Received: 1 February 2023 | Revised: 22 April 2023 | Accepted: 15 May 2023 | Published online: 24 May 2023

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2025, Vol. 4(1) 56–63
DOI: 10.47852/bonviewJCCE3202720

BON VIEW PUBLISHING

A Survey on Recent Advancements in Auto-Machine Learning with a Focus on Feature Engineering

Ravishankar S[1] and Gopi Battineni[2,3,*]

[1]*CloudThat Technologies Pvt. Ltd., India*

[2]*The Clinical Research Center, University of Camerino, Italy*

[3]*The Research Centre of the ECE Department, V. R. Siddhartha Engineering, India*

Abstract: A study on the recent trends and progress in the area of automated machine learning (AutoML) is done in detail in this paper. AutoML deals with the end-to-end automation of various steps in a machine learning pipeline. Some of the steps include feature selection, feature engineering, neural architecture search, hyperparameter optimization, and model selection. The time and the specialized skill set required to perform these tasks may be reduced to some extent with the help of automating all or some of these steps. Thus, a lot of research is going on in the area of AutoML, and the recent research articles add justice to the same. A review of existing literature on AutoML with a focus on feature engineering is presented in this paper to assist scientists in building better machine learning models "off the shelf" without extensive data science experience. The use of AutoML in different sectors will also be discussed, as will existing applications of AutoML. A review of published papers accompanied by describing work in AutoML from a computer science perspective was conducted.

Keywords: machine learning, feature engineering, model selection, performance metrics

1. Introduction

Feature engineering deals with the construction of new features by applying some transformations to the existing feature. Feature selection plays a crucial role in determining the most appropriate feature subset of a problem and is considered to be an optimization problem where meta-heuristic algorithms may be used to solve it. As feature selection is a nondeterministic polynomial-time (NP-hard) problem, a lot of meta-heuristics play an important role here. Neural architecture search (NAS) is an area to discover an optimal neural network architecture, thereby eliminating the need for manual design. Hyperparameter optimization (HPO) deals with finding the optimal set of discrete and continuous hyperparameters. Model selection is the art of finding out a suitable model from a list for solving a particular problem. In this paper, we aim to review these ML concepts in detail.

Dhiman et al. [1] developed a binary variant of the Emperor Penguin Optimizer (EPO) meta-heuristic approach to solve the discrete search problem. In this approach, the transition of locations of the emperor penguin is modeled using S-shaped and V-shaped transfer functions, which are then mapped into binary search space using the binarization method. Collision avoidance as well as drift mechanisms are used in Binary Emperor Penguin Optimizer (BPO). Then this meta-heuristic method was applied to solve the problem of feature selection as well. In a meta-heuristics approach, simulated annealing (SA) is combined along with a Generalized Normal Distribution Optimizer (GNDO) to overcome the issue of arriving at local optimal solutions [2].

The approach named Binary Simulated Normal Distribution Optimizer (BSNDO) uses SA to perform a local search to avoid early convergence and achieve better results in terms of accuracy for classification problems. The transfer function is used in meta-heuristics to transform the continuous nature of optimization problems into discrete, thereby helping in the feature selection process. S- and V-shaped transfer functions are two commonly used functions for this purpose, and both will output a probability of selecting a particular feature [3]. An improved butterfly optimization algorithm (BOA) is proposed to deal with high-dimensional feature selection problems [4]. The previous versions of the BOA could solve only those optimization problems with a smaller or limited number of features, whereas this velocity-based BOA (VBOA) could overcome the same problem by incorporating the velocity item and memory items into consideration. The next candidate solution was made based on the position update considering the above-mentioned items.

Automatic feature generation using deep neural networks resulted in a lot of irrelevant features and identical features despite several advantages over search-based methods. This problem was solved by mapping the feature interactions to a graph network where features were mapped to nodes and interactions were mapped to edges [5]. Local interactions between the features were

***Corresponding author:** Gopi Battineni, The Clinical Research Center, University of Camerino, Italy and The Research Centre of the ECE Department, V. R. Siddhartha Engineering, India. Email: gopi.battineni@unicam.it

captured by constructing an adjacency matrix called structure parameter matrix, and the interactions between the feature parameters were considered as message passing between the nodes of this matrix. Global feature interactions were captured using the reinforcement learning technique called Q-learning to automatically create an interaction layer based on the local attention-based interaction layer.

A formal definition of automated machine learning (AutoML) and a review of its use are provided in this paper using a variety of feature engineering techniques [6]. We will discuss key technologies in AutoML during this article, as well as how they can be used specifically in different industries. In this review, we organized the papers based on what authors are attempting to automate: automated feature engineering, HPO, pipeline optimization, and neural architecture research. These four categories will be discussed individually and how they can be applied.

The most recent developments in the AutoML field are organized into the following four sections. Section 2 gives an overview of the various NAS strategies, and Sections 3 and 4 talk about the HPO and MS techniques, respectively. Finally, the article ends with a conclusion Section 5.

2. Neural Architecture Search (NAS)

NAS refers to the combination of architecture optimization (AO) along with the determination of NAS space and model evaluation methods. The architecture of NAS is presented in Figure 1. An automatic search space and search strategy selection for a dataset based on the previous results by assuming that search space and search strategy are often combined [7].

The NAS architecture can be either scaled up or scaled out. An architecture that provides the capability to scale the performance and capacity of a single NAS system according to the needs of the user is referred to as a scale-up NAS architecture [8]. NAS controllers and storage need to be upgraded or added when scaling up a NAS system. There are two components to any NAS system that are commonly used:

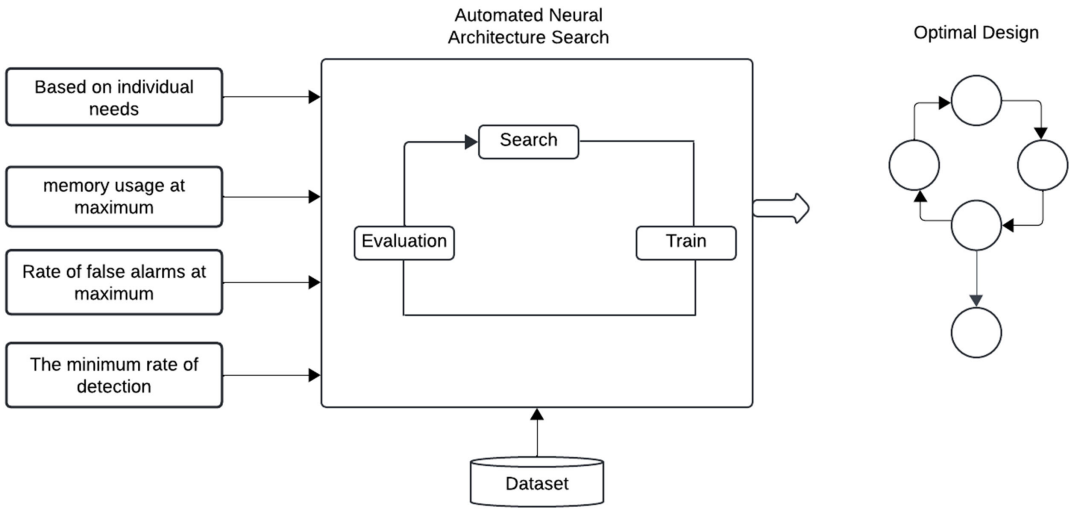
1) Controllers: These are computer systems that contain components such as networks, memory, and CPUs. File serving is managed by a specialized operating system installed on the controller. There is no restriction on how many storage devices can be connected to each controller. The controllers can perform all I/O processing, with some controllers performing all I/O processing, while others act as spares, or they can be active/active with all controllers accessing storage. Controllers configure RAID sets, create LUNs, install file systems, and export file shares [9].
2) Storage: Persistent storage of data is required. To support different requirements, NAS systems may have different types of storage devices. SATA, SAS, and SSD can all be supported by the same NAS system [10].

Meta-features based on a probing model are used for search space estimation, and a decision tree model is used for finding out the search strategy. The inference latency issue of the cell-based structure in CNN is also addressed here. The search space is suitable for multi-objective AutoML across various platforms. Layers Architecture Search Tree (LArST) is used to ensure search space learning, thereby transferring a basic cell structure from a benchmark dataset to any ImageNet dataset. Greedy and Progressive Architecture Search (GPAS) uses a greedy approach to find optimal architectures [11].

This phase is followed by an evaluation phase. CNNs and pretrained CNNs are not ideal for all image scene classification like remote sensing as they suffer from overfitting due to the limited availability of images. GPAS helps in finding the optimal architecture, but it takes a huge computational cost for searching the different architectures and arriving at an optimal one. Early stopping and partial training methods are deployed to alleviate the overfitting problem. Search happens in a progressive greedy manner and finds the optimal architectures using a gradient-based approach. Differential NAS (DNAS) finds layers and connections between the layers, which together form a block [12]. These blocks are then searched to find out the optimal one. Thus, the search space is already fixed. This causes less architectural flexibility and a poor ability to generalize. It also incurs huge computation costs and memory requirements.

Block Proposal Neural Architecture Search (BPNAS) introduced a new two-phase search mechanism namely Block Proposal Search (BPS) and Block Connection Search (BCS). An evolutionary algorithm named Latency Evolutionary NAS (LEvoNAS) is used to generate blocks with latency constraints. These blocks are then used by the DNAS to find out the optimal architecture with better search efficiency. Fast Network Adaptation via Parameter Remapping and Architecture Search (FNA++) is a method to modify the architecture and network

Figure 1
NAS architecture



parameters of a pretrained network to a new network [13]. This adaptation strategy was applied to some of the pretrained networks and NAS. It achieved faster search results for image segmentation and object detection tasks. The network adaptation strategy was implemented in manually designed mobile net v2 and Resnet networks. MIGO-NAS considers the search space as a multivariate geometric distribution [14]. It reduces the error estimation in natural gradients computed in multivariate geometric distribution by assigning a larger magnitude to the natural gradient with a higher architecture rank. The network generation is done using a dynamic programming approach, which is useful in devices with limitations in computing and memory. This approach helps to tackle the heavy computation cost and the poor generalization capability of NAS.

One category of multi-objective NAS is to find the optimal architecture search using some constraints [15]. This NAS method asks for user preferences in the architecture selection process. Users can give preference for accuracy and latency values. In the learning or training phase, it collects preferences from users and a coevolutionary algorithm is used to find optimal search space. In the prediction phase, a response is provided by the algorithm based on the user's adapting preferences. Authors developed a single-path NAS method where the search cost for CNN design is reduced significantly [16]. The one-shot supernet NAS method was improvised, hereby finding the optimal subset of Kernel weights to be used in each CNN layer. The accuracy–runtime trade-off was improved by performing a search on the Squeeze-and-Excitation path in the single-path encoding of one-shot supernet NAS. This method achieved an accuracy of 75.62% on the ImageNet dataset with a latency constraint of 80 ms. The NAS search cost was also minimized to eight iterations/epochs which approximately takes 24 TPU hours, which is faster compared to the already existing works.

One-shot NAS minimizes the amount of time that is required for training as weights are inherited from a supernet [17]. It uses a single-path training approach in the supernet training where only one path in the supernet is trained in each iteration. But at the same time, it suffers from catastrophic forgetting which leads to a decrease in accuracy due to the weight sharing. Catastrophic forgetting is the scenario where if the model is trained on a new task "X," then it forgets the learning that was done for task "Y." An approach in which this catastrophic forgetting is prevented by increasing the model's accuracy was presented by Kaushik et al. [18] and Kirkpatrick et al. [19]. A continual learning approach is used where catastrophic forgetting is prevented by adding a regularization term which will be the response from the previous task. Differentiable NAS evolved from the fact that it was able to transform the discrete architecture search space of NAS into a continuous space where the gradient descent method could be used to optimize the architecture concerning validation accuracy [17]. This discrete-to-continuous space mapping has not been proved theoretically.

Moreover, the architecture parameters were updated based on the performance reward, which leads to a rich-get-richer problem as architecture selection is biased as those with better performance in the earlier stages would be used for training quite often. This would also lead to a local optimum as those architectures at the earlier stages have a high chance of being sampled. An Exploration Enhancing Neural Architecture Search with Architecture Complementation (EENAS) method was proposed in which a variational autoencoder is used to map the architectures from discrete space to continuous. Further, a probabilistic exploration enhancement approach is used to address the rich-get-richer problem by intelligent search space exploration. The high computational cost of Neural AutoML is addressed by using the transfer Neural AutoML approach where knowledge transfer from the previous similar tasks is used to improve the computational speed [20]. RL-based architecture search methods are used to do parallel programming on different jobs. The search strategy is then transferred to do new jobs. One of the advantages of using this approach is the reduction of the time to converge for different datasets. Transfer learning approaches involve using the same hyperparameter combinations learned on one domain like NLP on other domains. Further enhancement to this approach involves addressing the meta-overfitting on small datasets and noise immunity.

3. Hyperparameter Optimization

Optimizing models is one of the most challenging aspects of implementing machine learning solutions. Machine learning and deep learning algorithms can improve models [21, 22]. Various branches of machine learning and deep learning have been dedicated to this. When optimizing hyperparameters for ML algorithms, the goal is to find those parameters that deliver the highest performance [23]. This will be done when compared to those that perform the least well on a validation set. An ML engineer sets hyperparameters before training, unlike model parameters. In a neural network, the weights are learned from training, whereas the number of trees is a hyperparameter. It is advisable to view hyperparameters as settings for an ML model that should be tuned so that it can solve a given problem optimally.

Optimizing hyperparameters (or tuning hyperparameters) is the process of identifying the correct combination of hyperparameters to optimize model performance [24]. A training process runs multiple trials at the same time. The training application is executed each time with the hyperparameters you selected, within the limits you specified. After this process has been completed, it will be possible to understand which hyperparameter values will give the most accurate results for given model [25–27]. The detailed explanation of hyperparameter tuning can be visualized in Figure 2.

Table 1 represents a comparison of some of the existing HPO methods against the computational cost, curse of dimensionality, adaptive nature, and configuration space. The HPO problem can be formulated as a constrained optimization problem if a limit on a second performance measure is known as maximal memory consumption [28]. It is required to reduce human effort, improve the performance of machine learning algorithms, and improve the reproducibility and fairness of scientific studies. The commonly used black-box function optimization methods such as Bayesian optimization suffer from high computational costs. The large models suffer from the curse of dimensionality, and function evaluation becomes expensive. The hyperparameter configuration space also consists of diverse types such as continuous and categorical. In the case of a neural network, the number of layers, the number of units per layer, the learning rate, and the momentum are some of the hyperparameters. The gradient-based loss functions cannot be applied in the case of hyperparameters.

Modern multi-fidelity methods are used, which deploy cheaper variants of the black-box function to approximate address the above-mentioned problems. The Grid Search method suffers from the curse of dimensionality, since the required number of function evaluations grows exponentially with the dimensionality of the configuration space [28]. The increase in the resolution of discretization results in an increase in the required number of function evaluations. It is nonadaptive as well. The Random Search (RS) method is also nonadaptive, since the search is not adapted by considering the available results. Bayesian optimization can be considered as the combination of a probabilistic surrogate model

Figure 2
Hyperparameter optimization

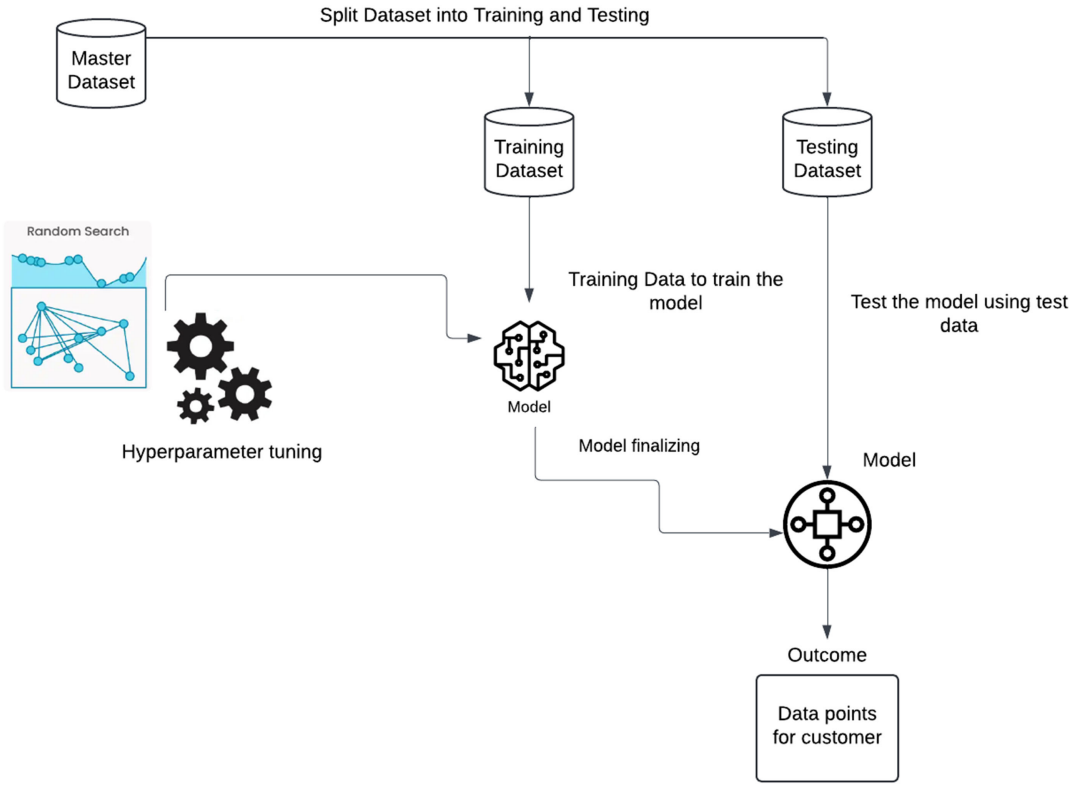


Table 1
Comparison of some of the existing HPO methods

Methods	Computational cost	Curse of dimensionality	Adaptive	Discrete configuration space
Grid Search [29]	High	Yes	No	No
Random Search [30]	Less	No	No	No
Bayesian [31]	Less	Yes	Yes	No
Gaussian Process [32]	High	No	Yes	No
Random Forest [30]	Less	Yes	Yes	Yes
Genetic Algorithm [33]	Less	No	Yes	No

and an acquisition function (which point to evaluate next) [34]. Some of the surrogate models are the Gaussian Process (GP) and Random Forest Sequential Model-Based Algorithm Configuration (RF-SMAC). GP is good for low-dimensional and numerical configuration spaces. The RF model is good for high-dimensional and discrete configuration spaces, and it takes less computational time compared to GP. Gradient Boosting Decision Tree can be an alternative to RF.

Some of the collaborative hyperparameter tuning methods are local search-based – ParamILS, estimation of distribution methods – REVAC, and surrogate-based [35]. The challenge involved is to determine optimal hyperparameters with high certainty. A model is built from experience using surrogate-based models. A new Bayesian optimization technique based on surrogate ranking and optimization techniques is developed, which can be applied to similar groups of objective functions. HyperNOMAD is a black-box optimization algorithm used to optimize the hyperparameters of a neural network. Hyperparameters can be floating point, integer, or categorical [36]. The different types of hyperparameters need to be optimized before the learning process. The objective functions can be expensive or noisy to evaluate in certain cases, where derivative-free optimization (DFO) can be deployed.

In cases where a derivative does not exist, then variants of DFO algorithms like model-based and direct search may be used. HyperNOMAD is a DFO algorithm that performs direct search and yields a better result by exploring large search spaces consisting of categorical variables. Yu et al. [37] proposed a model in which the efficiency of the HPO is improved by reducing the search space complexity. The approach also uses a genetic-based search utilizing the best hyperparameter configuration of the algorithms used for solving similar types of problems.

4. Model Selection

Table 2 shows a comparison of recent AutoML research works. The objective or method is mentioned in the first column, and the subsequent columns indicate the steps in an AutoML pipeline.

Table 2
Comparison of some of the existing AutoML research works

Objectives/methods	FE	MS	HPO	NAS	Datasets	Performance metrics	Ref
Recommendation/EnPSO	✖	✓	✖	✖	Movielens	RMSE, MSE, MAE	[38]
Classification/MLPlan	✓	✓	✖	✖	CIFAR 10, convex, MNIST	Accuracy	[39]
Classification/TPOT-NN	✓	✖	✖	✖	Spambase, ionosphere, breast-cancer-Wisconsin	Accuracy	[40]
Classification/ Deap-n-cheap/CNN	✖	✖	✓	✓	CIFAR-10,100, Fashion MNIST	Accuracy	[41]
Text Classification/autoBOT	✓	✖	✓	✖	kaggle/google	Accuracy, f1 score	[42]
Classification, Regression/GA/FC NN	✖	✓	✖	✖	MNIST, CMAPSS	Accuracy	[43]
Classification/PIL/ Laplacian Regularization	✖	✖	✓	✓	MNIST, CIFAR-10, openML	Accuracy, Training Time	[15]
Classification/ Genetic GNN	✖	✖	✓	✓	Cora, Citeseer, Pubmed	Accuracy, micro-f1 score	[44]
Classification/GNN/ skip connection	✖	✖	✓	✓	Cora, Citeseer, Pubmed, PPI	Accuracy, micro-f1 score	[45]
Classification/ SANE/DAS/GNN	✖	✖	✖	✓	Cora, Citeseer, Pubmed, PPI	Accuracy,	[46]
Classification/Search space optimization & Regularization/P-DART	✖	✖	✖	✓	CIFAR 10, Imagenet	Accuracy	[47]
Image Classification/ Text Modeling/ AutoDropout	✖	✖	✖	✓	CIFAR, Imagenet, Penn Treebank WikiText-2	Accuracy, Perplexity, BLEU score	[48]
Multi-objective/EA + Transfer Learning	✖	✖	✖	✓	ImageNet, C10, C100, CINIC-10, STL-10, Flowers102, Pets, DTD, Cars, Aircraft, Food-101	Model Accuracy, Size, CPU latency, GPU latency	[49]
CASH/TPOT-SH	✓	✓	✓	✖	Wine-quality-red, kdd cup, car-evaluation, cover type	Accuracy	[50]
RP-KNN	✖	✖	✓	✖	IMDB, POKER, Tweets	Accuracy, Runtime, Memory	[51]
PyGlove: Symbolic Programming	✖	✓	✓	✓	Symbolic Dataset	Accuracy	[52]
AutoML-Zero	✖	✓	✖	✖	CIFAR	Accuracy	[53]
Normalization/activation, image classification, instance segmentation, and GAN training	✖	✖	✓	✖	CIFAR, COCO	Accuracy	[54]

The benchmark datasets used along with the performance metrics are given in the last column.

MS deals with choosing the right algorithm for a particular task among the different existing ones. It is hard to distinguish or separate between the MS and HPO as both are combined in most of the works. A meta-learning-based approach model selection in case of classification problems was proposed by Yu et al. [37]. Meta-learning approaches suffer from human bias in MS, which is eliminated by incorporating reinforcement learning techniques. In this approach, a meta-learner accepts the meta-features as a feature vector and the optimal algorithm as a target. The meta-features are extracted using the reinforcement learning approach named Deep Q-network, and the random forest algorithm has been used as the classification model. The authors claim this is the first approach in model selection to utilize the reinforcement learning strategy. Despite the advantage that manual labor is not here, it suffers from a few issues like the limited number of meta-features, fixed set of algorithms, and the approach being applicable on tabular datasets.

Researchers proposed an AutoML technique for generating ensemble recommender systems [38]. This is the first research work in the field of recommender systems where AutoML is deployed. Ensemble models were used in the area of recommender systems and found to be useful compared to the existing methods. One problem in using the ensemble methods is the complex search space where an ensemble of a lot of models

needs to be evaluated and an optimal model needs to be found from those. Particle swarm optimization (PSO) is one of the meta-heuristics approaches which is used here to find out the optimal models from the search space. A comparison of other existing methods was performed based on the evaluation metrics such as RMSE, MSE, and MAE. The proposed method achieved better results in terms of these metrics.

In Collaborative Filtering for AutoML model selection, a matrix is constructed consisting of error values with models and hyperparameters on rows and columns [55]. A model is then fitted with inputs as models and hyperparameters and output as cross-validation (CV) errors. The model learns the latent meta-features and then predicts the CV error. Meta-features can be the number of samples or model performance [56, 57]. The problems addressed here are the time-constrained initialization and active learning.

4.1. Limitations

It is imperative to address a few significant limitations of the current work. To begin with, the database search did not capture all related papers; therefore, it was not able to retrieve all eligible articles. Identifying the relevant literature on AutoML combined with feature engineering techniques might not be possible with the limited works selected for this analysis. Meanwhile, in this review, we used the Google Scholar database, which limits the coverage of other journals related to the topic.

5. Conclusion

A brief literature review of the recent developments in the area of AutoML has been done in this paper. More emphasis has been on the steps of the ML pipeline like FS, FE NAS, HPO, and MS. AutoML makes the design decision on various phases of the ML pipeline easier compared to the traditional way. All parts of the ML pipeline could be automated so that users may build efficient applications. Machine learning algorithms need to be simplified further so that they can be applied by people with little or no background in machine learning. In this respect, AutoML represents a significant and promising step forward. AutoML is an excellent tool for developing models with acceptable discriminatory power. In biomedical environments, there are a number of limitations when using AutoML methods, including their inability to handle large and diverse datasets. In healthcare, AutoML has already been used in some cases, but more work is required for widespread adoption. Researchers interested in applying data science techniques to healthcare can use the survey as a basic guide. However, it is mentioned that there is still work to be done in this area.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Author Contribution Statement

Ravishankar S: Methodology, Software, Validation, Formal analysis, Writing – review & editing, Visualization, Supervision, Project administration. **Gopi Battineni:** Conceptualization, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

References

[1] Dhiman, G., Oliva, D., Kaur, A., Singh, K. K., Vimal, S., Sharma, A., & Cengiz, K. (2021). BEPO: A novel binary emperor penguin optimizer for automatic feature selection. *Knowledge-Based Systems*, *211*, 106560. https://doi.org/10.1016/j.knosys.2020.106560

[2] Ahmed, S., Sheikh, K. H., Mirjalili, S., & Sarkar, R. (2022). Binary simulated normal distribution optimizer for feature selection: Theory and application in COVID-19 datasets. *Expert Systems with Applications*, *200*, 116834. https://doi.org/10.1016/j.eswa.2022.116834

[3] Dokeroglu, T., Deniz, A., & Kiziloz, H. E. (2022). A comprehensive survey on recent metaheuristics for feature selection. *Neurocomputing, 494*, 269–296. https://doi.org/10.1016/j.neucom.2022.04.083

[4] Long, W., Xu, M., Jiao, J., Wu, T., Tang, M., & Cai, S. (2022). A velocity-based butterfly optimization algorithm for high-dimensional optimization and feature selection. *Expert Systems with Applications*, *201*, 117217. https://doi.org/10.1016/j.eswa.2022.117217

[5] Wu, Y., Xi, X., & He, J. (2022). AFGSL: Automatic feature generation based on graph structure learning. *Knowledge-Based Systems*, *238*, 107835. https://doi.org/10.1016/j.knosys.2021.107835

[6] Waring, J., Lindvall, C., & Umeton, R. (2020). Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial Intelligence in Medicine*, *104*, 101822. https://doi.org/10.1016/j.artmed.2020.101822

[7] Xue, C., Hu, M., Huang, X., & Li, C. G. (2022). Automated search space and search strategy selection for AutoML. *Pattern Recognition*, *124*, 108474. https://doi.org/10.1016/j.patcog.2021.108474

[8] Fang, J., Chen, Y., Zhang, X., Zhang, Q., Huang, C., Meng, G., ..., &, Wang, X. (2021). EAT-NAS: Elastic architecture transfer for accelerating large-scale neural architecture search. *Science China Information Sciences*, *64*(9), 192106. https://doi.org/10.1007/s11432-020-3112-8

[9] Park, K. M., Shin, D., & Yoo, Y. (2020). Evolutionary neural architecture search (NAS) using chromosome non-disjunction for Korean grammaticality tasks. *Applied Sciences*, *10*(10), 3457. https://doi.org/10.3390/app10103457

[10] Park, K. M., Shin, D., & Chi, S. D. (2021). Modified neural architecture search (NAS) using the chromosome non-disjunction. *Applied Sciences*, *11*(18), 8628. https://doi.org/10.3390/app11188628

[11] Peng, C., Li, Y., Jiao, L., & Shang, R. (2021). Efficient convolutional neural architecture search for remote sensing image scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, *59*(7), 6092–6105. https://doi.org/10.1109/TGRS.2020.3020424

[12] Zhang, M., Li, H., Pan, S., Chang, X., Zhou, C., Ge, Z., & Su, S. (2021). One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(9), 2921–2935. https://doi.org/10.1109/TPAMI.2020.3035351

[13] Fang, J., Sun, Y., Zhang, Q., Peng, K., Li, Y., Liu, W., & Wang, X. (2021). FNA++: Fast network adaptation via parameter remapping and architecture search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(9), 2990–3004. https://doi.org/10.1109/TPAMI.2020.3044416

[14] Zheng, X., Ji, R., Chen, Y., Wang, Q., Zhang, B., Chen, J., ..., & Tian, Y. (2021). MIGO-NAS: Towards fast and generalizable neural architecture search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(9), 2936–2952. https://doi.org/10.1109/TPAMI.2021.3065138

[15] Wang, K., & Guo, P. (2021). A robust automated machine learning system with pseudoinverse learning. *Cognitive Computation*, *13*, 724–735. https://doi.org/10.1007/s12559-021-09853-6

[16] Stamoulis, D., Ding, R., Wang, D., Lymberopoulos, D., Priyantha, B., Liu, J., & Marculescu, D. (2020). Single-path mobile AutoML: Efficient convnet design and nas hyperparameter optimization. *IEEE Journal of Selected Topics in Signal Processing*, *14*(4), 609–622. https://doi.org/10.1109/JSTSP.2020.2971421

[17] Zhang, M., Li, H., Pan, S., Chang, X., Ge, Z., & Su, S. (2020). Differentiable neural architecture search in equivalent space with exploration enhancement. In *34th Conference on Neural Information Processing Systems*, 1–11.

[18] Kaushik, P., Gain, A., Kortylewski, A., & Yuille, A. (2021). Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. *arXiv Preprint: 2102.11343*. https://doi.org/10.48550/arXiv.2102.11343

[19] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., . . . , & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526. https://doi.org/10.1073/pnas.1611835114

[20] Wong, C., Houlsby, N., Lu, Y., & Gesmundo, A. (2018). Transfer learning with neural AutoML. In *32nd International Conference on Neural Information Processing Systems*, 1–10.

[21] Elshawi, R., Maher, M., & Sakr, S. (2019). Automated machine learning: State-of-the-art and open challenges. *arXiv preprint:1906.02287*. https://doi.org/10.48550/arXiv.1906.02287

[22] Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, *415*, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061

[23] Gambella, C., Ghaddar, B., & Naoum-Sawaya, J. (2021). Optimization problems for machine learning: A survey. *European Journal of Operational Research*, *290*(3), 807–828. https://doi.org/10.1016/j.ejor.2020.08.045

[24] Hamdia, K. M., Zhuang, X., & Rabczuk, T. (2021). An efficient optimization approach for designing machine learning models based on genetic algorithm. *Neural Computing and Applications*, *33*(6), 1923–1933. https://doi.org/10.1007/s00521-020-05035-x

[25] Ali, Y. A., Awwad, E. M., Al-Razgan, M., & Maarouf, A. (2023). Hyperparameter search for machine learning algorithms for optimizing the computational complexity. *Processes*, *11*(2), 349. https://doi.org/10.3390/pr11020349

[26] Bacanin, N., Stoean, C., Zivkovic, M., Rakic, M., Strulak-Wójcikiewicz, R., & Stoean, R. (2023). On the benefits of using metaheuristics in the hyperparameter tuning of deep learning models for energy load forecasting. *Energies*, *16*(3), 1434. https://doi.org/10.3390/en16031434

[27] Daviran, M., Shamekhi, M., Ghezelbash, R., & Maghsoudi, A. (2023). Landslide susceptibility prediction using artificial neural networks, SVMs and random forest: Hyperparameters tuning by genetic optimization algorithm. *International Journal of Environmental Science and Technology*, *20*(1), 259–276. https://doi.org/10.1007/s13762-022-04491-3

[28] Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In F. Hutter, L. Kotthoff & J. Vanschoren (Eds.), *Automated machine learning: Methods, systems, challenges* (pp. 3–33). Springer. https://doi.org/10.1007/978-3-030-05318-5_1

[29] Shekar, B. H., & Dagnew, G. (2019). Grid search-based hyperparameter tuning and classification of microarray cancer data. In *2019 Second International Conference on Advanced Computational and Communication Paradigms*, 1–8. https://doi.org/10.1109/ICACCP.2019.8882943

[30] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(2), 281–305.

[31] Masum, M., Shahriar, H., Haddad, H., Faruk, M. J. H., Valero, M., Khan, M. A., . . . , & Wu, F. (2021). Bayesian hyperparameter optimization for deep neural network-based network intrusion detection. In *2021 IEEE International Conference on Big Data*, 5413–5419. https://doi.org/10.1109/BigData52589.2021.9671576

[32] Blum, M., & Riedmiller, M. A. (2013). Optimization of Gaussian process hyperparameters using Rprop. In *2013 European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 339–344.

[33] Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F. M., Rizzi, W., & Simonetto, L. (2018). Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Information Systems*, *74*, 67–83. https://doi.org/10.1016/j.is.2018.01.003

[34] Wu, J., Chen, X. Y., Zhang, H., Xiong, L. D., Lei, H., & Deng, S. H. (2019). Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology*, *17*(1), 26–40.

[35] Bardenet, R., Brendel, M., Kégl, B., & Sebag, M. (2013). Collaborative hyperparameter tuning. In *Proceedings of the 30th International Conference on Machine Learning*, *28*(2), 199–207.

[36] Lakhmiri, D., Digabel, S. L., & Tribes, C. (2021). HyperNOMAD: Hyperparameter optimization of deep neural networks using mesh adaptive direct search. *ACM Transactions on Mathematical Software*, *47*(3), 1–27. https://doi.org/10.1145/3450975

[37] Yu, H., Zhang, Q., Liu, T., Lu, J., Wen, Y., & Zhang, G. (2022). Meta-ADD: A meta-learning based pre-trained model for concept drift active detection. *Information Sciences*, *608*, 996–1009. https://doi.org/10.1016/j.ins.2022.07.022

[38] Gupta, G., & Katarya, R. (2021). EnPSO: An AutoML technique for generating ensemble recommender system. *Arabian Journal for Science and Engineering*, *46*(9), 8677–8695. https://doi.org/10.1007/s13369-021-05670-z

[39] Mohr, F., Wever, M., & Hüllermeier, E. (2018). ML-Plan: Automated machine learning via hierarchical planning. *Machine Learning*, *107*(8), 1495–1515. https://doi.org/10.1007/s10994-018-5735-z

[40] Romano, J. D., Le, T. T., Fu, W., & Moore, J. H. (2021). TPOT-NN: Augmenting tree-based automated machine learning with neural network estimators. *Genetic Programming and Evolvable Machines*, *22*(2), 207–227. https://doi.org/10.1007/s10710-021-09401-z

[41] Dey, S., Babakniya, S., Kanala, S. C., Paolieri, M., Golubchik, L., Beerel, P. A., & Chugg, K. M. (2021). Deep-n-Cheap: An automated efficient and extensible search framework for cost-effective deep learning. *SN Computer Science*, *2*(4), 265. https://doi.org/10.1007/s42979-021-00646-0

[42] Skrlj, B., Martinc, M., Lavrac, N., & Pollak, S. (2021). AutoBOT: Evolving neuro-symbolic representations for explainable low resource text classification. *Machine Learning*, *110*(5), 989–1028. https://doi.org/10.1007/s10994-021-05968-x

[43] Laredo, D., Ma, S. F., Leylaz, G., Schütze, O., & Sun, J. Q. (2020). Automatic model selection for fully connected neural networks. *International Journal of Dynamics and Control*, *8*(4), 1063–1079. https://doi.org/10.1007/s40435-020-00708-w

[44] Shi, M., Tang, Y., Zhu, X., Huang, Y., Wilson, D., Zhuang, Y., & Liu, J. (2022). Genetic-GNN: Evolutionary architecture search for graph neural networks. *Knowledge-Based Systems*, *247*, 108752. https://doi.org/10.1016/j.knosys.2022.108752

[45] Li, Y., & King, I. (2020). Autograph: Automated graph neural network. In *Neural Information Processing: 27th International Conference*, 189–201. https://doi.org/10.1007/978-3-030-63833-7_16

[46] Zhao, H., Yao, Q., & Tu, W. (2021). Search to aggregate neighborhood for graph neural network. In *2021 IEEE 37th International Conference on Data Engineering*, 552–563. https://doi.org/10.1109/ICDE51399.2021.00054

[47] Chen, X., Xie, L., Wu, J., & Tian, Q. (2021). Progressive DARTS: Bridging the optimization gap for nas in the wild. *International Journal of Computer Vision*, *129*(3), 638–655. https://doi.org/10.1007/s11263-020-01396-x

[48] Pham, H., & Le, Q. (2021). Autodropout: Learning dropout patterns to regularize deep networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(11), 9351–9359. https://doi.org/10.1609/aaai.v35i11.17127

[49] Lu, Z., Sreekumar, G., Goodman, E., Banzhaf, W., Deb, K., & Boddeti, V. N. (2021). Neural architecture transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(9), 2971–2989. https://doi.org/10.1109/TPAMI.2021.3052758

[50] Parmentier, L., Nicol, O., Jourdan, L., & Kessaci, M. E. (2019). TPOT-SH: A faster optimization algorithm to solve the AutoML problem on large datasets. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence*, 471–478. https://doi.org/10.1109/ICTAI.2019.00072

[51] Bahri, M., Veloso, B., Bifet, A., & Gama, J. (2020). AutoML for stream k-nearest neighbors classification. In *2020 IEEE International Conference on Big Data*, 597–602. https://doi.org/10.1109/BigData50022.2020.9378396

[52] Peng, D., Dong, X., Real, E., Tan, M., Lu, Y., Bender, G., . . . , Le, Q. (2020). PyGlove: Symbolic programming for automated machine learning. In *34th International Conference on Neural Information Processing Systems*, 1–13.

[53] Real, E., Liang, C., So, D., & Le, Q. (2020). AutoML-Zero: Evolving machine learning algorithms from scratch. In *Proceedings of the 37th International Conference on Machine Learning*, *119*.

[54] Liu, H., Brock, A., Simonyan, K., & Le, Q. (2020). Evolving normalization-activation layers. In *34th Conference on Neural Information Processing Systems*, 1–12.

[55] Yang, C., Akimoto, Y., Kim, D. W., & Udell, M. (2019). OBOE: Collaborative filtering for AutoML model selection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1173–1183. https://doi.org/10.1145/3292500.3330909

[56] Chiroma, H., Nickolas, P., Faruk, N., Alozie, E., Olayinka, I. F. Y., Adewole, K. S., . . . , & Adediran, Y. A. (2023). Large scale survey for radio propagation in developing machine learning model for path losses in communication systems. *Scientific African*, *19*, e01550. https://doi.org/10.1016/j.sciaf.2023.e01550

[57] Varghese, D. M., Arya, A., & Ahmad, S. (2023). Feature-engineering from protein sequences to predict interaction sites using machine learning. In L. Kurgan (Ed.), *Machine learning in bioinformatics of protein sequences: Algorithms, databases and resources for modern protein bioinformatics* (pp. 129–151). World Scientific. https://doi.org/10.1142/9789811258589_0005

How to Cite: Ravishankar, S., & Battineni, G. (2025). A Survey on Recent Advancements in Auto-Machine Learning with a Focus on Feature Engineering. *Journal of Computational and Cognitive Engineering*, *4*(1), 56–63. https://doi.org/10.47852/bonviewJCCE3202720