2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

NPC: Neural Predictive Control for Fuel-Efficient Autonomous Trucks

Jiaping Ren[1], Jiahao Xiang[1,2], Hongfei Gao[1], Jinchuan Zhang[1],
Yiming Ren[3], Yuexin Ma[3], Yi Wu[4], Ruigang Yang[1], *Fellow, IEEE*, Wei Li[1]

***Abstract*— Fuel efficiency is a crucial aspect of long-distance cargo transportation by oil-powered trucks that economize on costs and decrease carbon emissions. Current predictive control methods depend on an accurate model of vehicle dynamics and engine, including weight, drag coefficient, and the Brake-specific Fuel Consumption (BSFC) map of the engine. We propose a pure data-driven method, Neural Predictive Control (NPC), which does not use any physical model for the vehicle. After training with over 20,000 km of historical data, the novel proposed NVFormer implicitly models the relationship between vehicle dynamics, road slope, fuel consumption, and control commands using the attention mechanism. Based on the online sampled primitives from the past of the current freight trip and anchor-based future data synthesis, the NVFormer can infer optimal control command for reasonable fuel consumption. The physical model-free NPC outperforms the base PCC method with 2.41% and 3.45% more significant fuel saving in simulation and open-road highway testing, respectively.**

I. Introduction

Fuel costs are integral to the commercial vehicles' life cycle, particularly in the context of truck operations, where they comprise a substantial fraction of logistics companies' operational expenses. According to ATRI [1], fuel costs account for 28% of truck operational costs. Furthermore, road transportation is the major consumer of fossil fuel energy, contributing significantly to worldwide CO_2 and pollutant emissions [2]. Therefore, improving fuel consumption efficiency can not only reduce freight costs but also benefit the environment. However, the driving skill of human drivers in terms of fuel savings follows a distribution with large diversity and variance [3]–[5]. Thus, autonomous vehicles, or even advanced driving assistance systems (ADAS) are very promising to improve fuel consumption efficiency with a consistent high-performance driving strategy.

Recent studies [6]–[10] have demonstrated a significant fuel-saving advantage of autonomous vehicles. Regarding fuel efficiency for autonomous highway trucking, there is a substantial amount of researches [11]–[13] currently focusing on truck driving strategies. Hu et al. [14] develop an Adaptive Cruise Control (ACC) based on the Pulse-and-Gliding strategy to reduce fuel consumption, with a focus on car-following situations. Dynamic Programming has also been used for minimizing the energy consumption of trucks [15] using the local traffic information as the major optimization cost. Ren et al. [16] employ the linear quadratic regulator algorithm for predictive energy saving control based on future road slope, as well as the vehicle dynamic model. Among those fuel-saving solutions, predictive cruise control (PCC) [17]–[20] is the key line of research possessing the most significant potential for large-scale commercial applications.

The core of PCC-related works is an optimal control problem (OCP). Solutions for such an OCP using Pontryagin's minimum principle show high performance in fuel saving capability, as well as computation costs [21]–[24]. However, the performance in the real world is heavily dependent on the dynamic model and the engine/fuel model of the target vehicle. The accuracy of the parameters in the dynamic vehicle model, *e.g.* mass or weight, as well as the coefficient of resistance to drag and rolling, is always overlooked in laboratory experiments and research papers. However, it cannot be ignored in practical applications due to the huge impact on the result of PCC optimization. Technically, these vehicle parameters are estimated using classical system identification solutions [25], [26]. The engine control unit (ECU) of trucks used in our experiments can yield the estimated weight directly. We experimentally find that the ECU's weight estimation error is 8.5% in our autonomous fleet (747 trips with ground-truth weight from 17.7 tons to 51.85 tons are evaluated). Note that, diverse working conditions in freight tasks, *e.g.* different speed profiles and loads, terrains, weather, and climate would significantly increase the difficulty and error of vehicle parameter estimation.

To reduce the dependency on the accurate model of vehicle dynamics and engine, we propose a Neural Predictive Control method (see Fig. 1) for fuel-efficient autonomous trucks on hilly roads. NPC is a purely data-driven method, which is free of any physical model for the vehicle. In other words, weight, drag and rolling coefficient, and even the BSFC map of the engine are not needed anymore. Technically, a novel attention-based module NVFormer is designed to implicitly model the relationship between vehicle dynamics, road slope, fuel consumption, and control commands (speed, torque, *etc.*). Based on the online sampled primitives (data from the past of the current freight trip) and anchor-based future data synthesis, the NVFormer can accurately infer control commands for optimal fuel consumption. The physical model free NPC outperforms the base PCC method [18] in terms of not only the robustness over varying conditions but also the fuel-saving capability.

The main contributions of this paper are as follows:

- A pure data-driven NPC is proposed to solve the fuel-optimal control problem. NPC is free of any physical

[1]Inceptio Technology, Shanghai 200082, China, {jiaping.ren, hongfei.gao, jinchuan.zhang, ruigang.yang, wei.li}@inceptio.ai
[2]Tongji University, Shanghai 201800, China, xiang_jhao@163.com
[3]ShanghaiTech University, Shanghai 200120, China, {renym1, mayuexin}@shanghaitech.edu.cn
[4]Nanjing University of Posts and Telecommunications, Nanjing 210023, China, yiw@njupt.edu.cn

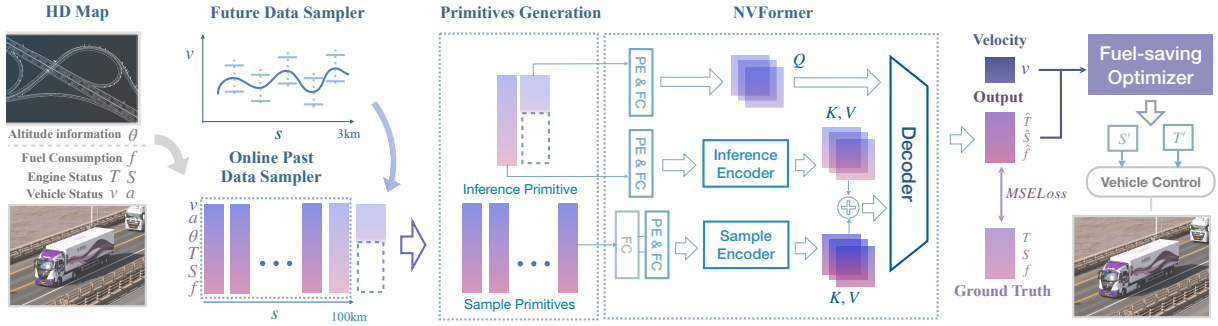

Fig. 1: Overview of the Neural Predictive Control (NPC) framework. Both offline and online past data consist of vehicle information from the embedded equipment and altitude information from the HD Map. The proposed NPC samples online past data (v,a,θ,T,S,f) and partial future data (v,a,θ), and generates inference primitive and sample primitives. Inference primitive comprises the latest data sequence along with the partial future data. Sample primitives are distilled from the online past data, excluding the latest data sequence. The NVFormer, which is trained on over 20,000 km offline past data with Mean Squared Error Loss (MSELoss), predicts the missing part (T,S,f) of the future data by the inference primitive and sample primitives. Fuel-saving optimizer utilizes the NVFormer to model the relationship between vehicle dynamics, road slope, fuel consumption, and control commands (speed, torque, *etc.*), and output the optimal truck control commands with reasonable fuel consumption.

model for the vehicle, *e.g.* dynamic model and even BSFC map of engine model.

- A novel attention-based module NVFormer is designed to implicitly model the relationship between vehicle dynamics, road slope, fuel consumption, and control commands (speed, torque, *etc.*).
- Compared to the PCC baseline [18], our NPC framework saves 2.41% fuel in 875.32km close-loop simulation testing and saves 3.45% in about 145km open-road highway testing with NVFormer which is trained by more than 20,000 km offline real-world vehicle data.

II. NPC Framework

In this section, we present the framework of Neural Predictive Control (NPC) for enhancing the fuel efficiency of autonomous trucks. Fig. 1 shows the overview of NPC. The input for NPC includes offline and online data with features vehicle speed (v), acceleration (a), slope (θ), engine torque (T), engine speed (S), and fuel consumption (f) correspondingly. For one freight trip, complete past data samples and partial future data with features v, a, θ are generated by the Online Past Data Sampler and the Future Data Sampler, respectively. Next, NPC yields the inference primitive and the sample primitives. Inference primitive comprises the latest data sequence along with partial future data. Sample primitives are distilled from the online past data, excluding the latest data sequence. We utilize the NVFormer to predict the missing part (T,S,f) of the future data by the inference primitive and sample primitives. With NVFormer, the Fuel-saving Optimizer can produce the optimal truck control commands that leverage fuel consumption while meeting transportation constraints.

A. Data Representation

NPC works on Frenet coordinates due to the predetermined lengths of commercial transportation routes. This coordinate system is extensively employed in autonomous planning tasks [27]. We use s to present distance, and $\{\eta^1,...,\eta^m\}$ to describe the data with m features: v,a,θ,T,S,f. The data sequence of feature i at a distance s is η^i_s. The distance at time t is s_t. We sample the data every Δs m. We use data chunks to represent the sequences of m data. The data chunk at s_t with length l is $\boldsymbol{\eta}(m,s_t,l) = \{\eta^i_s; i=1,...,m, s=s_t+\Delta s,...,s_t+l\Delta s\}$. We denote the $s \in \{s_t+\Delta s, s_t+\Delta s,...,s_t+l\Delta s\}$ as endpoint.

B. Online Past Data Sampler

The Online Past Data Sampler distills important samples from past data with features v,a,θ,T,S,f in current freight trip. We name such sequential sampled data as **primitive**, which encompasses representative vehicle states and environment information. The **inference primitive** comprises the latest data sequence along with the known partial future data (v,a,θ), collectively constituting local features (whether, road friction etc.) in NVFormer. **Sample primitive** is a data chunk sampled from the online past data, excluding the latest data sequence, which works as global context features (engine properties, vehicle shapes, tie friction characteristics etc.) in NVFormer. The length of sample primitives is l_h and the latest past data is $\boldsymbol{\eta}(m,s_t-l_h\Delta s,l_h)$ at time t. When selecting p sample primitives at time $t \in \{t_1,t_2,...,t_p\}$, the sample primitive k is $\boldsymbol{\eta}(m,s_{t_k},l_h)$.

1) Composition of Primitive: The primitive encompasses the following features: v, a, θ, T, S, and f. Specifically, the v, a, and θ represent the value at the endpoint over the distance interval Δs. The parameters T and S represent the average torque and engine speed, while f denotes the total fuel consumption during this interval. The NPC model implicitly learns the vehicle's status and dynamic characteristics with the above parameters. v and a could reflect the vehicle's kinetic information, while θ encapsulates gravitational potential energy. Additionally, we could generate control commands directly based on T and S.

2) Sample Primitives Selection: Sample primitives play a vital role in providing global context features. The vehicle information embedded within sample primitives enhances the prediction capabilities of the model. To provide better global information, sample primitives should encompass various working conditions and vehicle states. When the distance

between the current location and the location of sample primitives exceeds a threshold, the precision of global information drops to unusable. Hence, we establish a maximum effective distance of 100km for sample primitives. To reduce redundancy while maximizing the number of available primitives, we utilize a sliding window with a minimum step size of 1km and a primitive length of 2km. This design ensures the vehicle generates a new set of sample primitives approximately every 100km.

C. Future Data Sampler

The known future data includes the velocity status(v,a) generated by the future data sampler and slope (θ) obtained from the high-fidelity map. We predict T, S, and f by pre-trained NVFormer and obtain the complete future data. We use the data chunk $\boldsymbol{\eta}(m,0,l_f)$ to represent the future data, where l_f is the length of the future data chunk. We sample the speed for the future $l_f \Delta s$ m according to the altitude z (see Fig.2). Initially, we identify anchor points of the future altitude curve. Subsequently, we calculate speed limits at each anchor point using the reference speed line, which is created based on the current speed and the altitude curve, and the user-defined target speed line. Following this, we generate key points for each sampled speed line and get the future data chunk samples based on the key points.

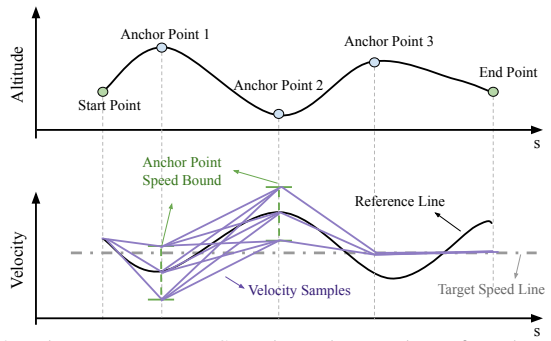

Fig. 2: The Future Data Sampler. The s-value of each anchor point refers to the position of the local maximum or minimum of the altitude curvature. The start point, anchor points, and end point together comprise the key points. Velocity curves are sampled within the speed bound of each anchor point.

1) Anchor Point Detection: An anchor point is a point with local maximum or minimum altitude. We detect anchor points by the condition:

$$\frac{\delta z}{\delta s} < \varepsilon, \tag{1}$$

where z is the altitude, ε is the near zero value. For each anchor point, we set the speed bound according to the target speed line defined by the user and the reference speed line created based on the current speed and the altitude curve. The closer, the greater the impact on decision-making at the current location. So, we first sample velocity around anchor points 1 and 2. As shown in Fig. 2, the speed bound of the anchor points 1 and 2 is defined by the reference line. Suppose the corresponding reference speed is v_f and the range is $\pm v_d$ m/s, the speed bound is $[v_f^i - v_d, v_f^i + v_d], i = 1,2$. The speed of the start point is v_f^s, and the speed bounds of the other anchor points (e.g., anchor point 3) and the end point is v_t, where v_t is the target speed. The start point, anchor points, and end point together comprise the key points.

2) Speed Sample Generation: We evenly divide each speed bound of the first two anchor points (anchor points 1, 2) into x parts, and get x^2 sampled speed series. The arbitrary speed series is $\{v_f^s, v_f^{1i}, v_f^{2j}, v_t, ..., v_t\}$, for $i \in \{1,2,...,x\}$ and $j \in \{1,2,...,x\}$. The speed samples can be generated by interpolating the key points' sampled speed series. Suppose the corresponding s values of the key-points is $\{0, s_{p_1}, ..., s_{p_k}, ..., l_f \Delta s\}$, s_{p_k} is the s value of the anchor point k. For $s_u \in [s_{p_k}, s_{p_{k+1}}]$, the uth interpolated speed of the ijth speed sample is

$$v_u^{ij} = \frac{s_{p_{k+1}} - s_u}{s_{p_{k+1}} - s_{p_k}} v_{p_k}^{ij} + \frac{s_u - s_{p_k}}{s_{p_{k+1}} - s_{p_k}} v_{p_{k+1}}^{ij}, \tag{2}$$

where $v_{p_k}^{ij}$ is the speed of the kth key-point of the ijth sampled speed series, s_u is the s value of the uth endpoint. Endpoint is the point with s value in $\{\Delta s, ..., u\Delta s, ..., l_f \Delta s\}, u \in \{1,2,...,l_f\}$.

3) Future Data Chunks: The future data chunk $\boldsymbol{\eta}(m, s_t, l_f)$ represents the composition for the future $l_f \Delta s$ m, where $\{\eta_s^i; i = 1,...,m, s = s_t + \Delta s, ...s_t + l_f \Delta s\}$. We divide the total $m = 6$ composition into the known composition v, a, θ with number $m_f = 3$ and the unknown composition T, S, f with number $m_r = 3$. We get velocities v from the velocity sampler, generate acceleration a from v, and get slope θ from the high-fidelity map. With the data chunks, we leverage the pre-trained NVFormer model to predict the unknown composition T, S, and f. Hence, $\eta_s^i = 0$ when $i > 3$ before prediction.

D. Neural Predictive Model

As illustrated in Fig. 1, the **NVFormer** model is built upon an Encoder-Decoder architecture, where the Encoder captures relevant information from sample primitives and inference primitive, and the decoder regresses the predictions. NVFormer is a transformer-based model since the multi-head attention mechanism serves a dual purpose. On the one hand, it can attend to the internal relationships within a single primitive. On the other hand, it can establish associations between the global context features of sample primitives and the local features of inference primitive, thereby providing additional information for predictions.

1) Encoder Module: The Encoder comprises a dual ensemble of Transformer Encoders [28], namely Sample Former and Inference Former. Sample Former is responsible for processing sample primitives and consists of n_s layer of vanilla Transformer Encoder Layers. Inference Former integrated a stack of n_i Transformer Encoder Layers devoted to processing historical feature series derived from inference primitive. The multi-head attention mechanism in the Transformer Encoder enhances the model's capacity to emphasize pivotal aspects within the input data.

2) Decoder Module: The decoder is structured with n_i layers of Transformer Decoder Layers. It assimilates insights emanating from the Encoder. Concurrently, a sequence mask is employed within the decoder to engage with forthcoming requests, fostering a heightened interdependence between the output and the sequential context of the data.

3) Model Workflow: Through the collaboration of the Future Data Sampler and the Online Past Data Sampler, we obtain a set of p sample primitives and a single inference primitive as inputs, where the future data sequences only encompass the known information (v, a, θ), while the unknown features (T, S, f) constitute the output of NVFormer.

In a single mini-batch, the shape of sample primitives is $p \times l_h \times m$. These dimensions undergo reduction via a fully connected (FC) layer, resulting in $l_h \times m$. Based on the description above, in an inference primitive, the shape of the latest data is $l_h \times m$. The input future sequence has m_f composition, and the sequence's shape is $l_f \times m_f$. For the output sequence, it is $l_f \times m_r$, where $m_r + m_f = m$. All data undergo an embedding process before being fed into the attention-based module. This embedding procedure encompasses both positional encoding [28] and an FC layer.

During the cross-attention module, the concatenated outputs of the Sample Former and Inference Former are employed as the Key (K) and Value (V) to the decoder module. The future input data sequences, with embedding and a sequence mask, serve as the future request Query (Q) for the decoder. Subsequently, the output of the decoder undergoes the FC layer, leading to the final output.

4) Loss Function: Since NVFormer aims to predict the engine status and fuel consumption, we utilize Mean Squared Error (MSE) as a loss function for training as depicted in Eq. 3, where N represents the sample quantity, y_i signifies the ground truth values, $\hat{y}_i$ denotes the predicted values.

$$\mathscr{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{3}$$

E. Fuel-saving Optimizer

Given the sample primitives, the latest data, and the future data samples, the fuel-saving optimizer selects the best output sequences through the utilization of the pre-trained NVFormer (refer to Algorithm 1). Specifically, the optimizer iterates through each future data sample. For each sample, we employ the pre-trained NVFormer to estimate the curves for torque, engine power, and fuel consumption. Subsequently, we compute the cost by the following equation:

$$C = w_1 \sum \eta_s^m + w_2(\left|v_{mean} - v_{target}\right|), \tag{4}$$

where $F = \sum_s \eta_s^m, s = \Delta s, ..., l_f \Delta s$ denotes the mth feature total fuel consumption. v_{mean} is the average velocity of the future data samples. v_{target} is the target speed. $|v_{mean} - v_t|$ is the speed difference between the v_{mean} and v_{target}. The closer between v_{mean} and v_{target}, the better to meet the transportation timing requirement. w_1 is the weight of fuel and w_2 is the weight of velocity. Since we focus more on fuel-saving efficiency, we set $w_1 = 1.0$ and $w_2 = 0.1$ in our implementation. The best future data sample is the one with the lowest cost.

Algorithm 1 Fuel-saving Optimizer

Require: $\eta(m, s_t - l_h\Delta s, l_h), \{\eta^i(m, s_t, l_f); i = 1, 2, ..., x^2\}$
$\boldsymbol{T} \leftarrow \mathbf{0}$, $\boldsymbol{S} \leftarrow \mathbf{0}$, $C_{best} \leftarrow 1000000$
for $i = 1$ to x^2 **do**
  Normalize $\eta(m, s_t - l\Delta s, l_h)$ and $\eta^i(m, s_t, l_f)$
  $\{\eta_{norms}^j; j > 3, s = s_t + \Delta s, ..., s_t + l_f\Delta s\} \leftarrow NVFormer(\eta(m, s_t - l\Delta s, l_f), \eta^i(m, s_t, l_f))$
  Denormalize $\{\eta_{norms}^j\}$ and get $\{\eta_{i,s}^j; j > 3, s \in [s_t + \Delta s, s_t + l_f\Delta s]\}$
  $C_i \leftarrow w_1 \sum_s \eta_{i,s}^6 + w_2(\left|\sum_s \eta_{i,s}^1 / l_f - v_{target}\right|)$
  if $C_i < C_{best}$ **then**
    $C_{best} \leftarrow C_i$, $\boldsymbol{T} \leftarrow \{\eta_{i,s}^4\}$, $\boldsymbol{S} \leftarrow \{\eta_{i,s}^5\}$
  end if
end for
return $\boldsymbol{T}$, $\boldsymbol{S}$

III. Experimental Results

To validate NPC's fuel-saving efficiency, we train and test the model on an offline dataset derived from real vehicle data. Subsequently, close-loop validation of NPC is performed on both simulation and open-road highway testing.

A. Experimental Results of NVFormer

We conducted NVFormer training and testing experiments on real-world Inceptio's autonomous truck data. Training the model on offline past data enables it to capture the characteristics of vehicle motion and engine behavior. Additionally, the comparative models included in this part are LSTM and Transformer.

1) Dataset: The existing datasets are not applicable for trucks [29]. Our data is derived from real commercial daily routes of Inceptio's autonomous trucks. All the data was obtained from 4 different vehicles with the same truck series from 26 trips of 4 routes, and the total mileage of all the data is more than 20,000 km. Each vehicle uses GPS, cameras, an inertial measurement unit, and a wire control chassis to record data. The length of each trip is approximately $600 \sim 850$km, and more than 95% of the mileage in these trips is on highways. To enhance the model's applicability across various external environments, these routes exhibit varying altitudes, encompassing flat terrains and slopes with different degrees of gradient. For instance, Fig. 3 illustrates two routes from our dataset. As shown in the figure, the northern part of Route (a) consists of mountainous terrain, while the southern part is characterized by plains. In Route (b), the northern section exhibits significant altitude changes, whereas the southern portion has relatively minor elevation variations. Furthermore, our data includes diverse weather conditions and different loads ($18.5 \sim 36.0$ ton, average 28.2 ton), aiming to increase the diversity of the dataset and enhance the generalization ability of the model. The dataset is used to train the vehicle dynamics, and the drivers' level of driving skill does not matter. The model can acquire

inherent vehicle characteristic parameters through the offline past dataset.

By employing a specified step size and point interval $\Delta s = 50$m, we employ a sliding window approach to sample multiple sets of data from each individual trip. We generated an offline dataset with about 140,000 primitives in aggregation. All the data is normalized within the range of 0 to 1 before being sent to the model. The ratio of training, validation, and test set is 7:1.5:1.5.

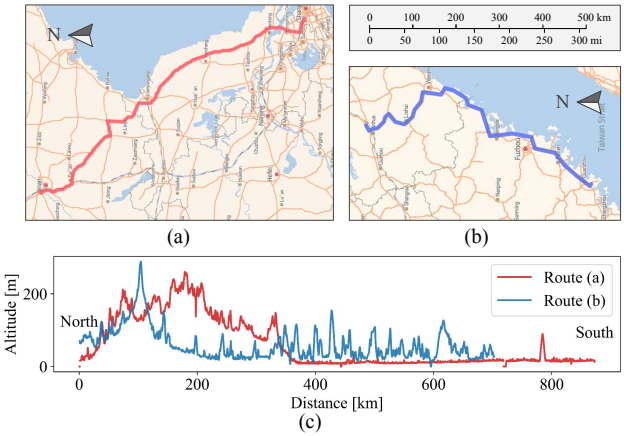

Fig. 3: Two of our datasets routes. Route (a) is located in East China, Route (b) is in Southeast China, and (c) represents their altitude variations from north to south.

2) Baselines and Experimental Settings: We choose an LSTM-based Encoder-Decoder architecture model (hereinafter referred to as LSTM) and the original Transformer model as baselines for comparative analysis, which could only receive the inference primitive as input. We use 2 encoder layers and 2 decoder layers for all models. Here Transformer and NVFormer are constructed with 8 multi-heads. For NVFormer, we set the number of sample primitives to 10 with 2 encoder layers in Sample Former. In both LSTM and Transformer architectures, additional FC layers are employed to facilitate the outputs. All models are constructed in the same embedding dimension and compared in data point interval $\Delta s = 50$ m, which leads to input length $l_h = 40$ and output length $l_f = 60$.

For all models, we conduct training for hundreds of epochs and select the model with the lowest validation loss after achieving stability for evaluation. A warm-up period of 10 epochs is employed for both Transformer and NVFormer. The initial learning rate for LSTM is 10^{-4} and for Transformer and NVFormer is 10^{-5}. The batch size of training with Adam optimizier [30] is set to 256. All experiments are implemented in PyTorch [31] and conducted on two NVIDIA RTX4090 24G GPUs. MAE and MSE are employed as evaluation metrics.

3) Results and Model Analysis: As demonstrated in Tab. I, in comparison to both LSTM-based and Transformer models, NVFormer exhibits a notable enhancement in predictive accuracy for engine torque. The prediction precision of engine speed and fuel consumption is also significantly elevated through the utilization of NVFormer. Incorporating sample primitives as historical information strengthens the model's capability to predict vehicular states. Within the context of the NPC framework, the provision of more accurate parameter predictions by the model affords NPC a more dependable reference for the vehicle's state.

Model	Torque [%]		Engine Speed [rpm]		Fuel [L/Δs]	
	MAE	MSE	MAE	MSE $\times 10^4$	MAE $\times 10^{-3}$	MSE $\times 10^{-7}$
LSTM-based	5.881	66.67	64.78	1.261	2.998	1.790
Transformer	4.792	40.44	66.17	1.170	2.974	1.780
NVFormer	**4.471**	**34.77**	**63.05**	**1.098**	**2.872**	**1.622**

TABLE I: Prediction performance of LSTM-based model, Transformer and NVFormer. On these three features, NVFormer exhibits the highest level of predictive accuracy.

Our model needs to be deployed onto operational autonomous vehicles. In the real-world scenario, our data points are spaced at $\Delta s = 50$ m intervals, translating to approximately 2 seconds of travel time during high-speed operation. Within the span of these 2 seconds, the NPC Future Data Sampler generates hundreds of velocity sampling lines between two consecutive data points. The NVFormer model, on the other hand, is tasked with processing these velocity lines within the limited time frame. Based on our model architecture design, NVFormer could be separately deployed. Since sample primitives are updated every 100 km, after updating sample primitives, we could use Sample Former to extract features and save the running results locally. After that, only Inference Former and NPC Decoder are running in real-time. This reduces the required computing power when running the model in real-time.

B. Close-loop Verification of NPC

We compare our NPC method with PCC [18] by close-loop verification in simulation and open-road highway. Building upon offline learning, the model within NPC can forecast future vehicle states based on data recorded by the Online Past Data Sampler. All models are executed using the C++ ONNX Runtime API [32].

1) Simulation: We use simulation to compare PCC, NPC with LSTM model, NPC with Transformer model, and NPC with NVFormer model on real-world road network. All simulation experiments are conducted on the personal computer with 128G memory and Intel Core i9-13900KF processor.

Scenario and Experimental Settings We create 15 different simulation scenarios from the real-world system, which include uphill, downhill, flat terrain, undulating road, and so on. Fig. 4 shows the altitude of these testing scenarios. Each scenario has a length of nearly 10 kilometers. Our simulation

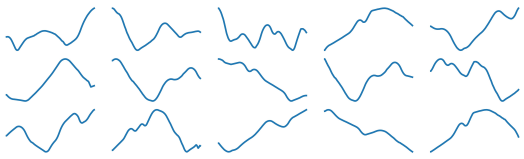

Fig. 4: The altitude curves of the 15 different simulation scenarios generated from the real-world map.

employs an in-house well-validated vehicle dynamics model to simulate the motion and performance of a vehicle under various conditions. We simulate fuel consumption interpolated from the BSFC Map [33]. In each scenario, we test the

Method	Fuel [L/100km]	Speed Diff. [m/s]	Cost	Fuel Saving [%]
PCC	25.70	0.83	25.78	-
NPC_{LSTM}	26.67	0.99	26.77	-3.77
$NPC_{Transformer}$	25.75	**0.53**	25.80	-0.19
$NPC_{NVFormer}$	**25.08**	0.69	**25.15**	**2.41**

TABLE II: Simulation Result. We simulate the four methods (PCC, NPC_{LSTM}, $NPC_{Transformer}$ and $NPC_{NVFormer}$) in 15 different scenarios, and set the target speed ranging from 19.44 m/s to 23.61 m/s. Fuel is the average value of all simulations with a total distance of 875.32 km for each method. Speed diff. is the average value of the difference between the average speed and the target speed for all simulations. $NPC_{NVFormer}$ has the lowest cost compared to the other three methods and saves 2.41% fuel compared to PCC.

fuel efficiency of each method with different target speed settings, which include 19.44 m/s, 20.28 m/s, 21.11 m/s, 21.94 m/s, 22.78 m/s, and 23.61 m/s.

Results To fairly compare the four methods (PCC, NPC_{LSTM}, $NPC_{Transformer}$ and $NPC_{NVFormer}$), we compute the fuel consumption by interpolation from the result. For each method, we conduct experiments with six different target speeds ranging from 19.44 m/s to 23.61 m/s in each scenario. We perform interpolation on simulated fuel consumption data at various target speeds to derive a fitting function, which is then used to estimate fuel consumption at 21.5 m/s. The estimated fuel consumption is treated as the fuel consumption of the corresponding method at the corresponding scenario (see Fig. 5). We also compute the speed difference, which is the difference between the average speed and the target speed. As we compute the fuel consumption and the speed difference for all methods at all scenarios, we compute the average fuel consumption F, the average speed difference Δv, and the cost C_{sim} for each method. Similar to Eq. 3, the cost function is the weighted sum of the fuel consumption and the speed difference, and

$$C_{sim} = w_1 F + w_2 \Delta v, \tag{5}$$

where $w_1 = 1.0$ and $w_2 = 0.1$. The simulation result is in Tab. II. Our $NPC_{NVFormer}$ has the lowest cost compared to PCC, NPC_{LSTM} and $NPC_{Transformer}$.

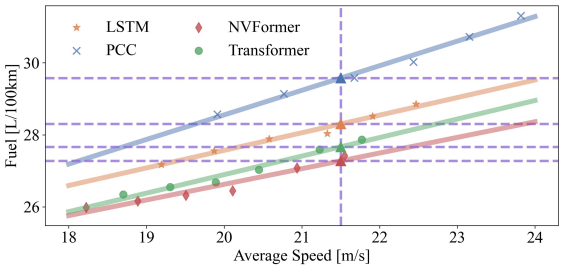

Fig. 5: An example of interpolating the fuel consumption at one scenario. We estimate the fuel consumption for each method by interpolating the simulated results at various target speeds from 19.44 m/s to 23.61 m/s.

2) Open-Road Highway Testing: The testing was conducted using autonomous driving trucks provided by Inceptio Technology. We made concerted efforts to control test variables, aiming to maintain a consistent testing environment.

Experimental Settings We conducted testing using the same truck type as an offline dataset for NVFormer. Both methods were subjected multiple times on the same highway section with identical load and driver on the same day. The length of the testing road is approximately 145km, consists of 36% flat terrain, and 57% slight uphills and downhills. When testing, the target speed was set to 20.83 m/s.

Results Tab. III indicates that NPC with NVFormer demonstrates a fuel efficiency improvement of nearly 1 liter per 100 kilometers compared to PCC in open-road testing. Despite a slightly lower average speed, the cost of NPC is still lower than PCC. We selected a road segment to conduct an in-depth analysis as shown in Fig. 6. This stretch of road encompasses three descents and two ascents. In the uphill segments, the torque output from NPC is comparatively lower than that of PCC. However, on uphill segments, this velocity differential is regained. In the gentle descent part, NPC maintained a lower torque output, allowing the vehicle to utilize gravitational potential energy to sustain its speed. NPC can achieve global control planning with higher fuel efficiency for the vehicle based on altitude information from a longer distance on the road.

Method	Fuel [L/100km]	Avg. Speed [m/s]	Cost	Fuel Saving [%]
PCC	27.50	20.07	27.58	-
$NPC_{NVFormer}$	**26.55**	19.13	**26.72**	**3.45**

TABLE III: Real-world Testing Result. Cost values are calculated by Eq.5 from simulation testing.

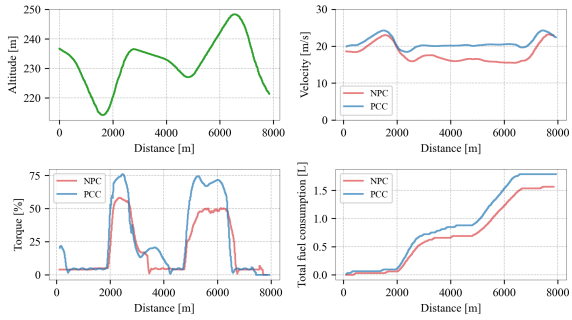

Fig. 6: Result of NPC and PCC in part of open-road highway testing. This 8-km long clip covers 2 complete uphill and downhill processes, and one of these three descents is relatively more gradual.

IV. Conclusion and Future Work

In this paper, we present a novel framework to estimate future torque and engine speed series with optimal fuel consumption under transportation timing constraints. To reduce the negative influence of inaccurate vehicle dynamics and engine models, the data-driven NPC method is proposed to take full advantage of a large amount of offline and online vehicle data. NPC performs better than the baseline PCC by the open-loop and close-loop verification. In the future, the efficiency and capability of the network can be enhanced to handle complex traffic on the fly while achieving a safe and fuel-saving autonomous freight trip. We will also learn from large-scale truck operational data to optimize the planner with a deep neural network directly instead of sampling future data to improve the computation efficiency of NPC.

REFERENCES

[1] A. Leslie and D. Murray, *An Analysis of the Operational Costs of Trucking: 2023 Update*. American Transportation Research Institute, 2023.

[2] D.-G. for Mobility and T. of European Commission, *EU transport in figures*. Publications Office of the European Union, 2022.

[3] M. Kamal, M. Mukai, J. Murata, and T. Kawabe, "Development of ecological driving system using model predictive control," in *2009 ICCAS-SICE*. IEEE, 2009, pp. 3549–3554.

[4] J. Van Mierlo, G. Maggetto, E. Van de Burgwal, and R. Gense, "Driving style and traffic measures-influence on vehicle emissions and fuel consumption," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 218, no. 1, pp. 43–50, 2004.

[5] H. J. Walnum and M. Simonsen, "Does driving behavior matter? an analysis of fuel consumption data from heavy-duty trucks," *Transportation Research Part D: Transport and Environment*, vol. 36, pp. 107–120, 2015.

[6] L. Zhang, T. Zhang, K. Peng, X. Zhao, and Z. Xu, "Can autonomous vehicles save fuel? findings from field experiments," *Journal of Advanced Transportation*, vol. 2022, 2022.

[7] S. Xu and H. Peng, "Design and comparison of fuel-saving speed planning algorithms for automated vehicles," *IEEE Access*, vol. 6, pp. 9070–9080, 2018.

[8] D. Fredette and U. Ozguner, "Dynamic eco-driving's fuel saving potential in traffic: Multi-vehicle simulation study comparing three representative methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2871–2879, 2018.

[9] R. Schmied, H. Waschl, and L. del Re, "Extension and experimental validation of fuel efficient predictive adaptive cruise control," in *2015 American Control Conference (ACC)*, 2015, pp. 4753–4758.

[10] H. Yang, H. Rakha, and M. V. Ala, "Eco-cooperative adaptive cruise control at signalized intersections considering queue effects," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1575–1585, 2016.

[11] M. Song, F. Chen, and X. Ma, "Organization of autonomous truck platoon considering energy saving and pavement fatigue," *Transportation Research Part D: Transport and Environment*, vol. 90, p. 102667, 2021.

[12] "Automotive platoon energy-saving: A review," *Renewable and Sustainable Energy Reviews*, vol. 179, p. 113268, 2023.

[13] S. Tsugawa, S. Jeschke, and S. E. Shladover, "A review of truck platooning projects for energy savings," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 68–77, 2016.

[14] H. Hu, C. He, H. Ma, C. Zou, H. Wu, X. Zhang, and J. Zhang, "Minimum fuel consumption strategy in autonomous adaptive cruise control scenarios," in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 6004–6009.

[15] J. Borek, B. Groelke, C. Earnhardt, and C. Vermillion, "Economic optimal control for minimizing fuel consumption of heavy-duty trucks in a highway environment," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 5, pp. 1652–1664, 2020.

[16] X. Ren, X. Lv, L. Zhang, L. Chen, H. Chu, and B. Gao, "Lqr-based predictive energy-saving control for commercial vehicles based on slope information," in *2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, 2022, pp. 1–6.

[17] F. Lattemann, K. Neiss, S. Terwen, and T. Connolly, "The predictive cruise control–a system to reduce fuel consumption of heavy duty trucks," *SAE transactions*, pp. 139–146, 2004.

[18] F. Zhang, Y. Yin, S. E. Li, Z. Xin, W. Li, and R. Yang, "Fuel efficient predictive cruise control for commercial vehicles with load variation," in *International Conference on Intelligent Transportation Engineering*. Springer, 2021, pp. 913–925.

[19] H. Chen, L. Guo, H. Ding, Y. Li, and B. Gao, "Real-time predictive cruise control for eco-driving taking into account traffic constraints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 8, pp. 2858–2868, 2019.

[20] E. Hellström, "Explicit use of road topography for model predictive cruise control in heavy trucks," 2005.

[21] B. Saerens, M. Diehl, and E. Van den Bulck, *Optimal Control Using Pontryagin's Maximum Principle and Dynamic Programming*. London: Springer London, 2010, pp. 119–138.

[22] K. Song, X. Wang, F. Li, M. Sorrentino, and B. Zheng, "Pontryagin's minimum principle-based real-time energy management strategy for fuel cell hybrid electric vehicle considering both fuel economy and power source durability," *Energy*, vol. 205, p. 118064, 2020.

[23] D. Shen, D. Karbowski, and A. Rousseau, "Fuel-optimal periodic control of passenger cars in cruise based on pontryagin's minimum principle," *IFAC-PapersOnLine*, vol. 51, no. 31, pp. 813–820, 2018, 5th IFAC Conference on Engine and Powertrain Control, Simulation and Modeling E-COSM 2018.

[24] S. Xu, S. E. Li, B. Cheng, and K. Li, "Instantaneous feedback control for a fuel-prioritized vehicle cruising system on highways with a varying slope," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1210–1220, 2017.

[25] E. Hellström, M. Ivarsson, J. Åslund, and L. Nielsen, "Look-ahead control for heavy trucks to minimize trip time and fuel consumption," *Control Engineering Practice*, vol. 17, no. 2, pp. 245–254, 2009.

[26] D. Jia, H. Chen, Z. Zheng, D. Watling, R. Connors, J. Gao, and Y. Li, "An enhanced predictive cruise control system design with data-driven traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8170–8183, 2022.

[27] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 987–993.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[29] S. Triest, M. Sivaprakasam, S. J. Wang, W. Wang, A. M. Johnson, and S. Scherer, "Tartandrive: A large-scale dataset for learning off-road dynamics models," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2546–2552.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[32] Microsoft, "Onnx rumtime," https://onnxruntime.ai/.

[33] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.