International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2019): 7.583

Benchmarking Machine Learning Tools and Development Process for Automotive Embedded Controls

Roopak Ingole[1], Ruchi Gupta Neema[2]

[1]Manager – Advanced Embedded Software Corporate Research & Technology Cummins Inc. Columbus IN, USA
Email: *roopak.ingole[at]cummins.com*

[2]Technology Consultant – Advanced Embedded Software, Corporate Research & Technology Cummins Inc. Columbus IN, USA
Email: *ruchigpt25[at]gmail.com*

Abstract: ***This study systematically benchmarks a variety of machine learning (ML) tools for their application in automotive embedded controls, with a particular focus on engine control units (ECUs). The advent of ML in the automotive industry has catalyzed significant advancements in vehicle automation, enhancing safety, efficiency, and performance. However, the deployment of ML technologies in embedded automotive systems presents unique challenges due to the stringent requirements for real-time processing and limited computational resources. In this research, we evaluate several ML tools and frameworks, including both commercial software and custom-developed algorithms, to determine their suitability for real-time automotive applications. Using criteria such as computational efficiency, memory usage, and ease of integration with existing automotive systems, we provide a comprehensive comparison and analysis. The tools examined range from high-level programming environments like Python and MATLAB to specific commercial services tailored for embedded systems. Our findings reveal significant variations in the performance and applicability of these tools in an automotive context. We also discuss the implications of these findings for the design and optimization of ML-driven automotive systems. The outcomes of this study offer valuable insights for automotive engineers and system designers, aiding in the selection of optimal ML tools that meet the dual demands of performance and practical implementation in embedded systems. This research underscores the potential of ML to revolutionize automotive systems and lays the groundwork for future innovations in this rapidly evolving field.***

Keywords: Machine Learning Tool, Automotive Embedded Controls

1. Introduction

The rapid evolution of machine learning (ML) technologies has significantly transformed various industries by enhancing their operational efficiencies and introducing advanced capabilities. Particularly in the automotive sector, ML has emerged as a pivotal technology, driving substantial innovations from autonomous vehicles to predictive maintenance systems. The integration of ML in automotive embedded controls promises enhanced safety, reliability, and performance, addressing both current and future challenges in vehicle automation.

As vehicles become increasingly intelligent, the complexity of embedded systems, including Electronic Control Units (ECUs), also escalates. These systems are critical for managing various vehicle functions such as engine performance, fuel efficiency, and safety features. ML algorithms, when embedded within these systems, can learn from operational data, adapt to new conditions, and perform complex decision-making processes autonomously. This capability not only improves the vehicle's performance but also ensures higher safety standards by minimizing human errors.

However, the implementation of ML in automotive systems is not without challenges. The selection of appropriate ML tools and algorithms that can operate within the constrained environment of automotive systems is crucial. Moreover, these tools must be compatible with real-time processing requirements and limited computational resources available onboard vehicles.

This paper aims to benchmark various ML tools and their applicability in developing robust embedded controls for automotive applications. By exploring different ML frameworks and evaluating their performance in real-world scenarios, this study provides insights into the optimal strategies for integrating ML into automotive embedded systems. We focus on several key aspects including algorithm efficiency, real-time data processing capabilities, and the integration of ML models with existing automotive software architectures.

Through a comprehensive analysis, this paper seeks to guide automotive engineers and system designers in selecting the most effective ML tools and strategies, thereby advancing the integration of artificial intelligence in automotive technologies. This research not only contributes to the theoretical knowledge in automotive ML applications but also provides a practical framework for future developments in this dynamic field.

2. Machine Learning in a nutshell

Machine learning (ML), a transformative branch of artificial intelligence (AI), has revolutionized the way computers learn and make decisions. Unlike traditional programming paradigms, where the logic and decision-making processes are explicitly defined by human programmers, machine learning enables computers to learn from data, identifying patterns and making decisions with minimal human

intervention. This shift towards data-driven decision-making represents a significant departure from conventional programming techniques, marking a new era in the development of intelligent systems [1].
The core of machine learning lies in its algorithms' ability to learn from and make predictions on data. These algorithms are designed to improve their performance as they are exposed to more data over time. The process, known as training, involves feeding large sets of data to an algorithm and allowing it to adjust and optimize its parameters to make accurate predictions or decisions. The outcome of this training process is a model that can be applied to new, unseen data to make predictions [2].

Machine learning models are broadly categorized into three types based on their learning approach: supervised learning, unsupervised learning, and reinforcement learning.

- Supervised learning is the most prevalent form, where the algorithm learns from a labeled dataset, meaning each example in the dataset is paired with the correct output. The algorithm makes predictions based on the input data and is corrected when its predictions are wrong, learning over time to make more accurate predictions. Applications of supervised learning include spam detection in emails and weather forecasting [3] [4].
- Unsupervised learning, in contrast, deals with data without labels. The algorithm learns patterns and structures from the data without any guidance on what outcomes it should predict. This type of learning is useful for discovering hidden patterns in data, such as customer segmentation in marketing analytics
- [5] [4].
- Reinforcement learning is a different approach where an agent learns to make decisions by performing actions in an environment to achieve a goal. The agent receives rewards or penalties based on its actions and learns to optimize its behavior for maximum reward. This learning method has been successfully applied in robotics, game playing, and navigation systems [4] [6].

The applications of machine learning are vast and have a profound impact on society. In healthcare, machine learning algorithms assist in diagnosing diseases, predicting patient outcomes, and personalizing treatment plans. In the financial sector, they are used for credit scoring, fraud detection, and algorithmic trading. The technology sector benefits from machine learning in improving search engine results, personalizing content recommendations, and enhancing user experiences in applications and services [7]. Automotive sector is heavily relying on use of machine learning for Autonomous Driving, Fault Isolation, and Predictive Analytics.

3. Application of ML in Automotive

The integration of machine learning (ML) into the automotive industry marks a significant leap forward in the evolution of vehicles, transforming them from mere modes of transportation to intelligent systems capable of autonomous decision-making, enhanced safety, and personalized driving experiences. Machine learning algorithms play a pivotal role in advancing automotive technologies, enabling cars to learn from vast amounts of data and improve their functions over time.

One of the most prominent applications of machine learning in the automotive sector is in the development of autonomous vehicles. Autonomous vehicles rely on sophisticated ML algorithms to process data from sensors and cameras to navigate safely through complex environments. These algorithms enable vehicles to make real-time decisions, such as identifying pedestrians, recognizing traffic signs, and avoiding obstacles, thereby increasing road safety and reducing human error [8]. Furthermore, machine learning facilitates the improvement of autonomous driving systems through continuous learning and adaptation to new driving conditions, enhancing the reliability and safety of AVs [9].

Machine learning also enhances vehicle safety features beyond autonomous driving. Advanced Driver Assistance Systems (ADAS), such as adaptive cruise control, lane-keeping assist, and automatic emergency braking, leverage ML algorithms to interpret sensor data and assist drivers in preventing accidents. These systems continuously learn from driving patterns and environmental conditions, improving their accuracy and reliability in predicting and mitigating potential hazards [10].

Predictive maintenance, powered by machine learning, uses data from vehicle sensors to predict potential failures before they occur, ensuring timely maintenance and reducing the likelihood of breakdowns.

Additionally, ML algorithms optimize energy efficiency and performance in conventional combustion, electric and hybrid vehicles. By learning driving patterns and environmental conditions, these algorithms can manage battery usage and energy regeneration more effectively, extending the vehicle's range and reducing energy consumption [11]. Use of neural network in the engine control system has proved to beneficial in the sensing technologies, where NN is used for developing virtual sensor and used it in conjunction with actual sensor helping either to avoid physical sensor or improve safety my means of redundancy.

4. Machine Learning in ECU

The Electronic Control Module (ECM) is the brain of an engine, responsible for managing various aspects of engine performance, fuel efficiency, and emissions. With advancements in machine learning (ML) technologies, the integration of ML algorithms into ECMs represents a significant leap in automotive engineering. This integration has enabled more adaptive, efficient, and cleaner engine performance than ever before.

Machine learning in the ECM can optimize engine parameters in real-time, adjusting to changing driving conditions, engine loads, and environmental factors. By processing data from sensors throughout the engine and vehicle, ML algorithms can make precise adjustments to fuel injection, ignition timing, and air intake, improving fuel efficiency and reducing emissions. These algorithms can predict the engine's needs based on historical data,

driving patterns, and real-time inputs, ensuring optimal performance under various conditions.

One of the critical areas where machine learning enhances ECM function is in predictive maintenance. By analyzing data trends and identifying anomalies in engine performance, ML algorithms can predict potential failures before they occur. This capability allows for proactive maintenance, reducing downtime and extending the engine's life. Predictive models can alert operators to issues such as filter clogs, fluid leaks, or wear in engine components, facilitating timely interventions.

The integration of ML into ECMs also supports the advancement of hybrid and electric vehicle technologies. Machine learning algorithms manage battery usage, regenerative braking, and energy distribution between the electric motor and internal combustion engine in hybrid systems, enhancing energy efficiency and vehicle range [11].

5. Machine Learning workflow for Engine Control Systems

At Cummins Corporate Research & Technology center, we started to evaluate the use of ML for embedded control system. For this we had two options, develop the process utilizing tools like Python [12] or MATLAB [13], or evaluate commercial service like Reality AI [14], C3 AI [15] or Palantir [16]. For setting up the process, we decided to evaluate both the paths by tackling the real problem and compare the results along the wide range of attributes. Inspired from the existing work from BMW [17] on oversteering detection, the problem we choose explore is to develop Clutch engagement detection model in manual transmission vehicle, will be discussed in depth in next section.

1) Clutch engagement in manual transmission

The accurate detection of clutch engagement in manual transmission vehicles is crucial for enhancing driving experience and vehicle performance (Figure 1). The manual transmission system requires the driver to engage and disengage the clutch manually, making the detection of clutch engagement pivotal for vehicle control systems. In manual transmission vehicles, clutch degradation is most commons problem due to miss-aligned clutch engagement where engine torque does not match appropriately with the transmission speed. Due to this clutch plate wears out and can pose a risk of damaging transmission or inefficient fuel economy. Hence it is important to detect exact timing of full clutch engagement before increasing the engine torque.

Traditional methods of detecting clutch engagement rely on physical sensors and heuristics, which may not always capture the nuanced stages of clutch engagement accurately. This method is not only cost inefficient but also requires service maintenance. Therefore, this research leverages machine learning (ML) techniques to offer a more precise and reliable detection mechanism, potentially improving manual transmission control strategies and contributing to the vehicle's safety features.

This study introduces a machine learning-based approach to identify the precise moments of clutch engagement, aiming to contribute to the development of a Clutch Engagement Start Control (CESC) feature for clutch protection. Utilizing a comprehensive dataset provided by the customer engineering team, various machine learning algorithms were evaluated for their effectiveness in recognizing two key events: the initial contact between clutch plates and full clutch engagement.

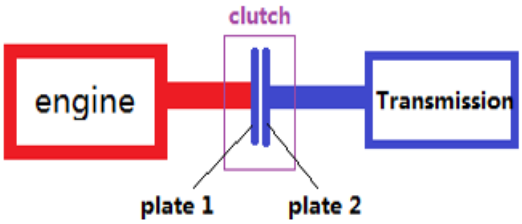

Figure 1: Clutch Assembly in powertrain

2) Use of home-grown tools (Python, Matlab)

The methodology adopted in this research is systematic and thorough, ensuring the collection of accurate data and the development of a reliable ML model. The process consists of four main stages (Figure 2):

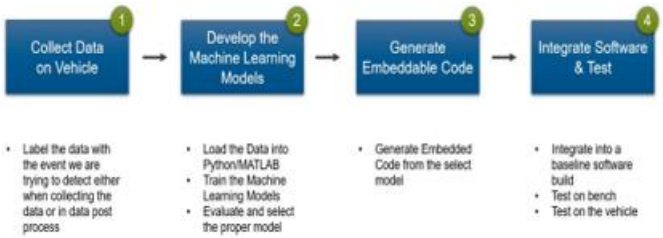

Figure 2: Machine Learning Model development process

a) Data Collection

Data was meticulously collected directly from vehicles equipped with manual transmission, focusing on the clutch's operational data during engagement and disengagement phases. The dataset includes various features such as clutch plate positions, engine speed, and torque, manually labeled with two events of interest: the beginning of clutch plates' contact and full clutch engagement. This foundational step ensured a robust dataset for training the ML models.

b) Machine Learning Process

The machine learning workflow (Figure 3) encompassed data preprocessing, feature extraction and selection, model training, optimization, and validation. Various preprocessing techniques were employed to prepare the dataset for model training, including re-labeling and addressing class imbalance. Feature extraction aimed to reduce the dimensionality of the dataset while retaining significant predictors of clutch engagement events. Utilizing advanced tools such as Python for data processing and combination of Python and MATLAB for ML model development (Figure 4), the study explored various algorithms for their suitability in accurately detecting clutch engagement events. Feature extraction and selection were critical in preparing the dataset for model training.

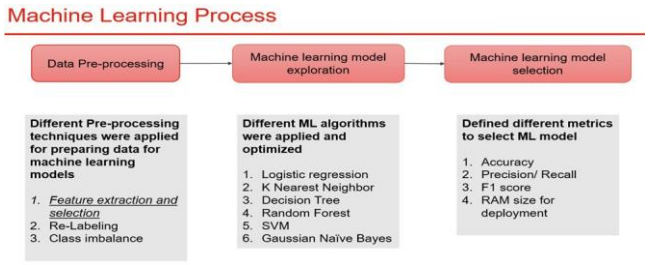

Figure 3: Machine Learning Process

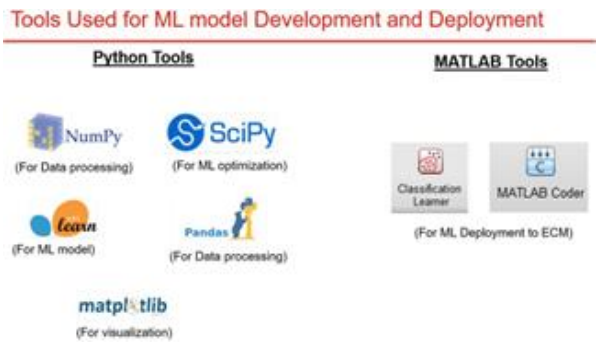

Figure 4: Machine Learning Tools

c) Model Exploration and Selection

Several machine learning algorithms were explored, including Logistic Regression, SGD Classifier, K Nearest Neighbor, Decision Tree, Random Forest, Support Vector Machine (SVM), and Gaussian Naïve Bayes. Utilizing rigorous cross-validation techniques like K-folds (Figure 5), each model's performance was evaluated based on accuracy, precision, recall, and F1-score, with a particular focus on the models' potential for embedded deployment in terms of computational efficiency and memory requirements. This phase was crucial in identifying the most effective algorithm for real- world application.

Figure 5: Machine Learning Model Optimization

d) Embeddable Code Generation

The final selected ML model was then processed using MATLAB Coder to generate embeddable code. This code could be directly integrated into the vehicle's Engine Control Module (ECM), enabling real-time clutch engagement detection.

Use of commercial tools like Reality AI

Along with in-house development of ML processes, we decided to benchmark this process with commercially available ML expert services. For this we choose to use Reality AI [14], headquartered in Columbia, Maryland, specializes in embedded AI and Tiny Machine Learning (TinyML) solutions aimed at enhancing non-visual sensing in various sectors including automotive, industrial, and commercial products. They merge machine learning with advanced signal processing math to offer efficient, fast machine learning inference capable of operating on small microcontrollers (MCUs). Reality AI's flagship product, Reality AI Tools® [18], is designed to support the full product development lifecycle with analytics derived from non- visual sensor data. Their AI solutions, which are inference- based, can be implemented across a wide range of endpoint AI applications such as industrial anomaly detection and automotive sound recognition using AI-built sensors.

As an evaluation and feasibility study, we provide Reality AI with the same problem space as Clutch engagement detection for manual transmission vehicle. We provided then will vehicle data and carved out a proof-of-concept development project. As a part of project, Reality AI will prepare a Data Coverage Matrix and Data Coverage Plan for a “Feasibility Proof-of-Concept”, will run its Readiness Assessment software on available data. Then, Reality AI will perform the initial data load, run, and test the first model and will show Cummins its results on developed ML model and its performance.

3) Integration of ML in ECU

The integration of Machine Learning model involves using the MATLAB Embedded Coder product to automatically generate C code from a selected machine learning (ML) model, followed by integrating this C code into an existing Electronic Control Unit (ECU) software framework. This method showcases a practical approach to embedding intelligent features into hardware devices, leveraging machine learning algorithms to enhance or add new functionalities. The journey of integration reveals significant insights into the technical and operational requirements needed to successfully implement ML models within constrained environments like ECUs, which are critical components in automotive, aerospace, and industrial systems. Below, we elaborate on the key learnings from this integration process:

a) Stack Requirements

Stack requirements refer to the size and configuration of the stack memory allocated for the program's execution. In the context of integrating ML models into ECUs, understanding the stack requirements is crucial because it directly impacts the reliability and performance of the system. ML models, depending on their complexity and the data they process, can have varying stack usage. Accurate estimation and optimization of stack size ensure that the system can manage the ML model's computational needs without risking stack overflow, which can lead to system crashes or unpredictable behavior.

b) RAM Requirements

RAM requirements pertain to the amount of memory needed by the ML model when integrated into the ECU software. This includes memory needed for the model's parameters (e.g., weights in a neural network), temporary variables, and any additional data structures used by the model. Efficient use of RAM is particularly important in embedded systems like ECUs, which typically have limited memory resources. Careful management and optimization of RAM usage can

help in achieving the desired balance between model performance and resource constraints, ensuring that the model runs smoothly without exhausting the system's memory.

c) Throughput Requirements

Throughput requirements focus on the system's capability to process data at a rate sufficient to meet application demands. This is a critical consideration when integrating ML models into ECUs because the effectiveness of the model often depends on its ability to process and analyze data in real-time or near-real- time.

d) Preemptive RTOS

Electronic control units run critical real-time application on Real-Time Operating System (RTOS). The nature of safety and time critical functionality in ECU highlights the importance of selecting an operating system that supports the efficient management of tasks and resources in a real-time environment. A preemptive RTOS allows tasks to be prioritized and managed in a way that ensures high-priority tasks are executed timely, which is essential for meeting the throughput requirements of ML models in ECUs. The RTOS must efficiently allocate CPU time to different tasks, including the execution of the ML model, ensuring that real-time performance targets are met. The way code is generated for selected ML model and inability be split into multiple functions, emphasizes the need for preemptive RTOS since it is required to run as atomic functionality while maintaining hard deadlines.

Integrating ML models into ECU software frameworks using tools like MATLAB Embedded Coder introduces a complex set of requirements that must be carefully managed to ensure successful deployment. Stack and RAM requirements necessitate a deep understanding of the model's memory usage and optimization strategies to fit within the constraints of embedded systems. Throughput requirements, particularly in systems running a preemptive RTOS, demand a meticulous approach to task scheduling and resource allocation to meet the real-time operational demands of the application. This journey not only underscores the technical challenges of embedding ML models in constrained environments but also highlights the importance of a systematic approach to understanding and addressing these challenges for effective integration.

4) Compare the results.

The evaluation of various machine learning algorithms revealed the Decision Tree algorithm as the most effective, with impressive performance metrics (Figure 6): an accuracy of 98.3%, precision of 0.87, recall of 0.82, and an F1-score of 0.85. Despite the Decision Tree's superior metrics, the Support Vector Machine (SVM) model was selected for deployment due to its lower RAM usage and reduced computational complexity, which are critical factors for in-vehicle systems. The SVM model demonstrated robust performance with an accuracy of 98.18%, precision of 0.90, recall of 0.74, and an F1-score of 0.81, indicating its suitability for detecting clutch engagement with high reliability while maintaining system efficiency.

Machine Learning Results Obtained

ML Classifier	Accuracy	Precision	Recall	F1-score
Logistic Regression	96%	0.78	0.35	0.48
SGD Classifier	97%	0.78	0.60	0.68
K Nearest Neighbour	97%	0.78	0.60	0.68
Decision Tree	98.3%	0.87	0.82	**0.85**
Random Forest	98.6%	0.88	0.78	**0.83**
Support Vector Machine (SVM)	98.18%	0.90	0.74	**0.81**
Guassian NB	50%	0.09	0.96	0.17

Figure 6: Machine Learning Model Performance

In the similar fashion, Reality AI provided the K-fold matrix as below, which indicates the similar outcome as in-house developed model (Figure 7).

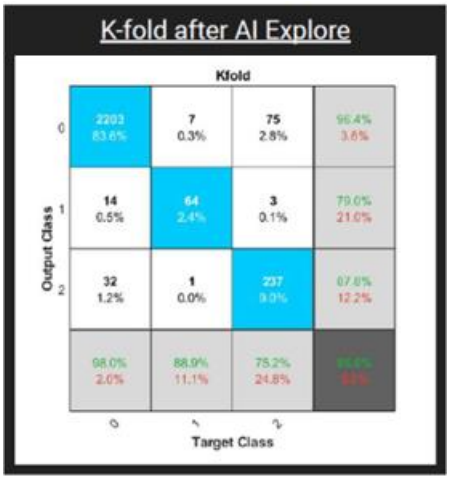

Figure 7: Reality AI Tools developed ML Model Performance

After comparing the results from in-house development with Reality AI tools, the results came remarkably close that confirms that our development process is on right track and have an option to use Reality AI tools to accelerate the model development. Apart from comparing performance of the model developed using different tools, we needed to compare other aspects of tools as shown in Figure 8. From this analysis we deduced that Reality AI is best to be used by non-AI experts to develop the models whereas MATLAB & Python bases tools can be used by AI experts.

Comparing Reality AI / MATLAB/ Python

Criteria	Weights	Python		MATLAB		Reality AI	
		Comments	Score	Comments	Score	Comments	Score
Data Exploration	8%	Offers wider set of choices in graphics package	10	Offers different library packages for visualization	9	All data exploration related work is done in backend E.g.: Deciding Class labels	8
Data type handling	5%	support various types of datatype	10	support different types of datatype	10	support various types of datatype	10
Data Preparation	15%	No support	1	No support	1	Really good Data Curate tool	8
Feature Engineering	10%	Good support but ML expertise needed	7	Excellent GUI (Classification Learner App)	8	Really good support but not Flexible	8
ML algorithm	25%	Good Support	8	Good Support (Excellent GUI)	9	Good but less algorithm (focused on deployment) (Not Flexible)	7
Hyper parameter Tuning	12%	Support different ways to do hyperparameter tuning	7	Excellent GUI (2019b)	8	Automated	8
Model Deployment	10%	There are ways to convert C code but they are not direct	8	MATLAB coder can generate C-code	10	Software supports c-code generation as they are using MATLAB in backend	10
Technical Expertise	3%	Requires technical expertise (coding in python)	2	Doesnot require lot of technical expertise (Excellent GUI)	9	Doesnot require lot of technical expertise	8
Documentation	3%	Open source (good documentation)	8	Documentation is very well written	8	Not properly documented	7
Support/Responsiveness	3%	Online community support	7	MATLAB gives good support	9	Technical Support is available	8
Cost	10%	Open Source (No cost)	10	Costly MATLAB license	2	Software has great cost	2
Overall Rating	100%	7.14		7.23		7.35	
Overall Comments		Flexible (can produce better results) ML expertise required		Excellent GUI Not Flexible Good for non-ML experts		Excellent data preparation and Feature Selection toolbox Not Flexible/Scalable Good for non-ML experts	

Figure 8: ML Tool Comparison Pugh Matrix

6. Future work

This benchmarking research marks a significant advancement in Machine Learning capability at Cummins. We used real life problem of the detection of clutch engagement in manual transmissions using machine learning algorithms. The successful application of this technology not only promises to enhance vehicle performance but also paves the way for the development of new safety and protection features, such as the CESC system. Future work will aim to refine the chosen model for even higher accuracy and explore its integration into a wider array of vehicle systems and applications. This study highlights the potential of machine learning in revolutionizing vehicle transmission systems, providing a foundation for further innovations in automotive technology. We can use exact same development process to develop the models for various powertrain problem spaces like combustion optimization, virtual sensors, engine parameter tuning, in-use emissions control, and embed those models in the ECM, EDGE devices or run those in the cloud.

7. Conclusion

The benchmarking study presented in this paper highlights the diverse capabilities and limitations of various machine learning (ML) tools when applied to automotive embedded controls, particularly within the realm of Electronic Control Units (ECUs). Our comparative analysis of both commercially available services and custom-developed ML frameworks, such as Python, MATLAB, and Reality AI Tools®, provided a clear perspective on their performance, integration challenges, and operational efficiencies in a highly constrained automotive environment.

The findings of this study underscore the importance of selecting ML tools that not only offer high computational efficiency and low memory usage but also adapt seamlessly to the real-time processing needs of automotive systems.

Moreover, the study confirmed the feasibility of integrating advanced ML models into the automotive ECUs without compromising the system's performance or safety. This integration is facilitated by tools like MATLAB's Embedded Coder, which allows for efficient translation of ML algorithms into deployable C code, ensuring that the models run effectively within the limited resources of embedded systems.

Future research will focus on refining these models to enhance their accuracy and reliability further. Additionally, exploring the integration of these ML models into a broader range of automotive systems will be crucial. This could include advanced predictive maintenance, optimized fuel consumption, and enhanced autonomous driving technologies.

Ultimately, this research not only contributes to the academic field by providing a rigorous evaluation framework for ML tools in automotive applications but also serves as a practical guide for engineers and developers aiming to implement AI technologies in automotive embedded systems. By continuing to explore and optimize these tools, we can further advance the capabilities of intelligent vehicles, paving the way for more innovative and efficient automotive solutions.

References

[1] T. M. Mitchell, "Machine Learning," in *Machine Learning*, McGraw- Hill, 1997.
[2] E. Alpaydin, "Introduction To Machine Learning," in *Introduction To Machine Learning*, MIT Press, 2020.
[3] G. James, D. Witten, T. Hastie, R. Tibshirani and J. Taylor, "An Introduction to Statistical Learning with Applications in R," in *An Introduction to Statistical Learning with Applications in R*, Springer, 2013.
[4] Coursera, "Types of Machine Learning," Coursera, [Online]. Available: https://www.coursera.org/articles/types-of-machine-learning.
[5] I. Goodfellow, Y. Bengio and A. Courville, "Deep Learning," in *Deep Learning*, MIT Press, 2016.
[6] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," in *Reinforcement Learning: An Introduction*, MIT Press, 2018.
[7] M. Jordan and T. Mitchell, "Machine learning: Trends, perspectives, and prospects.," *Science*, pp. 255-260, 2015.
[8] M. Bojarski, D. D. Testa, D. Dworkakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao and K. Zieba, "End to End Learning for Self-Driving Cars," *arXinL1604.07316*, 2016.
[9] S. Grigorescu, B. Trasnea, T. Cocias and G. Macesanu, "A survey of deep learning tehniques for autonomous driving," *Journal of Field Robotics*, pp. 362-386, 2019.
[10] C. BasuMallick, "What Is ADAS (Advanced Driver Assistance Systems)? Meaning, Working, Types, Importance, and Applications," Spiceworks, [Online]. Available: https://www.spiceworks.com/tech/iot/articles/what-is-adas/.
[11] W. Kempton and J. Tomic, "Vehicle-to-grid power fundamentals: Calculating capacity," *Journal of Power Sources*, vol. 1, no. 144, pp. 268-279, 2005.
[12] R. Roy, "Best Python libraries for Machine Learning," Geeks for Geeks, [Online]. Available: https://www.geeksforgeeks.org/best- python-

libraries-for-machine-learning/.
[13] The Mathworks Inc., "MATLAB for Machine Learning," The Mathworks Inc., [Online]. Available: https://www.mathworks.com/solutions/machine-learning.html?gclid=Cj0KCQjwiMmwBhDmARIsABeQ7xTsr92-k1QfQbgCkyCDpo5v8dQFRiGggqrC1E3QofREFAuJCV4AegcaAmkhEALw_wcB&ef_id=Cj0KCQjwiMmwBhDmARIsABeQ7xTsr92-k1QfQbgCkyCDpo5v8dQFRiGggqrC1E3QofREFAuJCV4AegcaAmkh EALw_wcB.
[14] Reality AI, "Reality AI Software for Real Time Analytics on MCUs & MPUs," Reality AI, [Online]. Available: https://www.renesas.com/us/en/products/microcontrollers- microprocessors/reality-ai.
[15] C3.AI, "C3 AI Platform," C3 AI, [Online]. Available: https://c3.ai/c3- ai-platform/.
[16] Palantir, "Palantir," Palantir, [Online]. Available: https://www.palantir.com/.
[17] T. Freudling, "Detecting Oversteering in BMW Automobiles with Machine Learning," BME Group, The Mathworks, Inc., 2018. [Online]. Available: https://www.mathworks.com/company/technical-articles/detecting-oversteering-in-bmw-automobiles-with-machine-learning.html?s_v1=26004&elqem=2639000_EM_WW_19- 01_NEWSLETTER_CG-DIGEST-AUTO&elqTrackId=5f8efd814ec94f15a064d0caef804b83&elq=35076 6a036e64caf9d5fb1.
[18] Reality AI, "Reality AI Tools," [Online]. Available: https://www.renesas.com/us/en/products/microcontrollers- microprocessors/reality-ai/reality-ai-tools.