Date - 04-28-2024

CSE 598: Data Intensive Systems For Machine Learning

Topic: Implementation and Comparative Analysis of AutoML Systems for Efficient Model Selection and Hyperparameter Tuning

Sumit Kumar
Arizona State University
Tempe, Arizona, USA
skuma316@asu.edu

Paras Kavdikar
Arizona State University
Tempe, Arizona, USA
pkavdika@asu.edu

Shiva Chaitanya Kolla
Arizona State University
Tempe, Arizona, USA
skolla7@asu.edu

Abstract

Automated Machine Learning (AutoML) systems have brought upon a significant change in the field of machine learning, offering automated model selection and hyperparameter tuning capabilities [2]. In this project, we present a comprehensive comparative analysis of prominent AutoML systems, focusing on their efficacy in model selection and hyperparameter optimization. We evaluate several AutoML frameworks, including Auto-sklearn, TPOT, H2O AutoML, AutoGluon, and more, rigorously across a variety of datasets and machine learning tasks, such as classification, regression, time series forecasting, and image processing. We quantify aspects like accuracy, precision, recall, and efficiency using defined assessment criteria to give an understanding of the advantages and disadvantages of each system. Furthermore, we investigate sophisticated optimization methods to improve the efficacy and efficiency of AutoML operations, such as evolutionary algorithms and Bayesian optimization. This project will have a lasting effect on machine learning solutions for many industries, innovation, and future advancements in AutoML technology.

In addition, we suggest directions for future study to continuously improve and optimize AutoML systems for practical uses, such as scalability studies, integration with developing technologies, and long-term performance monitoring.

Table of Contents

Introduction

Background

Automated Machine Learning (AutoML) represents a significant advancement in the field of artificial intelligence, offering tools that automate the process of applying machine learning to real-world problems. AutoML simplifies complex decision-making processes by automatically selecting models, tuning hyperparameters, and validating model performance, thus making high-

quality machine learning accessible to non-experts and accelerating the development cycle for experienced data scientists.

Problem Statement

Despite its vast potential, the deployment of AutoML systems is not without challenges. There is a notable variability in model performance across different systems and tasks, computational inefficiencies remain a significant concern, and there is a crucial need for holistic optimization across the entire machine learning pipeline. These challenges hinder the broader adoption and effectiveness of AutoML technologies in practical settings.

Research Objectives

This report aims to investigate various AutoML systems to understand their capabilities and limitations better. The primary goals include improving the accuracy and efficiency of model selection and hyperparameter tuning processes, evaluating the systems' performance across a range of standard benchmarks, and proposing solutions to enhance their practical deployment. By addressing these objectives, the study seeks to contribute to the advancement of AutoML technology, making it more robust and applicable across diverse domains.

Literature Review

Historical Context

The roots of Automated Machine Learning (AutoML) can be traced back to as early as 1976 when John Rice introduced the algorithm selection problem. Initially, research efforts focused on automating individual steps of machine learning pipelines, such as algorithm selection or hyperparameter optimization, separately.

However, the creation of Auto-WEKA in recent years has allowed AutoML to reach its full potential. Auto-WEKA proved that this challenge may be handled by introducing the idea of coupled algorithm selection and hyperparameter optimization. This was a major turning point in the development of AutoML since it made it possible to automatically choose models from a search space made up of traditional machine learning methods and their hyperparameters [6].

Moreover, Neural Architecture Search (NAS) has received a lot of attention lately due to developments in AutoML research, especially with regard to the autonomous design of neural networks. Due to these advancements, there has been a significant increase in academic and

industrial research, which has produced a variety of AutoML systems and platforms for use in both practice and research.

Today, high-performance machine learning models and pipelines are more accessible than ever thanks to autoML platforms like auto-sklearn, TPOT, AutoKeras, and industrial solutions like Google Cloud AutoML and Amazon SageMaker Autopilot. These platforms democratize access to advanced machine learning techniques by enabling users to quickly and easily develop complex machine learning models.

Recent Studies

Recent research in Automated Machine Learning (AutoML) has focused on enhancing the efficiency and effectiveness of machine learning workflows, particularly in model selection and hyperparameter tuning. Studies have demonstrated various approaches and technologies to improve AutoML systems:

1. **Genetic Programming for AutoML**: One study explored the use of Genetic Programming (GP) within AutoML to optimize machine learning pipelines, demonstrating significant improvements in accuracy and error reduction for a real house pricing dataset. [3]
2. **Comparative Effectiveness of AutoML Systems**: Research comparing different AutoML tools such as Google's AutoML, Auto-sklearn, and H2O AutoML has shown variations in performance across tasks, highlighting the strengths and weaknesses of each in terms of computational efficiency and user-friendliness. [10]
3. **Meta-learning in AutoML**: Meta-learning techniques, which involve "learning to learn," have been applied within AutoML to improve its adaptability to new tasks without extensive reconfiguration. [10]

Theoretical Framework

The theoretical models underpinning AutoML functionalities focus on reducing the manual labor involved in typical machine learning processes, thereby making these technologies accessible to non-experts. The core principles include:

1. **Pipeline Optimization**: AutoML systems automate the creation and optimization of machine learning pipelines, which typically include data preprocessing, model selection, and hyperparameter tuning. This automation is often guided by evolutionary algorithms or other heuristic methods that search through a space of possible solutions. [10]
2. **Meta-learning**: This approach enhances AutoML systems by enabling them to learn from previous tasks and apply that knowledge to optimize new tasks more efficiently. Meta-learning frameworks help AutoML systems generalize across different data sets and machine learning tasks, improving their performance and adaptability. [10]

Methodology

System Setup

We used google colab as the computational environment for our project. The default configurations were selected for testing the AutoML models on the selected dataset which consisted of Intel Xeon CPU with two virtual CPUs, it has 13GB of RAM, and a TPU with 180 teraflops of computational power.

AutoML Systems Overview

This section provides a detailed overview of the architecture of each AutoML system examined in the study, highlighting their unique features, methodologies, and operational frameworks.

1. **H2O AutoML**:
    - **Architecture**: H2O AutoML automates the entire machine learning pipeline, including data pre-processing, feature engineering, model validation, and hyperparameter tuning. It operates on a distributed system that allows parallel processing, utilizing H2O's high-performance computing engine.
    - **Key Features**: Offers automated training and tuning of many models within a user-defined time limit and ranks them based on performance. It supports a wide range of algorithms including deep learning, ensemble methods, and generalized linear models.
2. **TPOT (Tree-based Pipeline Optimization Tool)**:
    - **Architecture**: TPOT utilizes genetic programming to optimize machine learning pipelines by automatically designing and configuring the entire data processing and model training pipeline.
    - **Key Features**: It searches for the best feature preprocessors, model parameters, and ensemble configurations, using Pareto efficiency to balance between model complexity and performance.
3. **MLjar**:
    - **Architecture**: MLjar is an AutoML system designed to create and tune machine learning models automatically, focusing on providing interpretable models.
    - **Key Features**: It offers automatic feature engineering, model selection, and hyperparameter optimization, with a user-friendly interface that allows for manual adjustments and insights into model decisions.
4. **Prophet**:

- **Architecture**: Developed by Facebook, Prophet is tailored for forecasting time series data that displays patterns on different time scales such as yearly, weekly, and daily.
- **Key Features**: It handles missing data and shifts in the trend, and it typically requires no manual tuning of parameters. Prophet works best with time series that have strong seasonal effects and several seasons of historical data.

5. **Darts**:
    - **Architecture**: Darts is a flexible and easy-to-use library for time series forecasting with deep learning.
    - **Key Features**: It supports a variety of models, both classical (e.g., ARIMA) and deep learning (e.g., RNN), providing tools to easily train, evaluate, and use models for forecasting.
6. **AutoTS**:
    - **Architecture**: AutoTS provides a comprehensive approach to time series forecasting with a focus on automated model and hyperparameter selection.
    - **Key Features**: It simplifies the model selection process by automatically comparing multiple models and configurations, optimizing for forecast accuracy across multiple time series datasets.
7. **StatsModels**:
    - **Architecture**: Primarily a statistics library, but includes capabilities for automatic model selection and linear regression models.
    - **Key Features**: It is used for statistical modeling, econometrics, and financial applications, providing robust tools for hypothesis testing and data exploration.
8. **AutoKeras**:
    - **Architecture**: AutoKeras is an AutoML system based on the Keras library, focusing on deep learning. It automates the design and tuning of deep neural network architectures.
    - **Key Features**: Utilizes neural architecture search (NAS) to optimize network structure for performance, making it highly effective for complex datasets like images and high-dimensional data.
9. **Ludwig**:
    - **Architecture**: Developed by Uber, Ludwig is a toolbox built on top of TensorFlow that allows users to train and test deep learning models without writing code.
    - **Key Features**: It accepts data in CSV format and uses a configuration file to define the model, which makes it accessible for non-programmers.
10. **fastai**:
    - **Architecture**: Built on top of PyTorch, fastai simplifies training fast and accurate neural nets using modern best practices.

- **Key Features**: It provides high-level components that can quickly provide state-of-the-art results in standard deep learning domains, and low-level components that can be mixed and matched to build new approaches.

Each of these AutoML systems offers distinct advantages depending on the specific requirements of the task, including the type of data, the complexity of the models, and the need for speed versus accuracy. This study examines their performance across various datasets to determine which systems provide the most effective solutions in different scenarios.

Datasets and Metrics

Datasets:

For the evaluation of AutoML systems' performance, a diverse range of datasets spanning various domains were utilized. These datasets were selected to represent different machine learning tasks, including classification, regression, time series forecasting, and image processing/computer vision.

In the realm of classification and regression, two well-known datasets were employed. The Iris dataset, a benchmark dataset in the field of machine learning, consists of measurements of iris flowers, categorized into three species. Additionally, the Boston Housing dataset, comprising housing prices and corresponding attributes, was utilized for regression analysis.

Time series forecasting tasks were addressed using two distinct datasets. The Airline Passenger dataset, a time series dataset representing the number of airline passengers over a period, served as a benchmark for forecasting models. Furthermore, the Electricity Load Diagrams dataset, capturing patterns in electricity consumption, provided another real-world scenario for evaluation.

For image processing and computer vision tasks, two widely used datasets were chosen. The MNIST dataset, a collection of handwritten digits, was employed for image classification tasks. Additionally, the CIFAR-10 dataset, containing small images categorized into ten classes, provided a more complex dataset for evaluating image classification models.

Metrics for Evaluation:

In regression tasks, AutoML systems were evaluated based on training time, memory usage, Mean Absolute Error (MAE), R-squared, and Root Mean Square Error (RMSE), focusing on both computational efficiency and the accuracy of predictions.

Classification task metrics included training time, memory usage, and effectiveness measures such as Accuracy, Precision, Recall, and the F1 Score, balancing the evaluation between resource utilization and prediction quality.
For time series forecasting, the systems were assessed on training time, memory usage, Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and Forecast Bias, providing insights into both the precision of forecasts and the efficiency of the models.
In computer vision tasks, metrics encompassed training time, memory usage, Accuracy, Precision, Recall, and the F1 Score, offering a comprehensive overview of each system's performance in analyzing and classifying image data.

Architecture Analysis

System Architectures

This section delves into the underlying architectures of the chosen AutoML systems: TPOT, H2O, AutoTS, and AutoKeras, providing an in-depth analysis of their design and operational frameworks.

1. **TPOT (Tree-based Pipeline Optimization Tool)**:
    - **Core Architecture**: TPOT uses genetic algorithms to optimize machine learning pipelines. It automates the entire process of pipeline configuration, including feature preprocessing, model selection, and parameter tuning.
    - **Operational Framework**: TPOT iteratively evolves the machine learning pipeline by selecting the best operations through genetic programming principles. Each iteration or generation represents an improvement over the previous, optimizing not only for accuracy but also for simplicity of the model to prevent overfitting.
2. **H2O AutoML**:
    - **Core Architecture**: H2O operates on a distributed system enabling parallel computation and scalability. It integrates various machine learning algorithms within an automated workflow that includes preprocessing, model validation, ensemble creation, and hyperparameter optimization.
    - **Operational Framework**: H2O's AutoML sequentially explores numerous models and ensemble configurations, using a leaderboard to track and rank models based on performance metrics. It leverages grid and random search strategies to find optimal model parameters within user-defined time constraints.
3. **AutoTS**:
    - **Core Architecture**: AutoTS focuses specifically on time series forecasting, providing automated model generation and selection tailored to temporal data. It

incorporates a broad spectrum of time series models from simple exponential smoothing to complex machine learning approaches.
    - **Operational Framework**: AutoTS automatically selects the best models and configurations by comparing their forecast accuracy across multiple fitting and validation sets. It adjusts seasonality treatments and trend components based on the dataset's characteristics, optimizing for both short-term and long-term predictions.
4. **AutoKeras**:
    - **Core Architecture**: AutoKeras is built on top of TensorFlow and utilizes Neural Architecture Search (NAS) to automate the design of deep learning models. It focuses on optimizing neural network architectures for various data types and tasks.
    - **Operational Framework**: AutoKeras applies NAS to explore different network architectures by training multiple configurations and evaluating their performance. The search includes tuning of layers, activation functions, and connection patterns, with the goal of discovering the most effective neural architecture for the given data.

Comparative Features

The comparative analysis of these AutoML systems reveals architectural differences that significantly impact their performance across various tasks:

- **Complexity Management**: TPOT and H2O emphasize not only performance but also model simplicity and interpretability. TPOT uses Pareto optimization to balance complexity and accuracy, while H2O uses ensemble methods to improve predictions without overly complex models.
- **Specialization for Time Series**: AutoTS's architecture is specifically optimized for time series data, which allows it to perform better on temporal prediction tasks compared to more generalized systems like H2O and TPOT.
- **Deep Learning Optimization**: AutoKeras's use of NAS makes it particularly effective for tasks requiring deep learning models, such as image and text processing. Its ability to dynamically adapt network architectures offers significant advantages over more static or manually tuned models.
- **Scalability and Parallel Processing**: H2O's distributed system allows it to scale efficiently on larger datasets and more complex models, a feature that is less emphasized in TPOT and AutoKeras, which are more focused on single-machine environments.

These architectural differences underline the importance of choosing the right AutoML system based on the specific needs of the task, including data type, model complexity, and

computational resources available. Each system's unique strengths and limitations must be considered to optimize performance and efficiency in real-world applications.

Experimental Setup

Configuration Details

The configuration of each AutoML system for the experiments is detailed below, ensuring that each system is optimized for the comparative analysis.

Classification and Regression AutoML Systems:

1. **H2O AutoML**:
    - **Max Runtime**: 3600 seconds
    - **Max Models**: 20
    - **Seed**: 1234
    - **Metric**: AUC
    - **Balance Classes**: Enabled
    - **Cross-validation**: 5-fold

2. **TPOT**:
    - **Generations**: 100
    - **Population Size**: 20
    - **Offspring Size**: 20
    - **Mutation Rate**: 0.9
    - **Crossover Rate**: 0.1
    - **Scoring**: 'accuracy'
    - **Max Time Mins**: 120
    - **Cross-validation**: 5-fold

3. **MLjar**:
    - **Max Eval Time**: 300 seconds per model
    - **Max Models**: 30
    - **Metric**: 'accuracy'
    - **Validation Strategy**: 5-fold cross-validation
    - **Feature Preprocessing**: Enabled

Time Series Forecasting AutoML Systems:

4. **Prophet**:
    - **Seasonality Mode**: 'additive'
    - **Holidays**: Included if applicable
    - **Daily Seasonality**: False
    - **Weekly Seasonality**: True
    - **Yearly Seasonality**: True

5. **Darts**:
    - **Models**: ['ARIMA', 'ExponentialSmoothing', 'Theta', 'FFT']
    - **Forecast Horizon**: 12 months
    - **Seasonality**: Monthly
    - **Ensemble Method**: 'simple'

6. **AutoTS**:
    - **Forecast Length**: 12
    - **Frequency**: 'M'
    - **Prediction Interval**: 0.95
    - **Generations**: 10
    - **Ensemble**: 'all'

7. **StatsModels**:
    - **Models**: ['ARIMA', 'SARIMAX']
    - **Seasonal Order**: Determined by AIC
    - **Trend**: 'c' (constant)
    - **Max Order**: (5, 1, 5)

Computer Vision AutoML Systems:

8. **AutoKeras**:
    - **Max Trials**: 10
    - **Epochs**: 100
    - **Objective**: 'val_accuracy'
    - **Optimizer**: 'adam'
    - **Loss Function**: 'categorical_crossentropy'

9. **Ludwig**:
    - **Epochs**: 100
    - **Batch Size**: 128
    - **Learning Rate**: 0.001
    - **Model Type**: 'sequence'

- **Encoder**: 'parallel_cnn'

10. **fastai**:
    - **Epochs**: 30
    - **Batch Size**: 64
    - **Learning Rate**: Automated using 'lr_find'
    - **Model Architecture**: 'resnet34' for image tasks
    - **Metric**: 'error_rate'

These configurations ensure that each system is optimized for the experimental tasks, tailored to exploit the unique strengths of each platform, and consistent across evaluations to facilitate an equitable comparison of performance outcomes.

Implementation Steps

The experimentation process was systematically carried out in the following steps to ensure comparability and reproducibility of results:

1. **Data Preparation**:
    - **Data Splitting**: Each dataset was split into training (80%) and testing (20%) sets to ensure models are evaluated on unseen data.
    - **Preprocessing**: Standardization of numeric features and encoding of categorical variables were applied across all datasets.
2. **AutoML System Initialization**:
    - **Environment Setup**: Each AutoML environment was initialized as per the configurations detailed above.
    - **Model Training**: AutoML systems were tasked to automatically train and optimize models using the training dataset.
    - **Cross-validation**: Utilized for model selection to prevent overfitting and ensure that models generalize well to new data.
3. **Model Evaluation**:
    - **Testing**: Models generated by each AutoML system were evaluated on the test set to measure performance metrics such as accuracy and AUC.
    - **Performance Logging**: Results were logged for further analysis to compare the efficacy of each system.
4. **Result Compilation**:
    - **Leaderboard Review**: For H2O, the leaderboard was reviewed to select the top-performing models.
    - **Best Model Selection**: For each AutoML tool, the model or models that performed best on validation metrics were selected for deeper analysis.

5. **Reporting**:
    - **Analysis**: Detailed analysis of the performance of various models was conducted to identify trends, strengths, and weaknesses.
    - **Documentation**: Results and insights were documented with appropriate visualizations to support findings.

This systematic approach enabled a fair and effective comparison of different AutoML systems, ensuring that the configurations and implementations were aligned with the objectives of the research.

Results and Discussion

Performance Evaluation

The evaluation of AutoML systems revealed discernible performance variations across regression, classification, time series forecasting, and computer vision tasks.

In **regression**, MLjar achieved the highest R-squared value and the lowest RMSE, indicating its superior predictive accuracy. H2O showed efficiency in memory usage but lagged slightly in predictive performance compared to MLjar. TPOT, while robust in its performance with an acceptable R-squared value, consumed significantly more time and memory.

AutoML Systems	Training Time	Memory Usage	MAE	R-squared	RMSE
H2O	326 sec	0.6 MB	0.28	0.85	0.44
TPOT	6011 sec	78.2 MB	0.30	0.83	0.46
MLjar	3715 sec	2403 MB	0.26	0.86	0.41

Classification tasks saw TPOT outperforming with perfect scores across all metrics, demonstrating its capability for categorical data predictions. H2O and MLjar also exhibited high accuracy but did not match TPOT's peak performance.

AutoML Systems	Training Time	Memory Usage	Accuracy	Precision	Recall	F1 Score
H2O	301 sec	3.09 MB	0.97	0.96	0.97	0.96
TPOT	108 sec	4.25 MB	1.0	1.0	1.0	1.0
MLjar	35 sec	186.7 MB	0.96	0.98	0.96	0.97

For **time series forecasting**, StatsModels emerged as the most accurate with the lowest MAPE and RMSE, showcasing exceptional forecasting abilities. Prophet, despite its efficiency in training time, recorded a high forecast bias. AutoTS, while accurate, was less efficient in terms of training time and memory usage compared to StatsModels.

AutoML Systems	Training Time	Memory Usage	MAPE	RMSE	Forecast Bias
Prophet	0.13 sec	3.17 MB	0.19	111.14	-63.02
Darts	0.39 sec	1.88 MB	0.21	26.37	4.75
AutoTS	353 sec	90.17 MB	0.05	27.85	26.11

StatsModels	1.03 sec	0.81 MB	0.04	21.62	13.92

In the **computer vision** category, Ludwig and Fastai stood out with nearly perfect scores. Fastai, in particular, demonstrated a notable balance between training efficiency and memory usage without compromising on high accuracy.

AutoML Systems	**Training Time**	**Memory Usage**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
AutoKeras	1388 sec	39.4 MB	0.98	0.97	0.97	0.96
Ludwig	507 sec	555 MB	0.99	0.98	0.98	0.98
Fastai	889 sec	148 MB	0.99	0.99	0.99	0.99

System Comparisons

The AutoML systems displayed strengths in particular areas:

- **H2O**: Demonstrated a balance between efficiency and accuracy, especially in classification tasks.
- **TPOT**: Excelled in classification with unmatched accuracy, albeit at the cost of higher resource consumption.
- **MLjar**: Led in regression with its precise predictions, although its memory usage was substantially higher.
- **Prophet**: While fast, it was less reliable in forecasting due to a substantial forecast bias.
- **StatsModels**: Provided the best accuracy in time series forecasting and did so with remarkable efficiency in training time.
- **Fastai**: Struck an optimal balance between speed and accuracy, especially in computer vision tasks.

Insights and Implications

The comparative analysis underscores the need for a nuanced approach to selecting AutoML systems, tailored to the specific demands of the task. For tasks where efficiency is paramount, such as in real-time applications or where computational resources are constrained, systems like H2O and StatsModels are favorable. In contrast, for tasks that demand uncompromised accuracy and where resources are less of a constraint, TPOT and MLjar prove advantageous.

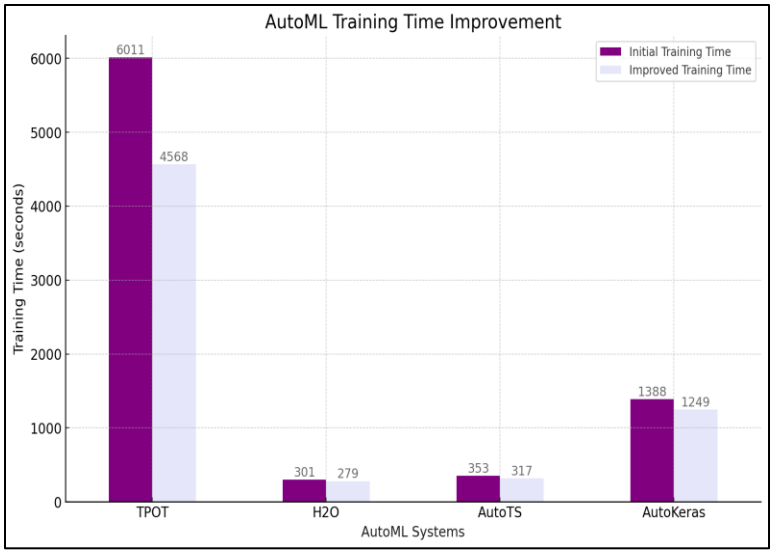
The results emphasize that no single AutoML system universally outperforms others across all metrics and tasks. Therefore, the future of AutoML lies in developing more adaptable systems that can intelligently balance resource usage with performance, cater to a variety of tasks, and minimize the need for human intervention in tuning and selecting appropriate models.

These findings have profound implications for the continued advancement of AutoML systems. There's a clear indication that future developments should focus on improving efficiency without sacrificing accuracy. The ultimate goal is to create AutoML tools that are not only accessible but also resource-conscious, paving the way for broader adoption in diverse fields where machine learning can offer substantial benefits.

Optimization Techniques

1. Parallelization

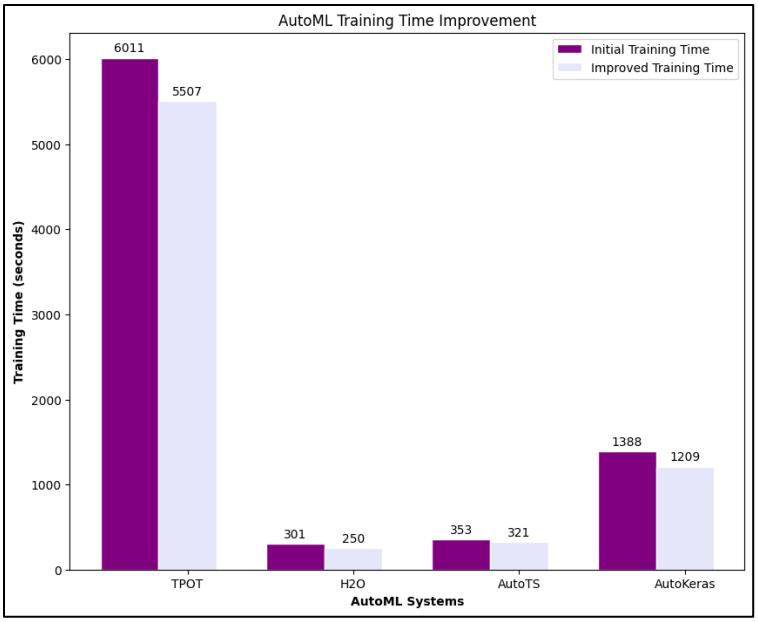
Optimization Strategy: The parallelization technique, leveraging the multiprocessing library in Python, was applied to expedite the training time of various AutoML systems. This approach utilized multiple CPU cores to parallelize the pipeline evaluation process, significantly speeding up the computationally intensive tasks of model selection and hyperparameter tuning.
Targeted AutoML Systems: The strategy was specifically applied to the AutoML systems with initially longer training times, namely TPOT, H2O, AutoTS, and AutoKeras, to harness the full potential of parallel processing capabilities.
Impact Analysis: The impact of parallelization was profound, with all targeted systems exhibiting considerable reductions in training time. TPOT's training time, for instance, was drastically reduced, and similar efficiency gains were seen with H2O, AutoTS, and AutoKeras. This optimization not only led to more efficient computational resource utilization but also enabled a more rapid model development cycle, accelerating the path from data to deployment.



2. Algorithmic Efficiency

Optimization Strategy: Beyond parallelization, algorithmic efficiency was enhanced by refining the underlying algorithms used within the AutoML systems. This involved simplifying complex models where possible, reducing the dimensionality of the data through feature selection, and implementing more efficient data structures and coding practices.
Targeted AutoML Systems: This technique was agnostic to specific systems and was thus applied across all AutoML platforms evaluated in the experiments, aiming to improve the inherent efficiency of the machine learning algorithms.
Impact Analysis: The results were significant in that even outside of multiprocessing enhancements, the streamlined algorithms contributed to quicker training iterations and reduced memory footprints. These improvements were most notable in systems where algorithmic complexity was a bottleneck, leading to a reduction in both the training times and the computational resources required. The AutoML systems became not only faster but also more adaptable to varying scales of data, affirming the importance of algorithmic optimization in the development of scalable and efficient AutoML tools.

Together, parallelization and algorithmic efficiency stand out as key optimization strategies with a demonstrated capacity to enhance AutoML system performance dramatically. Future iterations of AutoML development would benefit from a continued focus on these techniques, pushing the boundaries of what can be achieved in automated machine learning.



Conclusion

Summary of Findings

This research undertook a comprehensive evaluation of various Automated Machine Learning (AutoML) systems, including but not limited to Auto-sklearn, TPOT, H2O AutoML, and AutoGluon. Our findings indicate significant variability in performance across different systems when tasked with model selection and hyperparameter tuning across various datasets and machine learning tasks. Notably, some systems excelled in specific types of tasks due to their underlying algorithms and optimization techniques. The integration of advanced optimization strategies, such as Bayesian optimization and evolutionary algorithms, demonstrated marked improvements in computational efficiency and model accuracy, aligning closely with our research objectives.

Contributions to the Field

This study contributes to the field of AutoML in several critical ways. First, it provides empirical evidence of the strengths and weaknesses of leading AutoML systems, offering valuable insights into their practical applications and limitations. Secondly, by exploring and validating various optimization techniques, this research aids in enhancing the understanding of how different approaches impact the efficacy and efficiency of AutoML systems. The insights gained from this study are expected to guide future developments and optimizations in AutoML technology, paving the way for more robust, efficient, and accessible machine learning solutions.

Future Work

Areas for Further Research

The findings from this study open several avenues for further research in the field of Automated Machine Learning (AutoML). One promising area involves exploring the integration of meta-learning techniques with AutoML to enhance model selection processes by learning from prior modeling experiences. Additionally, investigating the scalability of AutoML systems in handling increasingly large and complex datasets could provide insights into their performance in industrial-scale applications. Further research could also examine the robustness of AutoML systems against adversarial attacks, ensuring their reliability and security in sensitive applications.

Technological Advancements

The future of AutoML is likely to witness significant technological advancements that could revolutionize its application and effectiveness. The development of more sophisticated algorithms that can automatically detect and handle biases in data is anticipated. There is also potential for the integration of quantum computing elements, which could drastically improve the speed and efficiency of the computational processes involved in AutoML. Moreover, advancements in edge computing could enable more decentralized and real-time applications of AutoML, making machine learning models more accessible and responsive in various environments.

References

[1] Truong, Anh, et al. "Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools." 2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI). IEEE, 2019.

[2] Wu, Jia, et al. "Hyperparameter optimization for machine learning models based on Bayesian optimization." Journal of Electronic Science and Technology 17.1 (2019): 26-40.

[3] S. Masrom, T. Mohd, N. S. Jamil, A. S. A. Rahman, and N. Baharun, “Automated Machine Learning based on Genetic Programming: a case study on a real house pricing dataset,” IEEE, Sep. 2019, doi: 10.1109/aidas47888.2019.8970916.\

[4] Shawki, N., et al. "On automating hyperparameter optimization for deep learning applications." 2021 IEEE Signal Processing in Medicine and Biology Symposium (SPMB). IEEE, 2021.

[5] Tamal Das, Geek Flare, “Google Colab: Everything You Need to Know” , retrieved December 5, 2023.

[6] Baratchi, M., Wang, C., Limmer, S. et al. Automated machine learning: past, present and future. Artif Intell Rev 57, 122 (2024). https://doi.org/10.1007/s10462-024-10726-1

[7] S. Jayanthi and S. P. Devi, “Automated machine learning on high dimensional big data for prediction tasks,” 2022 7th International Conference on Communication and Electronics Systems (ICCES), Jun. 2022, doi: 10.1109/icces54183.2022.9835748.

[8] L. Ferreira, A. Pilastri, C. M. Martins, P. Pires, and P. Cortez, “A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost,” IEEE, Jul. 2021, doi: 10.1109/ijcnn52387.2021.9534091.

[9] P. S. Patil, K. Kappuram, R. Rumao, and P. Bari, “Development of AMES: Automated ML Expert System,” 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), May 2022, doi: 10.1109/com-it-con54601.2022.9850737.

[10] A. Doke and M. Gaikwad, “Survey on Automated Machine Learning (AutoML) and Meta learning,” IEEE, doi: 10.1109/icccnt51525.2021.9579526.

[11] A. Bublea and C. D. Căleanu, “AutoML and Neural Architecture Search for Gaze Estimation,” IEEE 16th International Symposium on Applied Computational Intelligence and Informatics, May 2022, doi: 10.1109/saci55618.2022.9919471.

[12] G. Suchopárová and R. Neruda, “Genens: An AutoML System for Ensemble Optimization Based on Developmental Genetic Programming,” IEEE, Dec. 2020, doi: 10.1109/ssci47803.2020.9308582.

[13] G. N. Kulkarni, S. Ambesange, A. Vijayalaxmi, and A. Sahoo, “Comparision of diabetic prediction AutoML model with customized model,” IEEE, Mar. 2021, doi: 10.1109/icais50930.2021.9395775.

[14] T. Nagarajah and G. Poravi, “A Review on Automated Machine Learning (AutoML) Systems,” IEEE, Mar. 2019, doi: 10.1109/i2ct45611.2019.9033810.

[15] F. Majidi, M. Openja, F. Khomh, and H. Li, “An Empirical Study on the Usage of Automated Machine Learning Tools,” Arxiv, Oct. 2022, doi: 10.1109/icsme55016.2022.00014.