DiffPrep: Differentiable Data Preprocessing Pipeline Search for Learning over Tabular Data

IMTIAZ ALI, Otto-Friedrich University Bamberg, Germany

A well-designed data Preprocessing improves the quality of data that is used for the machine learning model, which makes it important to carefully craft a data preprocessing pipeline, a good design requires extensive knowledge and application development experience. Existing AutoML systems to some extent help to reduce the efforts but they are not only time-consuming but also have limited space to search for a better pipeline. DiffPrep offers a wider range of search space and delivers pipelines in a shorter time than all the existing systems. with more search space DiffPrep ensures that the pipeline selected generates high-quality preprocessed data and eventually results in maximum ML model performance.

CCS Concepts: • **Information systems** → **Data cleaning**; • **Computing methodologies** → **Machine learning**;

Additional Key Words and Phrases: Data Preprocessing, AutoML, Transformation type, Transformation Order, Transformation Operator, Gradient Descent

ACM Reference Format:
Imtiaz Ali. 2023. DiffPrep: Differentiable Data Preprocessing Pipeline Search for Learning over Tabular Data. *ACM Trans. Graph.* 37, 4, Article 111 (December 2023), 4 pages.

1 INTRODUCTION

In recent developments in Machine learning technologies more small and large-scale applications are coming out with time, each machine learning application passes through four different phases namely Data Acquisition, Data Preprocessing, Model Training, and Model testing. the weight of each of these processes is equal to the final output of the application. Data acquisition involves collecting data from different data sources using the latest data-warehouse capabilities and data collection tools it has been improved to the greatest extent. Data preprocessing helps to eliminate inconsistencies and to clean the data so overall application does not misbehave due to some too much to be true values. whereas in model training the data from preprocessing is split into test and training data sets and fed into the model for training of machine learning model which applies some model to data and helps to recognize a more incoming set of features efficiently.

Before data is passed to the Machine learning model it is looked for transformation types. *transformation types* defined as existing errors and discrepancies in the dataset [3] that might lead to inaccuracies in the model evaluation phase it includes missing value imputation, Normalization, outliers removal, and discretization. once

Author's address: Imtiaz Ali, imtiaz.ali@stud.uni-bamberg.de, Otto-Friedrich University Bamberg, Bamberg, Bavaria, Germany, 96047.

ACM 0730-0301/2023/12-ART111
https://doi.org/

Transformation Types	Transformation Operators	
Missing Value Imputation	Numerical Features	Mean Median Mode
	Categorical Features	Most Frequent Value Dummy Variable
Normalization	Standardization Min-Max Scaling Robust Scaling Max Absolute Scaling	
Outlier Removal	Z-Score (k) MAD (k) IQR (k)	
Discretization	Uniform (n) Quantile (n)	

Fig. 1. Data Transformation [8]

the transformation types are identified the transformation operator for data is selected. *transformation operator* is a statistical tool that is used to handle the transformation type such as for Missing value imputation if the data is numerical data mean, median, the mode can be used if the data is categorical data most frequent value can add similarly for Normalization the operator could be one of standardization, Min-Max scaling and so on. and in such a pattern *transformation order* can be defined as the order in which the transformation types are handled, each decision in data preprocessing contributes to the overall accuracy of application so if normalization is performed before handling the outliers the preprocessed data set could be different then when normalization is performed after and it can also lead to different performance results of the final application. fig. 2 shows how a typical data preprocessing pipeline.

several transformation operators for the same transformation type and fixing the order which leads us to a well preprocessed data leading to maximum performance is a difficult design decision. AutoML systems have been introduced to reduce the efforts of deciding the pipeline. autoML system takes in the dataset as an input and generates preprocessed data as an output. several such types of existing systems come with certain accuracy and limitations.

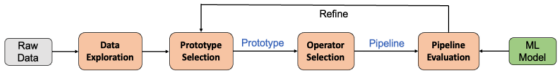

Fig. 2. Data Preprocessing Pipeline [8]

2 LIMITATIONS

Current systems that exist to solve the complicated design decision of defining the preprocessing pipeline include H20, Azure [1],

AutoSklearn [5], and Learn2Clean [2] have reduced the human effort to some extent but still come with some limitations, A search space [4]is defined as several parameters a system handles while deciding the pipeline. H2O [7] considers fixed transformation type, transformation operator, and order which leads H2O to shorter search space. H2O gives us preprocessed data but it applies the same preprocessing pipeline for all sorts of features. Azure considers the transformation operator and depending on the dataset it selects the most efficient operator, this widens the search space of Azure which means it needs to have as many pipelines as the number of operators exists for each transformation type. similarly, AutoSklearn provides flexibility in terms of both transformation type and transformation operator, the combination means that the search space is even bigger.

Furthermore, as the complexity grows the need to have better optimization methods arise that don't need to be trained each time and are fast with growing parameters. H2O uses Random search which is a computationally expensive method. Azure and AutoSklearn utilize Bayesian optimization which requires to be trained every time.

2.1 Limited Search space complexity

The systems cover parameters including transformation type, transformation order, and transformation operator to generate a pipeline, but for each feature or column of data same pipeline is applied. since we know that every feature is different and may or may not require a pipeline or may require a different pipeline because a different operator chosen might have a better effect. a separate pipeline for each feature might have a good impact on the overall performance of machine learning applications this leads to the conclusion that search space needs to be increased to consider the features as well. Search space has a direct proportion to preprocessed data.

Size of Search Space: Smallest ↓ Largest

Systems	Operator	Types	Order	Feature-wise	Optimization Method
H2O	×	×	×	×	Random Search
Azure	✓	×	×	×	Bayesian Optimization
Auto-Sklearn	✓	✓	×	×	Bayesian Optimization
Learn2Clean	✓	✓	✓	×	Q-Learning
DiffPrep-Fix	✓	✓	×	✓	**Bi-level Optimization with Gradient Descent**
DiffPrep-Flex	✓	✓	✓	✓	

Fig. 3. Existing AutoML systems [8]

2.2 Low Efficiency on optimization methods

As the search space grows optimization techniques such as Random search and Bayesian optimization become inefficient. The curse of dimensionality in the Bayesian method exponentially increases the amount of search space as the number of parameters grows. Optimization grows even more if the features are added to search space, if there are p possible pipelines to one feature in total there will be f to the power of p possible pipelines.

3 PROPOSAL

Cutting down the limitation there is a need to increase the search space as the existing system already covers the 3 parameters (transformation type, order, operator) introducing *DiffPrep* which will in addition to given parameters add the features into search space as well. so each feature will have the most effective pipeline.

As the features introduce more complexity to search space traditional methods become inefficient to handle the increased search space so *DiffPrep* uses a gradient-based optimization method to find the best pipeline among the search space.

4 PROBLEM STATEMENT

First of all, let's formulate the problem as Bi-level optimization, Bi-level optimization is an optimization framework that contains two levels of decision-making where one level serves as a constraint or input to the other level.

In this problem upper part we try to minimize the validation loss while the inner level helps to minimize the training loss.

$$\text{Outer level:} \quad \underset{pipline}{\text{argmin}}\ Loss\ (D_{val}, pipeline, model^*)$$

$$\text{Inner level:} \quad s.t.\ model^* = \underset{model}{\text{argmin}}\ Loss\ (D_{train}, pipeline, model)$$

"Bi-level Optimization"

Fig. 4. Bi level optimization [8]

As the search space is discrete and nondifferentiable it's hard to evaluate large search space.

5 PROPOSED SOLUTION

The key idea of this paper is to provide a way to transform the data into continuous and differentiable so that Gradient Descent [] can be applied. so to transform the data into discrete and differentiable ones we start with Parameterize the pipeline which converts the pipeline into a matrix and further relaxes the matrix so we get some continuous and differentiable data as a result.

5.1 Parametrization

So let's assume β_{ij} defines the selected operator for a specific transformation type. each transformation type T_i has m operators that can be applied T_i={ $f_{i,1}$, $f_{i,2}$, $f_{i,3}$, ... $f_{i,m}$ } and so for each transformation type there is a set of possible operators to chose from and if operator is chosen its value is 1 and 0 otherwise.

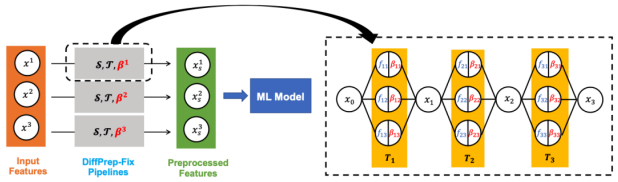

Fig. 5. DiffPrep with Fixed Order (DiffPrep-Fix) [8]

$$\beta = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{otherwise} \end{cases}$$

Fig 3. is mapped to a matrix where each column is a transformation type and every row is a transformation operator, maintaining a constraint which implies that the sum of each column is always 1 $\sum_{ij} = 1$.

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

5.2 Relaxation

As beta parameter allowed us to parameterize the search space but the data is still discrete and nondifferentiable. so to convert the data to continuous and differentiable softmax function is applied (the Softmax function converts the data column/vector from discrete to continuous maintaining the constraint)

$$\beta_{ij} = \frac{\exp(\tau_{ij})}{\sum_{k=1}^{n} \exp(\tau ik)}$$

5.3 Bi-level Optimization

As β is a continuous differential bi-level optimization can now be solved using gradient descent instead of enumerating all possible pipelines. the objective is to minimize the validation loss and parameters β for optimal pipeline
The equation for validation loss by τ is
$\boldsymbol{\tau} = \boldsymbol{\tau} - N_1 \nabla_{\boldsymbol{\tau}} \mathcal{L}_{\text{val}}(\beta(\boldsymbol{\tau}), \mathbf{w}^*)$, where N_1 is the learning rate, w is the parameters for model. to obtain optimal parameters (w^*) the inner optimization problem has to be solved first.
inner optimization is given as $\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}_{\text{train}}(\beta_1, \ldots, \beta_c, \mathbf{w})$ to solve this problem we use one-step gradient descent on the current existing parameters:
$\mathbf{w}^* \approx \mathbf{w}' = \mathbf{w} - N^2 \nabla_{\mathbf{w}} \mathcal{L}_{\text{train}}(\beta(\boldsymbol{\Theta}), \mathbf{w})$ where N_2 is the model training learning rate.
The summarized algorithm for fixed order is given in Figure 6.

Algorithm 2 DiffPrep-Fix

Require: Space of TF types and operators S, pre-defined prototype $\mathcal{T}$, training set D_{train}, validation set D_{val}
Ensure: Optimal pipeline parameters $\boldsymbol{\beta}$ and model parameters $\boldsymbol{w}$
1: Initialize $\boldsymbol{\tau}$ and $\boldsymbol{w}$
2: **while** *not converged* **do**
3: Fit TF operators on the transformed training data
4: Forward Propagation: Compute $L_{train}(\boldsymbol{\beta}, \boldsymbol{w}^+)$, $L_{train}(\boldsymbol{\beta}, \boldsymbol{w}^-)$, $L_{val}(\boldsymbol{\beta}, \boldsymbol{w}')$
5: Backward Propagation: Compute $\nabla_{\boldsymbol{\beta}} L_{train}(\boldsymbol{\beta}, \boldsymbol{w}^+)$, $\nabla_{\boldsymbol{\beta}} L_{train}(\boldsymbol{\beta}, \boldsymbol{w}^-)$, $\nabla_{\boldsymbol{\beta}} L_{val}(\boldsymbol{\beta}, \boldsymbol{w}')$
6: Compute $\nabla_{\boldsymbol{\tau}} L_{val}(\boldsymbol{\beta}, \boldsymbol{w}')$
7: Update $\boldsymbol{\tau}$: $\boldsymbol{\tau} = \boldsymbol{\tau} - \eta_1 \nabla_{\boldsymbol{\tau}} L_{val}(\boldsymbol{\beta}, \boldsymbol{w}')$
8: Update $\boldsymbol{w}$: $\boldsymbol{w} = \boldsymbol{w} - \eta_2 \nabla_{\boldsymbol{w}} L_{train}(\boldsymbol{\beta}(\boldsymbol{\tau}), \boldsymbol{w})$
9: **return** $\boldsymbol{\beta}(\boldsymbol{\tau}), \boldsymbol{w}$

Fig. 6. DiffPrep Algorithm [8]

also for DiffPrep-Flex similar approach has been followed and to choose the order in flex the same method is applied for order and parameterization and Relaxation is done and solved according to the gradient approach, and the algorithm for flexible order DiffPrep is given in Fig 7.

Algorithm 3 DiffPrep-Flex

Require: Space of TF types and operators S, training set D_{train}, validation set D_{val}
Ensure: Optimal prototype parameters $\boldsymbol{\alpha}$, pipeline parameters $\boldsymbol{\beta}$, and model parameters $\boldsymbol{w}$
1: Initialize $\boldsymbol{\theta}$, $\boldsymbol{\tau}$ and $\boldsymbol{w}$
2: **while** *not converged* **do**
3: Fit TF operators on the transformed training data
4: Forward Propagation: Compute $L_{train}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}^+)$, $L_{train}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}^-)$, $L_{val}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}')$
5: Backward Propagation: Compute $\nabla_{\boldsymbol{\alpha}} L_{train}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}^+)$,$\nabla_{\boldsymbol{\alpha}} L_{train}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}^-)$, $\nabla_{\boldsymbol{\alpha}} L_{val}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}')$, $\nabla_{\boldsymbol{\beta}} L_{train}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}^+)$, $\nabla_{\boldsymbol{\beta}} L_{train}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}^-)$, $\nabla_{\boldsymbol{\beta}} L_{val}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}')$
6: Compute $\nabla_{\boldsymbol{\tau}} L_{val}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}')$, $\nabla_{\boldsymbol{\theta}} L_{val}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}')$
7: Update $\boldsymbol{\tau}, \boldsymbol{\theta}$: $\boldsymbol{\tau} = \boldsymbol{\tau} - \eta_1 \nabla_{\boldsymbol{\tau}} L_{val}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}')$, $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta_1 \nabla_{\boldsymbol{\theta}} L_{val}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}')$
8: Update $\boldsymbol{w}$: $\boldsymbol{w} = \boldsymbol{w} - \eta_2 \nabla_{\boldsymbol{w}} L_{train}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w})$
9: **return** $\boldsymbol{\alpha}(\boldsymbol{\theta}), \boldsymbol{\beta}(\boldsymbol{\tau}), \boldsymbol{w}$

Fig. 7. DiffPrep-Flex [8]

5.4 General Architecture for DiffPrep

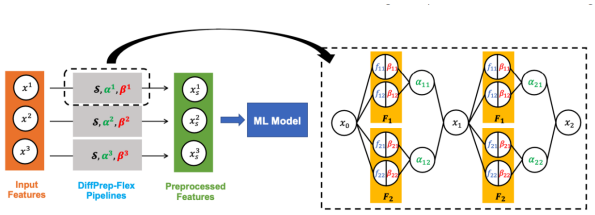

Fig. 8. DiffPrep-Flex Architecture [8]

6 EVALUATION RESULTS

Both the proposed solutions DiffPrep-Fix and DiffPre-Flex have been evaluated against other AutoML software systems to measure the effect of Preprocessed data on the overall performance of the machine-learning model.

6.1 Experiment setup

This experiment is conducted on an Intel-based environment more specifically (2.20GHz Intel Xeon(R) Gold 5120 CPU), with around 18 diverse datasets taken from openly available OpenML data repository. Furthermore, the Machine learning model was fixed to Logistic regression [6], and as optimization types are vast so is the Optimization operator. so the optimization operators are fixed to ones easily available such as (Missing value imputation, Normalization, Outlier removal, and Discretization). and it's compared to all the related existing systems AutoSklearn, Learn2Clean, Boost2Clean. The table shows the size, number of classes also number of outliers, and missing values detected using Z score. and dataset is randomly split into ratio of training (40%), validation (20%) and test (20 %).

6.2 Accuracy

14 out of 18 experiments have shown that the preprocessed data from DiffPrep performed better, and as the table in Fig 7 suggests the average increase in test accuracy among the systems compared to DiffPrep is around 6.6 percent.

Dataset	Data Characteristics					Test Accuracy						
	#Ex.	#Feat.	#classes	#MVs	#Out.	DEF	RS	AS	LC	BC	DP-Fix	DP-Flex
abalone	4177	9	28	0	200	0.24	0.243	0.216	0.186	0.168	0.238	**0.255**
ada_prior	4562	15	2	88	423	0.848	0.844	0.853	0.816	0.848	**0.854**	0.846
avila	20867	11	12	0	4458	0.553	0.598	0.615	0.597	0.585	**0.638**	0.63
connect-4	67557	43	3	0	45873	0.659	0.671	0.667	0.658	0.69	**0.732**	0.701
eeg	14980	15	2	0	209	0.589	0.658	0.657	0.641	0.659	**0.678**	0.677
google	9367	9	2	1639	109	0.586	0.627	**0.664**	0.549	0.616	0.645	0.641
house	1460	81	2	6965	617	0.928	0.938	**0.945**	0.812	0.928	0.932	**0.945**
jungle_chess	44819	7	3	0	0	0.668	0.669	0.678	0.676	0.667	**0.682**	**0.682**
micro	20000	21	5	0	8122	0.564	0.579	0.584	0.582	0.561	0.586	**0.588**
mozilla4	15545	6	2	0	290	0.855	0.922	**0.931**	0.854	0.93	0.923	0.922
obesity	2111	17	7	0	25	0.775	0.841	0.737	0.723	0.652	0.893	**0.896**
page-blocks	5473	11	5	0	1011	0.942	0.959	**0.969**	0.92	0.951	0.957	0.967
pbcseq	1945	19	2	1445	99	0.71	0.73	0.712	0.704	0.72	0.725	**0.743**
pol	15000	49	2	0	8754	0.884	0.879	0.877	0.737	0.903	0.904	**0.919**
run_or_walk	88588	7	2	0	8548	0.719	0.829	0.851	0.728	0.835	0.907	**0.917**
shuttle	58000	10	7	0	5341	0.964	0.996	0.998	0.997	0.997	**0.998**	0.997
uscensus	32561	15	2	4262	2812	0.848	0.84	0.851	0.786	0.848	**0.857**	0.852
wall-robot-nav	5456	25	4	0	1871	0.697	0.872	0.869	0.69	0.9	0.898	**0.914**

Fig. 9. Accuracy test [8]

6.3 Run Time Performance

As other systems were using Bayesian the run time for them is quite long opposite to DiffPrep. Fig 8 shows the red circles showing that in each bucket of data, DiffPrep takes almost half the time as the next best available optimization method.

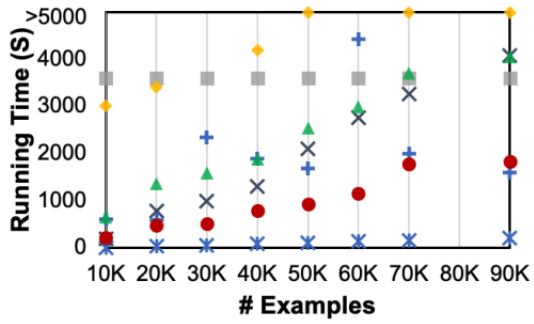

Fig. 10. Linear Regression [8]

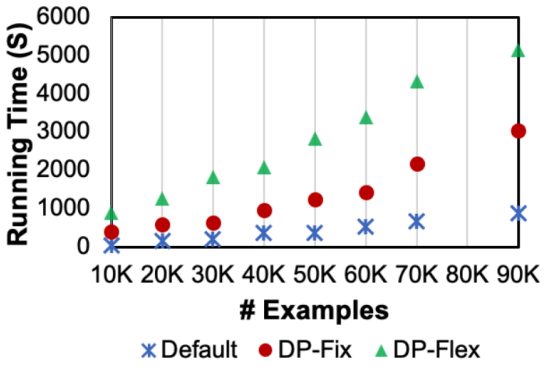

Fig. 11. 2 Layer NN [8]

6.4 Sensitivity Analysis

As the method targets to reduce the validation loss over-fitting could be the side effect of the approach on the validation set, Fig 12 shows the mean and standard error of some datasets, Use of either a very small training dataset or too small validation dataset can result in poor test accuracy due to overfitting.

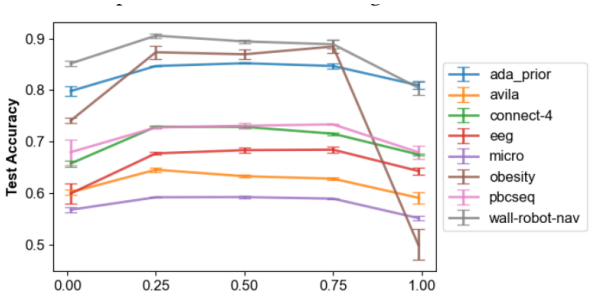

Fig. 12. Model test accuracy [8]

7 CONCLUSION

This paper proposes the automated tool DiffPrep in two variants DiffPrep-Fix (Fixed order) and DiffPre-Flex (order is decided by tool itself) as an alternative to the currently existing set AutoML solution, first of its kind to consider transformation operator, transformation type, transformation order and features in its search space. DiffPrep solves a basic search space problem by formulating the problem into a bi-level optimization problem and achieving the solution using gradient descent, and the results suggest that DiffPrep is better in terms of accuracy and run-time. as the proposed solution only works with a Differentiable end model. It is proposed for future works on how to find an optimal preprocessing pipeline for non-differentiable end models.

REFERENCES

[1] Microsoft Azure. Azure automl. Accessed: April 12, 2023.
[2] Laure Berti-Equille. Learn2clean: Optimizing the sequence of tasks for web data preparation. In *The World Wide Web Conference*, pages 2580–2586, 2019.
[3] Xu Chu, Ihab F Ilyas, Sanjay Krishnan, and Jiannan Wang. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 International Conference on Management of Data*, pages 2201–2206. ACM, 2016.
[4] Yuanzheng Ci, Chen Lin, Ming Sun, Boyu Chen, Hongwen Zhang, and Wanli Ouyang. Evolving search space for neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6659–6669, 2021.
[5] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: The next generation, 07 2020.
[6] Sanjay Krishnan and Eugene Wu. Alphaclean: Automatic generation of data cleaning pipelines. *ArXiv*, abs/1904.11827, 2019.
[7] Erin LeDell. H2o automl: Scalable automatic machine learning. 2020.
[8] Peng Li, Zhiyi Chen, Xu Chu, and Kexin Rong. Diffprep: Differentiable data preprocessing pipeline search for learning over tabular data. *Proc. ACM Manag. Data*, 1(2), jun 2023.