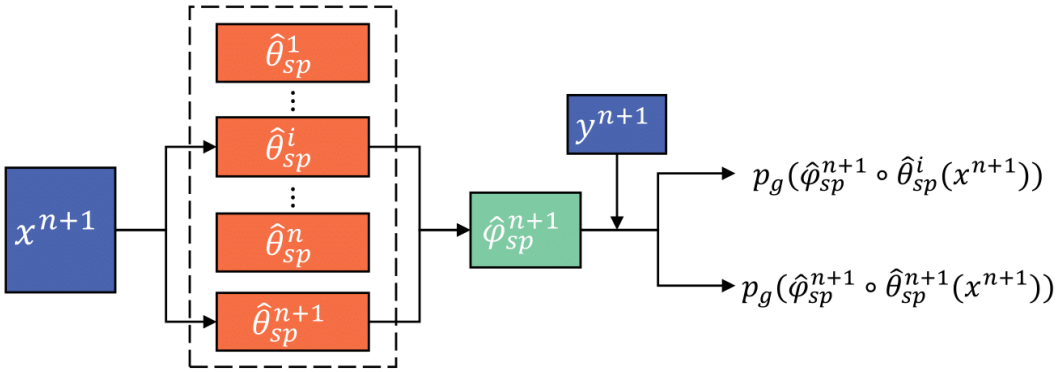


Fig. 4. Calculate the discriminative ability $p_g(\cdot)$ of specific extractor $\hat{\theta}_{sp}^{i}$.

self-attention model and output the representation of x^i (learnt embedding). Note that the attributes may be categorical and numerical in real-world data sets. In this paper, categorical attributes are embedded with an attribute-specific dictionary, while numerical attributes are embedded using a 2-layer perceptron.

3.2.2 Discriminative Ability. Intuitively, the shared features should be effective for all the learnt data sets, while the domain-specific features should be especially effective to the corresponding data set. This requires θ_{sp} to be highly discriminative to its own data set from the others by effectiveness. We quantify the effectiveness as the predicted possibility on the ground truth p_g, since a model is effective if it gives the correct prediction with high confidence. With regards to the extractors, effectiveness is defined as p_g based on the features they extract. Because only new data sets' labels are available at each increment, this method is applicable only to the new ones. Following this definition, we further introduce a score to reflect the discriminability of $\hat{\theta}_{sp}^{n+1}$ by comparing its effectiveness with the old specific extractor $\tilde{\theta}_{sp}^{i}, i \leq n$. As old data sets' specific feature spaces differ from the new one, $\tilde{\theta}_{sp}^{i}$ cannot extract meaningful features from X^{n+1}, which then defects the predictions based on it. By treating the predictions from $\tilde{\theta}_{sp}^{i}$ as bottom lines, we can tell if $\hat{\theta}_{sp}^{n+1}$ is more effective or not. The discriminability score s is thus defined as the subtraction between the p_g based on $\hat{\theta}_{sp}^{n+1}$ and the maximum p_g based on $\{\tilde{\theta}_{sp}^{i}\}_{i=1}^{n}$:

$$s(\hat{\theta}_{sp}^{n+1}; x^{n+1}) = p_g(\hat{\varphi}_{sp}^{n+1} \circ \hat{\theta}_{sp}^{n+1}(x^{n+1})) - \max_{i \leq n} \left\{ p_g(\hat{\varphi}_{sp}^{n+1} \circ \tilde{\theta}_{sp}^{i}(x^{n+1})) \right\},$$

where $x^{n+1} \in X^{n+1}$. The figure illustrating how to calculate p_g is in Fig. 4. Under this definition, large positive s indicates that $\hat{\theta}_{sp}^{n+1}$ is highly discriminative, while small or even negative s means $\hat{\theta}_{sp}^{n+1}$ is less effective, compared with domain-unrelated feature extractors. The maximum p_g from $\{\tilde{\theta}_{sp}^{i}\}_{i=1}^{n}$ is selected to raise a strict requirement on the discriminability of $\hat{\theta}_{sp}^{n+1}$.

3.3 Framework

In this subsection, we concretely introduce three loss functions as shown in Fig. 2. In Sec. 3.2.1, we introduce how to extract features from the input data X^{n+1}, e.g., $\hat{\theta}_s(x^{n+1})$ and $\hat{\theta}_{sp}^{n+1}(x^{n+1})$. After extraction, these features will be inputed into classifiers $\hat{\varphi}_s$ and $\hat{\varphi}_{sp}^{n+1}$ correspondingly and generate predictions $\hat{y}_s^{n+1} = \hat{\varphi}_s \circ \hat{\theta}_s(x^{n+1})$ and $\hat{y}_{sp}^{n+1} = \hat{\varphi}_{sp}^{n+1} \circ \hat{\theta}_{sp}^{n+1}(x^{n+1})$. Then, the final prediction $\hat{y}^{n+1}$ is

weighted sum of $\hat{y}_s^{n+1}$ and $\hat{y}_{sp}^{n+1}$ as:

$$\hat{y}^{n+1} = \alpha \hat{y}_s^{n+1} + (1-\alpha)\hat{y}_{sp}^{n+1}, \tag{7}$$

where α is a trade-off hyper-parameter. Note that the predictions for old data sets X^i, where $i \leq n$, are in the same manner. We next introduce three losses presented in Fig. 2.

Cross Entropy Loss L^e: Following the classic setting, we use cross entropy loss L^e to optimize shared and domain-specific parameters on the new data set:

$$L^e(\hat{y}_s^{n+1}, y^{n+1}) = -y^{n+1} \log \hat{y}_s^{n+1}, \tag{8}$$

$$L^e(\hat{y}_{sp}^{n+1}, y^{n+1}) = -y^{n+1} \log \hat{y}_{sp}^{n+1}, \tag{9}$$

where we discard the notation w.r.t. classes for simplicity.

Regularization Loss L^o: Apart from optimizing the new data set X^{n+1}, the model should also retain the old ones' performance. We use the regularization loss L^o to achieve this goal. Two alternatives of L^o, distillation loss L^o_{LwF} (see Eq. (4)) and EWC loss L^o_{EWC} (see Eq. (5)), can be employed. For L^o_{LwF}, it restores the prediction behavior of $\hat{\varphi}_s \circ \hat{\theta}_s$ to $\tilde{\varphi}_s \circ \tilde{\theta}_s$, so as to prevent forgetting. L^o based on L^o_{LwF} is as following:

$$L^o(\hat{y}_s, \tilde{y}_s) = L^o_{LwF}(\hat{\varphi}_s \circ \hat{\theta}_s(x^{n+1}), \tilde{\varphi}_s \circ \tilde{\theta}_s(x^{n+1})) \ . \tag{10}$$

When applying L^o_{EWC} that prevents forgetting by restricting the important parameters of old data sets from changing, L^o becomes following, where outputs are substituted by the corresponding models' parameters:

$$L^o(\hat{y}_s, \tilde{y}_s) = L^o_{EWC}(\hat{\varphi}_s \circ \hat{\theta}_s, \tilde{\varphi}_s \circ \tilde{\theta}_s). \tag{11}$$

Discriminative Loss L_{Dis}: In Sec 3.2, we define discriminability score s to measure how $\hat{\theta}_{sp}^{n+1}$ extracts more effective features than domain-unrelated extractors. Discriminative loss L_{Dis} utilizes this score to enhance $\hat{\theta}_{sp}^{n+1}$. It is defined as

$$L_{Dis}(\hat{y}_{sp}^{n+1}, y^{n+1}) = -\exp(-\gamma \cdot s(x^{n+1})) \cdot y^{n+1} \log(\hat{y}_{sp}^{n+1}), \tag{12}$$

where γ is a hyper-parameter and class notations are discarded. Under this definition, minimizing L_{Dis} requires both high p_g and large s, which reflects high discriminability of $\hat{\theta}_{sp}^{n+1}$.

In summary, when training on X^{n+1}, we use L^e to optimize the performance of $\hat{\theta}_s$ and $\hat{\theta}_{sp}^{n+1}$. For $\hat{\theta}_s$, the shared features should also be effective in the old data sets, which is ensured by the regularization loss L^o. On the other hand, $\hat{\theta}_{sp}^{n+1}$ should extract features especially effective in X^{n+1}, which is achieved by adding discriminative loss L_{Dis}. The final loss function can be writen as following.

$$L = L^e + \beta_1 L^o + \beta_2 L_{Dis}, \tag{13}$$

where β_1 and β_2 are factors controlling the weights of L^o and L_{Dis}.

4 EXPERIMENTS

In this section, Sec. 4.1 first introduces experiment setups. Sec. 4.2 presents the main experiments. Sec. 4.3. studies the robustness of the model towards data pre-processing settings. Sec. 4.4 analyzes the effectiveness and efficiency of ILEAHE compared with baselines. Then, Sec. 4.5 further shows the ablation studies of different components of our model. And Sec. 4.6 demonstrates the hyperparameter sensitivity. At last, Sec. 4.7 visualizes the features learnt by shared and specific extractors.

Table 2. The statistics of data sets. *Categorical* and *Numerical* are refer to categorical and numerical features, *Positive Rate* is the ratio of data with positive classes.

Name	Size	# Categorical	# Numerical	Classes	Positive Rate(%)
Bank	45,211	9	7	2	11.70
Shoppers	12,330	2	15	2	15.47
Income	32,561	8	6	2	24.08
BlastChar	7,043	17	3	2	26.54
Shrutime	10,000	4	6	2	20.37
Volkert	58,310	0	147	10	NA

Table 3. With considering the heterogeneous feature space, the following table compares our model with the baseline models. The relative advantages of our model are stated for each type of comparison. n refers to the number of data sets to learn.

Models	Test Effectiveness	Training Efficiency	Model Capacity	Require Previous Data
Ord-Joint	Best	$O(n^2)$	$O(1)$	Yes
Joint	Best	$O(n^2)$	$O(n)$	Yes
LwF	Bad	$O(n)$	$O(1)$	No
EWC	Medium	$O(n)$	$O(1)$	No
AFEC	Good	$O(n)$	$O(1)$	No
PCL	Bad	$O(n)$	$O(1)$	No
DMC	Bad	$O(n)$	$O(1)$	No
ACL	Bad	$O(n)$	$O(n)$	No
PNN	Good	$O(n)$	$O(n^2)$	No
MUC	Good	$O(n)$	$O(n)$	No
ILEAHE	Best	$O(n)$	$O(n)$	No

4.1 Experiment Setup

4.1.1 Dataset. We perform experiments on six data sets, whose characteristics are shown in Tab. 2. These data sets are publicly available at UCI, Kaggle, and AutoML (automl.chalearn.org/data). Conventionally, the input data sets of incremental learning belong to different classes, thus their feature spaces naturally differ from each other. This applies to the volkert data set, which has 10 classes. But as the first five data sets have only two classes, we can not synthesize sub data sets with regards to classes. Instead, we will randomly select various sets of attributes to form the sub data sets. Thus, even though the sub data sets belong to the same classes, the feature spaces are still heterogeneous. To further reduce the similarities among sub data sets, each of them randomly samples 65% of the original data set for training and 20% for testing. During training, we sequentially learn three sub data sets for the first five data sets. For volkert, we learn 2 classes at a step and in total 5 steps, where each sub data sets' attributes are randomly assigned as well.

4.1.2 Baselines. The adopted baselines are summarized in Tab. 3. And concrete implementations are listed as below.

- LwF [25] is our vanilla baseline. Use only a shared extractor to extract features from different data sets. L^o_{LwF} is applied to prevent forgetting at each increment.
- EWC [21] is another vanilla baseline. It identifies the important parameters for old data sets and restrict them from changing when learning the new one, so as to prevent forgetting.

- ACL [12] has a similar structure to ours, which also considers the shared and domain-specific feature spaces separately. ACL uses adversarial loss to maintain the effectiveness of θ_s to all data sets. We implemented their method without data replay, which corresponds to other baseline methods.
- MUC [27] is built upon LwF or EWC. Besides regularization, they use auxiliary classifiers to predict more effectively. MUC includes an unrelated data set to avoid the auxiliary classifiers being identical to each other. We implement MUC based on L^o_{LwF} and L^o_{EWC}, which are noted as MUC-LwF and MUC-EWC.
- PNN [39] builds a new branch of model for each new data set. Especially, the old models are laterally connected to the new model to assist training.
- AFEC [48] prevents forgetting via L^o_{EWC}. Besides, AFEC trains a supplementary model that learns new data set only, and by which let the major model actively forget redundant old knowledge.
- PCL [17] builds a classifier for each one of the incoming classes. It employs gradient regularization and knowledge transfer metrics to ensure the effectiveness of all the classifiers. But PCL does not release the code, thus no pre-trained model is available for our data sets. Without pre-trained models, the reproduction leads to unsatisfactory performance.
- DMC [51] trains a separate model for the new data set, then distills it and the old model into one finalized model. During distillation, DMC utilizes an external but domain-related data set to neutralize the finalized model from new and old data sets. Because each one of our data sets belongs to a different domain, this requirement is hard to satisfy and leads to lower performance.

In addition, we compare our model in incremental learning scenario with joint training paradigm.

- Joint-training (Joint) model uses the same shared-specific structure as ours and can access all the old data at each increment, which perfectly satisfies the constraint of Eq. (6). Thus it has the upper bound performance with respect to incremental learning setting. At the step $n+1$, all the learnt data sets, including X^{n+1}, will be the input and the model will jointly optimize on them.
- Ordinary Joint-training (Ord-Joint) model uses only the shared parameter to conduct joint training.

When conducting joint training, L^o and L_{Dis}, which are meant to approach the constraint Eq. (6), will be removed and only L^e used.

4.1.3 Evaluation Metrics. Following [12, 18, 29], we first use the average AUC (AAUC) to evaluate the models' performance on the classification task. Higher AAUC reflects better overall performance of the model during increments. Besides, we use average forgetting (AF) to show how much the model forgets old data set X^i after learning the subsequent data sets from X^{i+1} to X^{n+1}: $AF = {}^1/_n \sum_{i=1}^{n} (R_{i,i} - R_{n+1,i})$, where $R_{u,v}$ represents the tested AUC of data set X^v on u-th increment. Smaller AF means the model maintains previous performance better.

4.1.4 Implementation Details. We implement ILEAHE with PyTorch [36] and the experiments are run on a single RTX 2080 GPU. Our code is available at Github [1]. We first embed inputs with vectors of dimension 8. The Transformer encoder is employed as the feature extractor θ. The shared extractor is composed of 4 encoder layers with 2 attention heads, while the specific extractors reduce the encoder layers to 2. After feature extraction, fully connected (FC) layers will be applied as the classifier φ. We use 2 FC layers to generate prediction, the dimensions of layers are 8×64 and 64×2 respectively. The optimizer is AdamW [30]. We use hyperopt [2] to tune the hyperparameters and details are in Github.

[1] https://github.com/liuhanmo321/ILonHeteroFeatrueSpace

Table 4. The experiment results of our model and baselines. They are evaluated based on AAUC and AF and the reported results are averaged for 4 runs. For AAUC, the best result is noted by **Bold** and the second best is noted by Underline. Because the Joint methods serve as upper-bound performance and are not facing forgetting problem, they are excluded from comparison notations and AF measurements.

Model		AAUC							AF						
Category	Model Name	bank	blastchar	income	shoppers	shrutime	volkert	overall	bank	blastchar	income	shoppers	shrutime	volkert	overall
Joint	Ord-Joint	0.8639	0.8488	0.9102	0.9098	0.8207	0.9640	0.8862	Not Available						
	Joint	0.8635	0.8492	0.9104	0.9059	0.8200	0.9617	0.8851							
Regular-ization	LwF	0.8278	0.8364	0.8803	0.8943	0.7976	0.9232	0.8599	0.0490	0.0270	0.0462	0.0162	0.0222	0.0484	0.0348
	EWC	0.8548	0.8413	0.9070	0.9002	0.8104	0.9593	0.8788	0.0065	0.0071	0.0026	0.0045	0.0062	0.0071	0.0056
	AFEC	0.8560	0.8406	0.9064	0.9023	0.8125	0.9615	0.8799	0.0037	0.0025	0.0015	0.0034	0.0017	0.0040	0.0028
	PCL	0.5285	0.5107	0.4231	0.5104	0.4508	0.5285	0.4920	-0.0239	-0.0015	0.0342	-0.0061	0.0421	-0.0300	0.0024
	DMC	0.7064	0.7872	0.8436	0.7834	0.7309	0.6090	0.7434	0.0821	0.0293	0.0417	0.0888	0.0545	0.3044	0.1001
Model Adaptation	ACL	0.8436	0.8184	0.8910	0.8792	0.7957	0.9583	0.8644	0.0212	0.0347	0.0270	0.0408	0.0239	0.0062	0.0256
	PNN	**0.8629**	0.8439	**0.9090**	0.8979	0.8140	0.9631	0.8818	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	MUC-LwF	0.8229	0.8249	0.8696	0.8886	0.7861	0.9232	0.8525	0.0268	0.0294	0.0596	0.0158	0.0209	0.0111	0.0273
	MUC-EWC	0.8543	0.8374	0.9054	**0.9040**	0.8123	0.9634	0.8795	0.0015	0.0047	0.0014	0.0004	0.0006	0.0013	0.0017
Ours	ILEAHE-LwF	0.8611	**0.8464**	0.9086	0.9035	0.8146	**0.9654**	0.8833	0.0003	0.0011	0.0010	0.0012	0.0004	0.0001	0.0007
	ILEAHE-EWC	0.8622	**0.8464**	**0.9090**	**0.9040**	**0.8164**	**0.9654**	**0.8839**	0.0003	0.0000	0.0000	0.0017	0.0000	0.0000	0.0003

4.2 Main Result

As in Tab. 4, we compare baselines with ILEAHE. Note that ILEAHE-LwF and ILEAHE-EWC are two versions of our methods, which use L^o_{LwF} (see Eq. (10)) and L^o_{EWC} (see Eq. (11)) respectively.

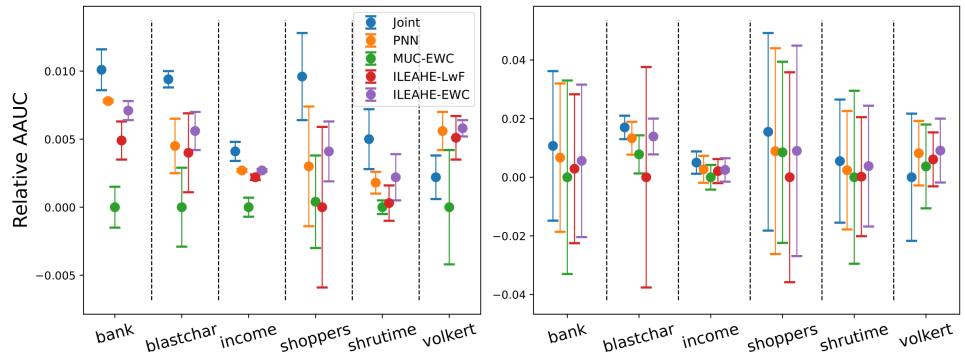
From the effectiveness view of classification results (i.e., AAUC), Joint methods have the upper-bound performance, because they require to access the old data sets when learning a new one and optimize on all data sets simultaneously. On the contrary, other methods receive new data sets only, thus face forgetting problems and have lower performances. Comparing the regularization methods with model adaptation ones, it can be seen that the latter ones generally have better performance, which is at the cost of increasing parameter size during data set increments. It is worth noting that MUC and ILEAHE both improve on L^o_{LwF} or L^o_{EWC}, but the AAUCs of MUC are lower than ILEAHE's. Because ILEAHE uses θ_{sp} to better describe the feature space, while MUC is based solely on θ_s. Compared with all the baselines, ILEAHE achieves the best results and approaches the performances of Joint paradigm. Especially for volkert, ILEAHE even outperforms Joint methods. Because the sub data sets of volkert vary in size, which is hard for joint optimization.

From the forgetting view of incremental learning models (i.e., AF), LwF, DMC and MUC-LwF have larger AF, because they only use θ_s and rely on distillation to prevent forgetting, which is unstable. ACL also suffers from forgetting because adversarial loss without assistance from old data is less effective. The negative AF of PCL shows its ability to improve the old performance when learning the new data sets. Because PCL actively transfers knowledge among old and new data sets when training the classifiers. AF of PNN is zero, because the parameters of each data set are independent and unchanged after optimization. For ILEAHE, benefiting from θ_{sp}, the forgetting issue from θ_s is largely suppressed and AF is close to zero.

4.3 Robustness Analysis to Data Pre-processing

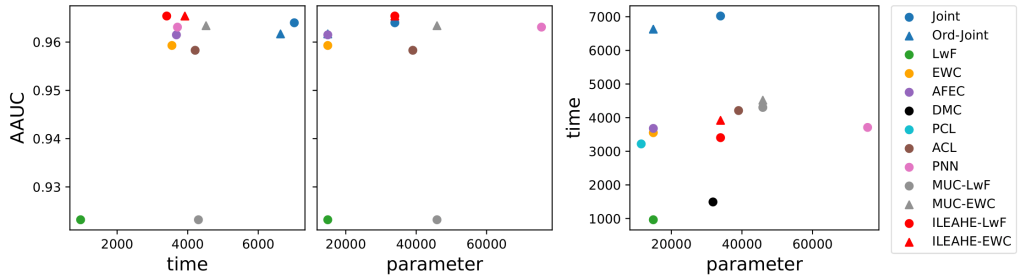
In our experiments, we randomly select attributes to form different sub data sets (i.e., data pre-processing) to mimic the incremental tabular learning in the real scenario. As shown in Fig. 5, we conduct more experiments to show the robustness of various models to data pre-processing. More detailed results (e.g., concrete AAUC and STD) can be found on Github.

4.3.1 Order of sub data sets. For one group of data sets (e.g., $\{X^1, X^2, X^3\}$), the orders of input data sets (e.g., $[X^1, X^3, X^2]$ and $[X^3, X^1, X^2]$) may affect the performance of the given model. As shown in Fig. 5a, we present the results of several promising models on the same group of data sets in 3 different orders. Compared with PNN and MUC-EWC, ILEAHE-EWC provides more consistent and better performance. Because L^o_{EWC} focuses on the change of parameters and is robust under different



(a) 3 orders (b) 10 sets of random attributes

Fig. 5. Compare the model performance and standard deviation under: (a) different orders, (b) different sets of attributes. For one dataset, relative AAUC is the difference between AAUC of one model and the lowest AAUC on that dataset. Standard deviation is noted as error bar.



(a) AAUC vs time/parameter (b) time vs parameter

Fig. 6. Effectiveness-efficiency comparison on volkert. PCL and DMC are omitted in AAUC comparison for their inferior AAUC performance (see Tab. 4).

orders. And θ_{sp} is optimized regardless of orders. However, the correlation between the old shared extractor and new data set varies with order, which affects L^o_{LwF} and destabilizes ILEAHE-LwF.

4.3.2 Selection of attributes. For a tabular data set, the predicting ability of its attributes varies from one another. If the predictive attributes are not selected for a sub data set, it will be hard to extract effective features. So that in Fig. 5b, we present the results on 10 sets of differently selected attributes to show the robustness of the models. By considering all datasets, ILEAHE-EWC has the most robust and best performance, as ensemble prediction from θ_s and θ_{sp} reduces uncertainty and brings effectiveness.

4.4 Efficiency Studies

In this part, we compare the efficiency between ILEAHE and baselines on volkert from aspects of training time cost and parameter size in Fig. 6. Full effectiveness-efficiency comparison results on all datasets are in Github.

In Fig. 6a, we compare the AAUC versus time cost and parameter size for all models. LwF and EWC, as vanilla baselines, are using shorter time and the smallest parameter size, yet have inferior AAUC. Joint and Ord-Joint access all datasets in training, thus they use the longest time to train. For strong baselines, PNN has mediate time cost but a much larger parameter size. On other hand, MUC

Table 5. Ablation study results. Check mark means the corresponding component of the model is applied. The reported results are averaged for 4 runs. Time is reported with Seconds.

#	Model Components					Data Set											
						bank		blastchar		income		shoppers		shrutime		volkert	
	θ_s	θ_{sp}	L^o_{LwF}	L^o_{EWC}	L_{Dis}	AAUC	time	AAUC	time	AAUC	time	AAUC	time	AAUC	time	AAUC	time
1	✓					0.7854	583	0.8171	152	0.8611	411	0.8789	185	0.7890	163	0.9174	3065
2	✓		✓			0.8205	457	0.8316	137	0.8708	207	0.8833	208	0.7950	184	0.9402	1593
3	✓			✓		0.8548	596	0.8355	151	0.9034	542	0.9020	257	0.8081	206	0.9632	2702
4		✓				0.8607	625	0.8438	157	0.9091	420	0.9004	215	0.8149	227	0.9558	3568
5		✓			✓	0.8612	573	0.8443	148	0.9094	381	0.9013	208	0.8175	200	0.9645	2629
6	✓	✓				0.8575	945	0.8443	190	0.9066	532	0.9000	270	0.8126	265	0.9646	3863
7	✓	✓			✓	0.8597	774	0.8445	218	0.9074	630	0.9019	259	0.8141	307	0.9646	3251
8	✓	✓	✓			0.8374	985	0.8421	208	0.9026	706	0.9012	265	0.8123	302	0.9577	3431
9	✓	✓		✓		0.8621	874	0.8415	193	0.9092	662	0.9020	311	0.8162	335	0.9647	3183
10	✓	✓	✓		✓	0.8611	1117	0.8464	204	0.9086	799	0.9035	293	0.8146	325	0.9654	2812
11	✓	✓		✓	✓	0.8622	974	0.8464	199	0.9090	793	0.9040	333	0.8164	334	0.9654	4846

takes a longer time and larger parameter size than ILEAHE. For the other baselines, although their efficiencies are better than ILEAHE, their AAUCs are all lower. Among all the baselines, ILEAHE has the best performance with medium time cost and parameter size. In short, ILEAHE trades a moderate amount of time and parameters for the best performance. This is better illustrated in Fig. 6b, which compares models in time and parameter. It shows that the baselines with higher efficiency (bottom left to ILEAHE in Fig. 6b) are inferior to ILEAHE in terms of effectiveness. To achieve comparable performance to ILEAHE, the other baselines either take a longer time or need more parameters (top right to ILEAHE in Fig. 6b).

4.5 Ablation Studies

In this part, we show the effectiveness brought by each components in our model, as well as the trade off on the efficiency. The ablated parts are listed in Tab. 5. Note that we discard some combinations since they cannot form a complete model for training.

4.5.1 Regularization Loss. Comparing the entries #1, #2 and #3, applying regularization loss releases the forgetting problem. However L^o_{LwF} is less effective than L^o_{EWC} in our experiments. Because the feature is extracted by using $\langle T \rangle$ to query the knowledge of input data through self-attention mechanism. For an unseen data set, the extractor has no knowledge about it, so that $\langle T \rangle$ can not provide meaningful features, which eventually defects prediction. Then it is misleading for L^o_{LwF} to use the old model's prediction on new data set to regularize the new model. L^o_{EWC}, however, avoids this problem by directly regularizing the parameters. Adding either L^o_{LwF} or L^o_{EWC} does not lead to a significant increase in time cost, as they help the model converge faster.

4.5.2 Discriminative Loss. By comparing entries #4 and #5, it is shown that L_{Dis} improves the specific parameters for all data sets. Although L_{Dis} is costly for computation, the model converges faster as the objective is more effective and compensates the cost spent on L_{Dis}.

4.5.3 Shared and Specific Extractors. Entries from #6 to #11 compare the additional effectiveness brought by combining the shared and specific parts. For #6 and #7, where no L^o is applied, their performances are inferior to using only specific extractor. Because the forgetting problem of θ_s negatively affects the ensemble prediction. Comparing #6, #8, #9 with #1, #2, #3 correspondingly, the combination with θ_{sp} notably improves θ_s. At last for #10 and #11, which are actually ILEAHE-LwF and ILEAHE-EWC, the performance becomes optimal. Because the parameter size is increased after combination, the time cost is also larger. However, compared with #6, adding either L^o or L_{Dis} does not increase the time cost significantly, since either of them can help model to converge and compensate the computation cost spent on it.

Table 6. Experiment of using MLP and RNN as basic extractor structures. The results are averaged for 4 runs. Bold for the best result and Underline for the second best. Comparison is within each extractor type and Joint is excluded from notations.

Extractor Type	Model	bank		blastchar		income		shoppers		shrutime		volkert		Avg AAUC for all datasets
		AAUC	std	AAUC	std	AAUC	std	AAUC	std	AAUC	std	AAUC	std	
RNN	Joint	0.8593	0.0012	0.8300	0.0015	0.9106	0.0020	0.8955	0.0026	0.8176	0.0015	0.9337	0.0015	0.8745
	PNN	0.8542	0.0061	**0.8261**	0.0035	0.9049	0.0034	**0.8889**	0.0015	**0.8124**	0.0009	0.9147	0.0527	0.8669
	MUC-EWC	0.8399	0.0096	0.8137	0.0073	0.9013	0.0034	0.8870	0.0022	0.7946	0.0040	0.9361	0.0396	0.8621
	ILEAHE-LwF	0.8529	0.0044	0.8181	0.0063	0.9045	0.0025	0.8884	0.0014	0.8008	0.0046	0.9035	0.0268	0.8614
	ILEAHE-EWC	**0.8549**	0.0021	0.8208	0.0026	**0.9069**	0.0007	0.8881	0.0019	0.7983	0.0032	**0.9371**	0.0082	**0.8677**
MLP	Joint	0.8598	0.0026	0.8486	0.0006	0.9038	0.0021	0.9133	0.0012	0.8264	0.0013	0.9793	0.0013	0.8885
	PNN	0.8528	0.0035	0.8314	0.0021	0.9041	0.0021	0.8784	0.0200	0.8040	0.0037	0.9733	0.0006	0.8740
	MUC-EWC	0.8670	0.0016	**0.8464**	0.0012	0.9090	0.0008	0.9089	0.0016	0.8150	0.0039	0.9653	0.0020	0.8853
	ILEAHE-LwF	0.8658	0.0009	0.8458	0.0000	**0.9102**	0.0004	0.9076	0.0010	**0.8154**	0.0046	0.9758	0.0003	0.8868
	ILEAHE-EWC	**0.8673**	0.0008	0.8461	0.0014	0.9098	0.0004	**0.9097**	0.0011	0.8144	0.0019	**0.9765**	0.0011	**0.8873**
Transformer	ILEAHE-LwF	0.8611	0.0021	0.8464	0.0028	0.9086	0.0011	0.9035	0.0050	0.8146	0.0001	0.9654	0.0006	0.8833
	ILEAHE-EWC	0.8622	0.0008	0.8464	0.0031	0.9090	0.0007	0.9040	0.0022	0.8164	0.0029	0.9654	0.0007	0.8839

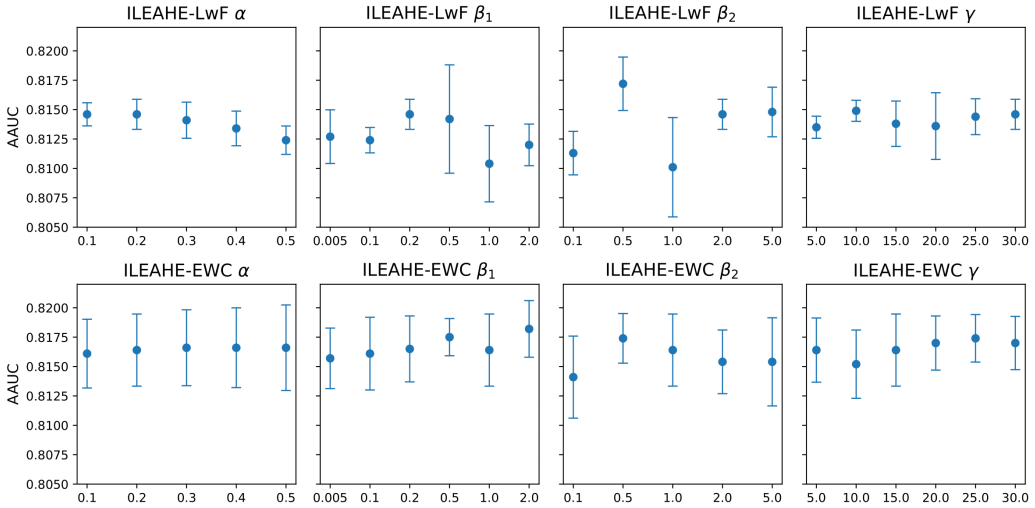


Fig. 7. Hyperparameter sensitivity of ILEAHE-LwF (Upper row) and ILEAHE-EWC (Lower row) on shrutime. The results are averaged for 4 runs.

4.5.4 Extractors Choices. In the main experiment, we use Transformer as the feature extractor. However, our method can plug in any structures of extractor. In Tab. 6, we provide experiments using RNN or MLP as the extractor structures and implement them into strong baselines. For both structures, ILEAHE provides good performance and can approach the joint paradigm. It supports that ILEAHE does not rely on the adopted feature extractor (e.g., Transformer) to achieve better performance than baselines.

4.6 Hyper-parameter Sensitiveness

The major hyperparameters for ILEAHE are α in Eq. (7), γ in Eq. (12), and β_1, β_2 in Eq. (13). As shown in Fig. 7, we study the parameter sensitivity of ILEAHE-LwF and ILEAHE-EWC on shrutime. Overall, compared with ILEAHE-EWC, ILEAHE-LwF is more sensitive to hyper-parameters.

- α controls the weight of prediction from shared part. Empirically, α is preferred to be smaller than 0.5. That is because the shared part usually faces forgetting problem. High weight for the shared part may be inferior to the domain-specific part after increments. This can be observed on both ILEAHE-LwF and ILEAHE-EWC.
- β_1 controls the weight of L^o. For L^o_{LwF}, β_1 tends to be small, otherwise the model may bias towards old data. L^o_{EWC} is not sensitive to β_1 because it leaves space for learning new data.

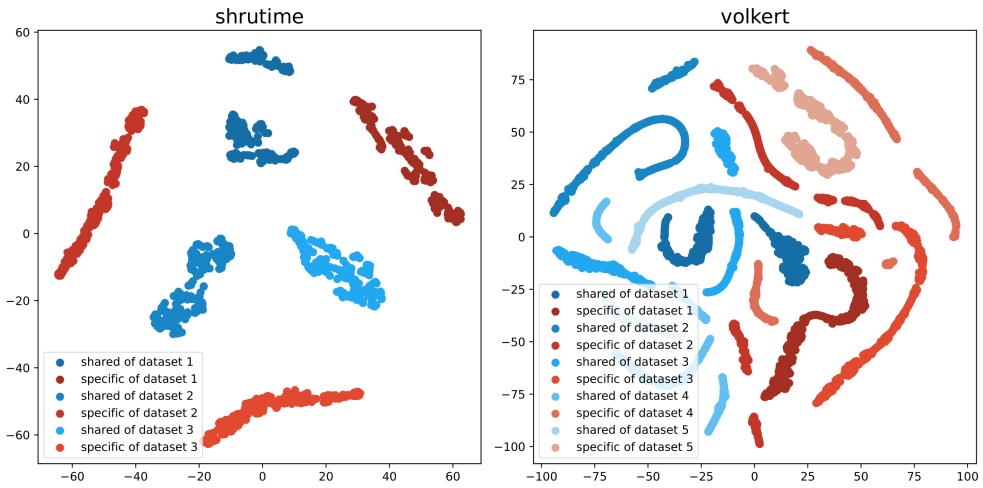


Fig. 8. Visualization of shared and specific features from ILEAHE-EWC. The shared features are in blue and specific features are in red.

- β_2 affects the weight of L_{Dis}. The selection is subject to datasets, as the discriminabilities of θ_{sp} vary with datasets. ILEAHE-LwF is sensitive to β_2, because large β_2 harms the optimization of L^o_{LwF}, while small β_2 defects L_{Dis}. ILEAHE-EWC is stable, again for the elasticity of L^o_{EWC} towards new data set.
- γ lifts the impact from discriminability score s to L_{Dis}. In our observation, two models are robust to γ. Because s exponentially affects L_{Dis}, the effect from γ becomes subtle when s is large. And this process is independent of L^o.

4.7 Visualization

ILEAHE adopts two kinds of feature extractors to extract the shared features across datasets and domain-specific features for each dataset. To check whether the extracted features are shared or domain-specific, we plot the features extracted by θ_s and θ_{sp} in Fig. 8 based on t-SNE [46]. It can be seen that the shared features (blue ones) gather together and separate from the specific ones (red ones). This supports that, in heterogeneous spaces, θ_s and θ_{sp} can handle the shared and domain-specific parts correspondingly.

5 CONCLUSION

In this paper, we proposes a novel incremental tabular learning framework ILEAHE that deals with the *Catastrophic Forgetting* problem on the heterogeneous feature space. ILEAHE extracts shared features and domain-specific features separately from each data set, where shared features are effective in all data sets and domain-specific features are especially effective in the corresponding data sets. It can significantly avoid forgetting by heterogeneously describing the feature spaces of each data sets during increments. At each increment, ILEAHE maintains the performance of the shared feature extractor for old data sets through regularization loss. To enhance the specific feature extractors, ILEAHE introduces the discriminability score to examine and improve their special effectiveness. ILEAHE empirically demonstrates its effectiveness and efficiency compared with baselines. For the future works, one direction worth trying is to adapt incremental learning to graph representation learning [24, 49], especially the scenario of knowledge graphs [9, 10].

ACKNOWLEDGMENTS

Lei Chen is partially supported by National Science Foundation of China (NSFC) under Grant No. U22B2060, the Hong Kong RGC GRF Project 16209519, CRF Project C6030-18G, C2004-21GF, AOE Project AoE/E-603/18, RIF Project R6020-19, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Microsoft Research Asia Collaborative Research Grant, HKUST-Webank joint research lab grant and HKUST Global Strategic Partnership Fund (2021 SJTU-HKUST). Shimin Di is supported by the JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust.

REFERENCES

[1] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In CVPR. 3366–3375.

[2] James Bergstra, Daniel Yamins, and David D. Cox. 2013. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In ICML, Vol. 28. JMLR.org, 115–123.

[3] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2021. Deep Neural Networks and Tabular Data: A Survey. CoRR abs/2110.01889 (2021). arXiv:2110.01889

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018. Efficient lifelong learning with a-gem. arXiv preprint arXiv:1812.00420 (2018).

[5] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. TPAMI (2021).

[6] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. 2019. Learning without memorizing. In CVPR. 5138–5146.

[7] Shimin Di, Jingshu Peng, Yanyan Shen, and Lei Chen. 2018. Transfer Learning via Feature Isomorphism Discovery. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (London, United Kingdom) (KDD '18). Association for Computing Machinery, New York, NY, USA, 1301–1309. https://doi.org/10.1145/3219819.3220029

[8] Shimin Di, Yanyan Shen, and Lei Chen. 2019. Relation Extraction via Domain-Aware Transfer Learning. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Anchorage, AK, USA) (KDD '19). Association for Computing Machinery, New York, NY, USA, 1348–1357. https://doi.org/10.1145/3292500.3330890

[9] Shimin Di, Quanming Yao, and Lei Chen. 2021. Searching to Sparsify Tensor Decomposition for N-ary Relational Data. In WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 4043–4054. https://doi.org/10.1145/3442381.3449853

[10] Shimin Di, Quanming Yao, Yongqi Zhang, and Lei Chen. 2021. Efficient Relation-aware Scoring Function Search for Knowledge Graph Embedding. In 37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021. IEEE, 1104–1115. https://doi.org/10.1109/ICDE51399.2021.00100

[11] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. 2020. Podnet: Pooled outputs distillation for small-tasks incremental learning. In ECCV. Springer, 86–102.

[12] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. 2020. Adversarial continual learning. In ECCV. Springer, 386–402.

[13] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. arXiv preprint arXiv:1701.08734 (2017).

[14] Robert M French. 1999. Catastrophic forgetting in connectionist networks. TiCS 3, 4 (1999), 128–135.

[15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. CoRR abs/1503.02531 (2015). arXiv:1503.02531

[16] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In CVPR. 831–839.

[17] Wenpeng Hu, Qi Qin, Mengyu Wang, Jinwen Ma, and Bing Liu. 2021. Continual Learning by Using Information of Each Class Holistically. In AAAI. AAAI Press, 7797–7805.

[18] Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. 2021. Distilling Causal Effect of Data in Class-Incremental Learning. In CVPR. 3957–3966.

[19] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. 2020. Memory-efficient incremental learning through feature adaptation. In ECCV. Springer, 699–715.

[20] Nitin Kamra, Umang Gupta, and Yan Liu. 2017. Deep generative dual memory network for continual learning. arXiv preprint arXiv:1710.10368 (2017).

[21] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. CoRR abs/1612.00796 (2016). arXiv:1612.00796

[22] Haoyang Li and Lei Chen. 2021. Cache-Based GNN System for Dynamic Graphs. In Proceedings of the 30th ACM International Conference on Information and Knowledge Management (Virtual Event, Queensland, Australia) (CIKM '21). Association for Computing Machinery, New York, NY, USA, 937–946. https://doi.org/10.1145/3459637.3482237

[23] Haoyang Li, Shimin Di, Zijian Li, Lei Chen, and Jiannong Cao. 2022. Black-box Adversarial Attack and Defense on Graph Neural Networks. In 2022 IEEE 38th International Conference on Data Engineering (ICDE). 1017–1030. https://doi.org/10.1109/ICDE53745.2022.00081

[24] Haoyang Li, Shimin Di, Zijian Li, Lei Chen, and Jiannong Cao. 2022. Black-box Adversarial Attack and Defense on Graph Neural Networks. In 38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022. IEEE, 1017–1030. https://doi.org/10.1109/ICDE53745.2022.00081

[25] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. TPAMI 40, 12 (2017), 2935–2947.

[26] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. 2018. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In ICPR. IEEE, 2262–2268.

[27] Yu Liu, Sarah Parisot, Gregory Slabaugh, Xu Jia, Ales Leonardis, and Tinne Tuytelaars. 2020. More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning. In ECCV. Springer, 699–716.

[28] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. 2020. Mnemonics training: Multi-class incremental learning without forgetting. In CVPR. 12245–12254.

[29] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In NeurIPS, Vol. 30. 6467–6476.

[30] Ilya Loshchilov and Frank Hutter. 2017. Fixing Weight Decay Regularization in Adam. CoRR abs/1711.05101 (2017). arXiv:1711.05101

[31] Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In CVPR. 7765–7773.

[32] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. 2020. Class-incremental learning: survey and performance evaluation on image classification. arXiv preprint arXiv:2010.15277 (2020).

[33] Zichen Miao, Ze Wang, Wei Chen, and Qiang Qiu. 2022. Continual Learning with Filter Atom Swapping. In ICLR.

[34] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering 22, 10 (2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

[35] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. Neural Networks 113 (2019), 54–71.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. NeurIPS 32 (2019).

[37] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In CVPR. 2001–2010.

[38] Amir Rosenfeld and John K Tsotsos. 2018. Incremental learning through deep adaptation. TPAMI 42, 3 (2018), 651–663.

[39] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016).

[40] Gobinda Saha, Isha Garg, and Kaushik Roy. 2021. Gradient Projection Memory for Continual Learning. In ICLR.

[41] Mathieu Salzmann, Carl Henrik Ek, Raquel Urtasun, and Trevor Darrell. 2010. Factorized orthogonal latent spaces. In AISTATS. JMLR Workshop and Conference Proceedings, 701–708.

[42] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In ICML. PMLR, 4548–4557.

[43] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. arXiv preprint arXiv:1705.08690 (2017).

[44] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. 2021. SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training. arXiv preprint arXiv:2106.01342 (2021).

[45] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. 2021. Layerwise optimization by gradient decomposition for continual learning. In CVPR. 9634–9643.
[46] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. Journal of machine learning research 9, 11 (2008).
[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In NeurIPS. 5998–6008.
[48] Liyuan Wang, Mingtian Zhang, Zhongfan Jia, Qian Li, Chenglong Bao, Kaisheng Ma, Jun Zhu, and Yi Zhong. 2021. AFEC: Active Forgetting of Negative Transfer in Continual Learning. In NeurIPS. 22379–22391.
[49] Zhili Wang, Shimin Di, and Lei Chen. 2021. Autogel: An automated graph neural network with explicit link information. Advances in Neural Information Processing Systems 34 (2021), 24509–24522.
[50] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In ICML. PMLR, 3987–3995.
[51] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. 2020. Class-incremental learning via deep model consolidation. In WACV. 1131–1140.

Received April 2022; revised July 2022; accepted August 2022