5 Supplementary Information

We describe the details of the FEAD system in this section. We start with our data acquisition system, which is based on a commercial L2 autonomous truck. We then formally define the objective for fuel optimization. It is followed by a complete dissection of our FEAD system, with detailed design motivation, exact formulation, and thorough analysis.

5.1 Data Acquisition

As a data-driven approach, it is crucial to obtain an abundance of real driving data. We utilize a fleet of commercial Level 2 (L2) autonomous trucks for this task. The truck tractors are manufactured by well-established commercial vehicle original equipment manufacturers (OEMs) within China. The sensor suite of these tractors comprises three LiDAR units, seven cameras, five mmWave radars, and a GPS unit (Fig. 5). A forward-facing long-range LiDAR is mounted above the front windshield, covering up to 200 meters with a 120-degree field of view. Three cameras with short-, medium-, and long-range capabilities are installed at the top of the windshield, while four additional cameras—including forward middle-range and rearward fisheye units—are mounted above the truck doors. Five long-range mmWave radars are situated on the bumper, providing forward, left-forward, right-forward, left-rear, and right-rear views, with GPS installed on the roof. An autonomous driving control unit is also installed to provide the computing and data storage capability.

The trucks operate in either AD or MD mode. AD mode can only be activated on highways, providing features such as autonomous navigation, lane keeping, automatic lane change, and cruise control. As a L2 system, a driver remains onboard at all times to intervene when necessary.

We collected data in two ways: first, by recording full driving data, including raw sensor data, at 10 Hz; and second, by recording sub-sampled data at 1 Hz. There are two types of full driving datasets. The first dataset, Large-Scale-Detailed (LSD), was collected on nationwide routes across China from March to December 2023. This data was used offline to train the NTM and NPM models. The second dataset, Laiwu-Detailed (LWD), was collected along a controlled test route between Laiwu City and Jinan City, in Shandong Province, China. This route was designed to evaluate the trained models' performance in controlled, open-road scenarios, enabling a comprehensive comparison of different baselines. The route contains no traffic signals, maintains a speed limit of at least 100 km/h, and features varied terrain, including flat sections, uphill climbs, and downhill slopes, as illustrated in the first subplot of Fig. 4b. We recorded multiple test runs with full data recording on this route to create the LWD dataset. The Large-Scale-Concise (LSC) dataset was used for a large-scale field deployment test. This test involved more than 800 heavy-duty trucks and over 1,000 professional drivers over eight months (January–August 2024), accumulating over 50.17 million kilometers of operational data. Similar to the LSD dataset, LSC was collected on nationwide routes, but during the different period of January to August 2024. In addition, it was collected online via Wi-Fi and *sub-sampled* to reduce transmission costs. Table 1 summarizes the data collected to train and evaluate the effectiveness of our approach.

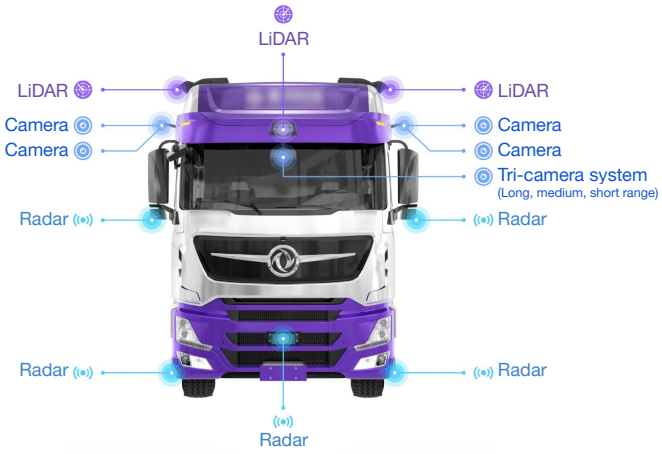

Fig. 5: Sensor suite configuration on a representative test truck. The system includes three LiDAR units (one long-range forward-facing), seven cameras (short-, medium-, and long-range), five mmWave radars (forward and rear coverage), and a roof-mounted GPS unit, providing comprehensive perception capabilities for autonomous highway driving.

5.2 Definition of Fuel Optimization

To enhance economic efficiency and reduce carbon dioxide emissions, lowering the fuel consumption of trucks is essential. However, for freight trucks, fuel savings must be achieved without compromising transportation timeliness. In practical applications, optimizing fuel consumption requires addressing both time constraints and boundary conditions. Firstly, due to spatiotemporal limitations in accessing geographic information, as well as hardware memory and computation time constraints, the optimization is restricted to a road segment with a fixed future distance. Furthermore, the analysis is conducted under highway driving conditions, where vehicle speeds are constrained by regulatory limits, and the relationship between speed and acceleration follows Newton's second law of motion. Additionally, an auxiliary constraint is introduced to penalize excessive braking, ensuring the optimization favors smooth and fuel-efficient driving behavior.

The optimization problem is formulated as follows:

$$
\begin{aligned}
\min_{\{B_i,P_i\}_{i=1}^n} \quad & \sum_{i=1}^{n} f_F(P_i)\,\Delta s + \beta B_i^2 \\
\text{s.t.} \quad & v_{\min}(s_i) \le v_i \le v_{\max}(s_i), \quad i = 0,\dots,n, \\
& \frac{1}{n}\sum_{i=1}^{n} v_i \ge v_{\text{trg}}, \\
& v_{i+1}^2 = v_i^2 + 2a_i\Delta s, \quad i = 0,\dots,n-1, \\
& a_i = f_{\text{D}}(B_i, P_i)
\end{aligned}
\tag{1}
$$

where $\mathbf{P} = \{P_i\}_{i=1}^n$ denotes the engine power sequence, $\mathbf{B} = \{B_i\}_{i=1}^n$ denotes the brake sequence, and both can be

Name	Large-Scale-Detailed (LSD)	Laiwu-Detailed (LWD)	Large-Scale-Concise (LSC)
Route	Nationwide, China	Laiwu Jinan, Shandong, China	Nationwide, China
Time Period	Mar. to Dec. 2023	Feb. to Jun. 2024	Jan. to Aug. 2024
Collection Frequency	10Hz	10Hz	1Hz
Collection Method	Disk copy	Disk copy	Online upload via 4G
Data Volume	~1.01 GB/s	~1.01 GB/s	~0.36 KB/s
Mileage	0.17 million km	16,000 km	50.20 million km
Usage	Model training	Evaluation	Large-scale evaluation

Table 1: Overview of the three datasets used in this study: Large-Scale-Detailed (LSD), Laiwu-Detailed (LWD), and Large-Scale-Concise (LSC), including their geographic scope, collection period, frequency, method, volume, mileage, and intended usage.

treated as control commands. $S = \sum_{i=1}^{n} \Delta s$ represents the optimization horizon in distance, which is set to $S = 5000$m in this study. The function $f_{\mathrm{F}}(P_i)$ represents the fuel consumption model, typically obtained from the manufacturer in conventional approaches. The coefficient β serves as a braking penalty factor, encouraging the minimization of braking. The lower and upper speed bounds, $v_{\min}(s_i)$ and $v_{\max}(s_i)$, are determined by road speed limits, traffic conditions, and vehicle constraints, with s_i denoting the position corresponding to v_i. The parameter v_{trg} corresponds to the target average speed, determined by the global speed planning method [39]. The vehicle speed sequence $\mathbf{v} = \{v_i\}_{i=1}^{n}$ is governed by the acceleration sequence $\mathbf{a} = \{a_i\}_{i=1}^{n}$, which is calculated by the dynamics model $f_{\mathrm{D}}(\mathbf{B}, \mathbf{P})$.

Conventional methods describe vehicle dynamics as $\mathbf{a} = f_{\mathrm{D}}(\mathbf{B}, \mathbf{P}; m, c_{\mathrm{r}}, c_{\mathrm{d}})$, where m is the vehicle mass, c_r is the rolling resistance coefficient, and c_d is the aerodynamic drag coefficient. In contrast, the FEAD system employs NTM as both the dynamics and fuel consumption model, jointly formulated as $[\mathbf{a}, \mathbf{F}] = \mathrm{NTM}(\mathbf{B}, \mathbf{P})$, where $\mathbf{F}$ denotes the fuel consumption sequence. From a fuel-efficiency standpoint, braking and propulsion do not occur simultaneously. To simplify the formulation, we define a unified control variable, termed the desired power $\tilde{\mathbf{P}}$, which represents both braking and propulsion operations:

$$\tilde{P}_i = \begin{cases} -B_i, & \text{if } B_i > 0 \quad \text{(braking)}, \\ P_i, & \text{if } B_i = 0 \quad \text{(propulsion)}. \end{cases} \tag{2}$$

Accordingly, in conventional methods, the models can be expressed as $\mathbf{F} = f_F(\mathbf{P})$ and $\mathbf{a} = f_D(\tilde{\mathbf{P}})$, whereas in FEAD, both are integrated as $[\mathbf{a}, \mathbf{F}] = \mathrm{NTM}(\tilde{\mathbf{P}})$. Rather than solving this optimization directly, FEAD employs a reinforcement learning framework (NPM) that learns an optimal policy by interacting with NTM in an iterative trial-and-error process.

5.3 FEAD system

Regarding the problem described in Equation 1, autonomous driving techniques can comprehensively benefit highway trucks in terms of fuel saving. We summarize the impacts of autonomous driving on fuel saving according to the effective scenarios.

5.3.1 Fuel-Efficient Full-Trip Speed Planning

Fuel-efficient full-trip speed planning relies on an operational speed strategy to determine target velocities, aiming to enhance fuel conservation for autonomous vehicles while meeting strict time constraints. The fuel consumption of an operational truck is closely linked to its cruising speed, while variations in speed profiles can also influence fuel efficiency, even when maintaining the same average speed on identical road conditions. Therefore, it is crucial to minimize the average speed within the allowable time constraints while optimizing the speed profile for maximum fuel efficiency, incorporating environmental factors such as terrain and traffic conditions. The above challenge can be formulated as an optimization problem [39], where the primary objective is to minimize fuel consumption and travel time by refining target speed profiles. To solve this problem, we adopt the Dynamic Programming (DP) approach as outlined in [40], enabling the determination of an optimal speed profile over the entire trip. By systematically evaluating all possible control sequences, DP minimizes the objective function, yielding an optimal speed trajectory that enhances fuel efficiency.

5.3.2 Fuel-Efficient Local Navigation

Fuel-efficient local traffic interaction is crucial for optimizing fuel consumption in autonomous vehicles and ADAS while maintaining safety and smooth traffic flow. To address local traffic interaction, we employ the behavior planner [43], which is composed of three main processes: guided branching, scenario realization, and evaluation. Guided branching predicts action sequences and the intentions of other participants, which together form traffic scenarios. Scenario realization then uses multi-agent forward simulation to execute these steps in real time. This enables the system to adapt to dynamic traffic conditions. Through this process, the model is enhanced to ensure safer and more efficient driving in complex, noisy traffic environments. Moreover, in car-following scenarios, it is important to manage the following distance. A large gap may lead to frequent cut-ins, while a smaller gap increases the risk of rear-end collisions and driver discomfort, often prompting excessive braking [41]. To mitigate these issues, we incorporate a cost related to the relaxation zone during the evaluation phase, ensuring the vehicle maintains an optimal following distance for fuel efficiency. Furthermore, lane changing, a fundamental

but complex driving maneuver, requires coordinated control of both longitudinal and lateral vehicle movements [42]. This maneuver allows the vehicle to adjust its driving conditions, optimizing speed and space satisfaction. To further enhance fuel efficiency, we introduce a braking-related cost term during the evaluation phase, encouraging decisions that minimize braking.

5.3.3 Fuel-Efficient Cruise Control

Fuel-efficient cruise control optimize speed profiles and regulate throttle and brake to minimize fuel consumption, while ensuring performance and meeting delivery time constraints. In this section, we introduce the traditional Predictive Cruise Control (PCC) [17, 33–35] and our proposed end-to-end control framework.

PCC enhances fuel efficiency by leveraging vehicle sensors and map data to predict road conditions, such as slopes and traffic signals, and adjust speed accordingly. This reduces unnecessary acceleration and braking, thereby optimizing fuel consumption. Traditional PCC methods involve three key components: fuel modeling, dynamics modeling, and policy formulation. Fuel modeling is used to construct the objective function, dynamics modeling generates acceleration and speed with the control commands, and policy formulation is responsible for deriving optimized speed profiles. While this approach simplifies computation, it often compromises robustness and accuracy by neglecting complex interdependencies within the system.

To tackle fuel efficiency challenges in commercial vehicles, we propose a modular end-to-end Neural Cruise Control (NCC) framework for optimization, consisting of two key modules: the Neural Truck Module (NTM) and the Neural Policy Module (NPM). NTM integrates offline training on historical engine-specific data with online refinement using real-time data (power, speed, fuel consumption) across varying road conditions. This dual-stage approach captures both persistent and transient dynamics and fuel consumption mapping. NPM replaces traditional control methods with reinforcement learning (RL), leveraging NTM's data-driven simulations to optimize operational task efficiency. During deployment, task-specific fuel models and dynamic representations enable adaptive policy tuning via pre-trained networks. The framework achieves sustained fuel savings and operational efficiency in dynamic environments through its integrated, scenario-generalizable design.

5.4 Neural Truck Module

NTM represents a paradigm shift in fuel-efficient autonomous driving by introducing a unified neural network architecture that implicitly and jointly models both the objective function (fuel consumption) and constraint conditions (vehicle dynamics). Unlike conventional approaches that rely on simplified, piecemeal approximations, NTM leverages high-dimensional feature representations to capture the complex, nonlinear interactions between a truck's fuel economy and its dynamic behavior, as formalized in Equation 1. This holistic design enables unprecedented accuracy and adaptability in real-world operating conditions.

Traditional fuel-saving control systems, grounded in optimization techniques, often resort to polynomial approximations of the fuel consumption model to render the problem computationally tractable. However, our extensive experimental results reveal that fuel consumption characteristics exhibit significant variability across operating conditions (e.g., engine load, terrain) and vehicle wear states, necessitating high-dimensional, nonlinear modeling. While this introduces non-convexity that challenges direct optimization, NTM's neural network-based framework seamlessly handles these complexities, overcoming a critical limitation of existing methods.

Furthermore, truck dynamics are influenced not only by intrinsic vehicle properties but also by a multitude of extrinsic factors—including trailer configurations, ambient wind speed and direction, and road surface adhesion—which are often ignored or oversimplified in physics-based models. NTM's data-driven approach inherently captures these interdependencies, providing a more robust and realistic representation of operational complexity. By integrating real-world sensor data, NTM dynamically adapts to varying conditions, ensuring consistent performance where conventional methods fail.

5.4.1 Design Principles of Neural Truck Module

The fuel-efficient strategies employed by skilled human drivers are inherently truck-centric, relying on an intuitive understanding of vehicle dynamics and environmental conditions. By analyzing these practices, we derive three core principles that form the foundation of NTM. Unlike traditional model-based approaches, NTM leverages neural networks to emulate and enhance human-like reasoning, enabling unprecedented accuracy and adaptability in fuel optimization.

- **End-to-end experiential modeling.** Expert drivers develop an implicit mapping between throttle input, vehicle speed, and fuel consumption through continuous experience. This end-to-end cognitive model allows them to anticipate fuel usage across varying road conditions without explicit knowledge of parameters like cargo weight. NTM replicates this capability through its PredictionBlock, which directly maps control inputs—engine power (P) and braking power (B)—to real-time acceleration (a) and fuel consumption (F). By eliminating intermediate simplifications (e.g., polynomial approximations), NTM captures nonlinear dynamics often overlooked by conventional optimization methods, resulting in a more accurate and efficient fuel model.
- **Truck-wise model acquisition.** Human drivers rapidly adapt to individual vehicle characteristics through iterative learning. Similarly, NTM incorporates an Eigen Block—a 64-dimensional feature encoder—trained on large-scale historical data to capture intrinsic vehicle traits such as engine response curves and chassis dynamics. This truck-wise encoding ensures predictions are tailored to specific vehicle platforms, avoiding the inaccuracies of generic models. The Eigen Block can be ensembled across different truck models, facilitating seamless knowledge transfer and scalability.

- **Task-level dynamic adaptation.** During each trip, expert drivers continuously recalibrate their strategies based on changing conditions (e.g., cargo load, weather). NTM emulates this through a Primitives Sample Block, which extracts representative in-trip signals (e.g., power-to-acceleration ratios) to dynamically adjust predictions. This real-time calibration allows NTM to adapt to extrinsic variations such as wind resistance or road friction, maintaining optimal efficiency under diverse operating conditions.

NTM's design principles not only mirror human expertise but also transcend its limitations by leveraging data-driven neural networks. This innovative approach enables robust, scalable, and highly accurate fuel modeling—critical for achieving sustainable autonomy in heavy-duty trucking. By integrating end-to-end learning, truck-specific encoding, and dynamic adaptation, NTM sets a new standard for fuel-efficient autonomous systems, offering significant improvements over traditional methods in both precision and practicality.

5.4.2 Pipeline of Neural Truck Module

NTM introduces a transformative approach to modeling truck dynamics and fuel consumption by seamlessly integrating intrinsic engine characteristics with external environmental influences in real-time. Unlike conventional vehicle dynamics identification methods that rely on solving differential equations from limited recent observations, NTM employs an innovative neural formulation to reconstruct time-varying vehicle behavior. Crucially, while traditional methods become unscalable due to requiring dedicated models for each vehicle or operational moment, NTM addresses this limitation through a unified feed-forward design that leverages a small number of representative data fragments, termed primitives, as reference inputs. Similar to how an expert driver infers vehicle load or wind resistance from minimal throttle events, these dynamically updated primitives compactly encode both vehicle-specific properties and environmental conditions. When conditioned on these primitives, a single NTM model can accurately predict acceleration and fuel consumption for any given engine and braking power command in a single forward pass, eliminating the need for vehicle-specific retraining while maintaining high precision.

During inference (Fig. 6 bottom), NTM executes an efficient three-stage process that begins with the Primitive Sample Block (PSB) selecting and clustering these representative primitives. These primitives are subsequently transformed by the Eigen Block into a low-dimensional eigen feature embedding, which captures essential vehicle-environment interactions. The final PredictionBlock then processes this embedding to generate precise acceleration and fuel consumption predictions, with the entire pipeline operating without any vehicle-specific or trip-specific fine-tuning—enabling true plug-and-play deployment across diverse truck platforms.

The training methodology (Fig. 6 top) employs a sophisticated two-stage approach that ensures both generalizability and platform-specific accuracy. The initial stage leverages primitive-based data structures from multiple vehicles to train the PSB's classification model in an unsupervised manner, while the second stage builds upon this foundation through supervised learning to train the Eigen Block and PredictionBlock. This hybrid approach enables the PredictionBlock to remain fully vehicle-agnostic, while the Eigen Block incorporates platform-specific dynamics through ensemble models trained on individual truck platforms with varying engines and powertrains. This innovative training paradigm allows NTM to capture the nuanced characteristics of different vehicle types while maintaining the practical scalability essential for real-world deployment across large, heterogeneous fleets.

5.4.3 Primitives Sample Block

The Primitives Sample Block (PSB) represents a foundational innovation within our NTM architecture, designed to select a minimal yet sufficient set of primitives capable of comprehensively capturing variations in vehicle dynamics and fuel consumption models arising from both engine characteristics and external environmental factors. From a full lifecycle perspective, a truck's dynamic and fuel-consumption behavior can be conceptually modeled as a superposition of five distinct submodels: the **base model** (determined by the vehicle's design specifications at production, shared across platforms with identical engines and powertrains), **instance models** (accounting for subtle manufacturing variations among vehicles of the same platform), **aging models** (capturing long-term drifts due to wear and component degradation, with discrete changes following maintenance), **task-related models** (addressing variations from transport-specific factors like payload or trailer type that remain constant within individual trips), and **environment-related models** (handling temporary variations caused by external conditions such as road surface, wind, or precipitation). While the base model is implicitly encoded within the Eigen Block, all other variations are dynamically represented through replaceable primitives in the PSB. Remarkably, by leveraging the expressive power of neural networks, we demonstrate that five minute-level in-trip primitives are sufficient to capture the majority of model variations across diverse temporal scales, representing a significant advancement in computational efficiency. Formally, we define a primitive as a feature extracted from a 2 km trajectory. This trajectory is partitioned into $N_{\text{sp}} = 40$ consecutive segments of equal length (50 m each). For each segment i, we extract a feature vector s_i that encapsulates the dynamics over that interval: $s_i = \{\left[d_j^{\text{st}}\ d_j^{\text{et}}\ d_j\right] \mid d_j \coloneqq \{\theta, v, a, \tilde{P}, F\}\}$, where d^{st}, d^{et} and d_j denote the key parameters (slope θ, speed v, acceleration a, desired power $\tilde{P}$, and fuel consumption F) for the start of the segment, the end of the segment, and the average over the segment. This sophisticated primitive formulation enables the PSB to maintain an optimal balance between representational completeness and computational efficiency, establishing a new standard for adaptive modeling in autonomous vehicle systems.

Primitives sample strategy

During vehicle operation, SPs are continuously accumulated, creating a dynamic repository of operational data. However, processing all accumulated SPs through the Eigen Block (EB) is computationally infeasible, necessitating an intelligent selection mechanism that limits quantity while maximizing

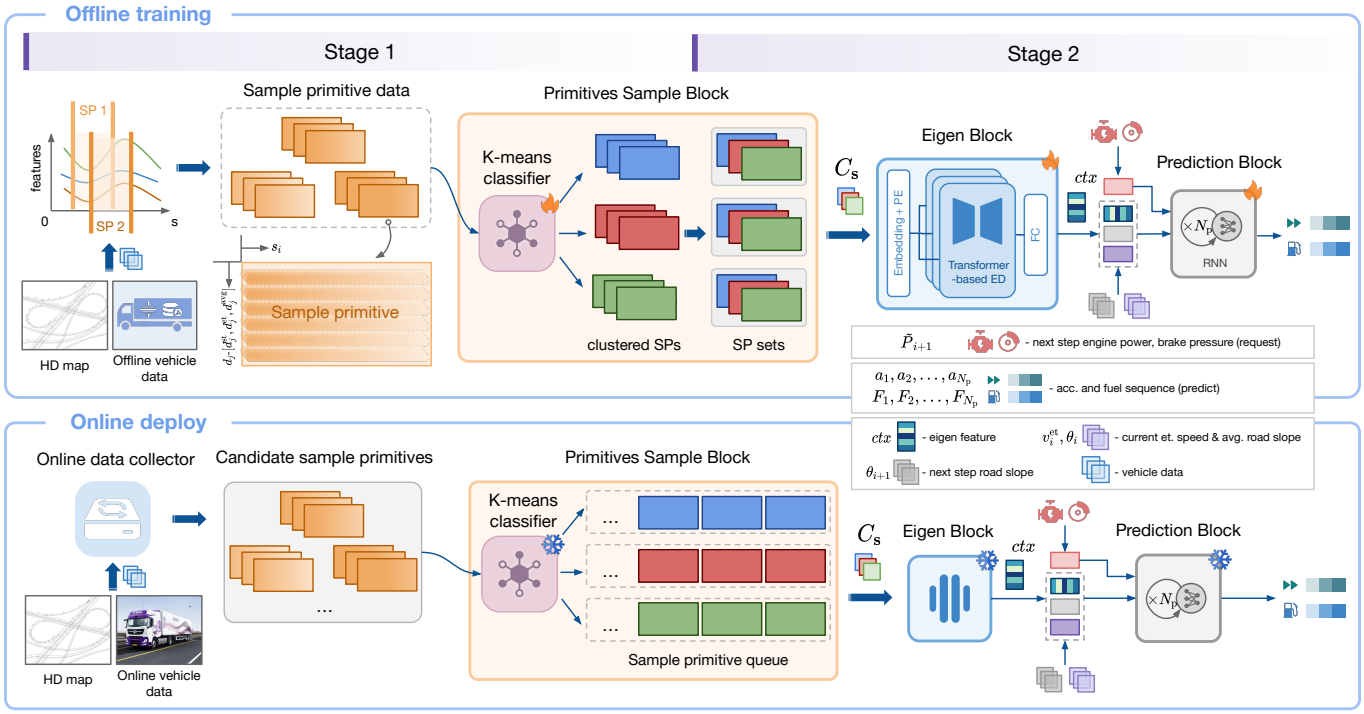

Fig. 6: Pipeline of the Neural Truck Module (NTM). The top panel illustrates the offline training phase, where vehicle-specific operational data and large-scale historical datasets are used to train the NTM. The bottom panel depicts the online deployment phase, in which the trained NTM adapts to real-time task and environmental variations to generate fuel-efficient driving policies.

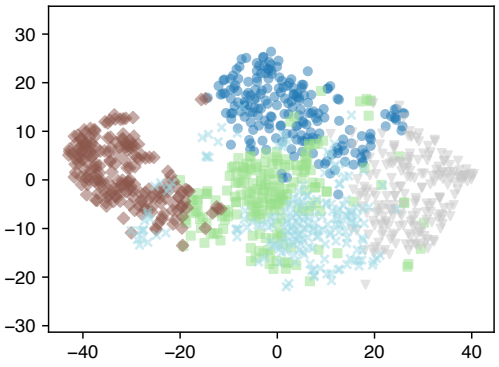

Fig. 7: t-SNE visualization of K-Means clustering (K=5) applied to 1,000 Sample Primitives (SPs). Each cluster is represented by a distinct color and marker type, demonstrating clear separation and validating the clustering strategy for selecting diverse SP representatives in the Primitive Sample Block (PSB) module.

informational diversity. The primary objective of the PSB is therefore to generate a maximally diverse SP set C_S that enables downstream EB processing to extract comprehensive vehicle intrinsic features across the full spectrum of operating conditions. To achieve this diversification, we implement a sophisticated clustering approach where each SP is assigned a category label using the K-Means algorithm (with $K = 5$ clusters), allowing the PSB module to select one representative SP from each category. This strategic selection ensures that the resulting SP set encompasses the full range of vehicle and environmental conditions without redundant overlap. Validation through t-SNE dimensionality reduction visualization (Fig. 7) confirmed that our K-Means classifier effectively categorizes SPs with clear inter-cluster separation across 1,000 samples.

To validate the effectiveness of our proposed selection strategy, we developed and evaluated three distinct SP selection strategies to optimize this process: Diverse-Close (DC), which selects SPs from each cluster based on their minimum distance to the cluster centroid; Recently (R), which simply selects the five most recent SPs without clustering; and Diverse-Recently (DR), which selects the most recent SP from each cluster after K-means categorization. Fig. 8 (top three rows) presents the visualized eigen features and the corresponding velocity and fuel consumption errors obtained from different SP selection strategies. All three experiments were derived from data collected by the same vehicle along an identical route, with different SP combinations producing distinct speed and fuel consumption profiles. Comparison of the first and third columns reveals a clear association between the fluctuations in eigen features and the selected SPs. It can be observed that selecting SPs using the DR strategy results in the smallest deviations in speed and fuel consumption from the ground truth, providing a more accurate representation of the vehicle characteristics. During the real-world operation of NTM, the DR strategy was employed, with a queue maintained for each cluster to facilitate SP selection.

Crucially, the PSB module incorporates a redundancy check that discards SPs exhibiting excessive overlap with current samples of the same label, ensuring minimal duplication while maintaining category coverage. During training,

we further enhance robustness through a one-to-many data augmentation strategy where the PSB module generates multiple distinct SP sets through repeated selection, enabling the model to learn consistent intrinsic features across varied SP combinations. The resulting NTM sample structure integrates a normalized current state, optimally diverse SP set, future engine power, and slope data as inputs, with corresponding motion states and fuel consumption values serving as ground truth—creating a comprehensive foundation for accurate vehicle dynamics modeling.

5.4.4 Eigen Block

The Eigen Block (EB) represents a critical innovation within our neural architecture, designed to address the simultaneous processing of historical vehicle data and real-time operational information. We have developed a sophisticated two-stage neural network model comprising the Eigen Block and Prediction Block to achieve this integration. During the training phase, the EB is engineered to discern and learn complex mapping relationships that capture both intrinsic vehicle characteristics and extrinsic environmental factors from the dataset. This capability enables the EB during inference to parse and extract essential eigen features from the Sample Primitives (SPs) that represent the vehicle's current operational state. Our implementation leverages a Transformer-based encoder-decoder architecture for the EB, specifically chosen for its exceptional ability to process sequential data through multi-head attention mechanisms. The multi-head self-attention (MHSA) and multi-head cross-attention (MHCA) components work in concert to effectively distill the intrinsic vehicle characteristics embedded within the SP set. This process can be formally represented as

$$ctx = f_{\mathrm{EB}}(\mathrm{MHCA}(\mathcal{Z}, \mathrm{MHSA}(C_{\mathcal{S}}))), \tag{3}$$

where $\mathcal{Z}$ denotes learnable queries of the transformer model, $C_{\mathcal{S}}$ represents the SP set, and the resulting eigen context $ctx \in \mathbb{R}^{N_c}$ constitutes a one-dimensional feature vector that comprehensively encodes the vehicle's intrinsic properties. This innovative approach enables our system to generate a rich, contextualized representation that captures both persistent vehicle characteristics and transient operational conditions, providing a robust foundation for accurate prediction and control decisions within the broader autonomous driving system.

Validation test of Eigen Block

Within the NTM framework, we hypothesize that the eigen features generated by the EB effectively capture both the intrinsic characteristics of the vehicle and external environmental influences. To validate this premise and assess the efficacy of the EB, we conducted a comprehensive analysis using the large-scale truck dataset, employing the trained EB in conjunction with various SP selection strategies within the PSB module. The eigen features extracted from each sample were subsequently visualized to enable clear interpretation and detailed analysis, as illustrated in Fig. 8. This experiment systematically evaluates the impact of diverse SP selection strategies, vehicle types, and route configurations on the performance of the EB, ensuring robustness across operational scenarios. As the vehicle progresses along its trajectory, the eigen features dynamically evolve; to discern overarching patterns, we computed the distance-based mean of these features for each trip. The resulting profiles exhibit consistent trends across these randomly selected trips, with a prominent peak observed at index 36, while variations in specific dimensions reflect trip-specific distinctions in eigen feature distributions, as depicted in Fig. 11. For this test, the SP overlap threshold was set to zero, meaning that for each sample, new SPs of a given class replace existing ones in the construction of $C_{\mathcal{S}}$; all eigen feature values were min-max normalized to a [0, 1] range for consistent visualization. As shown in Fig. 9, eigen feature values correlate strongly with vehicle weight, while sensitivity analyses in Fig. 10 confirm the stability of eigen context values, indicating the EB's ability to reliably encode input SPs and map vehicle characteristics to specific output contexts. Although the eigen context maintains overall uniformity throughout trips, observed variations are attributable to fluctuating external environmental conditions. This consistent and interpretable eigen context implicitly encapsulates critical vehicle parameters and historical data, thereby empowering the prediction module to generate accurate, context-aware inferences and forecasts, underscoring the novelty and practical impact of our approach for real-world autonomous driving applications.

5.4.5 Prediction Block

The Prediction Block (PB) serves as the critical forecasting component within our architecture, designed to infer the vehicle's future state across the next $N_{\mathrm{p}} = 100$ steps (equivalent to 5 km) by holistically processing historical, current, and anticipated trajectory information. To achieve this, we implement a Recurrent Neural Network (RNN) architecture that simultaneously models both fuel consumption and dynamic vehicle behavior, with each cell in the network corresponding to a discrete 50-meter distance interval. This design choice is strategically motivated by the RNN's lightweight computational footprint and superior efficiency in processing sequential data, making it exceptionally suitable for deployment on resource-constrained vehicle-mounted chips where operational efficiency is paramount. The core RNN cell is formally modeled as:

$$\left[a_{i+1}, F_{i+1}\right] = f_{\mathrm{PB}}(\tilde{P}_{i+1}, ctx, v_i^{\mathrm{et}}, \theta_i, \theta_{i+1}),\ i \in [0, N_{\mathrm{p}} - 1] \tag{4}$$

where F denotes the fuel consumption over a 50-meter interval, a represents average acceleration, θ is the road slope, $\tilde{P}$ the desired power, and v^{et} the segment's end speed. A key innovation in our approach is the incorporation of a physical constraint to maintain realism in velocity predictions: if the computed acceleration yields a negative velocity, we enforce a feasibility condition via:

$$v_{i+1}^{\mathrm{et}} = \sqrt{\max\left(0, 2a_{i+1}\Delta s + (v_i^{\mathrm{et}})^2\right)}, \tag{5}$$

where $\Delta s = 50$ meters, thereby eliminating non-physical outcomes and enhancing predictive robustness. This integrated design enables NTM to generate comprehensive future state projections, which are subsequently processed by NPM. The

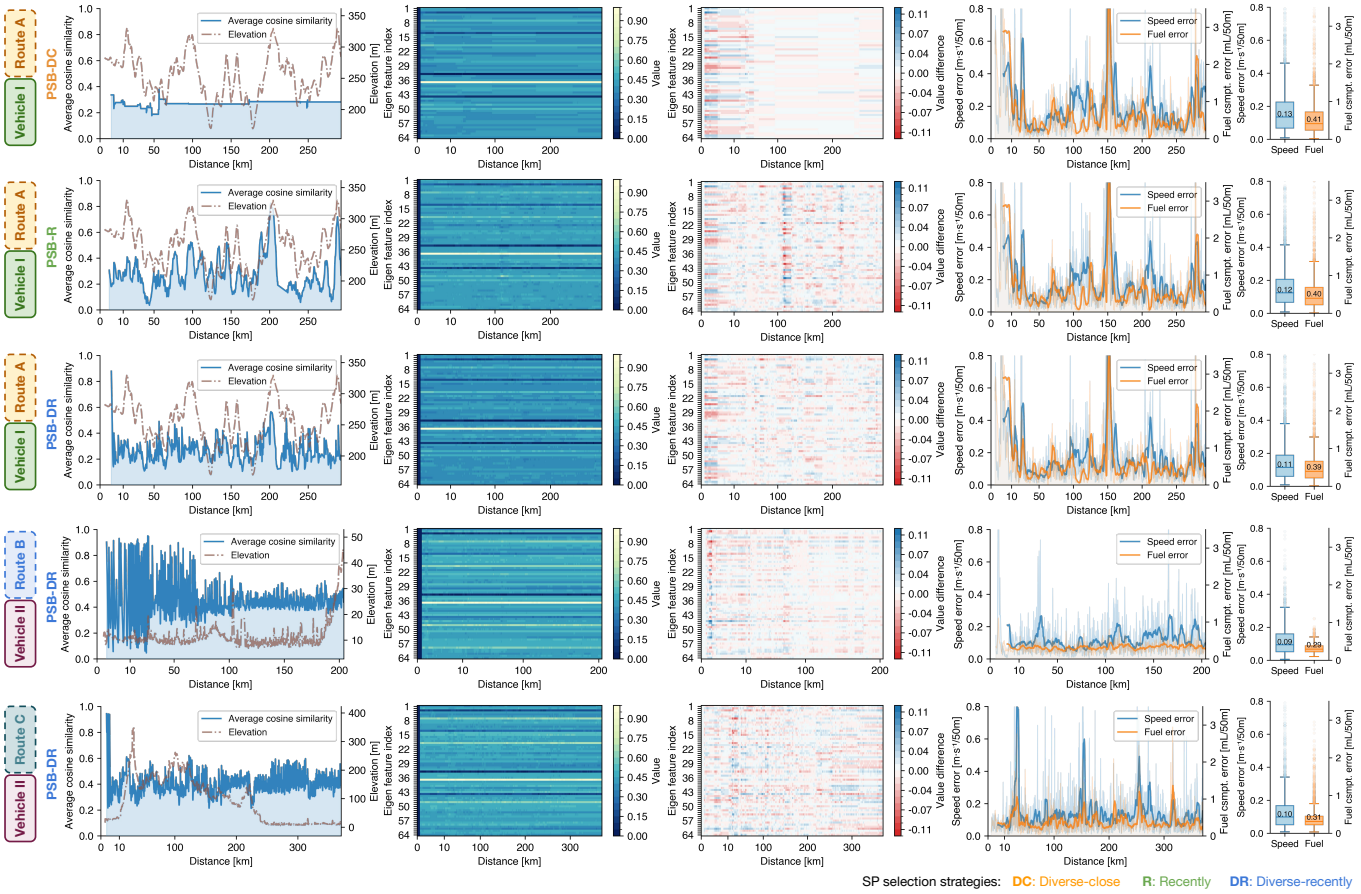

Fig. 8: Detailed analysis of input and output data of Eigen Block under three SP selection strategies - DC: Diverse-close, R: Recently, and DR: Diverse-recently. From left to right: (1) elevation profile per trip and average cosine similarity among SPs in the SP set at each distance point; (2) eigen feature values across the full trip; (3) deviation of eigen features from their mean value (per feature dimension); (4) prediction errors for speed and fuel consumption per sample over the trip; (5) statistical summary of prediction errors for speed and fuel consumption. In (1)-(4), the first 10 km of the x-axis are magnified to highlight early-stage dynamics. Feature values in (2) and (3) are normalized to $[0, 1]$. Numerical labels in (5) indicate mean values.

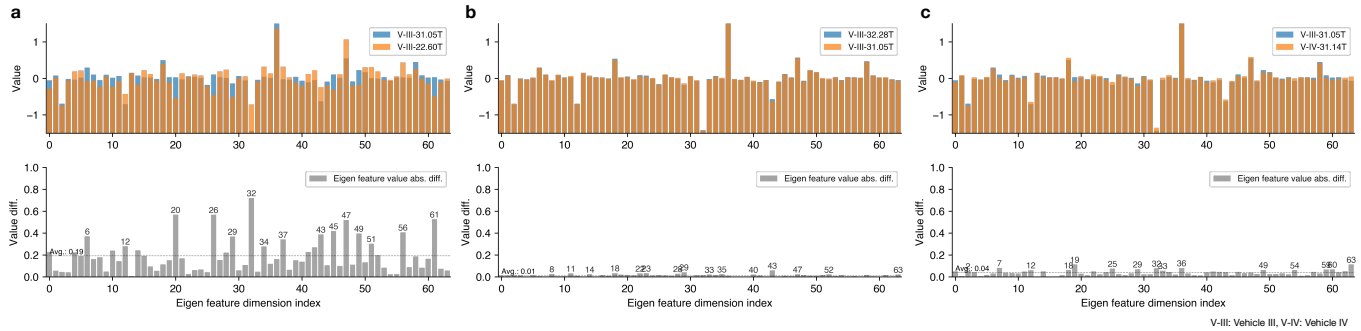

Fig. 9: Relationship between vehicle weight and eigen feature values. The upper part of each subplot displays eigen feature values averaged across all trips, while the lower part shows the corresponding absolute differences. Data are derived from real-world operational routes with consistent paths and similar dates. **a**, same vehicle (Vehicle III) under large weights variation. **b**, same vehicle (Vehicle III), under small weights variation. **c**, different vehicles (Vehicle III and Vehicle IV) with similar weights.

two-stage network architecture – featuring a complex, high-parameter EB for feature extraction and a simpler, efficient PB for forecasting – not only provides strong interpretability and modularity but also allows for deployment at varying computational frequencies, optimizing the trade-off between accuracy and real-time performance. This novel partitioning ensures that our system achieves state-of-the-art predictive capability while maintaining the practicality required for in-vehicle implementation.

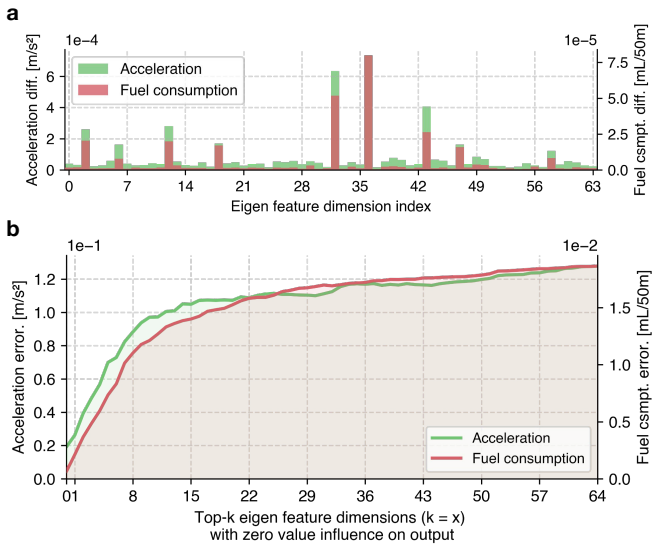

Fig. 10: Sensitivity analysis of eigen features, with analysis performed on the real-world operational routes dataset. **a**, Absolute changes in predicted acceleration and fuel consumption resulting after increasing each eigen feature dimension by 5%. **b**, Absolute deviations from ground truth after sequentially setting the top-k most sensitive eigen features (identified in **a**) to zero.

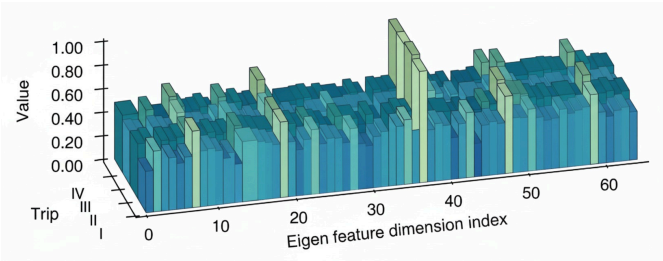

Fig. 11: Eigen context spatial profiles. Spatial averages of the 64-dimensional eigen features are shown for four randomly selected trips from the large-scale operational truck dataset. Each value represents a distance-based average of the corresponding feature dimension, illustrating variation across journeys.

5.4.6 Training Details

EB and PB are trained using the Large-Scale-Detailed dataset, from which a total of 1.77 million samples are extracted, each containing future fuel consumption and speed values as supervisory ground truth. Both networks are optimized using the mean squared error (MSE) loss function. The models are implemented in PyTorch and trained with the Adam optimizer, incorporating a weight decay of 0.08 and a multi-step learning rate scheduler initialized at 10^{-4}. Training proceeds for 100 epochs with a batch size of 128, taking approximately 34 hours to complete on a high-performance workstation equipped with an Intel i9-13900KF CPU, an NVIDIA RTX 4090 GPU, and 128 GB of RAM.

5.5 Neural Policy Module

NPM optimizes vehicle actions—including engine power, speed, and fuel consumption – to minimize fuel usage under time constraints. Unlike traditional PCC, which is limited to short-distance optimization (typically a few kilometers) and exhibits uncertain long-term performance, our approach uses Reinforcement Learning (RL) to enable long-horizon planning while incorporating long-term rewards. PCC requires multiple optimization iterations often constrained by real-time computational limits, especially when using, leading to suboptimal results, whereas RL facilitates deeper optimization through simulated environments, accelerating adaptation to real-world conditions. Moreover, while NTM offers superior modeling accuracy, its computational complexity challenges real-time PCC optimization; RL efficiently handles these complex models by continuously integrating environmental feedback, enabling adaptive policy refinement. Consequently, we integrate RL into our neural policy framework to achieve a robust and scalable solution for fuel-efficient autonomous driving

5.5.1 Reinforcement policy learning framework

The FEAD system formulates fuel-efficient driving as a reinforcement learning problem, where the autonomous truck functions as an intelligent agent interacting with its environment, a novel approach that significantly advances traditional control methodologies. Within this framework, the agent's decision-making process is governed by our NPM, which generates control actions $\mathbf{u} \in \mathcal{U}$ based on comprehensive observations $\mathbf{o} \in O$ of the vehicle's current state. The environment (ENV) incorporates our NTM to achieve high-fidelity simulation of vehicle dynamics and fuel consumption, creating a realistic training platform that accurately mirrors actual truck behavior under diverse operating conditions. This integration represents a key innovation, as it enables the policy to learn from physically accurate simulations while maintaining computational feasibility for real-time deployment.

A particularly innovative aspect of our framework is the strategic balance between prediction horizon and computational efficiency. Although NTM can predict vehicle behavior over a 5 km horizon, NPM's action horizon is deliberately limited to 500 meters based on three critical considerations that ensure practical implementation. First, accumulated speed estimation errors over longer distances would compromise optimization reliability. Second, longer horizons demand more complex models that conflict with the limited computational resources available in onboard systems. Third, downstream control modules typically manage only 200-300 meter horizons, making longer control sequences unnecessary in practice. This intentional design enables effective optimization across the full 5 km range through reward accumulation while maintaining real-time performance required for deployment.

The NPM policy will be optimized to maximize accumulated rewards over a 5 km horizon, aligning with our objective to enhance fuel efficiency across operationally meaningful distances while respecting practical constraints, as formalized in Equation 1.

Truck control action

An action represents the control decision made by the agent at each step, determined by its current observation and policy. In this work, each action is formulated as a sequence of desired power commands $\tilde{P}$ applied over the 500 m policy horizon. Since the trajectory is discretized into 50 m segments, the action comprises $N_\mathrm{u} = 10$ consecutive execution steps. Specifically, the action at step k, denoted as $\mathbf{u}_k \in \mathbb{R}^{N_\mathrm{u}}$, is defined as the sequence $\mathbf{u}_k = [\tilde{P}_{k+1}, \tilde{P}_{k+2}, \ldots, \tilde{P}_{k+N_\mathrm{u}}]$, where each element corresponds to the desired power command applied over one 50 m segment, collectively covering the full 500 m horizon. In the simulation, a perfect-control setting is assumed, where only one command $\tilde{P}_{k+1}$ is executed at each step.

Observation

An observation represents the integrated perception of the agent's internal state and external environmental information, captured at each step k as a comprehensive vector $\mathbf{o}_k = \{E_k, C_k, I_k, V_k\} \in \mathbb{R}^{N_\mathrm{o}}$ corresponding to the spatial position s_k. The first component, $E_k = ctx \in \mathbb{R}^{d_\mathrm{c}}$, encapsulates latent features extracted from the encoder of NTM, with a fixed dimension $d_\mathrm{c} = 64$ to ensure consistent representation of vehicle dynamics. The second component, $C_k = [\theta_k, v_k^{\mathrm{et}}]$, describes the current vehicle state, where θ_k denotes the average road slope over a 50 m segment and v_k^{et} represents the vehicle's speed at the segment end. The third component, $I_k = \boldsymbol{\theta}_k^f \in \mathbb{R}^{N_\mathrm{f}}$, incorporates future road information derived from high-definition maps, with each element encoding the average slope over 50 m segments across a 5 km horizon ($N_\mathrm{f} = 100$). The final component, $V_k = [v_{\mathrm{trg}}, v_k^{\mathrm{past}}]$, integrates velocity-related data, including the target speed v_{trg} and the historical average speed v_k^{past}. Collectively, these elements form a rich observational vector with a total dimension of $N_\mathrm{o} = 168$, enabling the agent to make informed decisions based on a holistic view of immediate and anticipated conditions.

Environment

The environment defines the world in which the agent operates, including all elements and interaction rules. It receives the agent's action and returns observation of the updated state and reward, formalized as $\mathbf{o}_{k+1}, r(\mathbf{o}_k, \mathbf{u}_k) = \mathrm{ENV}(\mathbf{o}_k, \mathbf{u}_k)$. We use this environment to train NPM by generating training scenarios from recorded data. As shown in Fig. 3, our data-driven NTM accurately simulates vehicle fuel consumption and dynamics, outperforming conventional BSFC maps and system identification methods. Thus, the trained NTM serves as the fuel and dynamics model within the environment. Using Equation 4 with observation $\mathbf{o}_k$ and action $\mathbf{u}_k$, we predict acceleration $\mathbf{a}_k$ and fuel consumption $\mathbf{F}_k$. From the acceleration, we derive v_{k+1}^{past} and v_{k+1}^{et} , while slope information θ_{k+1} and $\theta_{k+1}^{\mathrm{f}}$ is measured from the scenario. These updated parameters are used to construct $\mathbf{o}_{k+1}$. The objective is to minimize fuel consumption while meeting time constraints, adhering to speed limits, and maintaining power stability. Accordingly, the reward function is defined as:

$$r(\mathbf{o}_k, \mathbf{u}_k) = r^{\mathrm{obj}} + \lambda_1 r^{\mathrm{trg}} + \lambda_2 r^{\mathrm{lim}} + \lambda_3 r^{\mathrm{reg}}, \tag{6}$$

where $r^{\mathrm{obj}} = -\sum_{n=1}^{N_\mathrm{u}} F_{k+n} + \beta B_{k+n}$ is the primary objective from Equation 1, $r^{\mathrm{trg}} = -\sum_{n=1}^{N_\mathrm{u}} \max(0, v_{\mathrm{trg}} - v_{k+n}^{\mathrm{et}})$ penalizes speeds below the target v_{trg} , $r^{\mathrm{lim}} = -\sum_{n=1}^{N_\mathrm{u}} [\max(0, v_{\min} - v_{k+n}^{\mathrm{et}}) + \max(0, v_{k+n}^{\mathrm{et}} - v_{\max})]$ enforces speed constraints, and $r^{\mathrm{reg}} = -\sum_{n=1}^{N_\mathrm{u}} \max\left(0, (P_{k+n} - P_{k+n-1})^2 - \delta\right)$ promotes control smoothness by limiting power changes. The coefficients $\lambda_1, \lambda_2, \lambda_3$ are tunable weights.

Policy training

We adopt NPM as the policy within our RL framework, training it end-to-end in a closed-loop simulation using the soft actor-critic algorithm [45, 46]. The training process jointly optimizes a policy network NPM_ϕ and two Q-value networks Q_{ψ_1} and Q_{ψ_2}. The policy network defines a stochastic action distribution $\pi_\phi(\mathbf{u}_k \mid \mathbf{o}_k)$, with actions generated via the reparameterization trick:

$$\mathbf{u}_k = \mathrm{NPM}_\phi(\epsilon_k; \mathbf{o}_k), \epsilon_k \sim \mathcal{N}(0, 1). \tag{7}$$

During deployment, actions are selected deterministically as the mean of the learned distribution:

$$\mathbf{u}_k = \mathrm{NPM}_\phi(\mathbf{o}_k) = \mathbb{E}_{\pi_\phi(\cdot|\mathbf{o}_k)}[\mathbf{u}_k], \tag{8}$$

representing the expected action under the trained policy. The use of two Q-value networks helps mitigate overestimation bias and improves training stability.

The policy network NPM_ϕ is implemented as a two-layer multilayer perceptron (MLP). Its parameters ϕ are updated by minimizing the loss:

$$L_\pi(\phi) = \mathop{\mathbb{E}}_{\mathbf{o}_k \sim \mathcal{D},\, \epsilon_k \sim \mathcal{N}} \Big[\alpha \log\big(\pi_\phi(\mathrm{NPM}_\phi(\epsilon_k; \mathbf{o}_k) \mid \mathbf{o}_k)\big) - Q_\psi(\mathbf{o}_k, \mathrm{NPM}_\phi(\epsilon_k; \mathbf{o}_k)) \Big] \tag{9}$$

where $\mathcal{D}$ is the experience replay buffer and is used in gradient steps and α is a temperature parameter that balances exploration (via entropy) and exploitation (via reward), thereby regulating the stochasticity of the optimal policy. The Q-function $Q_\psi(\mathbf{o}_k, \mathbf{u}_k) = \min(Q_{\psi_1}, Q_{\psi_2})$ is computed as the minimum of two three-layer MLP critics Q_{ψ_1} and Q_{ψ_2} to further guard against overestimation. The Q-critics are trained by minimizing the soft Bellman residual:

$$L_Q(\psi_i) = \mathop{\mathbb{E}}_{(\mathbf{o}_k, \mathbf{u}_k) \sim \mathcal{D}} \left[\left(Q_{\psi_i}(\mathbf{o}_k, \mathbf{u}_k) - Q_{\psi_i'}(\mathbf{o}_k, \mathbf{u}_k)\right)^2 \right], i \in \{1, 2\}, \tag{10}$$

where the target value is given by:

$$Q_{\psi_i'}(\mathbf{o}_k, \mathbf{u}_k) = r(\mathbf{o}_k, \mathbf{u}_k) + \mathop{\mathbb{E}}_{\epsilon_k \sim \mathcal{N}} \Big[\min_{\psi_i'} Q_{\psi_i'}(\mathbf{o}_{k+1}, \mathrm{NPM}_\phi(\epsilon_{k+1}; \mathbf{o}_{k+1})) - \alpha \log(\pi_\phi(\mathrm{NPM}_\phi(\epsilon_{k+1}; \mathbf{o}_{k+1}) | \mathbf{o}_{k+1})) \Big].$$

Here, ψ_1', ψ_2' are the parameters of target networks, updated periodically to stabilize training. The temperature parameter α

regulates the algorithm's emphasis on entropy maximization versus reward optimization. If α is too small, the algorithm reduces to a standard Actor-Critic method, prioritizing reward accumulation without sufficient exploration. Conversely, an excessively large α causes the algorithm to focus solely on maximizing entropy, disregarding environmental rewards and compromising control performance. Thus, we adjust α automatically using the loss

$$L(\alpha) = \alpha \mathop{\mathbb{E}}_{\mathbf{o}_t \sim \mathcal{D}} \left[-\log(\pi_\phi(\cdot|\mathbf{o}_t)) + \dim(\mathbf{u}) \right]. \quad (11)$$

The RL algorithm outlined in Algorithm 1 alternates between gathering experience from the environment using the current policy and updating policy network NPM_ϕ, the two value networks Q_{ψ_i}, and the temperature parameter α via stochastic gradients computed from batches of size $b = 256$ sampled from the replay buffer $\mathcal{D}$.

Algorithm 1 RL training algorithm

Require: $\psi_1, \psi_2, \phi, \alpha$
  $\psi_1' \leftarrow \psi_1, \psi_2' \leftarrow \psi_2$
  $\mathcal{D} \leftarrow \emptyset$
  for each iteration **do**
    for each environment step **do**
      $\mathbf{u}_k = \mathrm{NPM}_\phi(\epsilon_k; \mathbf{o}_k), \epsilon_k \sim \mathcal{N}(0, 1)$,
      $\mathbf{o}_{k+1}, r(\mathbf{o}_k, \mathbf{u}_k) \leftarrow ENV(\mathbf{o}_k, \mathbf{u}_k)$
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{o}_k, \mathbf{u}_k, r(\mathbf{o}_k, \mathbf{u}_k), \mathbf{o}_{k+1})\}$
    end for
    for each gradient step **do**
      random select batch size b samples from $\mathcal{D}$
      compute $\nabla_\phi L_\pi(\phi), \nabla_{\psi_1} L_Q(\psi_1), \nabla_{\psi_2} L_Q(\psi_2), \nabla_\alpha L_\alpha$ from the selected b samples
      $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi L_\pi(\phi)$
      $\psi_i \leftarrow \psi_i - \lambda_Q \nabla_{\psi_i} L_Q(\psi_i)$ for $i \in \{1, 2\}$
      $\alpha \leftarrow \alpha - \lambda \nabla_\alpha L(\alpha)$
      $\psi_i' \leftarrow \tau\psi_i + (1 - \tau)\psi_i'$ for $i \in \{1, 2\}$
    end for
  end for

To ensure alignment with the optimization objective defined in Equation 1, we constructed 270 distinct simulation environments, each representing a 5 km route segment systematically extracted from 11 large-scale real-world driving scenarios spanning approximately 50 km each. This comprehensive environmental configuration enables the policy to learn optimal control strategies within the specified 5 km planning horizon while ensuring exposure to diverse operational conditions. The policy network undergoes training using the Adam optimization algorithm with a learning rate of 10^{-4}.

5.5.2 Policy Test for NPM

We conducted a detailed comparison and visualization of the strategies adopted by NPM and human drivers on Seg[5–25] of the Laiwu route, aiming to elucidate the fine-grained advantages of FEAD in local decision-making. Fig. 12 presents three subplots corresponding to different strategy configurations: a, full trip with FEAD; b, uphill with the MD strategy and downhill with FEAD; and c, uphill with FEAD and downhill with MD, In panels b and v, the power trajectories are constructed by concatenating real data from FEAD and Driver-WT, while the velocity and fuel consumption curves for all three subplots are computed by NTM. Due to the variations in power profiles, the mean velocities across the three cases differ. To enable a fair fuel consumption comparison, we performed linear normalization to align the mean velocities of all methods with that of Driver-WT-3, and the corresponding power, speed, and fuel consumptions are denoted as (SN).

As shown in the figure, during uphill driving, FEAD accelerates earlier than MD, effectively avoiding high-fuel-cost compensations caused by insufficient power. At the hilltop and in the subsequent flat or downhill sections, FEAD predominantly relies on inertial coasting, refraining from maintaining a small throttle input, whereas MD tends to sustain minor throttle engagement, leading to additional fuel consumption. These behaviors arise from NTM's precise modeling of vehicle dynamics and fuel consumption characteristics, enabling FEAD to optimize throttle engagement and release timing with greater accuracy, thereby achieving higher fuel efficiency within a single uphill–downhill cycle.

Under the condition of matched average speed, adopting MD-uphill + FEAD-downhill reduces fuel usage by 0.07 L (0.35 L/100 km); using FEAD-uphill + MD-downhill achieves a reduction of 0.16 L (0.8 L/100 km); and employing FEAD for both uphill and downhill yields the greatest improvement, saving 0.33 L (1.65 L/100 km). These results demonstrate that FEAD outperforms MD across both uphill and downhill scenarios, and that the combined strategy achieves synergistic fuel-saving effects beyond the linear sum of individual gains—highlighting the global coordination emerging from FEAD's local optimization capability.

5.6 The Optimal Control Baseline

5.6.1 Longitudinal Dynamics Model

Predictive Cruise Control (PCC) represents an advanced driving assistance system for autonomous trucks, designed to optimize fuel efficiency and vehicle performance through anticipatory road condition analysis. The core objective of PCC is to maximize fuel efficiency during both ascent and descent on graded terrains, necessitating a precise longitudinal behavior prediction model for heavy-duty vehicles. Within the PCC framework, accurate vehicle dynamics modeling is essential. Considering road gradient, aerodynamic drag, and rolling resistance, the longitudinal dynamics $\mathbf{a} = f_\mathrm{D}(\tilde{\mathbf{P}}; m, c_\mathrm{r}, c_\mathrm{d})$ are formulated through Newton's second law as:

$$a_i = \begin{cases} \frac{\eta \tilde{P}_i}{mv} - g\sin\theta_i - c_r g\cos\theta_i - \frac{c_d \rho_\mathrm{air} A v_i^2}{m}, & \text{if } \tilde{P}_i >= 0, \\ c_b \tilde{P}_i, & \text{if } \tilde{P}_i < 0 \end{cases} \quad (12)$$

where m is the vehicle mass, a represents the vehicle's acceleration, c_r is the driving resistance coefficient, c_d is the air drag coefficient, η is the engine power efficiency, g is the gravity acceleration, ρ_air is the air density, A is the front area of vehicle, c_b is the brake coefficient. where m denotes vehicle mass, a

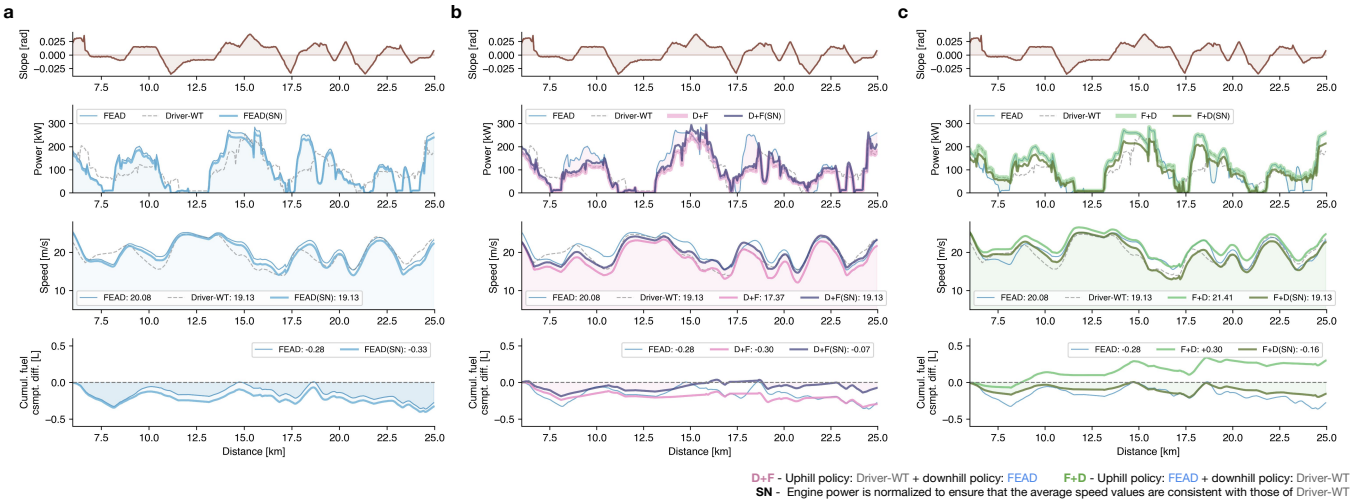

Fig. 12: Policy test for NPM. Extended version of Fig. 4d, illustrating the complete power and speed profiles, and fuel consumption deviations across Seg[5-25] for different methods.

represents acceleration, c_r is the driving resistance coefficient, c_d is the aerodynamic drag coefficient, η is engine efficiency, g is gravitational acceleration, ρ_air is air density, A is vehicle frontal area, and c_b is the brake coefficient. The critical parameters m, c_r, and c_d are estimated by system identification with the Kalman Filter methodology described in [47].

The optimal control formulation for eco-cruising on varying slopes must simultaneously address fuel efficiency, timeliness, and control smoothness. To enhance numerical stability during optimization, we incorporate a regularization term penalizing large power changes. Expanding upon Equation 1 and taking into account the vehicle's dynamics, speed limits and control limits, we formulate the complete optimal control problem as:

$$
\begin{aligned}
\min_{\{B_i,P_i\}_{i=1}^n} \quad & \sum_{i=1}^{n} (f_F(P_i) + k_p P_i^2)\,\Delta s + \beta B_i^2 \\
\text{s.t.} \quad & P_i \le P_\mathrm{max}, \quad i = 1,\dots,n, \\
& v_\mathrm{min}(s_i) \le v_i \le v_\mathrm{max}(s_i), \quad i = 0,\dots,n, \\
& \frac{1}{n}\sum_{i=1}^{n} v_i \ge v_\mathrm{trg}, \\
& v_0 = v_\mathrm{init}, \\
& v_{i+1}^2 = v_i^2 + 2a_i\Delta s, \quad i = 0,\dots,n-1,
\end{aligned}
\tag{13}
$$

where v_init denotes the initial speed and P_max the maximum engine power. To ensure stable convergence and facilitate the computation of an optimal solution, it is desirable for $f_\mathrm{F}(P_i)$ to be continuous, differentiable, and convex. The BSFC map provided by the manufacturer is a discrete lookup table describing the relationship among engine speed, torque, and fuel efficiency. To enable optimization in a continuous engine power space, the optimal fuel efficiency corresponding to each engine power value is extracted from the BSFC map. The resulting discrete pairs of engine power and optimal fuel efficiency are then fitted with a quartic polynomial, producing a smooth and continuous representation of fuel efficiency as a function of engine power. Consequently, $f_\mathrm{F}(P_i)$ is modeled as a fourth-order polynomial with respect to P_i.

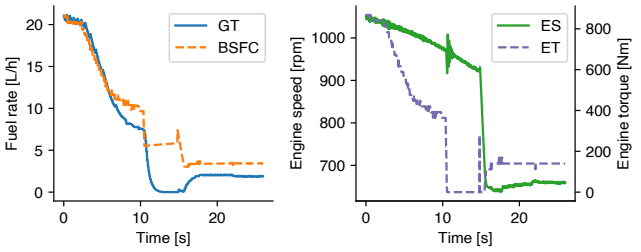

Fig. 13: Fuel consumption rate predicted by a BSFC map during rapid transitions in vehicle operation, such as abrupt acceleration or load changes. The BSFC model, typically calibrated under steady-state conditions, may exhibit increased error (left) during such dynamic events due to unaccounted engine transient effects in both engine speed and engine torques (right).

5.6.2 Predictive Cruise Control Optimizer

The original formulation in Equation 13 results in a nonlinear optimization problem that is inherently challenging to solve directly due to its nonconvex objective and nonlinear constraints. Such formulations typically face high computational costs and convergence difficulties, making them unsuitable for real-time or large-scale engineering applications. To overcome this limitation, we reformulate the problem as a quadratic programming (QP) problem. The key insight involves first discretizing the continuous-time vehicle dynamics into a finite horizon, yielding the discretized optimization objective function $J(\tilde{\mathbf{P}}) = \sum_{i=1}^{n}(f_F(P_i) + k_p P_i^2)\Delta s + \beta B_i^2$. Next, we locally approximate the nonlinear objective function and associated constraints through linearization and second-order Taylor expansion, specifically $J(\tilde{\mathbf{P}} + \epsilon) \approx J(\tilde{\mathbf{P}}) + \nabla J(\tilde{\mathbf{P}})^\top \epsilon + \frac{1}{2}\epsilon^\top \nabla^2 J(\tilde{\mathbf{P}})\epsilon$, resulting in a convex quadratic cost function

subject to linear equality and inequality constraints. This transformation produces a sequence of QP subproblems. To solve these efficiently, we employ the primal-dual interior-point method (IPM) [48], which is particularly suited for convex quadratic programming as it leverages the problem structure to achieve polynomial-time complexity and stable convergence. By iteratively solving these QP subproblems with IPM, the method converges to an approximate optimal solution while maintaining low computational complexity and numerical stability. This approach effectively balances accuracy and efficiency, ensuring that the optimized PCC strategy can be reliably deployed in real-world operational settings.

5.6.3 Vehicle Coastdown Experiment

Accurate fuel consumption control within the PCC framework critically depends on the precise values of the coefficients c_r and c_d in Equation (12). To determine these road load parameters for a fully loaded truck weighing 42,000 kg – consistent with the vehicle used in our FEAD experiments – we conducted a series of vehicle coastdown tests following the standardized procedures outlined in [49] and [50]. The experimental process began by warming up the test truck through a 30-minute driving period at 60 km/h to stabilize tire and driveline temperatures. The truck was then accelerated to 90 km/h, stabilized at this speed, and the accelerator pedal was released to idle before shifting the transmission into neutral. Data recording captured the speed versus distance coastdown curve until the truck came to a complete stop, and each run was repeated in the opposite direction to cancel out the effects of headwinds. Given the requirement for a long, constant-grade road segment to accommodate a full 90-0 km/h coastdown and the limitations of available test sites, we employed a split-run technique, dividing each coastdown into four phases: 87-70 km/h, 70-55 km/h, 55-30 km/h, and 30-0 km/h, as illustrated in Fig. 14a. To estimate c_r and c_d from the collected data, we formulated the energy equation during coasting as

$$\frac{1}{2}m(v_2^2 - v_1^2) + \int_{t_1}^{t_2} c_r mgv \, \mathrm{d}t + \int_{t_1}^{t_2} c_d A v^3 \, \mathrm{d}t = 0, \quad (14)$$

where v_1 and v_2 represent the longitudinal speeds at times t_1 and t_2 , respectively. This equation was rewritten in linear form as

$$\begin{bmatrix} H_1 & H_2 \end{bmatrix} \begin{bmatrix} c_r & c_d \end{bmatrix}^\top = Y, \quad (15)$$

with $H_1 = \int_{t_1}^{t_2} mgv \, \mathrm{d}t$, $H_2 = \int_{t_1}^{t_2} Av^3 \, \mathrm{d}t$, and $Y = -\frac{1}{2}(m + m_{\mathrm{rot}})(v_2^2 - v_1^2)$, transforming the parameter estimation into a multiple linear regression problem. Using the RANSAC algorithm [51] with a minimum sample size of 2, a relative residual threshold of 5%, and a maximum of 2000 iterations and imposing bounds of $[0.25, 0.50]$ for c_d and $[-0.01, 0.05]$ for c_r, we fitted the data from 8 sample points (two tests in opposite directions, each with four split runs). The results, shown in Fig. 14b, yielded estimated values of $c_d = 0.4037$ and $c_r = 0.0033$, with the RANSAC algorithm achieving a 75% inlier rate. These parameters are essential for optimizing fuel economy in the PCC system and are utilized throughout this study.

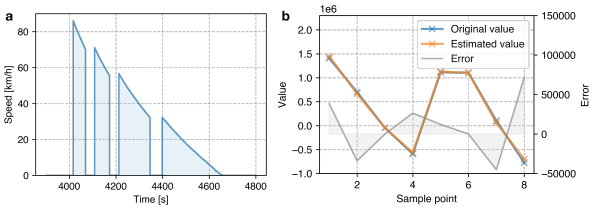

Fig. 14: Vehicle coastdown experiment. **a** Speed curve for the coastdown test. **b** Data fitting result.

5.7 The Low-level Controller

The acceleration sequence predicted by NTM, based on the engine power sequence generated by NPM, is transformed into executable vehicle control commands. However, the physical limitations of common truck actuators lead to discontinuities in the mapping between desired and executed accelerations. To address this, the Branch and Bound algorithm [52] is integrated with the MPC framework [53] to compute the optimal control acceleration sequence. Within this framework, the MPC formulation explicitly enforces constraints on acceleration, jerk, braking force, and vehicle speed. When the acceleration a_i exceeds the coasting acceleration, the required driving force is calculated according to Newton's second law, from which the corresponding torque is derived. Based on the pedal map provided by the manufacturer, the torque is then converted into a throttle opening. Conversely, when the acceleration a_i falls below the braking threshold ($-0.05\, m/s^2$), a deceleration command is directly issued.

5.8 Evaluation via Simulation

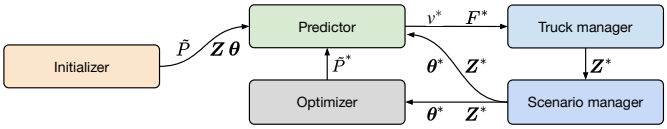

Fig. 15: Simulation pipeline. Initializer initializes vehicle state, control signals, and map data. Predictor estimates speed and fuel consumption using current state and control inputs. Truck Manager updates vehicle state based on predicted dynamics, and Scenario Manager handles map and scenario-related data. Optimizer computes optimal control signals for the next simulation step.

To train and evaluate optimization methods, we constructed a universal simulation framework (see Fig. 15). During initialization, the simulator extracts the desired power sequence $\tilde{\mathbf{P}}$, vehicle state $\boldsymbol{Z}$, and the most recent 5 km slope information $\boldsymbol{\theta}$ from each scenario. A predictor module then estimates speed and fuel consumption based on these inputs. The truck manager updates the vehicle state to $\boldsymbol{Z}^*$, while the scenario manager computes an updated slope sequence $\boldsymbol{\theta}$ from the map data. Finally, the optimizer determines the optimal engine power based on $\boldsymbol{Z}^*$ and the terrain profile. All simulations were executed on the high-performance workstation previously described.

In the FEAD configuration, the state vector $Z = [ctx, a, v, F, s]$ includes eigen context, acceleration, speed, fuel consumption, and longitudinal position. The predictor employs NTM to compute updated acceleration a^* and fuel consumption F^*, and the truck manager updates the state to $Z^* = [ctx, a^*, v^*, F^*, s^*]$. NPM, serving as the optimizer, computes the optimized engine power P^* for the next step. For conventional methods, the state is defined as $Z = [m, c_r, c_d, a, v, F, s]$, encompassing mass, resistance coefficients, and kinematic variables. The predictor uses Equation 12 and Equation 13 to update acceleration and fuel consumption, respectively, and the truck manager produces the updated state Z^*. The PCC optimizer then generates the next engine power command.

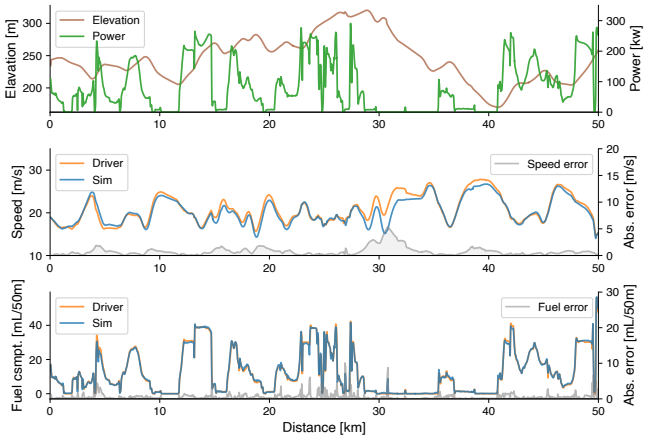

Fig. 16: Closed-loop simulation validation on the Laiwu route during driver command replay. The top subplot shows elevation profile and engine power input; the middle and bottom subplots compare actual (labeled Driver) and simulated (labeled Sim) speed profiles and fuel consumption rates, respectively.

Simulation accuracy is critical for reliable evaluation. We validated the FEAD closed-loop simulation on the Laiwu test route (Fig. 16), where inputting road gradient and engine power yielded simulated speed and fuel consumption values. Although some variance occurred between simulated and actual speeds ($R^2 = 0.86$), the results were highly aligned in most cases. Fuel consumption rates were nearly identical to actual values ($R^2 = 0.99$), confirming the simulation's accuracy.

5.8.1 Ablation study

To evaluate the effectiveness of the objective, dynamics, and policy modules, we extracted 10 scenarios from the Large-Scale-Detailed dataset and 1 scenario from the Laiwu-Detailed(LWD) dataset, resulting in a total of 11 test scenarios, each approximately 52 km in length. We implemented six configurations of the core modules, in which the objective, dynamics, and policy components were realized using different methods (see Table 2). These configurations were designed to analyze the contribution of each component to the overall performance of the operational speed planning system. Specifically, the objective module was implemented using either a fourth-order polynomial fitting of the BSFC map or NTM; the dynamics module employed either system identification or NTM; and the policy module adopted either NPM or PCC. The implementations of the BSFC map, system identification, and PCC are described in Sec. 5.6. Each configuration represents a hybrid integration of model-based and data-driven approaches.

Exp.	Objective	Dynamics	Policy	Avg. speed [m/s]	Delivery deadline	Fuel csmpt. [L/100km]
1	BSFC map	SI	PCC	20.23	√	26.30
2	BSFC map	SI	NPM	19.92	√	25.33
3	NTM	SI	NPM	20.43	√	24.04
4	BSFC map	NTM	NPM	20.58	√	23.96
5	NTM	NTM	PCC	20.67	√	23.30
6	**NTM**	**NTM**	**NPM**	**20.87**	√	**22.22**

Table 2: Ablation study of FEAM modules. Objectives, Dynamics, and Policy indicate the modules used in each experimental configuration. Objectives correspond to either the BSFC map or the NTM-based fuel estimator; Dynamics refer to either the system identification model or NTM-based dynamics model; and Policy represents either the rule-based Predictive Cruise Control (PCC) or the learned NPM policy. Avg. Speed [m/s] denotes the average vehicle speed, with values $\geq 19.25\,\mathrm{m/s}$ satisfying the delivery time constraint. Delivery deadline indicates whether the time requirement is met (✓). Fuel csmpt. [L/100 km] refers to the average fuel consumption per 100 km. All results are averaged over 11 simulation environments, each spanning approximately 52 km. Among all configurations, Exp. 6 (NTM+NTM+NPM) achieves the best fuel-saving performance.

In terms of implementation, Exp. 1 was optimized using CasADi, whereas Exp. 2, 3, 4, and 6 utilized pre-trained NPM models trained with the soft-SAC algorithm on the same scenario set as NTM+NPM. Exp. 5 was optimized using the Adam algorithm. All models were evaluated under identical NTM-based simulation environments comprising 11 scenarios, each approximately $50km$ in length.

By comparing Exp. 1 and Exp. 6 in Table 2, our FEAD system with NTM and NPM could reduce fuel consumption by 16.28% (4.28 L/100 km) compared to PCC. Based on the preceding discussion, NTM demonstrates greater fuel efficiency in both objectives and dynamics compared to the conventional fuel-saving pipeline. Comparing Exp. 2&3 with Exp. 2&4 demonstrates that the reduction in fuel consumption achieved by replacing the dynamics module is more pronounced. Upon comparing Exp. 1 with Exp. 2 in Table 2, it is observed that replacing PCC with NPM in the policy formulation, while keeping the objectives and dynamics constant, still manages to enhance average speed while reducing fuel consumption. Further employing NTM to replace the objectives and dynamics modules, the fuel-saving effect of NPM is not nullified; on the contrary, the overall system can achieve an additional reduction in fuel consumption of 3.32 L/100 km by comparing Exp. 2 and Exp. 6 in Table 2.

The results show that introducing data-driven estimation modules (NTM) and learned policies (NPM) consistently improve fuel efficiency while maintaining travel time. In particular, the fully learned configuration (NTM+NTM+NPM) achieves the best overall performance, indicating the advantage of joint optimization between perception, dynamics modeling, and policy learning.

5.8.2 Controlled open-road simulation

We compared NPM against PCC, a method recognized for its superior control performance in recent studies. PCC formulates the fuel-optimal control problem in real time by minimizing consumption along a given altitude profile derived from high-definition mapping data while satisfying temporal constraints. This process generates an optimized speed trajectory and command sequence to enhance efficiency. However, as a method grounded in optimal control theory, PCC relies on an accurate quantitative model of the underlying dynamics, which depends heavily on explicit environmental modeling. In practice, available environmental information is often limited to local regions, constraining PCC to locally optimal solutions and hindering its ability to achieve global optimality. In contrast, NPM leverages reinforcement learning (RL) by incorporating long-term rewards into the objective function, enabling the discovery of control strategies that may exhibit suboptimal short-term performance but yield greater fuel savings over extended distances.

To evaluate both approaches, we conducted tests on the 52 km Laiwu route. For a fair comparison, the PCC policy was computed using the BSFC map for fuel estimation and System Identification (System ID) for dynamics modeling, and the NPM policy was computed under the same modeling constraints. When comparing the fuel consumption of the resulting PCC and NPM policies, we used NTM as the evaluation metric due to its superior predictive accuracy (Fig. 3). The PCC policy recorded a fuel consumption of 27.76 L/100 km, while the NPM policy achieved a lower consumption of 25.91 L/100 km. As shown in Fig. 17, although NPM initially exhibited higher fuel consumption than PCC, its relative advantage grew with mileage, underscoring its capacity for sustained optimization over long hauls.

The PCC framework requires solving a complex, nonlinear, and nonconvex optimization problem that integrates vehicle dynamics, fuel efficiency, and speed constraints. This typically necessitates iterative numerical methods to approximate optimal solutions. However, real-time decision-making imposes strict computational limits, forcing a trade-off between solution accuracy and responsiveness. We analyzed the impact of iteration count on PCC's performance (Fig. 18a), measuring both fuel consumption and computation time. Initially, PCC's fuel consumption decreased significantly as the number of iterations increased, stabilizing after approximately 25 iterations—indicating convergence. However, computation time grew linearly with the iteration count. In real-world applications, the iteration limit is typically restricted to 15 due to runtime constraints, meaning optimization may terminate before full convergence. In contrast, NPM achieves lower

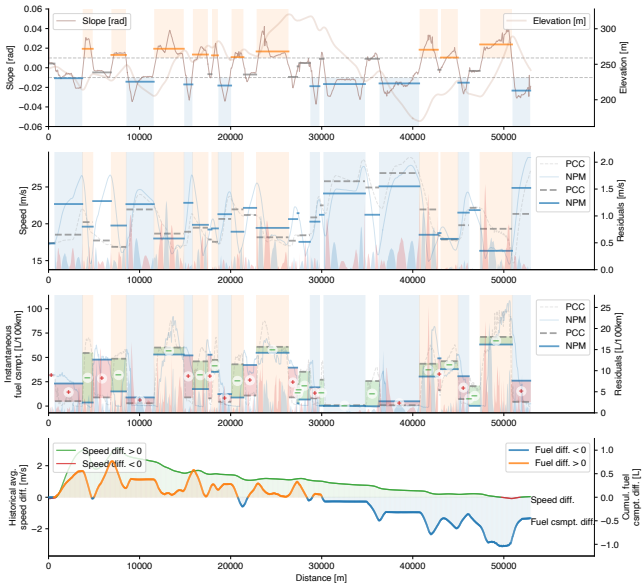

Fig. 17: Simulation-based comparison of speed and fuel consumption profiles between NPM and PCC on the Laiwu test route. The first subplot shows elevation and road slope variations, with orange ($\theta > 0.01$) and blue ($\theta < -0.01$) backgrounds indicating uphill and downhill segments. The second and third subplots display average speed and fuel consumption per slope section: solid blue (NPM) and dashed gray (PCC) lines represent actual values (thin) and normalized segment averages (thick) where normalization is performed by dividing actual values by road gradient to enable cross-slope comparison. Green/red rectangles highlight sections where NPM achieves higher/lower fuel efficiency relative to PCC. The final subplot shows historical average speed deviations (green/red: NPM faster/slower) and cumulative fuel consumption differences (blue/orange: NPM lower/higher).

overall fuel consumption without iterative computation during inference, drastically reducing runtime and highlighting its suitability for real-time deployment (Fig. 18a).

Although incorporating NTM into PCC has the potential to improve fuel modeling accuracy (Fig. 3), it also introduces a more extensive parameter space, which increases the problem's nonlinearity and nonconvexity. This amplifies the computational burden: while the original PCC (using BSFC and System ID) converges rapidly, PCC with NTM exhibits a slower convergence rate. As shown in Fig. 18b, the computational time required for 15 iterations of the original PCC is comparable to that for 10 iterations of PCC with NTM. Consequently, under real-time constraints, PCC with NTM is limited to just 10 iterations – a point at which its fuel efficiency is actually inferior to that of the original PCC after 15 iterations. Conversely, NPM with NTM achieves further reductions in fuel consumption without increasing computational complexity during inference. This combination delivers superior optimization performance while maintaining the runtime efficiency required for practical deployment.

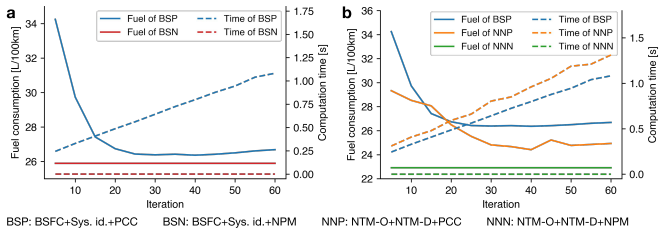

Fig. 18: Comparison of fuel consumption optimization performance between NPM and PCC under identical objective functions and dynamics models. Solid lines indicate fuel consumption, and dashed lines represent computation time, both plotted against the number of iterations. **a**, Using BSFC and System Identification for modeling: PCC's fuel consumption decreases with more iterations, stabilizing after approximately 25 iterations. In contrast, NPM achieves superior fuel efficiency without iterative optimization and requires less computation time. **b**, Using NTM for both modeling components: PCC with NTM achieves lower fuel consumption at convergence but demands increased computation time. NPM with NTM attains further reduced fuel consumption with relatively low computation time, consistently outperforming PCC with NTM in both efficiency and speed.

5.9 Large-Scale Evaluation

To rigorously evaluate the fuel-saving potential of the FEAD system under diverse, real-world conditions, we deployed the system across a fleet of more than 800 vehicles and conducted an extensive eight-month evaluation from January 1 to August 26, 2024. This large-scale implementation accumulated a total driving distance of 50.17 million kilometers with participation from over 1,000 professional drivers. To establish a robust performance baseline, drivers periodically assumed manual control of the vehicles, resulting in 15.96 million kilometers (31.81% of the total distance) driven manually and 34.21 million kilometers (68.19% of the total) under autonomous control using the FEAD system. Our analysis revealed that the FEAD system achieved an average fuel consumption of 23.44 L/100 km in autonomous driving (AD) mode, compared to 25.45 L/100 km under manual driving (MD), yielding an average fuel reduction of 2.01 L/100 km. Extrapolating this fuel reduction rate to the entire accumulated distance indicates that fully autonomous operation would have conserved 1 million liters of diesel and mitigated over 2,700 metric tons of CO_2 emissions [54]. Projecting these results to the U.S. heavy-duty truck fleet, which traveled 526 billion kilometers in 2021 [55], widespread adoption of autonomous driving could achieve annual fuel savings of 10.4 billion liters and reduce CO_2 emissions by more than 28 million metric tons.

Data analysis basis

Conventional synchronized fleet testing, which requires two trucks to operate in parallel under aligned conditions to enable precise fuel consumption comparison, is infeasible for large-scale evaluation scenarios. To ensure a fair and robust comparison of fuel efficiency between autonomous driving (AD) and manual driving (MD), we developed a comprehensive data processing and analysis methodology that aligns data across three critical dimensions: road conditions, vehicle-related variations (such as payload per trip), and environmental variations (including traffic flow and weather conditions). Given that comparisons must be conducted on identical road sections to ensure consistency in slope, surface friction, and other factors, we adopted a geo-spatial data organization approach, partitioning the entire test region into 4.9 km × 4.9 km geo-tiles. During testing, driving mode (AD or MD), instantaneous fuel consumption, GPS location, vehicle speed, and perception results were recorded at 1-second intervals and uploaded in real-time via a 4G network, with data subsequently grouped by geo-tile. Each geo-tile may contain bidirectional highway sections or multiple routes; data points from the same route and direction are defined as a comparable section, and each section can comprise multiple trips (referred to as section samples). A section sample is labeled as AD if over 95% of its mileage was driven autonomously and as MD if over 95% was driven manually. Environmental data –including temperature, wind speed, and precipitation – for each section sample were obtained through a commercial web service using timestamp and GPS coordinates and bound to the sample for integrated analysis. This structured approach enabled the construction of 17,320 valid geo-tiles with 34,170 unique sections, corresponding to 5.35 million accumulated section samples under AD mode (covering 26.53 million kilometers) and 2.17 million samples under MD mode (covering 10.91 million kilometers), ensuring a statistically robust and environmentally contextualized comparison.

Operational condition statistical analysis

As emphasized previously, conducting fuel consumption comparisons on identical road sections – ensuring consistency in slope, surface friction, and other geometric factors – is fundamental to a fair evaluation between autonomous driving (AD) and manual driving (MD). We therefore began by analyzing the road section distribution under both modes across our large-scale dataset. The geographic density distributions for AD and MD, illustrated in Fig. 20, exhibit highly similar spatial patterns, indicating that road-related variables were effectively controlled in our comparison. This alignment confirms that any observed differences in fuel efficiency are attributable to driving mode rather than infrastructural variations.

Beyond road conditions, we extended our analysis to vehicle load and environmental factors, extracting seven operational features to represent driving conditions: payload, wind speed, temperature, proportion of slippery road surfaces, proportion of snowy conditions, proportion of rainy conditions, and proportion of daytime driving. To quantify the distributional similarity of these features between AD and MD, we employed the Jensen-Shannon (JS) divergence, which ranges from 0 (identical distributions) to 1 (maximally dissimilar). As demonstrated in Fig. 21, the JS divergence values decrease consistently with increasing data volume, indicating that operational conditions—including payload and environmental factors—converge toward similarity at scale. This convergence ensures that these variables exert negligible influence on the fairness of fuel consumption comparisons between AD and MD in our large-scale evaluation.

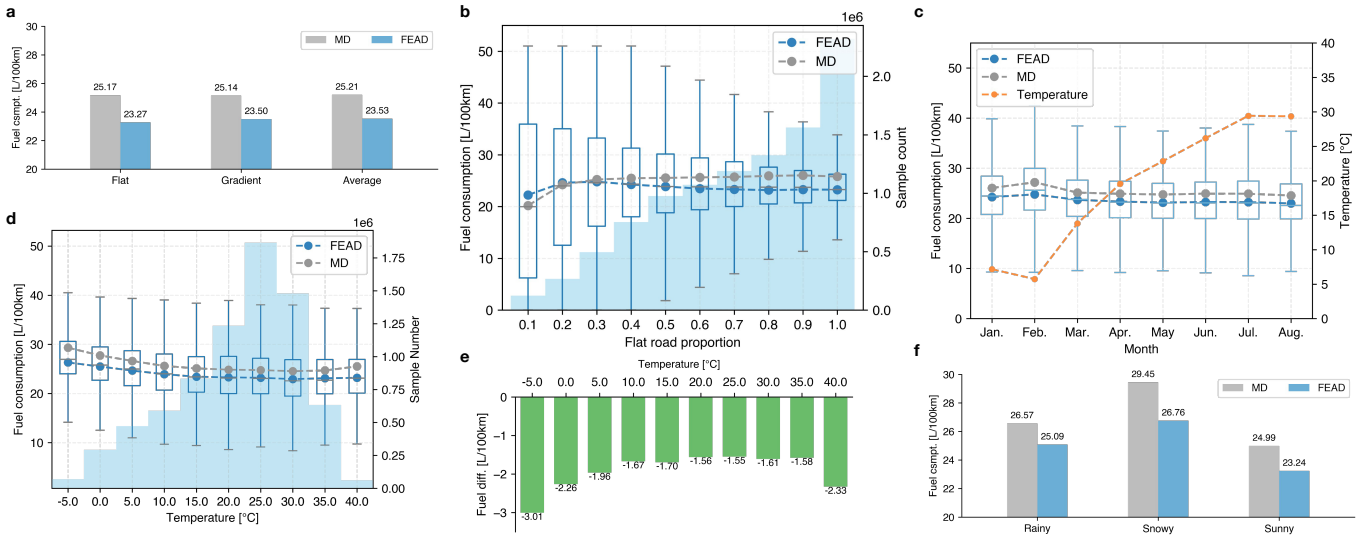

Fig. 19: Large-scale evaluation results of autonomous driving (AD) versus manual driving (MD). **a**, Comparison of AD and MD fuel consumption on flat and gradient roads. **b**, Fuel consumption as a function of flat-road proportion. **c**, Monthly fuel consumption: AD vs. MD. **d**, Fuel consumption at different ambient temperatures; **e**: Fuel consumption differences between AD and MD across temperature ranges. **f**, AD and MD fuel consumption under varying weather conditions (rain, snow, and sunny).

Fuel efficiency analysis: direct comparison

For fuel efficiency analysis, we adopted a direct road section-wise comparison approach. Each road section's fuel consumption was computed as the average across all corresponding AD or MD road section samples. To ensure statistical reliability and minimize outlier effects from anomalous operating conditions, we included only road sections with at least two samples for both AD and MD modes. Aggregating results across all valid road sections, we found that FEAD achieved an average fuel consumption of 24.27 L/100 km in autonomous mode, compared to 25.74 L/100 km under manual driving, yielding a reduction of 1.47 L/100 km (5.71%). This result underscores the efficacy of autonomous systems in enhancing fuel economy under real-world conditions.

Fuel efficiency analysis: difference-in-differences

To address the potential confounding effects of drivers' individual styles on fuel consumption, we implemented a difference-in-differences (DID) approach to isolate the impact of driving mode (autonomous driving, AD, versus manual driving, MD). In this design, vehicle V switches from manual to autonomous operation between two time periods, while vehicle W maintains manual driving throughout as a control. Fuel consumption was measured for vehicle V under manual operation in period t_1 (in road section R_1) and under autonomous operation in period t_2 (in road section R_2), with vehicle W driven manually in both periods under identical regional conditions (in road sections R_1 and R_2) to control for external factors like weather and terrain. Denoting fuel consumption as F_{V_1} and F_{V_2} for vehicle V, and F_{W_1} and F_{W_2} for vehicle W, the DID estimator is computed as $(F_{V_2} - F_{V_1}) - (F_{W_2} - F_{W_1})$, which subtracts changes under consistent manual driving from the treated vehicle's changes to isolate the mode-switch effect. Based on 7,206,989 sample pairs (6.94% of the full dataset) identified, results showed that AD reduced fuel consumption by an average of 1.57 L/100 km relative to MD, representing the pure effect after adjusting for temporal and regional variations.

Fuel efficiency analysis: regression

Fuel consumption is influenced not only by driver behavior but also by a range of external factors, including payload, road conditions, topography, ambient temperature, and wind. To ensure a valid comparison between autonomous driving (AD) and manual driving (MD), it is essential to control for these boundary conditions. While the difference-in-differences (DID) approach can partially address confounding variables, it relies on a limited subset of comparable samples – approximately 6.94% of the full dataset – which may constrain the representativeness of the results. To overcome this limitation, we adopted CatBoost, a machine learning algorithm designed for high-precision modeling in complex, heterogeneous datasets [44]. CatBoost utilizes a gradient boosting framework that integrates multiple weak learners to achieve robust nonlinear prediction capabilities. It excels in handling categorical variables through target encoding, which transforms discrete inputs into continuous representations, and reduces computational complexity and model bias via symmetric decision trees (oblivious trees) and an ordered boosting mechanism.

Due to divergent driving strategies, AD and MD can exhibit varying fuel consumption even at identical speeds (as illustrated in Fig. 12). In the large-scale dataset, data aggregation over 5 km intervals was necessary due to precision constraints, but this approach may obscure strategy-level details. To address this, we conducted separate fuel consumption regressions for AD and MD. Samples with an AD percentage exceeding 95% were used to train the CatBoost model for AD, while those with an AD percentage below 5% were used for MD. The AD model achieved a mean error of 0.015 L/100 km (0.06% of the mean), whereas the MD model had an error of 0.44 L/100 km (1.73%).

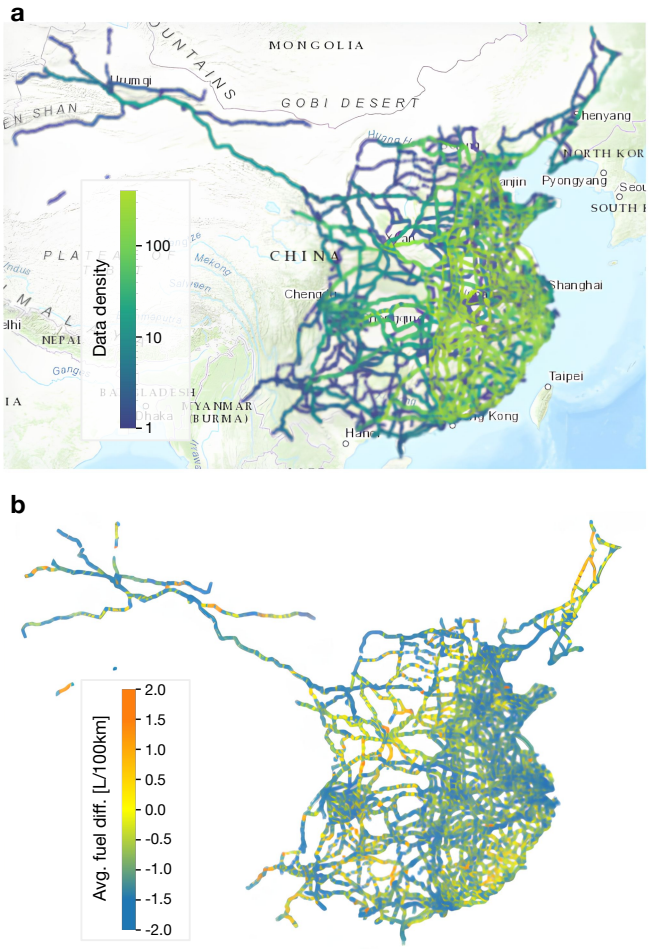

Fig. 20: Large-scale data. **a**, The spatial distribution of road sections across each region within the large-scale data. **b**, The average difference in fuel consumption between AD and MD across the corresponding region. A negative value denotes that AD demonstrates superior fuel efficiency, highlighting its potential advantages in sustainable energy utilization.

We assessed the consistency between predicted and observed values using the Spearman rank correlation coefficient [56], which ranges from -1 (inverse correlation) to 1 (perfect correlation). The AD model yielded a correlation of $\rho_s = 0.92$, and the MD model $\rho_s = 0.80$, indicating strong alignment in relative fuel rankings. The CatBoost models predicted average fuel consumption of 23.62 L/100 km for AD and 24.98 L/100 km for MD, resulting in a saving of 1.36 L/100 km (5.44%) under autonomous operation. Regional differences further support AD's superiority, as shown in Fig. 20b, where AD consistently outperforms MD in fuel efficiency across various areas.

Further data mining and analysis

We assessed the performance of AD relative to MD on both flat and gradient roads within the large-scale dataset. Road sections were classified as "flat" if the proportion of flat-road coverage exceeded 95%, and as "gradient" if this proportion fell below 5%. Under these criteria, AD consistently demonstrated superior fuel efficiency, achieving a reduction of 1.90 L/100 km on flat roads and 1.64 L/100 km on gradient roads, with an average

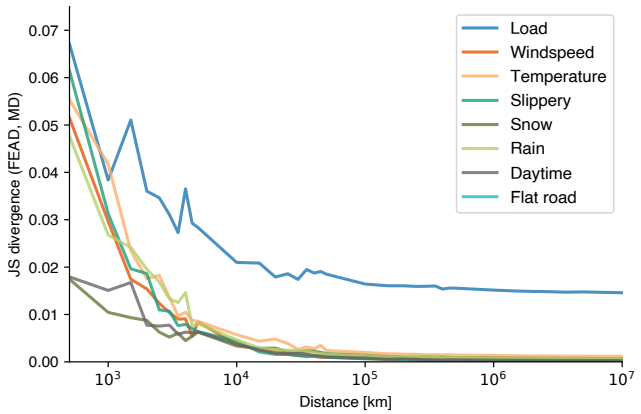

Fig. 21: The distribution similarity between AD and MD across eight features—load, wind speed, temperature, the proportion of slippery road surface, the proportion of snowy days, the proportion of rainy days, the proportion of daytime driving, and the proportion of flat-road driving—is quantified using Jensen–Shannon (JS) divergence, with values ranging from 0 (identical distributions) to 1 (maximally dissimilar).

fuel saving of 1.68 L/100 km across all road sections. Further analysis across a continuous range of flat-road coverage ratios revealed a statistically significant monotonic trend: the relative fuel efficiency advantage of AD over MD increased with the proportion of flat road (see Fig. 19b). This trend suggests that AD is better equipped to capitalize on reduced variability and resistance associated with flatter road profiles, likely due to its more accurate resistance modeling and adaptive throttle control, even when high-level driving strategies are similar between AD and MD.

Across all months, AD consistently maintained lower fuel consumption than MD (see Fig. 19c), with peak consumption occurring in February, attributable to lower temperatures. When temperatures ranged from –5°C to 15°C, fuel consumption decreased substantially with rising temperatures; however, in the 15°C to 35°C range, the effect of temperature on fuel usage was markedly attenuated (see Fig. 19d). Throughout the full temperature interval from –5°C to 35°C, AD exhibited superior fuel efficiency, and its consumption remained more stable than MD under extreme thermal conditions (see Fig. 19e). Temperature influences tire rubber hardness, tire pressure, and tire-road contact area, thereby affecting rolling resistance, while temperature-induced variations in air density alter aerodynamic drag. The enhanced accuracy of AD's resistance model in predicting these factors likely contributes to its improved fuel economy.

Under adverse weather conditions, AD achieved significantly lower fuel consumption: 25.09 L/100 km in rainy conditions (compared to MD's 26.57 L/100 km) and 26.76 L/100 km in snowy conditions (compared to MD's 29.45 L/100 km), when the proportion of rainy or snowy days exceeded 95% (see Fig. 19f). This improvement is attributed to AD's precise speed control and avoidance of abrupt acceleration and braking, whereas human drivers often make frequent speed adjustments due to caution or inexperience, increasing fuel

consumption. Braking energy loss was also evaluated, formulated as $E_{\text{brake}} = \int (a_{\text{coast}}(t) - a_{\text{local}}(t)) \cdot m \cdot v(t)\, dt$, where E_{brake} is the total braking energy loss in joules, $a_{\text{coast}}(t)$ is the natural acceleration during coasting, $a_{\text{local}}(t)$ is the actual acceleration, m is vehicle mass, $v(t)$ is speed, and dt is the time differential. AD's braking losses were 1.20 kWh per 100 km in rain and 0.52 kWh per 100 km in snow, compared to MD's losses of 2.46 kWh and 2.12 kWh, respectively. These reductions are statistically significant, indicating that AD dissipates less kinetic energy through braking and leverages its accurate resistance prediction model to conserve fuel more effectively under rainy and snowy conditions. Overall, AD achieves lower fuel consumption and braking energy loss, highlighting the effectiveness of its advanced control models.

Declarations

- Conflict of interest/Competing interests: The authors declare no competing interests.
- Authors' contributions: W.L. and R.Y. conceived the project. W.L., J.R., J.X., and Y.W. designed the core algorithms and experimental protocols. J.X. and Z.L. implemented the Neural Truck Module (NTM). J.X., Z.L., and X.G. conducted the NTM experiments and performed the corresponding analyses. J.R., Y.W., and S.C. implemented the Neural Policy Module (NPM) and carried out the associated experiments and analyses, with assistance and guidance from S.Z. in NPM design. J.Z. integrated the complete FEAD system into the autonomous driving software stack and deployed it on a real-world truck platform. X.C., J.P., and R.Y. provided critical technical mentorship and procedural guidance. W.L., J.P., and R.Y. wrote the manuscript, with input from all authors.