Journal of Universal Computer Science, vol. 31, no. 7 (2025), 668-682
submitted: 27/3/2024, accepted: 5/2/2025, appeared: 28/6/2025 CC BY 4.0

Meta-learning approach for variational autoencoder hyperparameter tuning

Michele Berti
(University of Trieste, Trieste, Italy
https://orcid.org/0009-0000-4609-1575, michele.berti.g@gmail.com)

Matheus Camilo da Silva
(University of Trieste, Trieste, Italy
https://orcid.org/0000-0002-1256-823X, max.mustermann@example.at)

Sebastiano Saccani
(Aindo Srl, Trieste, Italy
sebastiano@aindo.com)

Sylvio Barbon Junior
(University of Trieste, Trieste, Italy
https://orcid.org/0000-0002-4988-0702, sylvio.barbonjunior@units.it)

Abstract: Synthetic data generation is a promising alternative to traditional data anonymization, with Variational Autoencoders (VAEs) excelling at generating high-quality synthetic tabular datasets. However, VAE hyperparameter selection is often computationally expensive or suboptimal. We propose a meta-learning (MtL) method for hyperparameter recommendation, which achieves competitive performance to state-of-the-art Bayesian Optimization (BO) with median AUC values of 0.660 ± 0.038 (MtL) and 0.650 ± 0.041 (BO), showing no statistically significant difference. Notably, our approach reduces configuration time to under three minutes, compared to BO's multi-hour requirement, while also enabling incremental improvements through new data integration. This combination of efficiency, adaptability, and performance establishes MtL as a practical solution for hyperparameter tuning in synthetic data generation.

Keywords: HPO, Meta-learning, Variational Autoencoders, VAE
Categories: E.3,E.4,I.2.2, I.2.3
DOI: 10.3897/jucs.124087

1 Introduction

Deep generative models such as VAEs [Kingma and Welling 2022] have shown great potential in various tasks, including unsupervised disentangled representation learning [Locatello et al. 2019] and generative modelling of tabular or image datasets [Mami et al. 2022, Lucas and Verbeek 2019]. However, like any deep learning model, they have hyperparameters that greatly affect their performance, and their optimal configurations are often complex and dependent on the dataset properties [Brazdil et al. 2022]. Therefore, Hyperparameter Optimisation (HPO) is a crucial factor in obtaining optimal performance from these models.

HPO is often treated as an expensive black-box problem where only a few function evaluations are feasible, and no gradient is available. As such, BO [Jones et al. 1998] is a

popular and effective approach to obtain well-performing hyperparameter configurations [Bergstra et al. 2013, Eriksson et al. 2019, Snoek et al. 2012]. Nevertheless, BO exhibits potential drawbacks, including computational cost when applied to high dimensional problems, as the search space expands exponentially with the number of hyperparameters. Furthermore, the assumption of a smooth and continuous objective function, which underpins the BO method, may not always hold in practice, leading to challenges in finding the true optimum.

On the other hand, one of the available methodologies for HPO is MtL [Brazdil et al. 2008], which allows us to learn from past experiments and leverage this prior experience to design algorithms and optimise hyperparameters more effectively. The primary advantages of using MtL in hyperparameter tuning include the ability to efficiently reduce the search space [He et al. 2021] of promising hyperparameters and improve generalisation, which results in significant computational savings. Notably, MtL's improved generalisation is not limited to the specific dataset or task used for training, but rather, it enables transfer learning across multiple datasets and tasks [Franceschi et al. 2018, Li et al. 2018, Attig and Perner 2009, Martins et al. 2023].

Inspired by the approach adopted by Eggensperger et al. in [Eggensperger et al. 2015] and [Eggensperger et al. 2018] our proposed solution aims to tackle the problem of obtaining optimal hyperparameters for a given machine learning (ML) problem without the need for any expensive function evaluation. To achieve this, we leverage the knowledge of a large collection of previously solved problem instances with the same model.

The key idea is to treat each previously solved problem instance as an instance of a dataset and learn a meta-model that approximates the performance evaluations in different hyperparameter configurations for any new dataset. The meta-model can be optimised inexpensively to determine the optimal hyperparameters recommendation, thus reducing the overall computational requirement and avoiding any configuration evaluation.

Our contribution is a novel and task-specific hyperparameter recommender system designed for VAEs used in generating synthetic tabular datasets. VAEs have proven effective in synthesizing tabular data, as they can capture complex data distributions while maintaining privacy and utility, as highlighted in recent studies [Wan and Zhang and He 2017, Greco et al. 2020]. Specifically, we focused on tuning a small set of crucial hyperparameters to optimise performance: the coefficient β of the KL divergence term in the loss function, the latent dimension, and the encoder-decoder architectures. By carefully selecting and optimizing these parameters, our approach streamlines the tuning process, reducing computational overhead.

We compared our method's performance against BO, observing that while the median AUC achieved by our approach (0.660 ± 0.038) is slightly higher than that of BO (0.650 ± 0.041), the Wilcoxon signed-rank test indicates no statistically significant difference between the two methods. However, our method demonstrates a substantial advantage in computational efficiency. Unlike BO, which requires 30 VAE trainings—taking anywhere from 1 to 6 hours depending on the dataset—our approach completes the configuration suggestion process in under three minutes, with times ranging from 74 seconds to 2 minutes and 22 seconds. Additionally, our approach offers several advantages, including a reduced search space, effective exploration-exploitation trade-offs, improved generalization, and enhanced transfer learning capabilities. These strengths enable incremental enhancement of meta-databases (i.e., historical data) while minimizing computational overhead. Collectively, these benefits establish our method as a practical and efficient alternative for tuning VAEs, particularly in applications focused on generating synthetic tabular data.

2 Theoretical Background

The present study is grounded in three principal domains, namely VAEs, HPO, and MtL. The primary concepts and relevant literature related to this investigation are delineated in this section.

2.1 Variational Autoencoders

VAE's [Kingma and Welling 2022] are deep probabilistic generative models that made use of variational inferernce and an encoder-decoder structure to generate data following the same distribution of the original dataset. The encoder network maps the original data to a probability distribution in the latent space, while the decoder network maps the latent variables back to the original data space [Rezende et al. 2014].

The objective of training a VAE is to maximize the likelihood of the latent samples being drawn from the same distribution as the original data. This is achieved by minimising the reconstruction loss, which measures the difference between the original and reconstructed data. Moreover, VAEs impose a prior distribution on the latent variable using the Kullback-Leibler (KL) divergence. Specifically, the estimated distribution is compared to a predefined prior P, which is typically Gaussian. This regularisation technique enables the VAEs to learn a smooth latent space, which improves the interpretability and generative capability of the model.

The VAE loss function comprises two terms: the reconstruction loss and KL divergence. Mathematically, the VAE loss function can be expressed as:

$$L = -\sum_{x \in T} E_{z \sim q_\phi(z|x)}[\log p_\psi(x|z)] + \beta \cdot D_{KL}(q_\phi||P), \tag{1}$$

where x represents the original data, z the latent variable, $q_\phi(z|x)$ the encoder network, $p_\psi(x|z)$ the decoder network, T the training set, D_{KL} the KL divergence, and β is a hyperparameter introduced in [Higgins et al. 2017] that controls the strength of the KL divergence term. This parameter is extremely relevant. The two terms in the loss are in fact adversarial: minimising the reconstruction loss increases the KL term and vice versa. During training they converge to an equilibrium, and β controls the balance of this equilibrium.

2.2 Hyperparameter Optimisation

The performance of ML algorithms is significantly influenced by their hyperparameters [Lavesson and Davidsson 2005]. HPO or algorithm configuration (AC) is the task of finding the optimal hyperparameter settings for a particular task. Algorithm selection is a special form of HPO where the choice of algorithm is encoded as an additional hyperparameter [Thornton et al. 2013]. Combined algorithm selection and HPO (CASH) involve optimising the choice of algorithms and their hyperparameters simultaneously. The effect of hyperparameter configurations on performance is complex and dependent on the dataset properties.

Formally HPO is an optimisation problem [Hutter et al. 2019]: Given a problem instance or dataset d defined in the set of all possible problems D, and a chosen model m whose hyperparameters can be expressed as a parameter vector $\boldsymbol{\theta}$ defined in a parameter

space Θ, and a performance metric $\mathcal{L}_m : D \times \Theta \rightarrow \mathcal{R}$ that assesses the goodness of the model, finding the best hyperparameters $\boldsymbol{\theta}^*$, corresponds to solving the HPO problem:

$$\boldsymbol{\theta}^* = argmin_\theta \, \mathcal{L}_m(d, \boldsymbol{\theta}) \tag{2}$$

This problem is often efficiently tackled with the usage of BO [Jones et al. 1998] which reduces the number of function evaluations to obtain the optimal solution [Eriksson et al. 2019]. Our proposal instead uses MtL to suggest $\hat{\boldsymbol{\theta}}^*$, a surrogate for $\boldsymbol{\theta}^*$ obtained without the need for massive experimental evaluation.

2.3 Meta-Learning

Different alternatives have been proposed to address the HPO problem, one of which is MtL [Brazdil et al. 2008]. MtL has been successfully applied across various academic domains, as evidenced by several recent studies [Oyamada et al. 2023, Martins et al. 2023, Aguiar et al. 2022, Hospedales et al. 2022, Huisman et al. 2021]. The technique leverages ML methods to recommend optimal algorithms or configurations for new datasets. This is accomplished by incorporating knowledge gained from applying multiple algorithms or configurations to diverse datasets.

MtL involves learning at two levels: base-level and meta-level. To the base-level correspond traditional ML modelling: conventional algorithms are used to perform a typical ML task, (e.g. classification or regression) to solve the problem at hand. At the meta-level, the algorithms learn how to learn, inducing meta-models. The meta-models map the characteristics of each dataset to describe its performance when a set of algorithms or configurations are applied to it. Further details on MtL can be found in [Brazdil et al. 2008].

To conduct MtL experiments, datasets are first selected. Meta-features of each dataset are then extracted to create a meta-instance within a ”meta-dataset”. The algorithm or configuration space is then sampled for each dataset, and a performance metric (e.g., Accuracy, F-Score, Root Mean Squared Error, etc.) is used to determine which algorithm or configuration delivers the best performance. The performance evaluation is used to identify the target of the MtL procedure, also referenced as the meta-target.

Our approach considers all available data during meta-inference, not just measures of optimal configurations. For each performance measurement of a configuration, a corresponding meta-instance is created within the meta-dataset. These meta-instances are then used to train the meta-model covering different performances (meta-targets) identified.

3 Related Work

Different approaches have been studied to find the optimal dimension for the latent variable for a VAE, which is one of the most important hyperparameters of the model. Mai Ngoc and Hwang [Ngoc and Hwang 2020] suggested an empirical search by increasing the dimension until the reconstruction loss doesn’t improve anymore, while Bonheme and Grzes [Bonheme and Grzes 2022] presented a different algorithm that gain efficiency by stopping the trainings after few epochs. However, these methods only address one of the many hyperparameters that affect the performance of VAEs and not the others.

To tackle the general HPO problem, Bergstra et al. [Bergstra and Bengio 2012] suggested random search as a better alternative to grid search, while Snoek et al. [Snoek et al.

2012] showed that BO is able to outperform human experts in selecting hyperparameters. To improve the efficiency of BO, Deng et al. [Deng and Lindauer 2022] proposed some advancements that reduce the number of function evaluations needed to obtain a good hyperparameter configuration, which is the most relevant component of these procedures.

In the field of MtL, surrogate regression models have been extensively explored for predicting algorithm performance across various datasets based on dataset features [Brazdil et al. 2008, Aguiar et al. 2022]. However, these models require information on optimal parameters for every dataset, which may not always be readily available.

Inspired by the methodology proposed by Eggensperger et al. in [Eggensperger et al. 2015] and [Eggensperger et al. 2018], we present a novel methodology that constructs surrogate models to approximate the performance landscape. Subsequently, a cost-effective optimisation technique is employed to minimise the performance landscape. Our proposed approach does not necessitate configuration evaluation, resulting in swift results.

4 Meta-Learning for Variational Autoencoders

In recent years, synthetic data generation has gained significant traction across various domains, leveraging diverse methodologies such as generative adversarial networks (GANs), VAEs, and other deep learning approaches. Each of these methods offers unique advantages, making the choice of an appropriate technique crucial for the success of the application. This paper focuses on the use of VAEs for synthetic data generation, motivated by their ability to effectively capture the underlying distribution of the data while providing a straightforward framework for generating new data points. Unlike GANs, which often suffer from training instability and mode collapse, VAEs offer a more stable training process and a principled approach to learning latent representations. Additionally, VAEs facilitate better interpretability of the latent space, making them particularly suitable for applications where understanding the data generation process is as important as the data itself. Thus, while acknowledging the merits of alternative methods, our decision to utilize VAEs is driven by these compelling factors.

Our proposed approach aims to obtain optimal hyperparameters for a given ML problem without function evaluation. Our solution relies on leveraging the knowledge of a large collection of previously solved problem instances.

For the particular MtL problem at hand, it is essential to determine the meta-instances and meta-targets to construct the meta-model. In our proposal, a meta-instance refers to a VAE that has been trained with a specific dataset and a precise hyperparameter configuration. The corresponding meta-target is the quality of the dataset generated by the model. Subsequently, the meta-model is trained to anticipate the meta-target when provided with a new dataset and a set of hyperparameters.

More formally we consider a previously solved meta-instance (problem instance) as a dataset d_i, for which we have one or more performance evaluations $\mathcal{L}(d_i, \boldsymbol{\theta}_j)$ in some hyperparameter configuration $\boldsymbol{\theta}_j$, which may not necessarily correspond to the optimal $\boldsymbol{\theta}^*$. Each couple $(d_i, \boldsymbol{\theta}_j)$ is treated as a meta-instance, while the performance measurement is considered our meta-target for the MtL problem.

The objective is to learn a meta-model that approximates $\mathcal{L}(d, \boldsymbol{\theta})$. For this purpose, we first need to map each dataset d into its meta-feature representation $\hat{d}$. Using this representation, we can then induce a regression meta-model $\Psi(\hat{d}, \boldsymbol{\theta})$ that approximates $\mathcal{L}(d, \boldsymbol{\theta})$.

Evaluating this regression model $\Psi(\hat{d}, \boldsymbol{\theta})$ in a given configuration should be way cheaper than evaluating the original $\mathcal{L}(d, \boldsymbol{\theta})$ and can be therefore inexpensively optimized. By cost-effective optimisation of $\Psi(\hat{d}, \boldsymbol{\theta})$, we can obtain $\hat{\boldsymbol{\theta}}^*$, a surrogate for $\boldsymbol{\theta}^*$.

To achieve a good surrogate model Ψ, it is essential to have a large meta-dataset. This is due to the fact that a good sampling of $D \times \Theta$ is required to model the performance dependence on both meta-features and configurations, where D is the space of all possible meta-datasets (or problem) space and Θ is the configuration space.

An overview of the proposed methodology is shown in Figure 1. And we discuss each step in depth at the following subsections.

Figure 1: Overview of the proposed method

4.0.1 Generate Problem Instances

The specific instance of the VAE model that was used in the experiments was provided by the partner company Aindo. It is designed for generating synthetic tabular data including two additional layers, the embedding and inverse embedding layers, which map the input columns into an embedded space and map the synthesised data back into the original space. This instance of VAE is capable of handling columns of different data types, including text, categorical, numerical (float and integer), datetime, and UnixTime data.

For this work, only datasets made up of numerical and categorical columns have been considered due to the absence of literature on meta-features for general data types. Each column has a different pre-processing and post-processing pipeline according to the data type.

For numerical columns, a quantile transformer was used to transform the distribution of each column into a standard normal distribution. If the numeric column was an integer one, the generated samples were rounded to the closest integer value to maintain consistency with the original data type.

For categorical columns, a one-hot encoding procedure was used to transform the data into a numerical form that can be embedded. A threshold was applied to the maximum number of categories to reduce memory demand. Low-frequency categories are grouped into a single 'other' category, which was re-mapped to the original categories in the generated samples based on the initial marginal probabilities.

4.0.2 Extract Meta-features

We conducted experiments using the PyMFE(Python Meta-Feature Extractor) package [Alcobaça et al. 2020]. This package provides a comprehensive set of meta-features extraction routines. Through these routines we accomplished the mapping from the

dataset to a fixed size feature space, that was used to define the dataset part of the meta-instances. To enhance computational efficiency, we constrained the hyperparameter search space to focus on tuning only the network architecture, the weight of the KL term in the loss (β), and the latent dimension. The network architecture was encoded using three integer parameters, representing the width of the first and last layers as a base 2 logarithm of the number of neurons and the network depth. The number of units in the intermediate layers was determined through exponential interpolation, and we enforced reflection symmetry between the encoder and decoder architectures. By employing these constraints, we reduced the search space to five dimensions, improving the efficiency of our optimisation process. A summary of the 5 hyperparameters is presented in Table 1.

Hyperparameter	Description
β	Weight of the KL regularisation term in the loss function
latent dimension	Dimension of the latent variable
depth	Layers of the MLP encoder and decoder
minimum width	Width of the last (or first) layer of the encoder (or decoder)
maximum width	Width of the first (or last) layer of the encoder (or decoder)

Table 1: Hyperparameter summary.

To evaluate the performance of a configuration, thus obtaining the meta-target for the corresponding meta-instance, the trained VAE model was used to generate a synthetic dataset that aims to mimic the distribution of the one seen during training. We then randomly split the original and synthesised datasets into halves that were used for training and evaluation of a discriminator. The discriminator is a binary classifier trained to correctly classify original instances of the dataset and the ones that have been synthesised by the VAE. To fulfill this task we relied on a XGBoost [Chen and Guestrin 2016] model, a widely-used algorithm known for its accurate modelling ability. The employed instance was characterised by a maximum tree depth of 6 and 10 estimators. The performance of this discriminator on the evaluation set represents the desired meta-target. In fact, the worse a model is able to distinguish between original and synthesised data, the better the VAE performed. Between the possible evaluation metrics for binary classification, the ROC-AUC was used.

4.0.3 Optimal Configuration Search

The data used to train the meta-model was provided by Aindo. The meta-dataset is composed of 23009 configuration performance measurements regarding 423 artificially crafted tabular datasets. The datasets were generated by sampling from Gaussian mixture models and introducing non-linearities. The resulting data were then binned to obtain categorical values. The evaluation set is composed of 45 datasets generated with the same procedure. Table 2 presents the details regarding the proposed meta-dataset structure.

The training and all the other computations were performed on a computing node provided by Aindo, mounting an AMD Ryzen Threadripper 2950X 16-Core Processor and a NVIDIA RTX A5000 GPU. The training exploited the computing capabilities of the available GPU accelerator.

	Value sampling method	Values
Row number	$\lfloor \exp(u * (\ln(M) - \ln(m)) + \ln(m)) \rfloor$	$[m, M] = [10^3, 10^6]$
Column number	$\lfloor \exp(u * (\ln(M) - \ln(m)) + \ln(m)) \rfloor$	$[m, M] = [4, 39]$
Number Gaussian Components	c	$C = \{1, 2, 3, 4, 5\}$
Gaussian mean	$(t_4^1, _, t_4^n) \times 30$	$t_4^i \in \mathbb{R}$
Gaussian Covariance Matrix	$\Sigma = \tilde{\Sigma}^t \tilde{\Sigma}$	$\tilde{\Sigma} = ((u^{11}, _, u^{1n}), _, (u^{n1}, _, u^{nn}))$
Non linear transformation	c^* if $u^* < p_{nl} \sim U(0,1)$	$C = \{(x \to x^2), (x \to x^3), (x \to \tanh(x)), (x \to ReLU(x))\}$
Categorical Binning Number	$\lfloor \exp(u * (\ln(M) - \ln(m)) + \ln(m)) \rfloor$ if $u^* < p_c \sim U(0,1)$	$[m, M] = [2, 10^3]$

*The value is extracted for every column in the dataset

Table 2: Meta-dataset details: the variable u stands for a random variable sampled uniformly from $U(0,1)$, the variable c stands for a random extraction form the set C, t_i for a sampled value from a student t distribution with i d.o.f.

4.0.4 Surrogate Model

The Surrogate Model or meta-model was built using the XGBoost [Chen and Guestrin 2016] algorithm. This model is trained for the purpose of approximating the performance of the model based on the meta-instance data (hyperparameter configuration and meta-features of the dataset). The hyperparameters of the surrogate model are presented in Table 3.

Parameter	Value	Description
n_estimators	1500	Number of trees in the ensemble.
max_depth	7	Maximum depth of a tree.
eta	0.015	Learning rate.
subsample	0.65	Subsampling ratio of the training instances when constructing each tree.
colsample_bytree	0.9	Subsampling ratio of columns when constructing each tree.

Table 3: Meta model parameters.

The choice of XGBoost as a surrogate model for the meta-dataset is grounded in its proven efficacy in handling structured data and its robust performance across various

predictive modeling tasks. These attributes are particularly beneficial for our meta-dataset, which requires a model that can efficiently manage high-dimensional features and deliver reliable performance. Therefore, the selection of XGBoost aligns with our objective to achieve high predictive accuracy while maintaining model interpretability and computational efficiency.

To assess the significance of the meta-features, we conducted a SHAP (SHapley Additive exPlanations) analysis [Mitchell et al. 2020], which is a game-theoretic approach that explains the output of any ML model. These values represent the influence of the absolute value on the model's prediction. We recognised the need to extract only a limited number of meta-features, which are presented in Table 4.

Meta-feature name	Description
nr_attr	number of attributes
cat_ratio	fraction of categorical columns
mad	median absolute deviation [Ali and Smith 2006]
sparsity	discreteness of the input data set [Salama et al. 2013]
attr_conc	concentration coefficient [Kalousis and Hilario 2001]
eigenvalues	eigenvalues of covariance matrix [Ali and Smith 2006]
attr_entr	entropy of attributes [Michie et al. 1994]
two_itemset	correlation information of the attribute pairs [Song et al. 2012]
t_mean	truncated mean

Table 4: Meta-features: for each of the meta-features presented in the table and the following aggregated statistics have been considered: mean, standard, kurtosis, quantiles {0, 0.25, 0.5, 0.75, 1}, range and skewness.

4.0.5 Optimisation Baseline

For each instance in the test set, we employed Gaussian process-based BO to obtain an optimal value for the model hyperparameters. The minimisation technique relied on the scikit-optimize package [Head et al. 2020]. The search was initialised with two fixed configurations (as presented in Table 5) and eight randomly selected ones, followed by a 20 evaluation budget search. We considered these results as a baseline to compare against the performance of the recommended configuration obtained by minimising our surrogate model.

5 Results

5.1 Recommendation Performance

As stated in the previous section our performance metric is the ROC-AUC of a discriminator. The objective of the proposed model is to suggest the hyperparameter for a realistic synthetic data generator, therefore the best solutions are the one with lower values for the discriminator ROC AUC. More specifically the desired values ROC-ACU

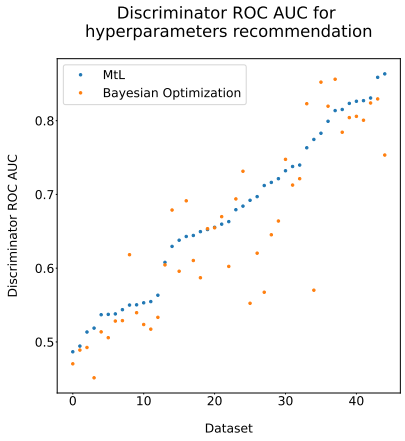
	Configuration 1 (cfg_1):	**Configuration 2** (cfg_2):
latent dimension	30	10
beta	1	1
depth	2	2
minimum width	7	5
maximum width	7	5

Table 5: Hyperparameter configurations for the first two fixed configurations.

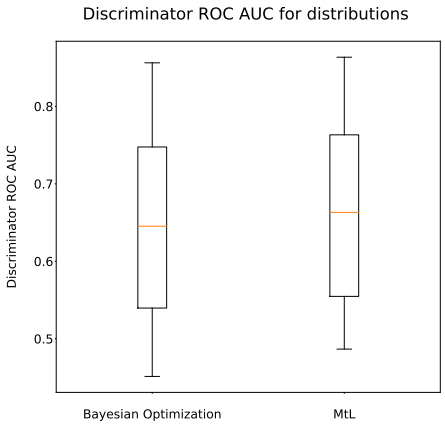
is 0.5, which correspond to the performance of a random classifier that is not able to distinguish between the two datasets. For this reason, we can consider that an optimal result is found when the ROC-AUC value is below 0.6. When comparing the results of different methodologies for HPO, in our case, we have to look for evidence of one method performing better than the other.

We observed that our hyperparameter recommender system provided statistically comparable model recommendations to those obtained with BO using 30 evaluations.

The Wilcoxon signed rank test failed to reject the null hypothesis (p=0.003) that the baseline population (median (MD) = 0.650 $\pm$ 0.041, median absolute deviation (MAD) = 0.100) is not greater than the population obtained with our system (MD = 0.660 $\pm$ 0.038, MAD = 0.110). A paired comparison of the results obtained by the two methodologies are presented in Fig. 2a while the box and whiskers plot in Fig. 2b presents a comparison between the two populations.



(a) Comparison of the best ROC-AUC values obtained using the MtL recommender pipeline and the BO baseline. Both methods achieve suboptimal results (ROC-AUC $<$ 0.6) for only a small number of datasets.



(b) Box-and-whisker plot comparing ROC-AUC distributions from MtL and BO. Despite MtL showing a slightly higher median, the Wilcoxon test (p = 0.003) indicates no significant difference between the two methods.

Figure 2: Performance comparison of the MtL recommender pipeline and Bayesian Optimization (BO): (a) Best ROC-AUC values across datasets; (b) Distribution of ROC-AUC results.

5.2 Time cost

Our MtL method has demonstrated comparable performance to BO. However, these two approaches differ significantly in terms of execution time, in which our recommender system required only a few minutes from meta-feature extraction to configuration suggestion, whereas BO required 30 VAE trainings that took several hours to complete.

The time required for BO with a budget of 30 evaluations corresponds roughly to the processing time demanded by 30 VAE training, required from a minimum of 1h (1:00:43) to a maximum of 6h (6:00:12) depending on the dataset. The time required to generate the MtL recommendation varied between a maximum of less than 3 minutes (0:02:22) and a minimum of 74 seconds (0:01:14). These values are illustrated in Fig. 3.

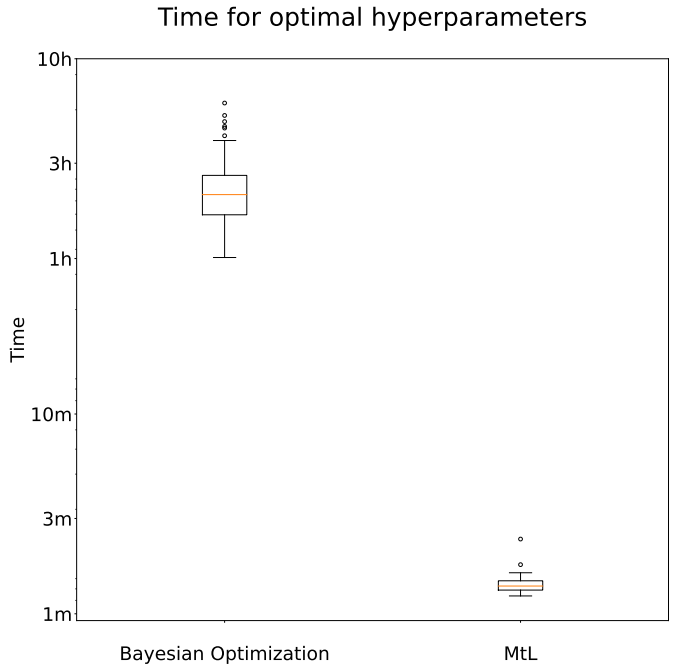


Figure 3: The box and whiskers shown in the present Figure compare the distributions of computation time required from the BO baseline and the MtL suggestion. It can be seen that there's a huge gap between the quantities. In fact the time axis is presented in logarithmic scale due to the high differences in the time scales for the different quantities.

5.3 Proposal Limitations

By modelling the knowledge gained from previously solved problems, our system can guide the search for optimal configurations. However, while the meta-model provides

reasonable insights, it cannot fully explain the dependence of the meta-target in terms of configuration and meta-features.

To improve the accuracy of the meta-model, more data is needed and more informative meta-features are required. Specifically, real-world datasets and optimal configurations on available datasets are needed to enhance the precision of the recommendations and the generalisation capability of the model. Additionally, further research on meta-features is necessary to increase the informative power of dataset representation that the meta-model can exploit.

Despite these limitations, our system has the potential to significantly reduce human and computational effort and can be further improved for enhanced performance and it can be updated with new data for increased accuracy over time.

An extension of this tool could also be adapted for use with multi-table relational databases, leading to the development of an AutoML tool for relational databases' generative models. However, such an extension would require developing new techniques for meta-feature extraction that take into account the relationships between tables in a relational database.

6 Conclusions and Future Work

This paper proposes a MtL procedure for recommending hyperparameters for VAEs. Our experimental results show that our approach achieves performance statistically similar to that of the BO technique used to train it while offering the added advantage of efficiency. Specifically, our approach is able to provide recommendations in a matter of minutes rather than hours.

The system developed can serve as a foundation for an AutoML tool for synthetic tabular data generation. Future research could be conducted to identify additional informative meta-features that could enhance the performance of the regression model. Additionally, the tool could be extended to handle other data types such as datetime and text columns, which would involve identifying a set of meta-features to describe the properties of these data types.

Acknowledgements

We would like to acknowledge the support of Aindo for providing the dataset and computational resources used in this study. Additionally, we extend our gratitude to the Aindo research team for developing the generative model used in our experiments. Their expertise and experience with VAEs and synthetic data were invaluable, and we greatly appreciate the helpful insights they provided throughout the research process.

References

[Aguiar et al. 2022] Aguiar, G.J., Santana, E.J., de Carvalho, A.C.P.L., Barbon Junior, S.: "Using meta-learning for multi-target regression"; Inf. Sci. 584 (2022), 665-684. https://doi.org/10.1016/j.ins.2021.11.003.

[Alcobaça et al. 2020] Alcobaça, A., Siqueira, F., Rivolli, A., Garcia, L.P.F., Oliva, J.T., de Carvalho, A.C.P.L.F.: "MFE: Towards reproducible meta-feature extraction"; J. Mach. Learn. Res. 21, 111 (2020), 1–5.

[Ali and Smith 2006] Ali, S., Smith, K.A.: "On learning algorithm selection for classification"; Appl. Soft Comput. 6, 2 (2006).

[Attig and Perner 2009] Attig, A., Perner, P.: "A study on the case image description for learning the model of the watershed segmentation"; Trans. Case-Based Reasoning 2, 1 (2009).

[Bergstra and Bengio 2012] Bergstra, J., Bengio, Y.: "Random search for hyper-parameter optimization"; J. Mach. Learn. Res. 13, 2 (2012). http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf.

[Bergstra et al. 2013] Bergstra, J., Yamins, D., Cox, D.: "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures"; Proc. 30^{th} Int. Conf. Mach. Learn., PMLR 28, 1 (2013), 115–123. http://proceedings.mlr.press/v28/bergstra13.pdf.

[Bonheme and Grzes 2022] Bonheme, L., Grzes, M.: "FONDUE: an algorithm to find the optimal dimensionality of the latent representations of variational autoencoders"; arXiv preprint (2022). https://arxiv.org/abs/2209.12806.

[Brazdil et al. 2008] Brazdil, P., Giraud Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining. Springer (2008).

[Brazdil et al. 2022] Brazdil, P., van Rijn, J.N., Soares, C., Vanschoren, J.: Metalearning: applications to automated machine learning and data mining. Springer Nature (2022).

[Chen and Guestrin 2016] Chen, T., Guestrin, C.: "XGBoost: A Scalable Tree Boosting System"; Proc. 22^{nd} ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (2016), 785–794. https://doi.org/10.1145/2939672.2939785.

[Deng and Lindauer 2022] Deng, D., Lindauer, M.: "Searching in the Forest for Local Bayesian Optimization"; Proc. ECMLPKDD Workshop Meta-Knowledge Transfer, PMLR 191 (2022), 38–50. https://proceedings.mlr.press/v191/deng22a/deng22a.pdf.

[Eggensperger et al. 2015] Eggensperger, K., Hutter, F., Hoos, H., Leyton-Brown, K.: "Efficient Benchmarking of Hyperparameter Optimizers via Surrogates"; Proc. AAAI Conf. Artif. Intell. 29, 1 (2015). https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9375.

[Eggensperger et al. 2018] Eggensperger, K., Lindauer, M., Hoos, H.H., Hutter, F., Leyton-Brown, K.: "Efficient benchmarking of algorithm configurators via model-based surrogates"; Mach. Learn. 107 (2018), 15–41. https://link.springer.com/article/10.1007/s10994-017-5657-4.

[Eriksson et al. 2019] Eriksson, D., Pearce, M., Gardner, J., Turner, R.D., Poloczek, M.: "Scalable Global Optimization via Local Bayesian Optimization"; Adv. Neural Inf. Process. Syst. 32 (2019).

[Franceschi et al. 2018] Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., Pontil, M.: "Bilevel Programming for Hyperparameter Optimization and Meta-Learning"; Proc. 35^{th} Int. Conf. Mach. Learn., PMLR 80 (2018), 1568–1577. http://proceedings.mlr.press/v80/franceschi18a/franceschi18a.pdf.

[Greco et al. 2020] Greco, G., Guzzo, A., Nardiello, G.: "FD-VAE: A feature driven VAE architecture for flexible synthetic data generation"; Int. Conf. Database Expert Syst. Appl. (2020).

[Head et al. 2020] Head, T., Kumar, M., Nahrstaedt, H., Louppe, G., Shcherbatyi, I.: "scikit-optimize/scikit-optimize"; Zenodo, version v0.8.1 (Sep 2020). https://doi.org/10.5281/zenodo.4014775.

[He et al. 2021] He, X., Zhao, K., Chu, X.: "AutoML: A survey of the state-of-the-art"; Knowl.-Based Syst. 212 (2021). https://doi.org/10.1016/j.knosys.2020.106622.

[Higgins et al. 2017] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework"; Int. Conf. Learn. Represent. (2017).

[Hospedales et al. 2022] Hospedales, T.M., Antoniou, A., Micaelli, P., Storkey, A.J.: "Meta-Learning in Neural Networks: A Survey"; IEEE Trans. Pattern Anal. Mach. Intell. 44, 9 (2022), 5149–5169. https://doi.org/10.1109/TPAMI.2021.3079209.

[Huisman et al. 2021] Huisman, M., van Rijn, J.N., Plaat, A.: "A survey of deep meta-learning"; Artif. Intell. Rev. 54, 6 (2021), 4483–4541. https://doi.org/10.1007/s10462-021-10004-4.

[Hutter et al. 2019] Hutter, F., Kotthoff, L., Vanschoren, J.: Automated Machine Learning: Methods, Systems, Challenges. Springer Nature (2019).

[Jones et al. 1998] Jones, D.R., Schonlau, M., Welch, W.J.: "Efficient global optimization of expensive black-box functions"; J. Glob. Optim. 13, 4 (1998), 455.

[Kingma and Welling 2022] Kingma, D.P., Welling, M.: "Auto-Encoding Variational Bayes"; arXiv preprint (2022). arXiv:1312.6114.

[Kalousis and Hilario 2001] Kalousis, A., Hilario, M.: "Model selection via meta-learning: a comparative study"; Int. J. Artif. Intell. Tools 10, 4 (2001).

[Lavesson and Davidsson 2005] Lavesson, N., Davidsson, P.: "Quantifying the Impact of Learning Algorithm Parameter Tuning (short version)"; Mälardalen University (2005).

[Li et al. 2018] Li, D., Yang, Y., Song, Y.-Z., Hospedales, T.: "Learning to Generalize: Meta-Learning for Domain Generalization"; Proc. AAAI Conf. Artif. Intell. 32 (2018). https://ojs.aaai.org/index.php/AAAI/article/view/11596.

[Locatello et al. 2019] Locatello, F., Abbati, G., Rainforth, T., Bauer, S., Schölkopf, B., Bachem, O.: "On the Fairness of Disentangled Representations"; Adv. Neural Inf. Process. Syst. 32 (2019). https://papers.nips.cc/paper/2019/file/27ed56c20c07c55e7c409560ecf3eb11-Paper.pdf.

[Lucas and Verbeek 2019] Lucas, T., Verbeek, J.: "Auxiliary Guided Autoregressive Variational Autoencoders"; Mach. Learn. Knowl. Discov. Databases (2019), 443–458. https://doi.org/10.1007/978-3-030-10925-7.

[Mami et al. 2022] Mami, C.A., Coser, A., Medvet, E., Boudewijn, A.T.P., Volpe, M., Whitworth, M., Svara, B., Sgroi, G., Panfilo, D., Saccani, S.: "Generating Realistic Synthetic Relational Data through Graph Variational Autoencoders"; arXiv preprint (2022). arXiv:2211.16889.

[Martins et al. 2023] Martins, V.E., Cano, A., Barbon Jr, S.: "Meta-learning for dynamic tuning of active learning on stream classification"; Pattern Recognit. 138 (2023), 109359.

[Michie et al. 1994] Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine learning, neural and statistical classification. (1994).

[Mitchell et al. 2020] Mitchell, R., Frank, E., Holmes, G.: "Gputreeshap: Fast parallel tree interpretability"; arXiv preprint (2020). arXiv:2010.13972.

[Ngoc and Hwang 2020] Ngoc, K.M., Hwang, M.: "Finding the Best k for the Dimension of the Latent Space in Autoencoders"; Comput. Collective Intell. (2020), 453–464.

[Oyamada et al. 2023] Oyamada, R.S., Shimomura, L.C., Barbon Jr, S., Kaster, D.S.: "A meta-learning configuration framework for graph-based similarity search indexes"; Inf. Syst. 112 (2023), 102123.

[Rezende et al. 2014] Rezende, D.J., Mohamed, S., Wierstra, D.: "Stochastic Backpropagation and Approximate Inference in Deep Generative Models"; arXiv e-prints (2014). arXiv:1401.4082.

[Salama et al. 2013] Salama, M.A., Hassanien, A.E., Revett, K.: "Employment of neural network and rough set in meta-learning"; Memetic Comput. 5, 3 (2013).

[Song et al. 2012] Song, Q., Wang, G., Wang, C.: "Automatic recommendation of classification algorithms based on data set characteristics"; Pattern Recognit. 45, 7 (2012).

[Snoek et al. 2012] Snoek, J., Larochelle, H., Adams, R.P.: "Practical Bayesian Optimization of Machine Learning Algorithms"; Adv. Neural Inf. Process. Syst. 25 (2012).

[Thornton et al. 2013] Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms"; Proc. 19^{th} ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (2013), 847–855.

[Vanschoren 2018] Vanschoren, J.: "Meta-Learning: A Survey"; arXiv preprint (2018). arXiv:1810.03548. https://arxiv.org/abs/1810.03548.

[Wan and Zhang and He 2017] Wan, Z., Zhang, Y., He, H.: "Variational autoencoder based synthetic data generation for imbalanced learning"; Proc. IEEE Symp. Ser. Comput. Intell. (2017).