Rethinking of Encoder-based Warm-start Methods in Hyperparameter Optimization

Dawid Płudowski[1,*] Antoni Zajko[1,*] Anna Kozak[1] Katarzyna Woźnica[1]

[*]Equal contribution.
[1]Warsaw University of Technology

Abstract Effectively representing heterogeneous tabular datasets for meta-learning purposes remains an open problem. Previous approaches rely on predefined meta-features, for example, statistical measures or landmarkers. The emergence of dataset encoders opens new possibilities for the extraction of meta-features because they do not involve any handmade design. Moreover, they are proven to generate dataset representations with desired spatial properties. In this research, we evaluate an encoder-based approach to one of the most established meta-tasks – warm-starting of the Bayesian Hyperparameter Optimization. To broaden our analysis we introduce a new approach for representation learning on tabular data based on [Iwata and Kumagai, 2020]. The validation on over 100 datasets from UCI and an independent metaMIMIC set of datasets highlights the nuanced challenges in representation learning. We show that general representations may not suffice for some meta-tasks where requirements are not explicitly considered during extraction.

1 Introduction

The meta-learning for tabular data poses a unique challenge for the machine learning community as there is no universal way to compare two datasets. Yet, such data remains ubiquitous across various domains [Davenport and Kalakota, 2019, Alanazi, 2022]. To resolve this issue, new methods for data representation were developed [Vanschoren, 2019, Jomaa et al., 2021].

Problem motivation. So far existing approaches to datasets' representation rely primarily on handcrafted meta-features which are often based on statistical measures, information theory, or landmarkers [Rivolli et al., 2022]. The emergence of encoders tailored to heterogeneous datasets solved this problem by limiting the need for manually chosen meta-features since they only require a loss function that formalizes the desired properties of the representations. One of the first encoder-based approaches was Dataset2Vec [Jomaa et al., 2021] which introduced:

Requirement 1. *Ensuring close representations for batches of observations within the same dataset while maintaining distinct representations for observations across different datasets.*

By imposing this condition, the authors assume that the resulting representations will reflect the internal complex structure of the data. This is, however, not explicit whether such representations are useful in meta-learning tasks, like Bayesian Optimization (BO).

Contributions. In this work, we: (1) conducted a comprehensive analysis of encoder-based representations using two distinct sets of datasets on the BO warm-start problem, (2) proposed new dataset encoder which is rooted in Requirement 1 and inspired by the few-shot architecture introduced in [Iwata and Kumagai, 2020].

The validation of encoder-based representations' applicability in meta-tasks highlights the nuanced challenges in representation learning. **Merely adhering to Requirement 1 through encoder-based representations proves to be inadequate for broader meta-learning.**

2 Problem setting

In this section, we provide the formulation of the problem of employing dataset representation in meta-learning tasks. In Figure 1, we show the defined components and their workflow. First, we train encoder ϕ on OpenML's subset $\mathcal{D}_{rep}$. The encoder is either *liltab* or D2V. Next, we create representations of datasets from the train and test set $\mathcal{D}_{train}, \mathcal{D}_{test}$. The origin of these datasets is either the UCI repository or metaMIMIC, depending on the experiment setting. In the next step, we create a mapping between the datasets from $\mathcal{D}_{train}$ and the corresponding best configurations of hyperparameters (HP) for the predictive algorithm, elastic net, and XGBoost. Having this mapping and the distances from a single dataset from test set $\mathcal{D}_i$ to datasets from $\mathcal{D}_{train}$, we run BO with warm-start chosen among best configurations for 10 closest datasets from $\mathcal{D}_{train}$.

As shown in the Figure 1, our experiment setting follows a two-step process of solving meta-tasks provided in [Jomaa et al., 2021]: (1) finding the representation in the common vector space of heterogeneous tabular datasets, (2) using representation as an input to the meta-model which solves the meta-task.

We use the BO warm-start in our experiments due to its popularity in research [Bergstra et al., 2011a, Nicoletta Del Buono et al., 2020, Xiyuan Liu et al., 2020, Victoria and Maragatham, 2021] and its significance in the speeding up the BO process. The obtained results are compared with baselines that do not use a data structure to propose a candidate for warm-start.
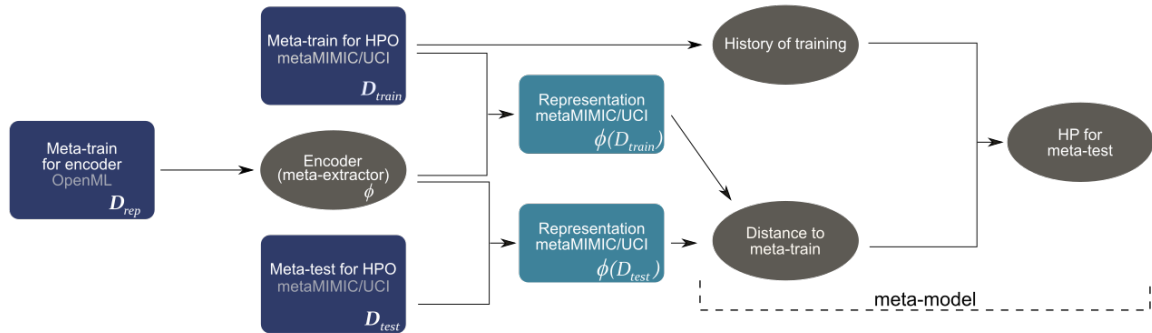

Figure 1: The workflow of the performed experiment. We denote datasets using dark blue color, their representations using light blue, and all other components using grey.

3 Considered approaches

Next to the Dataset2Vec, we use *liltab*-based encoder which meets Requirement 1. Both work on heterogeneous data, which is crucial when working with tabular data. Their performance is contrasted with two baseline methods independent of dataset representations.

3.1 Encoders

Dataset2Vec (D2V) is an encoder of the whole dataset proposed in [Jomaa et al., 2021]. It is one of the approaches to encoding heterogeneous tabular data that uses neural networks instead of statistical measures. It is a DeepSet-based method [Zaheer et al., 2017] composed of feed-forward neural networks. It extracts interdependencies between features and targets and joint distributions inside the data in a three-step process which results in fixed-size encoding. In addition to D2V, we propose the encoder based on [Iwata and Kumagai, 2020]. The crucial feature of Iwata's architecture is the ability to perform predictions on heterogeneous tabular data in the few-shot setting. It is obtained by using an inference network as an encoder for the support set. In our package *liltab*, we implemented both the original architecture from [Iwata and Kumagai, 2020] and the modified inference network as an encoder. For training purposes, we used a contrastive learning approach. For a more detailed description, see Appendix A.

3.2 Baselines

As baselines for the usability of encoder-based representations, we use two methods: BO without warm-start and warm-start based on the ranks of HP configurations. We define ranks by the number of tasks in which the HP configuration was the best. When calculating ranks we only treat the given configuration as the best one for the specific task when it is strictly the best among all considered settings i.e. there is only a single best task.

4 Experiments

We select three sets of datasets for the experiment purposes. All of them contain only binary classification tasks: (1) OpenML's subset [Vanschoren et al., 2013] used in [Iwata and Kumagai, 2020], (2) UCI's subset [Markelle Kelly, 2021] used in [Jomaa et al., 2021], (3) Over 5000 few-shot tasks generated from metaMIMIC [Woźnica et al., 2023] collected from physionet.org. Their usage is presented in Figure 1. The detailed description can be found in the Appendices B and D.

We train both encoders on OpenML datasets. The details of the training process are listed in Appendix E. Full reproduction of the training encoders, as well as the following experiments, can be done with code on GitHub[1].

We evaluate the quality of the representations in two ways: visually using T-SNE [Van der Maaten and Hinton, 2008] and quantitatively using two metrics. One of them is the accuracy in the meta-task of classification whether two samples (both in terms of rows and columns) originate from the same dataset. The other one is Caliński-Harabasz (CH) index [Caliński, 1974] where in the place of the cluster labels we put the label of the origin of the dataset's batch. This measures how far from each other are representations depending on the origin dataset of the batch. Having these metrics, we measure the ability of encoders to distinguish between datasets and their representations' spatial properties. Therefore, they reflect meeting the Requirement 1.

Next, we assess the applicability of the meta-extractors in the HPO warm-start meta-task. To achieve that we performed optimizations with encoder-based warm-start, heuristic-based warmstart, and random initialization. The gain of specific approaches was computed using Average Distance To the Minimum (ADTM) for the scaled ROC-AUC score which resulted in separate metrics for each iteration of the HPO. The experiment was performed on data from the UCI repository and metaMIMIC. We use cross-validation concerning the selection of validation datasets. For metaMIMIC and UCI data, we use 4 folds and 5 folds respectively. Each optimization consists of 30 iterations with 10 warm-start iterations.

4.1 Experimental setup

During our research on the warm-start, we focus on searching for the best HP for the elastic net [Pedregosa et al., 2011] and XGBoost [Chen and Guestrin, 2016] classifiers. For the experiment's purpose, we generate a random set of the hyperparameters of the elastic net and XGBoost, each consisting of 100 configurations. We use XGBoost only on UCI datasets, as metaMIMIC tasks would be too small for this model. The definition of the ranges and distributions of the HP are shown in Appendix F. Then, we train and evaluate the model with every configuration of HP on each task. For encoder-based approaches as warm-start points, we select the best configuration on the 10 datasets closest in terms of distance function $\| \cdot \|_2$. For the selection of the rest of the points, we use the Tree-structured Parzen Estimator [Bergstra et al., 2011b].

5 Results

This section presents two different assessment methods of tabular data representation. The first of them is the similarity of latent vectors corresponding to similar datasets, i.e., meeting Requirement 1. The second one is the metrics obtained from warm-start in HP optimization.

---

[1]https://github.com/azoz01/rethinking_encoder_warmstart

5.1 Encodings similarity of datasets

In Figure 2, we show the T-SNE representation of vectors obtained by *liltab* and D2V. For this purpose, we choose three tasks from the metaMIMIC dataset where two of them - `hypertensive` and `diabetes` share 7 out of 10 features and third - `hypotension` which has only 3 features in common with the rest. Each point on this figure corresponds to the validation set of the single few-shot task generated from metaMIMIC. The analogous results are present for UCI datasets in Appendix G. In Table 1 we present corresponding metrics. Both encoders perform similarly, clustering the majority of datasets very well which indicates that they fulfill Requirement 1.
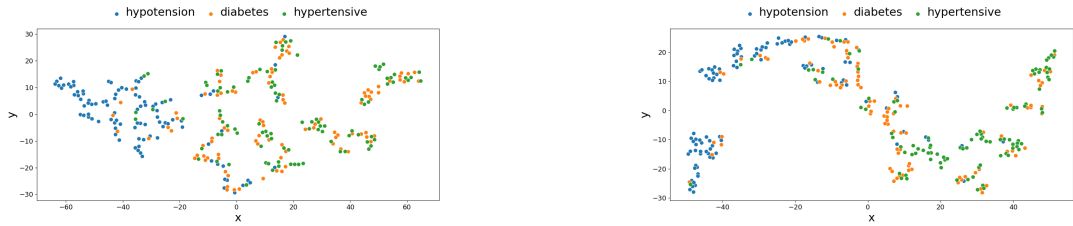
(a) The representations obtained with *liltab*.

(b) The representations obtained with D2V.

Figure 2: T-SNE visualization of three encoded datasets' representations.

Table 1: Metrics that assess the quality of representations generated by encoders. Each evaluation was repeated 15 times. We present the averaged values. Accuracy above 0.5 indicates the ability to distinguish between datasets to some extent.

Encoder	Set of datasets	Accuracy	CH index
D2V	metaMIMIC	0.59 ± 0.006	64.85 ± 5.03
liltab	metaMIMIC	0.58 ± 0.007	70.77 ± 2.90
D2V	UCI	0.72 ± 0.006	241.27 ± 5.10
liltab	UCI	0.65 ± 0.008	136.37 ± 8.55

5.2 Transferability of HP

To determine the quality of the proposed warm-start, we present on Figure 3a trajectories of ADTM for each cross validation fold with averages and confidence intervals for each considered method. In Figures 3b and 3c we show the statistical significance of our results aggregated across all folds on critical distance plots. There is a significant difference in the results of the specific methods. It can be noticed that the rank method is significantly better than other methods. Moreover, both encoders are indistinguishable in terms of performance on the 10th and 30th iteration. Corresponding plots for UCI are presented in Appendix G. In all cases, baselines either were statistically undistinguishable from encoder-based approaches or even outperformed them. **This contradicts the assumption that the similarity of general datasets' representations implies that they share the best HP configurations**.

6 Conclusion

In our study, we provide results showing that using encoder-based representations has no significant gain over simpler methods in the task of choosing warm-start points in BO. Moreover, our results show that meta-learning methods do not necessarily outperform simple random initialization. To make the evidence stronger, we propose a new method to extract data representation from the model proposed in [Iwata and Kumagai, 2020]. We evaluate its performance and compare it to
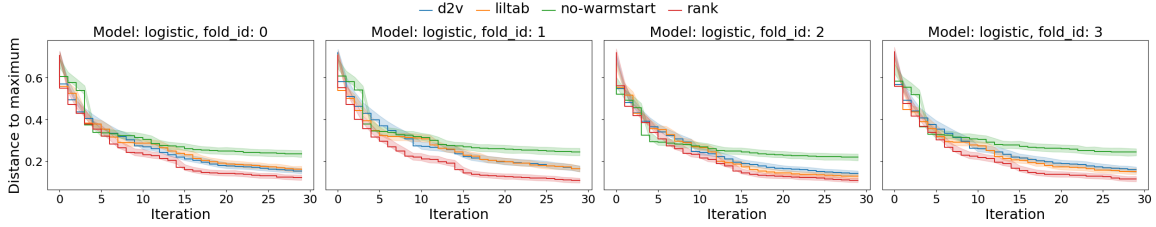
(a) Plots of ADTM of metaMIMIC data HPO with warm-start comparison. A lower score means the performance of a method is closer to the best score obtained on a specific dataset on average.

(b) Critical distance plot for logistic regression in 10th iteration.

(c) Critical distance plot for logistic regression in 30th iteration.

Figure 3: Results for the elastic net on metaMIMIC. Each fold in ADTM plots on Figure 3a presents results for a specific split of datasets to train and validate the subset. Here, the rank method results in the lowest average distance to the maximum ROC-AUC score for specific datasets. The position on the Critical distance on Figures 3b and 3c scale denotes test statistic in the Friedman test. Methods that are connected with horizontal lines are statistically indistinguishable.

Dataset2Vec, an established dataset encoder. Both of them met Requirement 1. The experiments show that they have comparable results on most warm-start tasks and both of them yield no significant improvement in considered meta-task.

7 Limitation and Broader Impact

Here, we want to emphasize some of the limitations of our work. Additionally, we summarize the potential future directions in this field.

We only used two encoders because few can work on heterogeneous data. However, considering the benchmark provided in [Zhu et al., 2023], we choose the two best models that work on varying features and target spaces. To perform experiments, we used three significantly different sets of datasets. Despite their versatility, there is a need to re-perform experiments on a more extensive data portfolio to ensure independence of the results and the data. Concerning reproducibility, one of the sets cannot be published (metaMIMIC) but detailed instructions to reproduce are provided in original paper [Woźnica et al., 2023].

We believe that creating new heuristics and encoders is the next direction in this study. The results shown in this paper suggest that this field needs to be explored more. In particular, there is a great need to create encoders or heuristic-based methods that perform in all tasks better than random sampling starting points.

Social impact statement
After careful reflection, we have determined that this work presents no notable negative impacts on society or the environment.

References

Tomoharu Iwata and Atsutoshi Kumagai. Meta-learning from Tasks with Heterogeneous Attribute Spaces. In *Advances in Neural Information Processing Systems*, volume 33, pages 6053–6063, 2020.

Thomas Davenport and Ravi Kalakota. The potential for artificial intelligence in healthcare. *Future Healthcare Journal*, 6(2):94–98, 2019. doi: 10.7861/futurehosp.6-2-94.

Abdullah Alanazi. Using machine learning for healthcare challenges and opportunities. *Informatics in Medicine Unlocked*, 30:100924, 2022. doi: 10.1016/j.imu.2022.100924.

Joaquin Vanschoren. Meta-learning. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Automated Machine Learning: Methods, Systems, Challenges*, pages 35–61. Springer International Publishing, 2019. doi: 10.1007/978-3-030-05318-5_2.

Hadi S Jomaa, Lars Schmidt-Thieme, and Josif Grabocka. Dataset2Vec: Learning Dataset Meta-Features. *Data Mining and Knowledge Discovery*, 35:964–985, 2021. doi: https://doi.org/10.1007/s10618-021-00737-9.

Adriano Rivolli, Luís P. F. Garcia, Carlos Soares, Joaquin Vanschoren, and André C. P. L. F. de Carvalho. Meta-features for meta-learning. *Knowledge-Based Systems*, 240:108101, 2022. doi: 10.1016/j.knosys.2021.108101.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, volume 24, 2011a.

Nicoletta Del Buono, Flavia Esposito, and Laura Selicato. Methods for Hyperparameters Optimization in Learning Approaches: An Overview. pages 100–112, 2020. doi: 10.1007/978-3-030-64583-0_11.

Xiyuan Liu, XiYuan Liu, Jia Wu, Jia Wu, Senpeng Chen, and SenPeng Chen. Efficient Hyperparameters optimization Through Model-based Reinforcement Learning and Meta-Learning. *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1036–1041, 2020. doi: 10.1109/hpcc-smartcity-dss50907.2020.00139.

A Helen Victoria and Ganesh Maragatham. Automatic tuning of hyperparameters using Bayesian optimization. *Evolving Systems*, 12(1):217–223, 2021. doi: 10.1007/s12530-020-09345-2.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198.

Kolby Nottingham Markelle Kelly, Rachel Longjohn. UCI Machine Learning Repository, 2021. URL https://archive.ics.uci.edu.

Katarzyna Woźnica, Mateusz Grzyb, Zuzanna Trafas, and Przemysław Biecek. Consolidated learning: a domain-specific model-free optimization strategy with validation on metaMIMIC benchmarks. *Machine Learning*, pages 1–25, 2023. doi: 10.1007/s10994-023-06359-0. Publisher: Springer.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

Harabasz Caliński. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 1974. doi: 10.1080/03610927408827101.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. doi: 10.1145/2939672.2939785.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011b. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

Max Zhu, Katarzyna Kobalczyk, Andrija Petrovic, Mladen Nikolic, Mihaela van der Schaar, Boris Delibasic, and Petro Lio. Tabular few-shot generalization across heterogeneous feature spaces. *arXiv preprint arXiv:2311.10051*, 2023.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep Metric Learning via Lifted Structured Feature Embedding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4004–4012, 2015.

Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. MIMIC-IV, 2023.

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes]

   (c) Did you discuss any potential negative societal impacts of your work? [Yes]

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? https://2022.automl.cc/ethics-accessibility/ [Yes]

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [Yes]

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [Yes]

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes]

   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [Yes]

   (e) Did you report the statistical significance of your results? [Yes]

   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes]

(g) Did you compare performance over time and describe how you selected the maximum duration? [No] The methods presented in this paper are meant to be significantly faster than single evaluation in Bayes Optimization and thus, we do not set time but iteration limit.

(h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

(i) Did you run ablation studies to assess the impact of different components of your approach? [No] We do not propose any new method in this paper except simple heuristic that are too simple to perform ablation studies on them.

3. With respect to the code used to obtain your results...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [No] We fulfilled all requirements from this points except providing all data, as metaMIMIC dataset cannot be published.

(b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes]

(c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes]

(d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes]

(e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes]

4. If you used existing assets (e.g., code, data, models)...

(a) Did you cite the creators of used assets? [Yes]

(b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [Yes]

(c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]

5. If you created/released new assets (e.g., code, data, models)...

(a) Did you mention the license of the new assets (e.g., as part of your code submission)? [N/A]

(b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [N/A]

6. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

7. If you included theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

A Liltab's representations of datasets

Here, we describe representations generated by the *liltab* which is part of the architecture introduced [Iwata and Kumagai, 2020], namely the inference network. Please note that in the original paper, there are three parts of the proposed network while we show only the first two of them. This is because the second part creates a representation for each of the records in the dataset. On the contrary, the representations produced by the third part are created attribute-wise and as a result, are harder to interpret. In the original paper, the whole architecture was designed to serve in few-shot learning tasks. However, we do not use any of the few-shot learning features in our work so we do not use terms related to this technique in the description below.

A.1 Network architecture

Let's denote $[X, Y]$ as task, x_n as n-th observation in it and x_{ni} as i-th attribute of n-th observation. In our research, we focus only on the binary target so $y_n \in \{0, 1\}$ denotes n-th target value. We use here N as the number of observations in the task and I as the number of attributes so $n = 1, \ldots, N$ and $i = 1, \ldots, I$.

Let's denote feed-forward neural networks as $f_\odot$ when their function is to encode information about attributes and targets and $g_\odot$ when their function is to aggregate the outputs of $f_\odot$ networks. Vector $\bar{v}_i$ and scalar $\bar{c}_i$ store information about the marginal distribution of each attribute and a target. Their values are obtained by feed forward networks $f_{\bar{v}}$, $f_{\bar{c}}$, $g_{\bar{v}}$, $g_{\bar{c}}$, which is presented in Figure 4. The formal formulation of this step is presented below:

$$\bar{v}_i = g_{\bar{v}} \left(\frac{1}{N} \sum_{n=1}^{N} f_{\bar{v}}(x_{ni}) \right), \quad \bar{c} = g_{\bar{c}} \left(\frac{1}{N} \sum_{n=1}^{N} f_{\bar{c}}(y_n) \right). \tag{1}$$

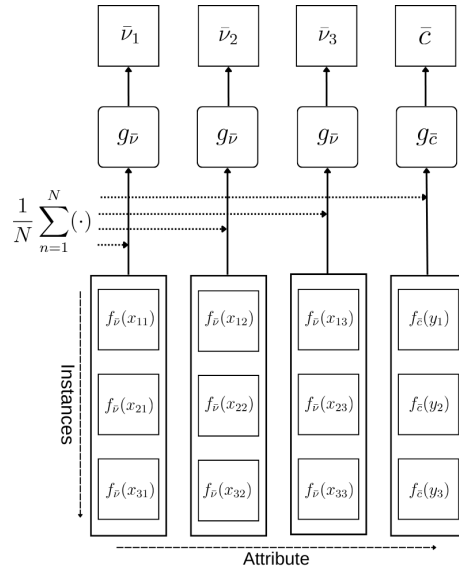Figure 4: First step of the inference network. Here, the inference network learns about the empirical marginal distributions of the attributes based on the support set.

Further encoding u_n provides the possibility to provide additional information about relationships between attributes and targets. What is more, this representation is created for each observation (as shown in Figure 5). This step can be formulated as below:

$$u_n = g_u \left(\frac{1}{I} \sum_{i=1}^{I} f_u([\bar{v}_i, x_{ni}]) + f_u([\bar{c}, y_n]) \right), \qquad (2)$$

where $[\cdot]$ denotes the concatenation. Please note that the representations $u = (u_1, \ldots, u_N)$ is in $\mathbb{R}^{N \times p}$, where p denotes the size of representation and is treated as the architecture's hyperparameter. The presented architecture can store information about marginal distributions and interactions between variables in the dataset that is provided on input. Thus, by providing an appropriate loss function, one can train such a model to distinguish between different datasets or subsets of datasets.
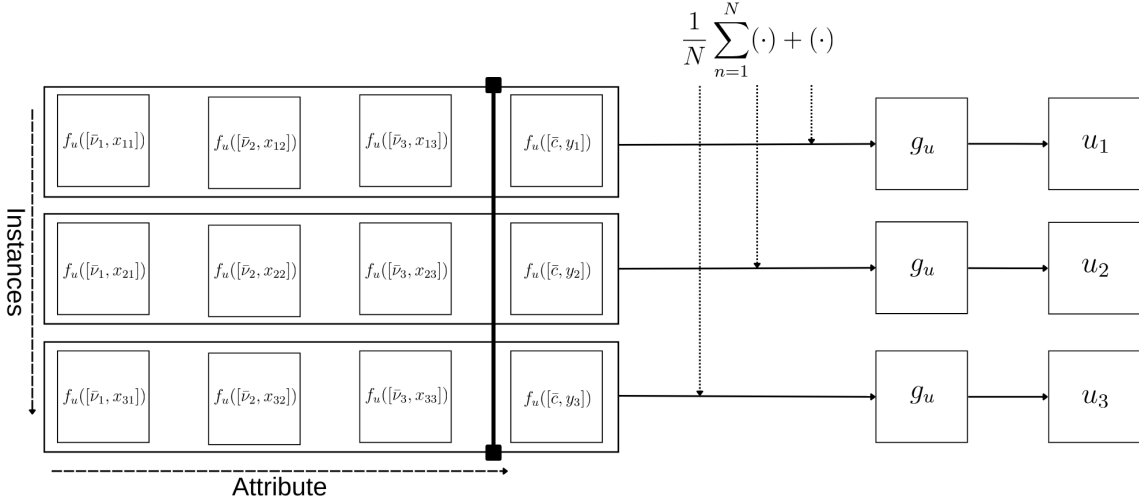

Figure 5: Second step of the inference network. Here, the inference network learns the relationships between the attributes and the target based on the support set.

A.2 Network training

During the original training procedure, the authors of [Iwata and Kumagai, 2020] wanted to maximize its predictive performance. However, representations produced by an inference network occur to be uninformative on real-world data and often are vectors with few non-zero coordinates. To overcome that issue, we propose a new encoder-based technique that targets this part of the model.

In our training procedure, each batch of data is created from samples from several different datasets, where the numbers of samples for each dataset are equal. Next, observations from one dataset in the batch are encoded to their representation. During the calculation of the loss function, we focus on representations of specific observations and train the network to produce encodings that are close to each other when they correspond to the same dataset and far away when they relate to different datasets. To achieve that, we propose the contrastive learning approach with loss function inspired by [Song et al., 2015]:

$$loss = \frac{1}{|\mathcal{N}| + |\mathcal{S}|} \left(\sum_{i,j \in \mathcal{S}} \|\phi(x_i) - \phi(x_j)\|_2 - \sum_{i,j \in \mathcal{N}} \|\phi(x_i) - \phi(x_j)\|_2 \right),$$

where x_i denotes specific observation, ϕ is a *liltab*'s encoder function, $\mathcal{S}$ is a set of pairs of indices of observations that belong to the same dataset, and $\mathcal{N}$ denotes pairs of indices of observations that are from different datasets. Note that we index by observations, not by subsets in batch.

B Data

The set of datasets from OpenML are binary classification tasks created from regression ones when a classification target is obtained as an indicator of whether the regression target is above or below the average. The datasets are preprocessed and standardized, so the distribution of ones and zeros is equal. A complete list of these datasets is in Appendix C.

The next set of data is supplied from the GitHub repository of D2V. Some of the datasets are multilabel classification tasks with several classes varying from 2 to 10. However, we decided to cast classes so that there are only 2 classes in every task. The reason for such modification is that the tasks on which the encoders are trained are binary only. We also remove datasets with more than 10000 observations or 20 features. Appendix D shows a complete list of used datasets.

The last set of datasets is prepared on the metaMIMIC datasets [Woźnica et al., 2023]. It contains fully preprocessed data from [Johnson et al., 2023], with 12 binary classification tasks, each describing the presence of a specific disease. As tasks originally contain more than 100 features, selecting the 10 most important for each task provides the ability to measure the similarity of tasks by simply calculating the fraction of shared most significant features. During our study, we would like to consider several small datasets, as it is the preferred data type for *liltab*. Due to that fact, we split each metaMIMIC task into multiple smaller tasks. For every small task used in the warm-start evaluation, we specify 4 observations as the training set and 29 observations as the validation set. Both training and validation sets are sampled equally concerning the target value. With this methodology, we yielded over 5000 small tasks in total. MetaMIMIC has not yet been used in this domain, especially as the source of small subsets of data. However, it is a promising type of set of tasks for the dataset similarity as each task has some common features with the others.

C Datasets from OpenML

Table 2: Datasets from OpenML used for encoder training.

name	number of rows	number of columns
diggle_table_a1	48	5
fri_c4_100_25	100	26
chatfield_4	235	13
edm	154	18
vineyard	52	3
pollution	60	16
fri_c3_100_10	100	11
visualizing_hamster	73	6
pyrim	74	28
visualizing_slope	44	4
hip	54	8
chscase_geyser1	222	3
rabe_148	66	6
bodyfat	252	15
chscase_vine1	52	10
sleuth_ex1714	47	8
echocardiogram-uci	132	8
hutsof99_child_witness	42	16
fri_c3_250_10	250	11
chscase_funds	185	2
qsartox	16	24
qsabr2	15	10
sleep	62	8
longley	16	7
heart	270	14
mu284	284	10
ICU	200	20
qsbralks	13	22
analcatdata_uktrainacc	31	16
slump	103	10
rabe_131	50	6
gascons	27	5
rabe_265	51	7
rabe_166	40	2
baskball	96	5
bolts	40	7
chscase_demand	27	11
qsfsr2	19	10
fri_c2_100_5	100	6
sleuth_ex1605	62	6
qsfsr1	20	10
visualizing_ethanol	88	3

name	number of rows	number of columns
EgyptianSkulls	150	5
transplant	131	3
pwLinear	200	11
diabetes_numeric	43	3
autoPrice	159	16
treepipit	86	10
branin	225	3
machine_cpu	209	7
detroit	13	14
fri_c2_250_25	250	26
qsbr_y2	25	10
humans_numeric	75	15
visualizing_environmental	111	4
USCrime	47	14
rabe_176	70	4
fri_c0_250_5	250	6
rabe_266	120	3

D Datasets from UCI

Table 3: Datasets from UCI used for HPO warmstart evaluation.

name	number of rows	number of columns
chess-krvkp	3196	37
post-operative	90	9
contrac	1473	10
pittsburg-bridges-REL-L	103	8
seeds	210	8
ecoli	336	8
acute-nephritis	120	7
haberman-survival	306	4
cylinder-bands	512	36
pittsburg-bridges-TYPE	105	8
ringnorm	7400	21
tic-tac-toe	958	10
led-display	1000	8
hill-valley	1212	101
waveform	5000	22
credit-approval	690	16
dermatology	366	35
statlog-heart	270	14
echocardiogram	131	11
thyroid	7200	22
planning	182	13
spect	265	23
musk-1	476	167
wall-following	5456	25
ozone	2536	73
hayes-roth	160	4
breast-cancer-wisc-prog	198	34
yeast	1484	9
energy-y2	768	9
wine-quality-red	1599	12
teaching	151	6
musk-2	6598	167
zoo	101	17
statlog-landsat	6435	37
molec-biol-promoter	106	58
heart-switzerland	123	13
congressional-voting	435	17
statlog-image	2310	19
oocytes_trisopterus_states_5b	912	33
oocytes_merluccius_nucleus_4d	1022	42
wine-quality-white	4898	12
mushroom	8124	22

name	number of rows	number of columns
conn-bench-sonar-mines-rocks	208	61
vertebral-column-3clases	310	7
heart-hungarian	294	13
low-res-spect	531	101
semeion	1593	257
iris	150	5
ilpd-indian-liver	583	10
optical	5620	63
horse-colic	368	26
waveform-noise	5000	41
lenses	24	5
glass	214	10
mammographic	961	6
twonorm	7400	21
balance-scale	625	5
abalone	4177	9
pittsburg-bridges-T-OR-D	102	8
bank	4521	17
hepatitis	155	20
breast-cancer-wisc	699	10
ionosphere	351	34
flags	194	29
image-segmentation	2310	19
breast-tissue	106	10
monks-2	601	7
pittsburg-bridges-SPAN	92	8
trains	10	30
heart-cleveland	303	14
spectf	267	45
page-blocks	5473	11
statlog-vehicle	846	19
monks-3	554	7
blood	748	5
oocytes_merluccius_states_2f	1022	26
statlog-australian-credit	690	15
energy-y1	768	9
heart-va	200	13
steel-plates	1941	28
breast-cancer	286	10
lung-cancer	32	57
wine	178	14
spambase	4601	58
oocytes_trisopterus_nucleus_2f	912	26
pima	768	9
annealing	898	32
breast-cancer-wisc-diag	569	31

name	number of rows	number of columns
lymphography	148	19
car	1728	7
cardiotocography-10clases	2126	22
titanic	2201	4
acute-inflammation	120	7
pittsburg-bridges-MATERIAL	106	8
monks-1	556	7
fertility	100	10
balloons	16	5
vertebral-column-2clases	310	7
synthetic-control	600	61
parkinsons	195	23
statlog-german-credit	1000	25
molec-biol-splice	3190	61
cardiotocography-3clases	2126	22

E Hyperparameters of encoders

Table 4: Hyperparameters of *liltab* encoder.

Hyperparameter	Description	Value
num_epochs	Maximum number of epochs	100000
learning_rate	Learning rate	0.0001
weight_decay	Weight decay	0
batch_size	Batch size	37
early_stopping_epochs	Epochs without loss decrease before stop of training	2500
hidden_representation_size	Dimensionality of output of intermediate networks	32
n_hidden_layers	Number of hidden layers in intermediate networks	3
hidden_size	Size of hidden layers in intermediate networks	32
dropout_rate	Dropout rate in all intermediate networks	0.1

Table 5: Hyperparameters of Dataset2Vec encoder.

| Hyperparameter	Description	Value
gamma	Scaling factor in metric-based classification	1
num_epochs	Maximum number of epochs	100000
learning_rate	Learning rate	0.001
weight_decay	Weight decay	0.0001
batch_size	Batch size	16
train_n_batches	Number of batches in one epoch	100
early_stopping_epochs	Epochs without loss decrease before stop of training	500
f_dense_hidden_size	Size of input/output layers in residual block in first layer	16
f_res_hidden_size	Size of hidden layers in residual block in first layer	16
f_res_n_hidden	Number of hidden layers in residual block in first layer	3
f_dense_out_hidden_size	Output size of first layer	16
f_block_repetitions	Number of residual blocks in first layer	3
g_layers_sizes	Sizes of hidden layers in second layer	[32, 16, 8]
h_dense_hidden_size	Size of input/output layers in residual block in third layer	32
h_res_hidden_size	Size of hidden layers in residual block in third layer	32
h_res_n_hidden	Number of hidden layers in residual block in third layer	3
h_dense_out_hidden_size	Output size of last layer	16
h_block_repetitions	Number of residual blocks in third layer	3

F Hyperparameters grids used for HPO

Table 6: Hyperparameters grid used for HPO of elastic net algorithm.

Condition	Hyperparameter	Values' range	Distribution
n/a	tol	[0.0001, 0.001]	loguniform
n/a	C	[0.0001, 10000]	loguniform
n/a	solver	[lbfgs, liblinear, newton-cg, newton-cholesky, sag, saga]	categorical
solver = liblinear	intercept scaling	[0.001, 1]	uniform
solver = liblinear	penalty	[l1, l2]	categorical
solver = liblinear and penalty = l2	dual	[true, false]	categorical
solver = saga	penalty	[elasticnet, l1, l2, null]	categorical
solver = saga	l1 ratio	[0, 1]	uniform
solver ≠ saga and solver ≠ liblinear	penalty	[l2, null]	categorical

Table 7: Hyperparameters grid used for HPO of XGBoost algorithm.

| Hyperparameter	Values' range	Distribution
no of estimators	[1, 1000]	uniform (int)
learning rate	[0, 1]	uniform
booster	[gblinear, gbtree]	categorical
subsample	[0.5, 1]	uniform
max depth	[6, 15]	uniform (int)
min child weight	[2, 256]	uniform
colsample bytree	[0.2, 1]	uniform
colsample bylevel	[0.2, 1]	uniform

G Results on the UCI datasets
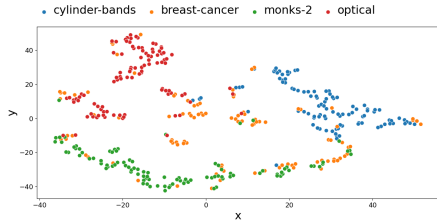
(a) The representation obtained with *liltab*.
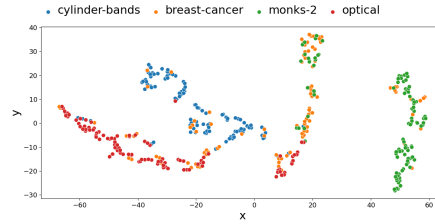

(b) The representation obtained with D2V.

Figure 6: T-SNE visualization of four encoded datasets' representations. Each point represents an encoded subset of data sampled from one of the datasets.
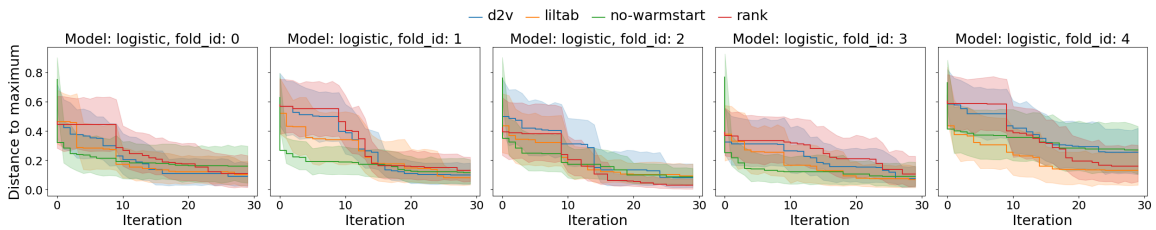
(a) Plots of ADTM of UCI data HPO with warm-start comparison. A lower score means the performance of a method is closer to the best score obtained on a specific dataset on average.
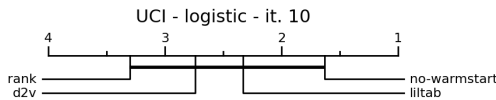

(b) Critical distance plot for logistic regression in 10th iteration.
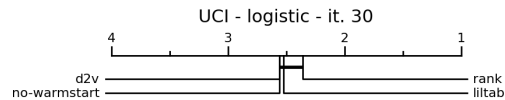
(c) Critical distance plot for logistic regression in 30th iteration.

Figure 7: Results for UCI datasets, elastic net. Each fold in ADTM plots on Figure 7a presents results for a specific split of datasets to train and validation the subset. The position on the critical distance scale in Figures 7b and 7c denotes test statistics in the Friedman test. Methods that are connected with horizontal lines are statistically indistinguishable. It shows that the difference in performance between methods is statistically irrelevant.
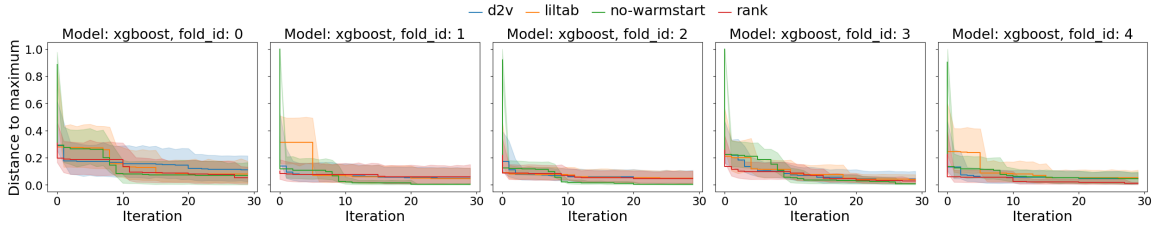
(a) Plots of ADTM of UCI data HPO with warm-start comparison. A lower score means the performance of a method is closer to the best score obtained on a specific dataset on average.

(b) Critical distance plot for XGBoost in 10th iteration.

(c) Critical distance plot for XGBoost in 30th iteration.

Figure 8: Results for UCI, XGBoost. Each fold in ADTM plots on Figure 8a presents results for a specific split of datasets to train and validation the subset. The position on the critical distance scale in Figures 8b and 8c denotes test statistics in the Friedman test. Methods that are connected with horizontal lines are statistically indistinguishable. It shows that the difference in performance between methods is statistically irrelevant.