ICE-T: Interactions-aware Cross-column Contrastive Embedding for Heterogeneous Tabular Datasets

Tomas Tokar[1, 2], **Scott Sanner**[2, 3]

[1]Wondeur AI, [2]University of Toronto, [3]Vector Institute for AI

Abstract

Finding high-quality representations of heterogeneous tabular datasets is crucial for their effective use in downstream machine learning tasks. Contrastive representation learning (CRL) methods have been previously shown to provide a straightforward way to learn such representations across various data domains. Current tabular CRL methods learn joint embeddings of data instances (tabular rows) by minimizing a contrastive loss between the original instance and its perturbations. Unlike existing tabular CRL methods, we propose leveraging frameworks established in multimodal representation learning, treating each tabular column as a distinct modality. A naive approach that applies a pairwise contrastive loss to tabular columns is not only prohibitively expensive as the number of columns increases, but as we demonstrate, it also fails to capture interactions between variables. Instead, we propose a novel method called ICE-T that learns each columnar embedding by contrasting it with aggregate embeddings of the complementary part of the table, thus capturing interactions and scaling linearly with the number of columns. Unlike existing tabular CRL methods, ICE-T allows for column-specific embeddings to be obtained independently of the rest of the table, enabling the inference of missing values and translation between columnar variables. We provide a comprehensive evaluation of ICE-T across diverse datasets, demonstrating that it generally surpasses the performance of the state-of-the-art alternatives.

Introduction

Heterogeneous tabular datasets constitute an extremely important, yet often overlooked, data class, which remains to be challenging for application of deep neural networks (Shwartz-Ziv and Armon 2022). As with other data types the key is to find good quality data representations that facilitate the downstream tasks (Bengio, Courville, and Vincent 2013). Contrastive representation learning (CRL) offers a very straightforward way for learning such representations, without the need of associated data labels (Le-Khac, Healy, and Smeaton 2020; Ericsson et al. 2022).

Multimodal CRL Recently, there has been increasing attention on a type of CRL methods known as multimodal CRL. Multimodal CRL has traditionally been applied to data comprising multiple distinct modalities, such as image-text or audio-video pairs, from which it derives its name (Deldari et al. 2022; Zong, Mac Aodha, and Hospedales 2023). Multimodal CRL offers a unique way of learning modality-specific embeddings that are coordinated with embeddings from other modalities while allowing inputs of one modality to be embedded independently of the others (Guo, Wang, and Wang 2019). Therefore, embeddings can be obtained even in the absence of some modalities and can be used to

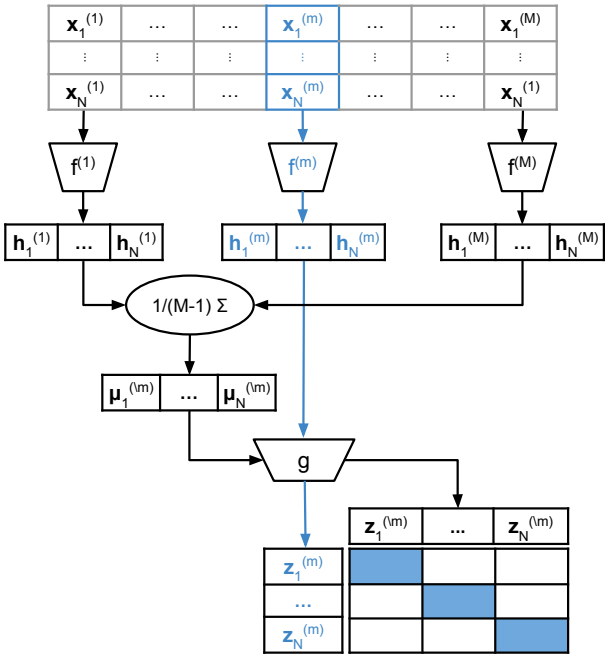

Figure 1: A schematic representation of ICE-T. Tabular entries are passed through variable-specific encoders $f^{(m)}$ to produce intermediate latent representations $\mathbf{h}^{(m)}$. For each variable m we compute the associated *anchor* $\mu^{(\setminus m)}$ as the average of the representations of the remaining variables. These resulting vectors are then projected via a shared neural subnetwork g to produce the final embeddings. The cosine similarities of these embeddings are used to compute the contrastive loss for each variable.

infer the values of the missing ones—a task often referred to as *modality translation* (Kaur, Pannu, and Malhi 2021). The idea of adopting these methods for learning representations of heterogeneous tabular data is thus very appealing.

Motivation Intuitively, heterogeneous tabular data can be treated as multimodal data, where each variable (tabular column) is considered a single modality. The central idea is to learn variable-specific mappings that project inputs into a common latent space so that the embeddings of inputs are similar if they belong to the same instance (tabular row) while dissimilar otherwise. A naive approach would involve computing similarities between all pairs of variable-specific embeddings. However, such a pairwise contrasting scales quadratically with the number of variables and becomes prohibitively costly as the number of variables increases. Moreover, pairwise contrastive learning fails to capture interactions between variables, which can corrupt the resulting embeddings (e.g, dataset depicted in Figure 2). This situation is common in tabular data, which typically contain interacting variables, and can render pairwise embedding approaches inapplicable (Borisov et al. 2022).

Proposed method To address this problem, we propose a novel CRL approach called **I**nteractions-aware Cross-column **C**ontrastive **E**mbedding for heterogeneous **T**abular datasets (ICE-T) (Figure 1). ICE-T contrasts column-specific embeddings with the embedded aggregates of the remaining columns in *linear time* and allows to account for interactions among variables, while preserving the advantages of multimodal CRL. ICE-T is a simple, versatile, and data-agnostic approach specifically designed for heterogeneous tabular data but can be easily applied to other domains. Overall, ICE-T offers superior or generally competitive performance compared to other methods.

Contributions (1) We offer a simple, easily reproducible example that illustrates the failure of multimodal CRL methods to capture interactions among variables (2) We introduce a novel method called ICE-T that addresses this limitation (3) We provide a comprehensive experimental comparison between ICE-T, five CRL methods and shallow learning benchmarks across a range of datasets, including frequently overlooked image/text–tabular data; measuring performance in four tasks (i) cross-modal translation, (ii) clustering, (iii) supervised learning and (iv) transfer learning.

Related Work

Previous efforts to apply CRL to heterogeneous tabular data utilized joint CRL, where tabular rows undergo random transformations to produce their perturbed analogs, and the latent representations are learned by contrasting the original data against their perturbed analogs. This approach is best exemplified by **Scarf** (Bahri et al. 2021) and **SubTab** (Ucar, Hajiramezanali, and Edwards 2021). In Scarf, rows are perturbed by replacing inputs from a random subset of variables by values drawn from the respective variable marginals. In SubTab rows are perturbed by randomly dividing variables to create overlapping subsets. In addition to contrastive embedding loss, the authors of SubTab propose using additional distance loss and reconstruction loss.

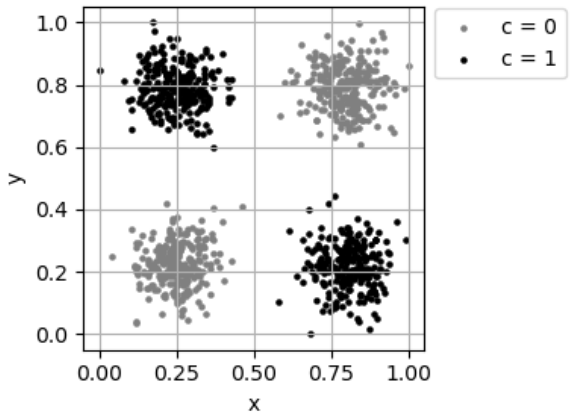

Figure 2: The XOR blobs can form a heterogeneous tabular dataset with three variables: x, y and c, where each variable depends on the interaction between the other two. As we demonstrate, this interaction poses a challenging problem for multimodal CRL methods.

Unlike these methods, we propose to borrow frameworks established in the domain of multimodal representation learning and to approach tabular data as multimodal datasets, where each tabular column is treated as a single modality. From this perspective we identified three methods that are closely related to our line of research:

CLIP (Radford et al. 2021) produces text and image embeddings that are similar if the input text and image are related, and dissimilar otherwise. It consists of modality-specific encoders, followed by linear projections into a common latent space, which are trained under the InfoNCE loss (Oord, Li, and Vinyals 2018), using cosine similarity, across a very large collection of image-text pairs. While the term "CLIP" usually refers to a specific model, it may denote a CRL method that trains modality-specific mappings only through pairwise cosine similarity across modalities (Table 1). Under this generalization, CLIP provides a versatile approach that can be applied to any type of modalities, as exemplified by (Alayrac et al. 2020; Wang et al. 2021; Guzhov et al. 2022).

GMC (Poklukar et al. 2022) was originally proposed for multimodal time-series data where modality-specific mappings are trained in coordination with joint representation learning. It consists of modality-specific encoders followed by deep neural projection heads to map inputs into a common latent space. Additionally, GMC projects inputs into the same latent space jointly through a designated neural subnetwork. The objective of GMC is to maximize the cosine similarity between the modality-specific embeddings and the joint embedding belonging to the same instance, while minimizing the similarity among embeddings belonging to different instances. The minimized term includes pairwise similarities between modality-specific embeddings, the similarity between joint and modality-specific embeddings, and the

Method	Batch loss
Scarf	$\mathcal{L} = \sum_i^B -\log \frac{\exp\left(s(\mathbf{z}_i^{(1:M)}, \tilde{\mathbf{z}}_i^{(1:M)})/\tau\right)}{\sum_j \exp\left(s(\mathbf{z}_i^{(1:M)}, \tilde{\mathbf{z}}_j^{(1:M)})/\tau\right)}$
SubTab	$\mathcal{L} = \sum_i^B \sum_{\mathcal{A},\mathcal{B}} \Big[-\log \frac{\exp\left(s(\mathbf{z}_i^{(\mathcal{A})}, \mathbf{z}_i^{(\mathcal{B})})/\tau\right)}{\sum_j \mathbb{1}_{j\neq i} \exp\left(s(\mathbf{z}_i^{(\mathcal{A})}, \mathbf{z}_j^{(\mathcal{B})})/\tau\right)} + (\mathbf{z}_i^{(\mathcal{A})} - \mathbf{z}_i^{(\mathcal{B})})^2 \Big]$
CLIP	$\mathcal{L} = \sum_i^B \sum_m^M \sum_{n\neq m}^M -\log \frac{\exp\left(s(\mathbf{z}_i^{(m)}, \mathbf{z}_i^{(n)})/\tau\right)}{\sum_j \exp\left(s(\mathbf{z}_i^{(m)}, \mathbf{z}_j^{(n)})/\tau\right)}$
GMC	$\mathcal{L} = \sum_i^B \sum_m^M -\log \frac{\exp\left(s(\mathbf{z}_i^{(m)}, \mathbf{z}_i^{(1:M)})/\tau\right)}{\sum_{j\neq i} \exp\left(s(\mathbf{z}_i^{(m)}, \mathbf{z}_j^{(1:M)})/\tau\right) + \exp\left(s(\mathbf{z}_i^{(m)}, \mathbf{z}_i^{(m)})/\tau\right) + \exp\left(s(\mathbf{z}_i^{(1:M)}, \mathbf{z}_i^{(1:M)})/\tau\right)}$
MCN	$\mathcal{L} = \sum_i^B \Big[\sum_m^M \sum_{n\neq m}^M -\log \frac{\exp\left(s(\mathbf{z}_i^{(m)}, \mathbf{z}_i^{(n)}) - \delta\right)}{\exp\left(s(\mathbf{z}_i^{(m)}, \mathbf{z}_i^{(n)}) - \delta\right) + \sum_{j\neq i} \exp\left(s(\mathbf{z}_i^{(m)}, \mathbf{z}_j^{(n)})\right)} - \sum_m^M \log \frac{\exp\left(s(\mathbf{z}_i^{(m)}, C_i') - \delta\right)}{\sum_k^K \exp\left(s(\mathbf{z}_i^{(m)}, C_k)\right)} \Big]$
ICE-T	$\mathcal{L} = \sum_i^B \sum_m^M -\log \frac{\exp\left(s(\mathbf{z}_i^{(m)}, \mathbf{z}_i^{(\setminus m)})/\tau\right)}{\sum_j \exp\left(s(\mathbf{z}_i^{(m)}, \mathbf{z}_j^{(\setminus m)})/\tau\right)}$

Table 1: Unified comparison of contrastive loss functions used by state-of-the-art methods vs. ICE-T (ours); s indicates cosine similarity, τ is learnable parameter $\mathbf{z}^{(1:M)}$ indicates joint instance embedding; $\mathbf{z}^{(\mathcal{A})}$ and $\mathbf{z}^{(\mathcal{B})}$ indicate partial instance embeddings; δ is a hyperparameter; C_i' indicates the centroid that is nearest to the i-th fused multimodal feature and C_k is the k-th centroid; g is a neural function.

similarity between pairs of joint embeddings.

MCN (Chen et al. 2021) combines multimodal CRL with latent space clustering. It encodes inputs via modality-specific encoders and associated linear projections. Subsequently, MCN calculates joint representations by aggregating modality-specific embeddings together via an arithmetic mean. These joint embeddings, referred to by the authors as *fused multimodal features*, are subjected to online k-means clustering (Cohen-Addad et al. 2021). Euclidean distance between the features and a set of k centroids is computed and each feature is assigned the nearest centroid. In addition to maximizing pairwise similarity among the modality-specific embeddings, MCN aims to maximize the similarity between these embeddings and the centroids that are nearest to the respective multimodal features; the authors also add the reconstruction loss to the overall loss of the model.

Algorithm 1: ICE-T – batch loss computation.

Inputs:
$\{x^{(1)}, \ldots, x^{(M)}\}$ {Data batch}
$\{f_{(1)}, \ldots, f^{(M)}\}$ {Mappings}
g, l {Projection, Loss function}
$\Sigma \leftarrow 0$ {Initialize sum}
for $m = 1$ **to** M **do**
  $h^{(m)} \leftarrow f^{(m)}(x^{(m)})$ {Intermediate representations}
  $\Sigma \leftarrow \Sigma + h^{(m)}$ {Update sum}
end for
$L \leftarrow 0$ {Initialize loss}
for $m = 1$ **to** M **do**
  $\mu^{(\setminus m)} \leftarrow (\Sigma - h^{(m)})/(M-1)$ {Calculate anchor}
  $z_1 \leftarrow g(h^{(m)})$ {Apply g}
  $z_2 \leftarrow g(\mu^{(\setminus m)})$
  $s \leftarrow s(z_1, z_2)$ {Get similarities}
  $L \leftarrow L + l(s)$ {Update loss}
end for

Interactions-aware Cross-column Contrastive Embedding

Preliminaries

Let $\mathbf{x}_i$ denote the i-th data instance, comprising M inputs from different variables: $\mathbf{x}_i = (\mathbf{x}_i^{(m)})_{m=1}^M$. Assume that each input $\mathbf{x}^{(m)}$ comes from a distinct variable-specific input space $\mathcal{X}^{(m)}$: $\mathbf{x}^{(m)} \in \mathcal{X}^{(m)}$. Furthermore, consider $\mathbf{x} \sim p(\mathbf{x})$ and $\mathbf{x}^{(m)} \sim p(\mathbf{x}^{(m)})$, where $p(\mathbf{x})$ denotes the joint data distribution, and $p(\mathbf{x}^{(m)})$ denotes the marginal distribution of given variable m; and let $p(\mathbf{x}^{(m)}|\mathbf{x}^{(\setminus m)})$ denote the conditional probability of $\mathbf{x}^{(m)}$ given the inputs $\mathbf{x}^{(\setminus m)} = (\mathbf{x}^{(n)})_{n\neq m}^M$.

Rationale

The central idea of ICE-T is to take intermediate hidden representations $\mathbf{h}^{(m)}$ obtained by variable-specific encoders $f^{(m)}$ and fuse the intermediate representations of the remaining variables into a single vector $\mu^{(\setminus m)}$, which we refer to as the *anchor*. The anchors are computed via the arithmetic mean: $\mu^{(\setminus m)} = 1/(M-1)\sum_{n\neq m} \mathbf{h}^{(n)}$, where M is the total number of variables. Integrating by averaging, instead of learning a joint representation via a fixed architecture (i) encourages additive embedding representations and (ii) makes the method robust against missing inputs and hence more versatile.

The two hidden vectors $\mathbf{h}^{(m)}$ and $\mu^{(\setminus m)}$ are passed through the identical neural subnetwork g to obtain final latent representations $\mathbf{z}^{(m)} = g(\mathbf{h}^{(m)})$ and $\mathbf{z}^{(\setminus m)} = g(\mu^{(\setminus m)})$, which are then compared by cosine similarity s. We will use $s^{(m)}(i,j)$ to denote similarity between the pair of latent vectors belonging to i-th and j-th instance: $s^{(m)}(i,j) :=$

Method	Embedding form		
	$\mathbf{z}^{(m)}$	$\mathbf{z}^{(1:M)}$	$\mathbf{z}^{(\mathcal{A})}$
Scarf		✓	
SubTab		✓	
CLIP	✓	!	!
GMC	✓	✓	!
MCN	✓	✓	!
ICE-T	✓	✓	✓

Table 2: ICE-T provides variable-specific $\mathbf{z}^{(m)}$, joint instance $\mathbf{z}^{(1:M)}$ and also partial instance embeddings $\mathbf{z}^{(\mathcal{A})}$ for any subset $\mathcal{A}$ of input variables generically, whereas some methods allow *post hoc* embeddings aggregation ("!").

$s(g(\mathbf{h}_i^{(m)}), g(\mu_j^{(\setminus m)}))$. The aim is to maximize the similarity between the embedding vectors and the matching anchors $s^{(m)}(i,i)$, while minimizing the non-matching ones $s^{(m)}(i,j)$.

For this purpose, we adopt the InfoNCE loss (Oord, Li, and Vinyals 2018), so the loss incurred from the i-th instance on variable m is given by:

$$l_i^{(m)} = -\log \frac{\exp(s^{(m)}(i,i)/\tau)}{\sum_j \exp(s^{(m)}(i,j)/\tau)} \tag{1}$$

Here τ is the "temperature", a trainable parameter controlling the sensitivity of the loss across the range of similarity values. The total batch loss is then the sum of losses incurred across all instances and modalities: $L = \sum_{(i,m)} l_i^{(m)}$ (cf. Algorithm 1).

Probabilistic Interpretation

We seek variable-specific neural mappings $f^{(m)} : \mathcal{X}^{(m)} \to \mathbb{R}^q$ and an additional neural mapping $g : \mathbb{R}^q \to \mathbb{R}^d$, such that for any $\mathbf{x}_i^{(m)} \in \mathcal{X}^{(m)}$, for $\forall m \in \{1, \dots M\}$, produce embeddings: $\mathbf{z}_i^{(m)} = g(f^{(m)}(\mathbf{x}_i^{(m)}))$; such that

$$p(\mathbf{x}^{(m)}|\mathbf{x}^{(\setminus m)}) \propto s(\mathbf{z}^{(m)}, \mathbf{z}^{(\setminus m)}) \tag{2}$$

where $s : \mathbb{R}^{M \times d} \to \mathbb{R}$ is a similarity function that quantifies the similarity between the variable-specific embedding $\mathbf{z}^{(m)}$ and the embedding of the remaining variables $\mathbf{z}^{(\setminus m)} = g\big(1/(M-1)\sum_{n \neq m} f^{(n)}(\mathbf{z}^{(n)})\big)$. Equation 2 thus implies that we want to maximize the similarity between the embeddings of inputs that are likely to be associated and minimize it otherwise.

Benefits

Embedding versatility Once trained, ICE-T can provide variable-specific embeddings $\mathbf{z}^{(m)}$, joint complete instance embeddings $\mathbf{z}^{(1:M)}$, and joint partial instance embeddings $\mathbf{z}^{(\mathcal{A})}$ of a values from any subset $\mathcal{A} \in \mathcal{P}(M)$, where $\mathcal{P}(M)$ is the power set of the variables it was trained on. This versatility is thanks to the use of arithmetic aggregation of the intermediate representation, instead of a fixed neural architecture (a used by, for example, in (Poklukar et al. 2022)). This gives ICE-T an advantage over many other methods, which require *post hoc* aggregation of the variable-specific embeddings (cf. Table 2).

Modality translation The important advantage of most multimodal CRL methods, including ICE-T, is support for modality translation. In the context of heterogeneous tabular datasets this can be viewed as a form of missing values imputation, where the value of one variable is estimated based on the values of the other modalities of the given instance. This can be formulated as the assignment:

$$\hat{\mathbf{x}}^{(m)} = \underset{\mathbf{x}^{(m)} \in \mathcal{X}^{(m)}}{\operatorname{argmax}} \; p(\mathbf{x}^{(m)}|\mathbf{x}^{(\mathcal{A})}) \tag{3}$$

where $\mathcal{A}$ is a subset of evidence variables so that $\mathcal{A} \in \mathcal{P}(M)$ and $m \notin \mathcal{A}$. Assuming $p(\mathbf{x}^{(m)}|\mathbf{x}^{(\mathcal{A})}) \propto s(\mathbf{z}^{(m)}, \mathbf{z}^{(\mathcal{A})})$, the above assignment converts to:

$$\mathbf{x}^{(m)} = \underset{\mathbf{x}^{(m)} \in \mathcal{X}^{(m)}}{\operatorname{argmax}} \; s\big(\mathbf{z}^{(m)}, \mathbf{z}^{(\mathcal{A})}\big) \tag{4}$$

which can be solved provided the input space $\mathcal{X}^{(m)}$ is well defined and can be sampled efficiently.

Linear scaling Methods that rely on pairwise contrasting do not scale well with the large number of variables due to the quadratic increase in the number of pairwise contrastive losses. Unlike these, ICE-T *scales linearly with the number of variables*, as can be seen in Algorithm 1, where we can compute the sum of all embeddings in one linear pass over modalities and then calculate the loss in the second linear pass by contrasting the modality-specific embeddings against their respective anchors computed in a leave-one-out manner. This provides an important computational benefit to our method.

Data-agnostic ICE-T can be readily adapted to accommodate various data types, including image or text variables, by selecting appropriate neural architectures to implement mappings $f^{(m)}$. This flexibility allows it to be applied to idiomatic tabular datasets containing images, text, or other modalities.

Experimental Design

Data

We used a collection of 44 real-world tabular datasets from the benchmark introduced in (Grinsztajn, Oyallon, and Varoquaux 2022)[1]. Moreover, to demonstrate applicability of ours and other methods on tables containing images and text, we used 2 additional image-tabular datasets and 2 text-tabular datasets; thus, in total, we used 48 datasets (cf. Table 3). Each dataset includes one categorical (classification), or numerical (regression) target variable (response).

To minimize the effects of data preparation, we restricted ourselves only to minimal data processing involving ordinal encoding of the categorical and rescaling/log-transforming some numerical variables (cf. Supplementary Materials). We

[1]We excluded the largest dataset (delays_zurich_airport) due to computational restrictions.

Name	Samples	Variables			
		Num	Cat	Txt	Img
Classification					
albert	58252	0	32	0	0
bank_marketing	10578	7	1	0	0
bioresponse	3434	419	1	0	0
california	20634	8	1	0	0
clothing	23486	2	5	2	0
compas	4966	3	9	0	0
covertype	566602	10	1	0	0
credit	16714	10	1	0	0
defaults	13272	20	2	0	0
diabetes	71090	7	1	0	0
electricity	38474	7	2	0	0
eye_movements	7608	20	4	0	0
heloc	10000	22	1	0	0
higgs	940160	24	1	0	0
jannis	57580	54	1	0	0
kickstarter	108128	3	4	3	0
miniboone	72998	50	1	0	0
road_safety	111762	14	17	0	0
skin_cancer	13354	1	4	0	1
streetview	11054	2	1	0	1
telescope	13376	10	1	0	0
Regression					
abalone	4177	8	1	0	0
ailerons	13750	34	0	0	0
airlines	1000000	4	2	0	0
allstate	188318	15	110	0	0
bike_sharing	17379	5	2	0	0
brazilian_houses	10692	8	4	0	0
cpu	8192	22	0	0	0
diamonds	53940	8	2	0	0
elevators	16599	17	0	0	0
house	22784	17	0	0	0
house_sales	21613	14	3	0	0
houses	20640	9	0	0	0
medical_charges	163065	4	0	0	0
mercedes	4209	1	359	0	0
miami_housing	13932	14	0	0	0
nyc_taxi	581835	4	13	0	0
pol	15000	27	0	0	0
seattle_crime	52031	1	4	0	0
sgemm	241600	4	6	0	0
soil	8641	4	1	0	0
sulfur	10081	7	0	0	0
superconduct	21263	80	0	0	0
supreme	4052	3	5	0	0
topo	8885	253	3	0	0
ukair	394299	3	4	0	0
wine_quality	6497	12	0	0	0
yprop	8885	43	0	0	0

Table 3: The 48 datasets used in this work and their respective number of samples and the number of variables by type.

randomly split the data into training, validation and testing portions, allocating 10% of the data for validation and another 10% for testing. For each dataset we performed multiple experimental replicates (5 for tabular and 3 for img/text-tabular data), each time using new random split and then averaged the obtained results.

Models and Training

We compared ICE-T against the five state-of-the-art CRL methods described in the previous section: (i) Scarf (ii) SubTab with contrastive and distance loss (iii) CLIP (iv) GMC and (v) MCN with contrastive and clustering loss. Note, since CLIP and MCN scale quadratically with the number of variables, we decided to exclude them from experiments on datasets with more than 25 variables, which would otherwise result in excessive computation runtimes.

For each *method* we trained multiple *models* using different configurations of hyperparameters controlling the model size and its training. Models were trained on the training sets of the data for up to 100 epochs using early stopping with patience set to 5 epochs and validation loss as the criterion.

Evaluation

Once trained, the models were applied to the training, validation and testing portion of the data to produce respective embeddings. We then evaluated the quality of the embedding vectors with respect to three common downstream tasks (i) *imputation* which we approach as cross-modal translation, (ii) *clustering*, and (iii) *supervised learning* – performed across test sets. Finally, we evaluated models in the role of pre-trained encoders, i.e. with respect to their support for (iv) *transfer-learning*. Note, for each method we report the best model performance as achieved in the given task.

Imputation (modality translation) The task aims to estimate values of one variable (query) in the dataset given the value of the remaining variables (evidence). We iterated over test set data columns, permuting and subsequently estimating the values of a single "query" column, by selecting the value whose embedding maximized the similarities to the embeddings of the remaining variables. In the case of ICE-T, the estimate was done as described by Equation 4, whereas for the remaining models, the estimate was made by maximizing sum of the pairwise similarities. The estimates were then compared to the original values to evaluate the quality of recovery (i.e., imputation). We used mean squared error (MSE) to evaluate the quality of imputation of the numerical variables and balanced accuracy score (Brodersen et al. 2010) for the categorical ones. To obtain a unified score across all variables, the results were first scaled to the $[0, 1]$ interval, so that higher values indicate better performance, and then averaged into an *imputation score*. Note that Scarf and SubTab do not support modality translation and were excluded from this evaluation.

Clustering We evaluated how well the obtained embedding vectors support downstream data clustering. For each dataset, a regular k-means clustering model was trained on the training set embeddings. The quality of clusters was evaluated by the *silhouette score* (Shahapure and Nicholas 2020). The hyperparameter k controlling the number of clusters was selected to maximize the silhouette score on the validation set. The test set silhouette score was used as the

evaluation criterion. We also performed k-means clustering using the raw data, instead of embedding vectors, to serve as a shallow benchmark (not applied to txt/img-tabular data).

Supervised learning Similarly to clustering, we evaluated how well the obtained embeddings support downstream supervised learning. For each dataset, we trained a KNN classifier, or regressor, using the training set embeddings and the associated targets. Predictive performance was evaluated by the balanced accuracy score for classification tasks or MSE for regression tasks. Hyperparameter k was selected to maximize predictive performance on validation set. The resulting test set performance was then used as the evaluation criterion. We performed KNN using the raw data as a shallow benchmark (not applied to txt/img-tabular data).

Transfer learning Pre-trained models were used as an encoders, on top of which we added neural prediction head, implemented as a single hidden layer, ReLU-activated MLP, forming a neural predictor. The predictor was trained on the training set for up to 100 epochs using early stopping with patience set to 5 epochs and validation loss as the criterion. Once trained, we evaluated the predictive performance of the resulting model on the test set. The predictions were evaluated by the balanced accuracy score for classification tasks and MSE for regression tasks. As an additional control, we also employed a vanilla MLP predictor with a single hidden layer, i.e., without any pre-trained components. For benchmarking, we used the prediction performance achieved by XGBoost (Chen and Guestrin 2016) with default parametrization (not applied to txt/img-tabular data).

Average relative score The task-specific performance of ICE-T and other methods, as achieved on a given dataset, were scaled to the $[0, 1]$ interval so that the higher value of this *relative score* indicates better performance. To quantify the overall task-specific performance, the resulting values were subsequently averaged across datasets into an *average relative score* (cf. Supplementary Materials).

Results

Synthetic data experiment

Consider the dataset illustrated in Figure 2, referred to as Gaussian XOR "blobs", or noisy XOR (Duch 2007). It is evident that the conditional class probability cannot be factorized into a product of conditionals: $p(c|x, y) \neq p(c|x)p(c|y)$. Similar non-factorizability holds for the two coordinates: $p(x|c, y)$ and $p(y|c, x)$. Hence, we hypothesize that learning mappings f by pairwise similarity comparison across the three variables will result in poor embeddings, which will be manifested by a failure to predict any of the three variables from the embeddings of the remaining two.

To validate this hypothesis, we generated training and testing sets consisting of 1,000 and 100 noisy XOR points, respectively; and tested whether the embeddings obtained by ICE-T support the imputation task (modality translation) better than the embeddings from other multimodal contrastive methods. The obtained results show that ICE-T indeed greatly outperforms the other methods, confirming our hypothesis (cf. Table 4).

Method	ACC ↑ c	MSE ↓ x	MSE ↓ y	Imputation score
CLIP	0.750	0.271	0.106	0.562
GMC	0.683	0.206	0.122	0.565
MCN	0.733	0.179	0.148	0.589
ICE-T	**0.900**	**0.008**	**0.006**	**0.982**

Table 4: The performance of ICE-T compared to other multimodal CRL methods in the synthetic data experiment (cf. Figure 2). The embedding vectors produced by ICE-T allow to predict each of the three variables using the remaining two better than those from other methods, demonstrating its ability to capture cross-columnar interactions in tabular data.

	Imputation	Clustering	Supervised learning	Transfer learning
Scarf		**0.674**	0.434	0.474
SubTab		0.344	0.522	0.332
CLIP	0.561	0.345	0.571	0.544
GMC	0.536	0.553	0.495	0.544
MCN	0.378	0.306	0.636	0.533
ICE-T	**0.604**	0.617	**0.687**	**0.824**

Table 5: The average relative score (higher is better, cf. Evaluation) in the four tasks as achieved across the 48 real-world datasets. ICE-T gives the best performance in imputation, supervised and transfer learning; and is second in clustering. Note, Scarf and SubTab do not support modality transfer and could not be evaluated for imputation.

Real-world data experiments

For compactness, the results obtained across the 48 real-world datasets were conveyed as win-loss matrices (cf. Figure 3) and as a boxplots depicting the relative scores (cf. Figure 4) (the raw numbers are provided in Supplementary Tables 2–5). The overall performance across the datasets was quantified by average relative score (cf. Table 5). Based on the obtained results we summarize our findings as follows:

- Compared to other methods, embedding vectors produced by ICE-T give better support for imputation and supervised learning, but not clustering, for which Scarf provides a better alternative. This is likely because Scarf, unlike other methods, learns to exert embedding jointly by contrasting entire data instances, which may result in better global alignment of the embedding vectors.
- ICE-T serves in transfer learning better than other methods and transferring pre-trained ICE-T to neural predictor improves performance beyond that of XGBoost (benchmark). Interestingly, in similar experiments performed on independent tabular benchmark, Scarf surpassed XGBoost on approx. 60%, and the MLP used as a negative control on 45% of datasets (Bahri et al. 2021), strongly corroborating our results (cf. Figure 3, right).

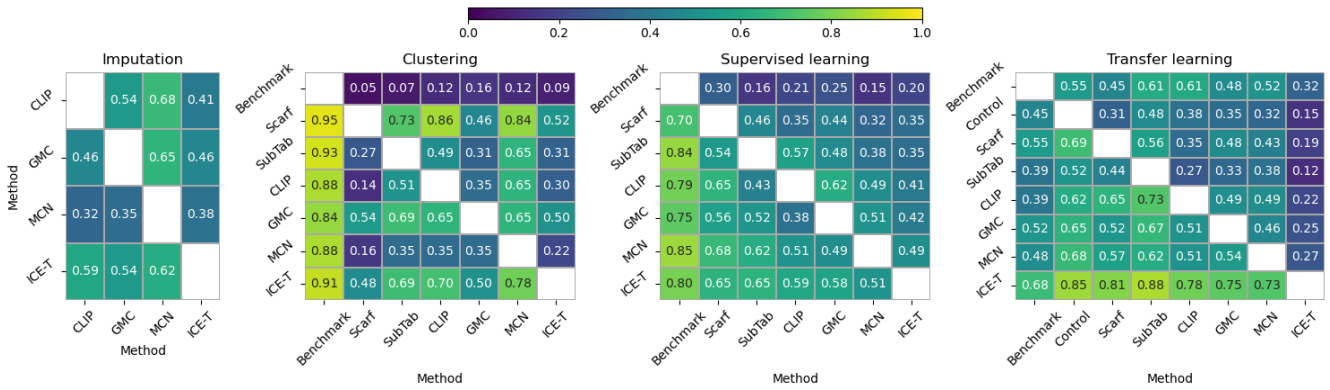

Figure 3: Heatmaps of win-loss matrices, where each value indicates the fraction of datasets where the method in the given row surpassed the method in the given column. Note that ICE-T (bottom row) surpassed other methods on the majority of the datasets (values > 0.5) in support of imputation, supervised and transfer learning; and performed reasonably well in clustering.

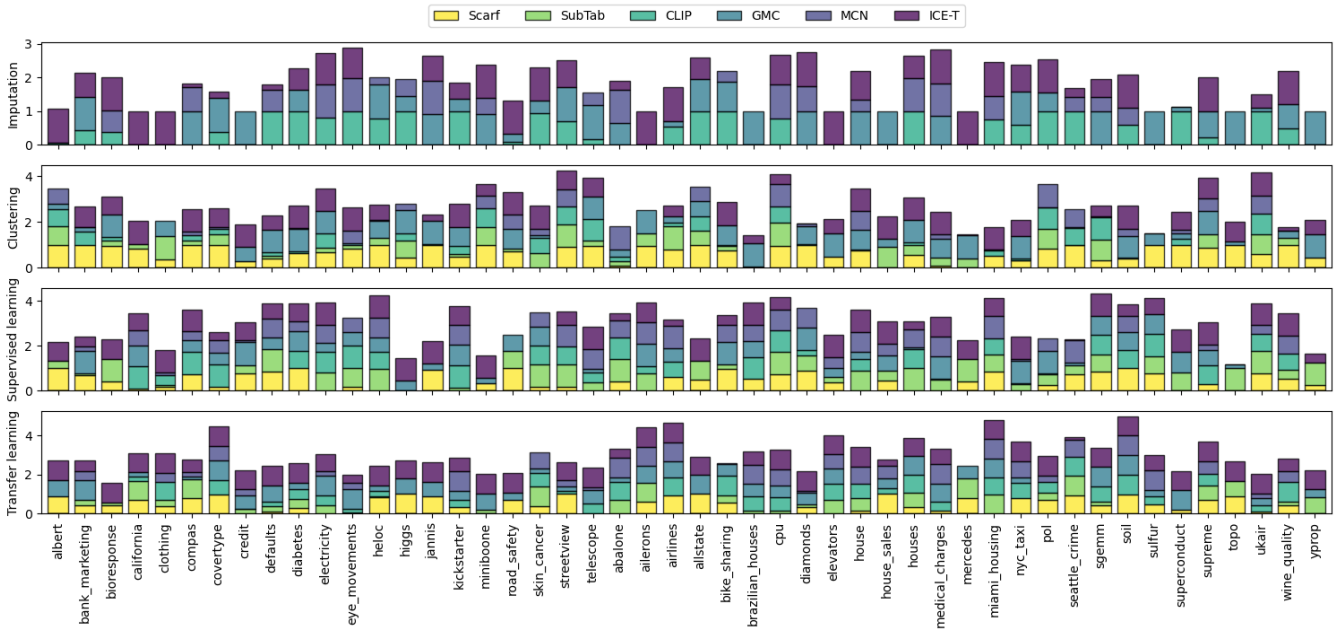

Figure 4: For each task, we scaled the results obtained on the given dataset to $[0, 1]$ interval, so that 1 is assigned to the best performing method, and 0 to the least performing method. The resulting quantities are depicted as segments in a stacked bar plot. A larger segment indicates better performance relative to other methods. As can be seen, ICE-T either outperforms other methods (the largest segment in the given bar), or generally provides good performance, across the majority of the datasets.

Discussion

We propose to approach heterogeneous tabular data as a type of multimodal data, where each variable, i.e., each tabular column, is treated as single modality. Multimodal CRL on tabular data has the potential to provide important advantages over existing methods.

However, unlike multimodal data, most tabular data are characterized by a large number of variables. This significantly penalizes methods using pairwise variable contrasting, which may become prohibitively expensive with growing number of variables. Moreover, in tabular data, interactions among variables are likely to occur.

This motivated ICE-T, whose *key insight is that its loss contrasts each modality with the embedding of a mean of intermediate representation of other modalities*, allowing it to perform in *linear time* and, as we demonstrated, to *capture variable interactions*. Based on our results, we conclude that ICE-T provides better support in most downstream tasks.

Supplementary materials and the code are available at https://github.com/tomastokar/ICET

Acknowledgements

This work was supported by Mitacs Industrial Upskilling Award (IT29286).

References

Alayrac, J.-B.; Recasens, A.; Schneider, R.; Arandjelović, R.; Ramapuram, J.; De Fauw, J.; Smaira, L.; Dieleman, S.; and Zisserman, A. 2020. Self-supervised multimodal versatile networks. *Advances in Neural Information Processing Systems*, 33: 25–37.

Bahri, D.; Jiang, H.; Tay, Y.; and Metzler, D. 2021. Scarf: Self-Supervised Contrastive Learning using Random Feature Corruption. In *International Conference on Learning Representations*.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828.

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

Brodersen, K. H.; Ong, C. S.; Stephan, K. E.; and Buhmann, J. M. 2010. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, 3121–3124. IEEE.

Chen, B.; Rouditchenko, A.; Duarte, K.; Kuehne, H.; Thomas, S.; Boggust, A.; Panda, R.; Kingsbury, B.; Feris, R.; Harwath, D.; et al. 2021. Multimodal clustering networks for self-supervised learning from unlabeled videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 8012–8021.

Chen, T.; and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Cohen-Addad, V.; Guedj, B.; Kanade, V.; and Rom, G. 2021. Online k-means clustering. In *International Conference on Artificial Intelligence and Statistics*, 1126–1134. PMLR.

Deldari, S.; Xue, H.; Saeed, A.; He, J.; Smith, D. V.; and Salim, F. D. 2022. Beyond just vision: A review on self-supervised representation learning on multimodal and temporal data. *arXiv preprint arXiv:2206.02353*.

Duch, W. 2007. Learning data structures with inherent complex logic: neurocognitive perspective. *Man-Machine Systems and Cybernetics (CIMMACS'07), Tenerife, Canary Islands, Spain*, 294–303.

Ericsson, L.; Gouk, H.; Loy, C. C.; and Hospedales, T. M. 2022. Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3): 42–62.

Grinsztajn, L.; Oyallon, E.; and Varoquaux, G. 2022. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35: 507–520.

Guo, W.; Wang, J.; and Wang, S. 2019. Deep multimodal representation learning: A survey. *Ieee Access*, 7: 63373–63394.

Guzhov, A.; Raue, F.; Hees, J.; and Dengel, A. 2022. Audioclip: Extending clip to image, text and audio. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 976–980. IEEE.

Kaur, P.; Pannu, H. S.; and Malhi, A. K. 2021. Comparative analysis on cross-modal information retrieval: A review. *Computer Science Review*, 39: 100336.

Le-Khac, P. H.; Healy, G.; and Smeaton, A. F. 2020. Contrastive representation learning: A framework and review. *Ieee Access*, 8: 193907–193934.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Poklukar, P.; Vasco, M.; Yin, H.; Melo, F. S.; Paiva, A.; and Kragic, D. 2022. Geometric multimodal contrastive representation learning. In *International Conference on Machine Learning*, 17782–17800. PMLR.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Shahapure, K. R.; and Nicholas, C. 2020. Cluster quality analysis using silhouette score. In *2020 IEEE 7th international conference on data science and advanced analytics (DSAA)*, 747–748. IEEE.

Shwartz-Ziv, R.; and Armon, A. 2022. Tabular data: Deep learning is not all you need. *Information Fusion*, 81: 84–90.

Ucar, T.; Hajiramezanali, E.; and Edwards, L. 2021. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34: 18853–18865.

Wang, L.; Luc, P.; Recasens, A.; Alayrac, J.-B.; and Oord, A. v. d. 2021. Multimodal self-supervised learning of general audio representations. *arXiv preprint arXiv:2104.12807*.

Zong, Y.; Mac Aodha, O.; and Hospedales, T. 2023. Self-supervised multimodal learning: A survey. *arXiv preprint arXiv:2304.01008*.

Supplementary Materials

1 Data preprocessing

The datasets that were used in our experiments were obtained from:

- `https://huggingface.co/datasets/inria-soda/tabular-benchmark`
- `https://huggingface.co/datasets/marmal88/skin_cancer`
- `https://huggingface.co/datasets/stochastic/random\_streetview_images_pano_v0.0.2`
- `https://huggingface.co/datasets/Censius-AI/ECommerce-Women-Clothing-Reviews`
- `https://huggingface.co/datasets/james-burton/kick_starter_funding`

using the Python library `datasets`. Categorical variables were numerically encoded using `OrdinalEncoder` (`sklearn`) and the selected numerical variables were processed by log-transformation, or min-max scaling using `MinMaxScaler` (`sklearn`) (cf. Supplementary Table 1). No additional data processing was performed.

2 Implementation details

2.1 Encoders

To ensure a fair comparison across the methods, in all our experiments, we employed identical encoders. The architecture of each encoder was tailored to a specific variable type. Unless stated otherwise, the encoders were implemented as follows:

- **Categorical encoder** was implemented as a single embedding layer with the embedding dimension controlled by hyperparameter `encoder_dim`.
- **Numerical encoder** was implemented as a ReLU-activated MLP, with the following number of neurons {1, `hidden_dim`, `encoder_dim`}.
- **Image encoder** was implemented as a pre-trained ResNet50 [3], followed by linear projection to reduce output dimension to `encoder_dim`;
- **Text encoder** was implemented as a pre-trained BERT transformer [2], followed by linear projection to reduce output dimension to `encoder_dim`. The BERT tokenizer was used to tokenize the inputs before passing them to the encoder.

2.2 Models

The intermediate representation vectors, once produced by the respective encoders, were then processed through method-specific downstream computations, which were implemented based on the information from the respective papers, and, if applicable, using the provided code.

Scarf Representation vectors were concatenated and subsequently passed to a neural subnetwork g (in the original paper referred to as the *pretraining head*). This subnetwork was implemented as a ReLU-activated MLP, with the number of neurons set to {`encoder_dim` $\times$ M, `encoder_dim`, `latent_dim`}; where is M is number of variables and `latent_dim` is hyper parameter denoting number of latent space dimensions. The contrastive loss was computed from the similarities between the resulting embeddings and the embeddings of the corrupted analogs. Note that when creating the corrupted views, unlike in the original implementation, we sampled from the batch marginals instead of dataset marginals. This is a minor change that allows substantially faster runtime when applied to image/text-tabular datasets.

SubTab Representation vectors were grouped into $k = 3$ overlapping blocks, with the 75% overlap. These blocks were subsequently concatenated and passed to a neural subnetwork E, implemented as a ReLU-activated MLP with the number of neurons set to {`encoder_dim` × M, `encoder_dim`, `latent_dim`} to produce block embeddings. The contrastive loss was computed from the similarities between the resulting embeddings; along the distance loss between the embeddings.

CLIP No further steps were applied. The intermediate representation vectors produced by encoders were used as the final embeddings, from which the pairwise contrastive loss was computed.

GMC Joint representation vectors were produced by passing inputs jointly through designated subnetwork $f^{(1:M)}$, which consisted of variable specific encoders analogous to those described above (without sharing weights), followed by linear projection to reduce dimensions to `encoder_dim`. Variable-specific and joint representation vectors were passed through *projection head* g, implemented as SiLU-activated (swished) MLP, with the number of neurons set to {`encoder_dim`, `encoder_dim`, `latent_dim`}, to produce their respective embeddings.

MCN The intermediate representation vectors produced by the encoders were averaged to, so-called, *fused multimodal features*, which were then subjected to online k-means clustering [1]. Parameter k of the online k-means clustering was set to `batch_size`/20, but not less than 2.

ICE-T The neural subnetwork g was implemented as a ReLU-activated MLP with the number of neurons set to {`phi_dim`, `phi_dim`, `latent_dim`}.

2.3 Hyperparameters

The hypermarameters used in our experiments include: `hidden_dim`, `encoder_dim`, `phi_dim`, `latent_dim`, `learning_rate` and `batch_size`. The range of values tried per hyperparameter across the datasets in our experiments are available in the code supplement (`/src/config.yml`). The performance resulting from each hyperparameters configuration across the datasets can be found in code supplement (`/results/dataset_name.csv`).

2.4 Hardware & Software

All our experiments were performed using multiple `NVIDIA RTX A6000` GPU cards, `CUDA v12.2`, on `Ubuntu v20.04.6 LTS`. All the code was written in Python programming language and was run using `Python v3.8.10`, `torch v2.1.0`, `numpy v1.24.4`, `sklearn v1.3.2` and `pandas v2.0.3`. For more details see the `requirements.txt` file in the code supplement.

2.5 ICE-T

```python
class ICETEmbedder(nn.Module):
    def __init__(self, mappings: dict, phi_dims: list, hidden_dim: int, latent_dim: int,
    tau_init: float = 0.07):
        super(ICETEmbedder, self).__init__()
        # Base mappings
        self.mappings = nn.ModuleDict(mappings)
        # Phi function
        self.phi = MLP(hidden_dim, latent_dim, phi_dims)
        # Temperature
        self.tau = nn.Parameter(torch.log(torch.tensor(tau_init)))

    def sim_func(self, x, y):
        a = F.normalize(x, dim = 1)
        b = F.normalize(y, dim = 1)
        sim = torch.matmul(a, b.T)
        sim = sim / torch.exp(self.tau)
        return sim
```

```
    def loss_func(self, sims):
        labs = torch.arange(sims.shape[0], device=sims.device)
        loss = F.cross_entropy(sims, labs, reduction='mean')
        return loss

    def calc_loss(self, H, h):
        # Sum latent representations
        H_sum = h * len(H)
        # Denominator
        k = len(H) - 1
        # Calc loss
        loss = 0.
        for hv in H.values():
            # Calc anchor
            mu = (H_sum - hv) / k
            # Project by phi
            zv, z_mu = self.phi(hv), self.phi(mu)
            # Calc similarities
            sims = self.sim_func(zv, z_mu)
            # Calc loss
            loss += self.loss_func(sims)
        return loss

    def transfer(self, query_var: str, candidates: torch.Tensor, evidence: dict):
        # Get latent representations of evidence
        H = [self.mappings[v](x) for v, x in evidence.items()]
        # Joint representation of evidence
        mu = sum(H) / len(H)
        # Get latent representation of query
        h = self.mappings[query_var](candidates)
        # Make projections
        z, z_mu = self.phi(h), self.phi(mu)
        # Calc similarity
        sim = self.sim_func(z, z_mu)
        # Select candidate value
        estimate = candidates[torch.argmax(sim, dim = 0)]
        return estimate

    def forward(self, X: dict, calc_loss = True):
        # Get interemdiate representations
        H = {v:mapping(X[v]) for v, mapping in self.mappings.items()}
        # Get joint intermediate representation
        h = sum([v for v in H.values()])/len(H)
        # Get joint representation
        z = self.phi(h)
        # Calculate loss
        if calc_loss:
            loss = self.calc_loss(H, h)
        else:
            loss = None
        return z, loss
```

Listing 1: Python implementation of ICE-T

3 Supplementary tables

Dataset	Processing step	Variable
airlines	min-max scaling	'CRSDepTime'
		'CRSArrTime'
		'Distance'
		'DepDelay'
allstate	min-max scaling	'cont1' – 'cont14'
		'loss'
brazilian_houses	log-transform	'hoa_(BRL)'
		'rent_amount_(BRL)'
		'property_tax_(BRL)'
		'fire_insurance_(BRL)'
		'total_(BRL)'
covertype	min-max scaling	'x1' – 'x10'
defaults	log-transform	'x1'
		'x18' – 'x23'
	min-max scaling	'x12' – 'x17'
house	log-transform	'P1'
houses	log-transform	'total rooms'
		'total bedrooms'
		'population'
		'households'
higgs	min-max scaling	'lepton_pT'
		'lepton_eta'
		'lepton_phi'
		'missing_energy_magnitude'
		'missing_energy_phi'
		'jet_1_pt' – 'jet_4_pt'
		'jet_1_eta' – 'jet_4_eta'
		'jet_1_phi' – 'jet_4_phi'
		'm_jj'
		'm_jjj'
		'm_lv'
		'm_jlv'
		'm_bb'
		'm_wbb'
		'm_wwbb'
medical_charges	log-transform	'Average Covered Charges'
		'Average Medicare Payments'
miniboone	min-max scaling	'ParticleID 19'
nyc_taxi	min-max scaling	'passenger_count'
		'tolls_amount'
		'total_amount'
		'tip_amount'
road_safety	min-max scaling	'Location_Easting_OSGR'
		'Location_Northing_OSGR'
	log-transform	'Engine_Capacity_(CC)'
soil	min-max scaling	'northing'
		'easting'
		'resistivity'
		'track'

Supplmentary Table 1: The numerical variables subjected to min-max scaling, or log-transformation.

	Method			
Dataset	CLIP	GMC	MCN	**ICE-T**
abalone	0.983	0.983	0.982	0.999
ailerons	0.797	0.887	0.732	0.844
airlines	0.814	0.782	0.838	0.869
albert		0.396		0.648
allstate		0.217		0.735
bank_marketing	0.722	0.974	0.900	0.755
bike_sharing	0.857	0.889	0.835	0.846
bioresponse		0.989		0.984
brazilian_houses	0.889	0.769	0.845	0.789
california	0.923	0.852	0.722	0.848
clothing	0.729	0.416	0.801	0.772
compas	0.966	0.596	0.960	0.937
covertype	0.937	0.955	0.887	0.868
cpu	0.954	0.920	0.923	0.891
credit	0.831	0.968	0.982	0.942
defaults	0.920	0.881	0.857	0.887
diabetes	0.751	0.978	0.866	0.998
diamonds	0.802	0.812	0.797	0.862
electricity	0.942	0.879	0.836	0.949
elevators	0.868	0.946	0.674	0.895
eye_movements	0.801	0.830	0.808	0.795
heloc	0.926	0.946	0.957	0.934
higgs		0.918		0.940
house	0.971	0.963	0.959	0.981
house_sales	0.886	0.876	0.674	0.810
houses	0.884	0.879	0.862	0.850
jannis		0.927		0.780
kickstarter	0.890	0.551	0.981	0.939
medical_charges	0.918	0.999	0.979	0.999
mercedes		0.046		0.730
miami_housing	0.655	0.933	0.753	0.887
miniboone		0.956		0.930
nyc_taxi	0.862	0.485	0.856	0.734
pol	0.872	0.963	0.973	0.977
road_safety		0.430		0.743
seattle_crime	0.984	0.936	0.982	1.000
sgemm	0.803	0.891	0.677	0.846
skin_cancer	0.992	0.773	0.504	0.986
soil	0.857	0.759	0.800	0.786
streetview	0.852	1.000	0.914	0.931
sulfur	0.950	0.877	0.937	0.998
superconduct		0.855		0.818
supreme	0.818	0.607	0.578	0.576
telescope	0.922	0.970	0.902	0.990
topo		0.977		0.895
ukair	0.766	0.643	0.628	0.684
wine_quality	0.916	0.938	0.867	0.967
yprop		0.970		0.961

Supplmentary Table 2: Imputation

	Method						
Dataset	Benchmark	Scarf	SubTab	CLIP	GMC	MCN	**ICE-T**
	Silhouette score - higher is better						
abalone	0.640	0.934	0.872	0.862	0.688	0.828	0.618
ailerons	0.518	0.749	0.569	0.676	0.603	0.570	0.730
airlines	0.242	0.851	0.441	0.358	0.888	0.280	0.742
albert	0.024	0.387	0.284		0.252		0.413
allstate	0.074	0.599	0.639		0.619		0.575
bank_marketing	0.831	0.916	0.827	0.830	0.806	0.824	0.916
bike_sharing	0.418	0.848	0.591	0.476	0.367	0.415	0.753
bioresponse	0.104	0.552	0.500		0.603		0.671
brazilian_houses	0.352	0.482	0.391	0.404	0.680	0.352	0.557
california	0.703	0.889	0.711	0.730	0.987	0.716	0.982
clothing		0.773	0.554	0.750	0.891	0.476	0.909
compas	0.555	0.761	0.566	0.521	0.546	0.686	0.812
covertype	0.329	0.863	0.490	0.319	0.725	0.339	0.687
cpu	0.379	0.473	0.571	0.437	0.642	0.429	0.343
credit	0.734	0.997	0.906	0.905	0.998	0.904	0.931
defaults	0.318	0.574	0.446	0.519	0.704	0.401	0.760
diabetes	0.429	0.644	0.554	0.576	0.253	0.459	0.463
diamonds	0.231	0.631	0.401	0.355	0.690	0.596	0.744
electricity	0.576	0.564	0.833	0.854	0.628	0.677	0.998
elevators	0.556	0.584	0.596	0.569	0.477	0.567	0.577
eye_movements	0.553	0.914	0.792	0.910	0.920	0.756	0.885
heloc	0.289	0.342	0.437	0.422	0.504	0.971	0.288
higgs	0.115	0.842	0.686		0.858		0.476
house	0.490	0.760	0.819	0.546	0.591	0.499	0.649
house_sales	0.947	0.957	0.945	0.947	0.947	0.946	0.927
houses	0.525	0.772	0.613	0.580	0.803	0.558	0.846
jannis	0.227	0.391	0.361		0.788		0.505
kickstarter		0.987	0.994	0.935	0.788	0.985	0.879
medical_charges	0.800	0.959	0.800	0.805	0.926	0.818	0.801
mercedes	0.135	0.474	0.187		0.771		0.554
miami_housing	0.347	0.761	0.430	0.410	0.815	0.786	0.876
miniboone	0.646	0.994	1.000		0.996		1.000
nyc_taxi	0.090	0.524	0.470	0.266	0.795	0.201	0.772
pol	0.173	0.333	0.417	0.305	0.581	0.359	0.629
road_safety	0.297	0.358	0.596		0.937		0.392
seattle_crime	0.185	0.326	0.208	0.267	0.212	0.211	0.442
sgemm	0.828	0.859	0.835	0.839	0.912	0.835	0.888
skin_cancer		0.969	0.971	0.988	0.849	0.993	0.849
soil	0.635	0.994	0.682	0.906	0.703	0.925	0.681
streetview		0.819	0.959	0.987	0.738	0.745	0.851
sulfur	0.489	0.620	0.533	0.545	0.738	0.607	0.754
superconduct	0.535	0.739	0.620		0.678		0.621
supreme	0.575	0.920	0.523	0.621	0.628	0.577	0.841
telescope	0.435	0.732	0.640	0.474	0.763	0.640	0.729
topo	0.304	0.864	0.496		0.558		0.813
ukair	0.701	0.640	0.831	0.873	0.107	0.793	0.968
wine_quality	0.510	0.691	0.576	0.526	0.576	0.526	0.556
yprop	0.288	0.478	0.343		0.638		0.531

Supplmentary Table 3: Clustering

	Method						
Dataset	Benchmark	Scarf	SubTab	CLIP	GMC	MCN	**ICE-T**
Balanced accuracy score (Classification) – higher is better							
albert	59.307	61.819	60.814		60.342		61.622
bank_marketing	75.664	76.729	75.497	75.409	77.277	75.792	76.276
bioresponse	72.862	69.453	73.011		66.829		72.173
california	62.433	67.873	66.104	82.291	80.639	77.441	78.223
clothing		40.803	40.202	42.743	39.700	40.375	46.294
compas	65.251	66.378	65.712	66.600	66.141	66.087	66.573
covertype	81.749	64.767	61.607	81.292	71.793	72.935	68.549
credit	56.977	73.162	67.234	61.209	76.824	62.900	73.311
defaults	66.983	70.741	71.138	68.380	69.777	70.743	70.205
diabetes	54.675	57.978	56.188	57.550	57.756	57.027	57.608
electricity	77.509	78.768	82.393	82.958	80.631	82.424	83.330
eye_movements	55.701	56.748	59.604	60.150	58.434	58.805	56.119
heloc	68.189	67.679	68.853	68.602	68.433	68.738	68.888
higgs	53.092	53.085	53.350		58.309		64.890
jannis	64.003	72.804	69.209		70.338		73.152
kickstarter		53.337	53.763	57.094	56.819	56.716	56.457
miniboone	87.114	89.863	89.041		89.521		91.398
road_safety	58.925	72.709	70.714		70.670		64.671
skin_cancer		61.266	65.955	64.830	65.062	63.886	60.173
streetview		87.954	90.913	90.062	87.260	90.789	89.480
telescope	77.112	77.287	78.645	78.934	77.969	80.584	81.035
MSE (Regression) – lower is better							
abalone	4.69957	5.31256	4.32163	4.37282	5.95043	4.72089	5.38935
ailerons	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
airlines	0.13671	0.01859	0.01871	0.01859	0.01860	0.01852	0.01865
allstate	0.05829	0.00331	0.00321		0.00344		0.00317
bike_sharing	21032.67723	10434.51400	12173.03800	12643.51400	10324.40200	10863.28400	11613.92900
brazilian_houses	0.00038	0.00051	0.00103	0.00013	0.00031	0.00033	0.00007
cpu	9.52330	8.45528	7.63315	7.82178	10.76382	7.83379	8.98616
diamonds	0.04037	0.02019	0.02396	0.03312	0.01756	0.02049	0.03846
elevators	0.00004	0.00004	0.00004	0.00004	0.00004	0.00003	0.00003
house	0.42876	0.53404	0.43986	0.47898	0.50120	0.44088	0.42927
house_sales	0.12313	0.11423	0.11395	0.11756	0.11229	0.11384	0.10987
houses	0.09530	0.15153	0.07202	0.08437	0.14501	0.08823	0.12270
medical_charges	0.02028	0.02144	0.01616	0.02122	0.01109	0.01232	0.01241
mercedes	75.09708	59.27377	43.83995		70.46438		48.39973
miami_housing	0.05099	0.03833	0.04331	0.04673	0.08245	0.03124	0.04209
nyc_taxi	0.06974	0.00572	0.00520	0.00568	0.00400	0.00554	0.00403
pol	60.49084	57.50370	50.97037	64.06667	35.07755	47.69167	65.23950
seattle_crime	160450.58896	156052.69000	156762.48000	157684.94000	157440.58000	155453.16000	157572.48000
sgemm	0.00024	0.00027	0.00027	0.00027	0.00027	0.00031	0.00026
soil	0.00598	0.00000	0.00016	0.00003	0.00003	0.00004	0.00008
sulfur	0.00037	0.00022	0.00022	0.00018	0.00020	0.00035	0.00022
superconduct	134.12197	130.40073	108.44905		104.98537		102.89780
supreme	0.00735	0.00633	0.00714	0.00495	0.00536	0.00657	0.00452
topo	0.00080	0.00082	0.00081		0.00082		0.00082
ukair	0.15621	0.16689	0.15702	0.16710	0.19882	0.18213	0.15919
wine_quality	0.64618	0.56382	0.57846	0.54574	0.62226	0.53675	0.51462
yprop	0.00078	0.00078	0.00074		0.00080		0.00077

Supplmentary Table 4: Supervised learning

	Method							
Dataset	Benchmark	Control	Scarf	SubTab	CLIP	GMC	MCN	**ICE-T**
Classification (ACC) – higher is better								
albert	65.047	65.181	65.560	65.379		65.550		65.584
bank_marketing	78.321	78.456	78.579	77.982	77.158	80.357	78.698	78.814
bioresponse	77.677	75.920	77.949	77.354		77.002		79.221
california	90.584	80.932	80.977	81.108	80.734	80.745	80.636	81.120
clothing		37.933	38.452	37.999	41.178	36.518	38.673	41.463
compas	66.789	68.589	68.838	68.992	68.296	68.382	68.463	68.742
covertype	85.289	87.615	92.624	77.242	88.715	92.715	88.649	92.806
credit	76.320	77.423	76.900	77.113	76.914	77.566	77.184	77.852
defaults	69.842	72.477	72.548	72.643	72.610	72.953	72.494	73.031
diabetes	60.404	61.024	60.931	60.991	60.998	60.943	60.848	61.146
electricity	90.827	81.659	81.807	82.873	83.039	84.259	82.416	83.893
eye_movements	66.645	56.760	55.920	56.078	56.279	57.923	56.544	56.799
heloc	70.902	69.512	70.944	69.955	70.279	70.234	69.930	71.108
higgs	73.538	73.706	74.985	72.168		74.447		74.796
jannis	78.862	78.066	79.597	77.164		79.143		79.855
kickstarter		63.774	62.200	60.216	62.569	63.055	66.431	64.686
miniboone	93.855	93.905	93.918	94.070		94.535		94.672
road_safety	78.225	77.763	79.140	77.843		78.534		79.690
skin_cancer		96.487	95.404	96.468	95.977	95.135	96.186	94.748
streetview		59.338	80.257	67.441	69.276	70.629	71.825	78.739
telescope	85.972	84.020	83.357	83.351	85.072	85.737	83.809	86.768
Regression (MSE) – lower is better								
abalone	5.05978	3.91145	4.05962	3.91939	3.86525	4.01733	3.85508	3.96275
ailerons	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
airlines	0.12965	0.01696	0.01693	0.01709	0.01693	0.01694	0.01693	0.01691
allstate	0.04847	0.00247	0.00242	0.00264		0.00242		0.00243
bike_sharing	10929.90035	9875.94700	9741.88400	9787.36900	9651.51000	9737.07600	9845.23000	9862.71600
brazilian_houses	0.00003	0.00008	0.00007	0.00007	0.00003	0.00004	0.00002	0.00004
cpu	6.33514	8.69058	8.53732	8.18825	6.96726	7.25649	6.67351	6.30104
diamonds	0.00830	0.01619	0.01366	0.01460	0.01511	0.01269	0.01467	0.01093
elevators	0.00001	0.00004	0.00005	0.00004	0.00004	0.00004	0.00004	0.00003
house	0.37756	0.47359	0.47892	0.45266	0.44590	0.48754	0.43751	0.43130
house_sales	0.03019	0.15205	0.08681	0.19229	0.15862	0.13981	0.12547	0.16002
houses	0.05121	0.11072	0.10974	0.10562	0.10344	0.10236	0.11376	0.10352
medical_charges	0.00779	0.00684	0.00699	0.00701	0.00693	0.00684	0.00683	0.00687
mercedes	69.05531	42.33849	43.48200	43.08921		43.74935		44.94259
miami_housing	0.02473	0.09113	1.01151	0.12189	0.22357	0.11804	0.09870	0.11192
nyc_taxi	0.05003	0.00272	0.00275	0.00363	0.00276	0.00336	0.00269	0.00253
pol	26.52296	38.61242	32.18196	37.99794	34.84072	45.15300	38.72781	26.15045
seattle_crime	147748.56894	146966.23000	147000.86000	146991.89000	146976.47000	147479.90000	147045.56000	147406.19000
sgemm	0.00029	0.00063	0.00053	0.00066	0.00037	0.00028	0.00073	0.00029
soil	0.00151	0.00001	0.00001	0.00081	0.00001	0.00001	0.00001	0.00001
sulfur	0.00091	0.00129	0.00101	0.00119	0.00104	0.00108	0.00082	0.00090
superconduct	98.50537	256.64734	246.62310	251.30844		229.29940		230.40335
supreme	0.00756	0.29500	0.29459	0.29417	0.29533	0.29988	0.29496	0.29220
topo	0.00083	0.00075	0.00075	0.00075		0.00077		0.00075
ukair	0.13846	0.14539	0.14442	0.14684	0.14124	0.13942	0.14231	0.12723
wine_quality	0.51409	0.51571	0.51537	0.51922	0.50477	0.52282	0.51265	0.51159
yprop	0.00086	0.00076	0.00077	0.00076		0.00076		0.00075

Supplmentary Table 5: Transfer learning

4 Statistical analysis

We tested the statistical significance of the obtained results under the alternative hypothesis that the performance obtained by the ICE-T across the 48 datasets, is greater than the one obtained by the baseline methods collectively, using the one sided T-test. The obtained p-values (lower the better) are summarized in the table below. The results show that ICE-T outperforms the baselines very significantly ($p < 0.01$) in three out of four tasks and with lower significance ($p = 0.065$) in one of the tasks.

Task	**p-value**	**Statistic**
Imputation	6.506e-02	1.528
Clustering	6.488e-03	2.552
Supervised learning	3.003e-03	2.832
Transfer learning	4.415e-10	6.921

Supplmentary Table 6: Results of the statistical evaluation.

5 Scalability statement

We would like to emphasize that certain extremely high-dimensional tabular datasets may remain computationally challenging even despite ICE-T's favorable scaling. However, this limitation reflects a general trade-off for all CRL methods, which employ column, or modality-specific embeddings. We emphasize that ICE-T was successfully tested on datasets with several hundred columns and up to 1 million rows (Table 3 in the manuscript), which we believe is representative of the majority of real-world tabular datasets commonly encountered in practice.

References

[1] Vincent Cohen-Addad et al. "Online k-means clustering". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1126–1134.

[2] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[3] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.