Systematic hyperparameter selection in Machine Learning-based engine control to minimize calibration effort

Control Engineering Practice 140 (2023) 105666

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Systematic hyperparameter selection in Machine Learning-based engine control to minimize calibration effort

Prasoon Garg [a,*], Emilia Silvas [a,b], Frank Willems [a,b]

[a] *Eindhoven University of Technology, Department of Mechanical Engineering, Control Systems Technology, 5600 MB Eindhoven, The Netherlands*
[b] *TNO Traffic and Transport, 5708 JZ Helmond, The Netherlands*

ARTICLE INFO

Keywords:
Engine control calibration
Preview control
Machine Learning
Long short-term memory neural network

ABSTRACT

For automotive powertrain control systems, the calibration effort is exploding due to growing system complexity and increasingly strict legal requirements for greenhouse gas and real-world pollutant emissions. These powertrain systems are characterized by their highly dynamic operation, so transient performance is key. Currently applied control methods require tuning of an increasing number of look-up tables and of parameters in the applied models. Especially for transient control this state-of-the-art calibration process is unsystematic and requires a large development effort. Also, embedding models in a controller can set challenging requirements to production control hardware. In this work, we assess the potential of Machine Learning to dramatically reduce the calibration effort in transient air path control development. This is not only done for the existing benchmark controller, but also for a new preview controller. In order to efficiently realize preview, a strategy is proposed where the existing reference signal is shifted in time. These reference signals are then modeled as a function of engine torque demand using a Long Short-Term Memory (LSTM) neural network, which can capture the dynamic input–output relationship. A multi-objective optimization problem is defined to systematically select hyperparameters that optimize the trade-off between model accuracy, system performance, calibration effort and computational requirements. This problem is solved using an exhaustive search approach. The control system performance is validated over a transient driving cycle. For the LSTM-based controllers, the proposed calibration approach achieves a significant reduction of 71% in the control calibration effort compared to the benchmark process. The expert effort and turbocharger experiments used in calibrating transient compensation maps in physics-based feedforward controller are replaced by little simulation time and parametrization effort in ML-based controller, which requires significantly less expert effort and system knowledge compared to benchmark process. The best trade-off between multi-objective cost terms is achieved with one layer and 32 cells LSTM neural network for both non-preview and preview control. For non-preview control, a comparable control system performance is achieved with the LSTM-based controller, while 5% reduction in cumulative NO_x emissions and similar fuel consumption is achieved with preview controller.

1. Introduction

Energy-efficient and cleaner automotive powertrains are required to meet strict regulations for greenhouse gas emissions and real-world pollutant emissions. To meet these regulations, the powertrain control system should achieve optimal and robust performance under wide range of operating conditions, disturbances, uncertainties, system aging and manufacturing tolerances. Current research includes advanced air and fuel path technologies and increasing levels of powertrain electrification for light-duty applications and study of waste heat recovery systems and advanced combustion concepts running on low carbon biofuels and hydrogen for heavy-duty transport. Due to the growing system complexity and real-world performance requirements, the control development effort for future powertrain control system will become unacceptable with traditional map-based control approaches (Atkinson, 2014). Transient control is especially challenging for automotive powertrains.

Model-based control calibration has the potential to reduce development time and reduce expensive engine testing (Mentink et al., 2013). It is currently considered the state-of-the-art approach. Three different applications are identified:

* Corresponding author.
E-mail address: p.garg.1@tue.nl (P. Garg).

https://doi.org/10.1016/j.conengprac.2023.105666
Received 27 January 2023; Received in revised form 6 July 2023; Accepted 10 August 2023
Available online 23 August 2023

Table 1
Comparison of model-based control approaches. PB is physics-based, FF is feedforward, FB is feedback and MPC is model predictive control.

	Studied benchmark (Moergastel et al., 2019)	Model for off-line optimization (Sequenz, 2013), (Isermann, 2014)	Model-embedded control (Karlsson et al., 2010), (Stewart et al., 2010) (Zhao et al., 2013) (Liu et al., 2021)	LSTM-based controller without preview (Norouzi et al., 2022) (Peng et al., 2023)	LSTM-based controller with preview
Criteria/Controller type	PB FF and PID-based FB	Map-based	MPC	LSTM-based FF and PID-based FB	LSTM-based FF and PID-based FB
Optimal performance	0	–	+	0	+
Robust performance	0	–	+	0	0
Calibration effort	0	–	+	+	+
Computational requirements	0	+	–	+	+

1. Engine models are used for off-line optimization of feedforward and reference setpoint maps (Isermann, 2014);
2. Embedding of engine models in feedforward (FF) controllers (Alfieri et al., 2009; Moergastel et al., 2019) or observers;
3. Model-embedded control, such as model predictive control (MPC) (Karlsson et al., 2010; Moriyasu et al., 2019; Stewart et al., 2010; Zhao et al., 2013), economic MPC (Liu et al., 2021).

Model-based control allows more control development at the desk (i.e., virtual environment) and, therefore, requires lower experimentation effort compared to traditional map-based approach. Moreover, physics-based (PB) models embedded in FF controllers or PB model-embedded control are more robust than map-based approach in extrapolating beyond the tested operation envelope. Although, feedback control can compensate for certain disturbances, it offers limited compensation to all the uncertainties in the real-world. MPC and economic MPC computes control inputs on-line with preview and show potential to reduce calibration effort, however, these methods are computationally expensive due to requirement of large computation power and memory to store engine model.

Parametrization of the embedded maps and models in controllers using Supervised Learning (SL) methods have potential to reduce the calibration effort (Garg et al., 2021). SL methods have been extensively studied for engine modeling and can accurately model complex engine processes (Aliramezani et al., 2022; Garg et al., 2021). With SL methods, a significantly small number of parameters i.e., hyperparameters are required to be calibrated by an expert, while the model parameters are learned using numerical optimization. Especially time-series SL methods, for example, Long Short-Term Memory (LSTM) neural network can capture time-dependent input–output relationship, which makes these neural networks suitable for transient control. Fig. 1 illustrates the potential reduction in calibration effort using a LSTM neural network for parametrizing the FF controller without preview.

Recent studies have applied SL methods to parametrize control policy for implementation on the engine control unit (ECU). MPC control policy is parametrized using multi-layer perceptron (MLP) (Moriyasu et al., 2019; Peng et al., 2023) and LSTM neural network (Norouzi et al., 2022; Peng et al., 2023) for diesel engine control. These studies, however, have only focused on the ML model accuracy and associated control system performance. The hyperparameter settings have a significant impact the ML model accuracy, training times, inference time and memory requirements, therefore, it is important to optimize their settings for efficient and optimal control development (Goodfellow et al., 2016). In our previous work, we applied two SL methods i.e., support vector machines and neural networks to parametrize the feedforward controller for steady-state engine control (Sastry V M et al., 2022). The impact of model choice on the model inaccuracy, control system performance, number of calibration parameters, inference time and memory was studied. However, the study was limited to fixed model capacity of both SL models and steady-state engine control. Moreover, we did not evaluate the calibration effort of using SL-based controllers.

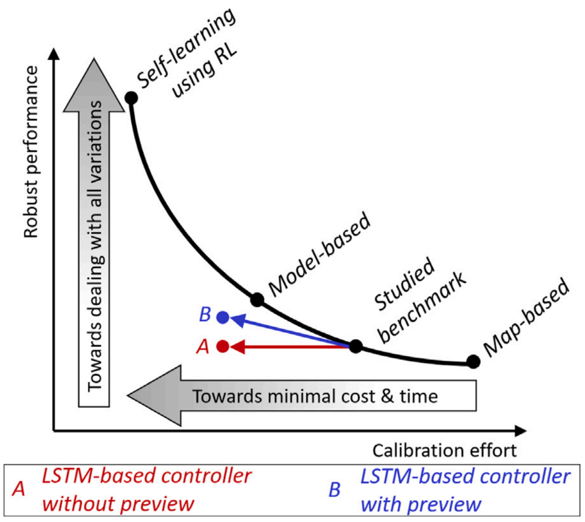
Self-learning using RL
Robust performance
Towards dealing with all variations
Model-based
Studied benchmark
Map-based
B
A
Towards minimal cost & time
Calibration effort
A LSTM-based controller without preview
B LSTM-based controller with preview

Fig. 1. Illustration of calibration effort and robust performance for different calibration approaches and potential benefit of LSTM-based controllers.

Therefore, an approach is not yet well established that optimizes for all above mentioned performance metrics for calibrating ML-based engine control. Self-learning powertrains show potential to further reduce the calibration effort and achieve robust performance by automating the calibration process and adapting control settings on-line as illustrated in Fig. 1. Reinforcement Learning is identified as a potential solution for developing self-learning controls (Garg et al., 2021).

Further improvement in transient performance of engines can be achieved using preview control, which can compensate for delays in the system response due to its dynamics, using future knowledge about changing references or disturbances. Preview control has been extensively studied in the past few decades for wide range of robotics (Kanzaki et al., 2005) and automotive applications (Yu et al., 2015). Several studies have investigated MPC for preview control of engines (Norouzi et al., 2021), however, it is difficult to implement on-line due to its large computational complexity. Therefore, there is a need for a method that can accurately implement the preview control policy with minimal calibration effort.

In this paper, a calibration approach for non-preview and preview transient control of diesel engine air-path system using LSTM neural network-based controllers is proposed. Fig. 1 illustrates the potential benefit in calibration effort and robustness using LSTM neural network-based controllers. Table 1 presents a relative comparison of the studied benchmark controller (Moergastel et al., 2019) with existing applications of model-based control calibration and potential outputs with LSTM-based control. The controller is designed by solving a multi-objective optimization problem. The novel contributions of this work are the following:

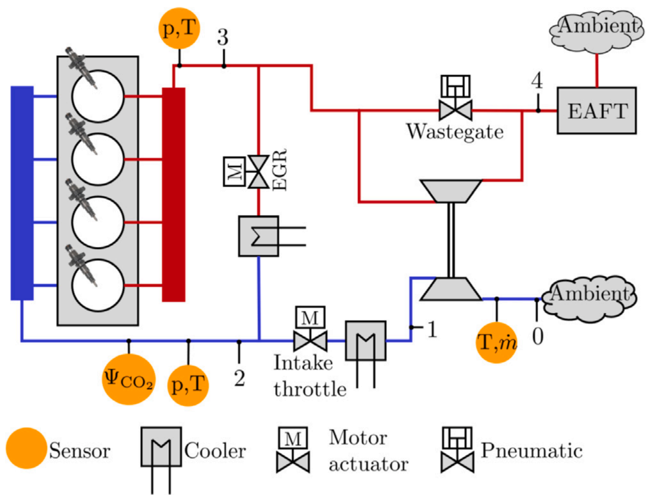

Fig. 2. Schematic of the studied Euro-6 engine layout. Used symbols: EAFT is exhaust after-treatment, $\dot{m}$ is mass flow rate, T is temperature, p is pressure and Ψ is concentration.

1. A time-efficient strategy to generate preview control signals for diesel engine air-path system, where the reference trajectories for non-preview control are shifted in time to generate feedforward control actions with preview;
2. A systematic hyperparameter selection method in ML-based controllers for non-preview and preview transient control of diesel engine air-path by solving a multi-objective optimization problem.

This work is organized as follows. describes the engine specifications and the diesel engine air-path control problem. Section 3 presents the proposed calibration process for non-preview and preview transient control of diesel engines. In Section 4, the calibration problem is defined. Section 5 presents the calibration results. The conclusions and future work are drawn in Section 6.

2. Engine controller calibration

In the control development process, controller calibration concentrates on determining the parameters in the designed controller, such that the desired system performance is realized. In this section, first the engine control problem is introduced for the studied engine. Second, the benchmark controller and its associated calibration process are presented.

2.1. Engine control problem

Fig. 2 shows a schematic of the studied Euro-6 diesel engine. This engine is equipped with a common rail fuel injection system, a Exhaust Gas Recirculation (EGR) system and a turbocharger with wastegate. The EGR system dilutes the intake air with residual gases from the combustion process in order to reduce engine-out NO_x emissions. The turbocharger wastegate controls the intake manifold pressure, which is required to produce the desired engine torque. Engine specifications are summarized in Table 2.

The high-level objective of the diesel engine control system is to realize the driver's torque demand with minimal fuel consumption while satisfying constraints related to emissions and safety. In the studied engine, the following control inputs $\mathbf{u}$ are available:

- Fuel path: start of injection u_{SOI}, duration of injection u_{DOI}, and rail pressure $u_{p_{rail}}$;
- Air path: wastegate valve u_{wg}, EGR valve u_{egr} and intake throttle valve u_{th};

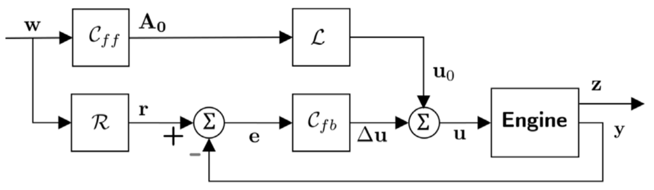

Fig. 3. Schematic of the benchmark air-path control system. $\mathbf{w}$ is a vector of external inputs i.e., engine torque and engine speed demand , $\mathbf{r}$ is reference signals, $\mathbf{e}$ is the tracking error, $\mathbf{A}_0$ is the nominal feedforward controller action for the valve opening areas, $\mathcal{L}$ is the linearization block, which translates valve opening area to valve open position, C_{ff} is the feedforward controller, C_{fb} is the feedforward controller, $\mathbf{u}_0$ is the nominal feedforward action for valve open position, $\Delta\mathbf{u}$ is the feedback controller action, $\mathbf{u}$ is the total control action, $\mathbf{z}$ is the performance output and $\mathbf{y}$ are measured signals.

Table 2
Engine specifications.

Parameter	Value
Engine type	Direct injection diesel engine
Displacement volume [l]	2.0
Number of cylinders	Inline 4
Compression ratio	15.8
Air-path subsystem	- Turbocharger with wastegate - Charge air cooling - High pressure, cooled EGR
Fuel path subsystem	Common rail direct injection

The driver's engine torque demand $M_{e,dem}$ and engine speed N_e are the considered external inputs $\mathbf{w}$.

2.2. Benchmark air path controller

To realize the high-level control objectives under different operating conditions, precise control of the fuel and air paths is required. In this study, we focus on an air path control using $\mathbf{u} = [u_{wg}\ u_{egr}\ u_{th}]^T$. Due to the coupled gas dynamics of intake and exhaust manifolds, this is a challenging multi-variable control problem, which also requires significant calibration effort. The benchmark air-path controller is shown in Fig. 3. This modern production-type controller is based on a classical control system design, which combines a feedforward controller C_{ff}, PID-based feedback controller C_{fb} and reference generator $\mathcal{R}$ (Moergastel et al., 2019). Here, the feedforward controller is based on a physics-based dynamic turbocharger model. The benchmark controller has two control modes:

1. p_2 **mode** for improved torque response at larger engine loads. In this mode, boost pressure r_{p_2} and oxygen concentration in the intake manifold $r_{\Psi_{O_2,2}}$ are controlled by the wastegate and EGR valves;
2. dp **mode** for controlling NO_x at low engine loads. In this mode, pressure difference between intake and exhaust manifold r_{dp} and $r_{\Psi_{O_2,2}}$ are controlled by wastegate and EGR valves.

The reference generator $\mathcal{R}$ is mainly based on static engine maps to determine $\mathbf{r}(\mathbf{w}) = [r_{p_2}\ r_{dp}\ r_{\Psi_{O_2,2}}]^T$. However, for the intake oxygen concentration, a dynamic compensation is added to the map value. This dynamic compensation is a first-order filter derived by a physics-based model. This is done to improve torque response during fast transients. The third air path actuator i.e., intake throttle, is used to ensure positive pressure differential between exhaust and intake manifold for EGR flow. The intake throttle is actuated if a negative pressure difference is observed.

2.3. Benchmark calibration process

The calibration process for the benchmark air path controller is illustrated in Fig. 4. It begins after the completion of the control concept

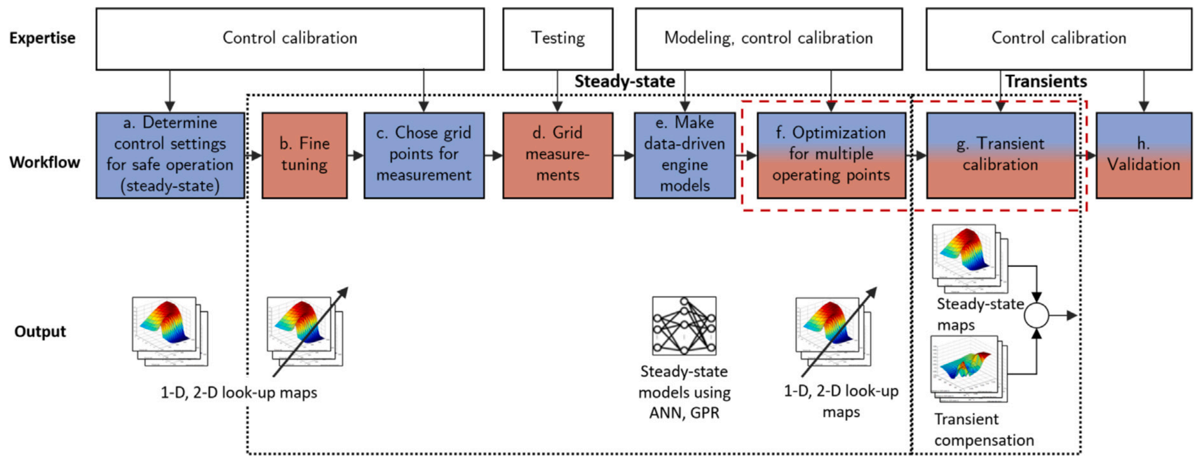

Fig. 4. Workflow of the benchmark calibration process for steady-state and transient engine operation. Dashed block indicates the steps in the process, which takes the major fraction of calibration effort. Steps in blue are performed in off-line environment (i.e., at desk), steps in red are performed in on-line environment (i.e., at test bench) and steps in blue-red indicates both off-line and on-line environment. Arrow indicates that the parameters in the map are updated. 1-D is one-dimensional, ANN is artificial neural network, GPR is Gaussian process regression, MiL is model-in-the-loop, SiL is software-in-the-loop, HiL is hardware-in-the-loop and RCP is rapid control prototyping. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

definition and control system design phases in the V-development cycle for engine control development. Firstly, for steady-state engine operation, base settings of control parameters for map-based C_{ff} and $\mathcal{R}$ are determined by a calibration engineer using physics in an off-line environment (a). These settings are then fine-tuned on an engine test-bench followed by a design of experiments (c) and measurements on an engine test-bench (d). Based on these measurements, a steady-state, data-driven engine model is generated using Supervised Learning (SL) methods. This model is used to calibrate off-line the reference maps $\mathbf{r}$ in step (e). This concludes the calibration of the steady-state control functions. For transient performance, the physics-based model in the C_{ff} is then calibrated using data from turbocharger tests (f). Detailed information on the state-of-the-art, model-based engine calibration process can be found in Isermann (2014). Transient compensation in $\mathcal{R}$ is calibrated in simulation and further fine-tuned on engine test-bench. The identified workflow steps with most effort are: *optimization for multiple operating points* and *transient calibration*, indicated by a red box in Fig. 4. More precisely, these two steps require 58% of the total experimentation and expert time for the benchmark process.

3. Method for calibration effort reduction

In this section, a new approach using ML-based controller is proposed. Supervised Learning (SL) can be used to parametrize the embedded maps and models in benchmark controllers. This choice of controller design can reduce the expert effort and experimentation time in the steps f and g of the calibration process shown in Fig. 4. Multiple maps can be substituted by a SL model, such that a significantly smaller number of parameters i.e., hyperparameters need to be tuned by an expert for steady-state and transient control, which reduces the expert effort. Moreover, turbocharger experiments used to tune transient compensation in physics-based feedforward controller are not required, which reduces the experimentation time. Besides the standard (non-preview) controller, a preview-based controller is introduced, such that not only the calibration effort is reduced, but also system performance is enhanced, as illustrated in Fig. 1.

3.1. Impact on calibration process

The proposed calibration process using ML-based controller is shown in Fig. 5. Contrary to measurements of steady-state engine operation, components and transient calibration in the benchmark case, dynamic engine measurements (d) are done in the new calibration process. This modification eliminates the experiments required for calibration of control parameters. Dynamic engine measurements can be made by simultaneously perturbing the available actuators, for example, using a Amplitude-modulated Pseudo Random Binary Signal (APRBS) at different operating points. This requires less time compared to steady-state grid measurements (Isermann, 2014). This data can used to develop dynamic engine models for emissions, mass flow rates and pressures in the intake and exhaust manifolds. However, in this work, a mean-value engine model is used as a dynamic model for method development.

For non-preview control, these dynamic engine models are used in an off-line optimization framework on a personal computer to determine optimal reference trajectories $\mathbf{r}$ and corresponding $\mathbf{A_0}$ for acceleration steps using methods such as, dynamic programming and model predictive control. Using the generated data, the input–output correlations are then captured using a ML model.

3.2. LSTM-based air-path controller

Fig. 6 shows the proposed control architecture with the new machine learning (ML)-based controller $\mathcal{M}_{wp}$.

In order to learn temporal relationship between inputs containing history and preview information, a ML model that can describe time-dependencies is chosen. Recurrent neural networks (RNN) can learn these dependencies, however, they suffer from vanishing or exploding gradients for which either the weights are oscillating or learning takes very long time for extended time intervals (Hochreiter & Schmidhuber, 1997). Long short-term memory (LSTM) neural networks are a gated RNN and do not suffer from limitations of generic RNN. Through use of gates, LSTM can learn long-term dependencies and are suitable for modeling dynamic engine control policies and reference trajectories.

3.3. Preview control

Currently, the air path dynamics are limiting transient engine performance. In a simulation study, the benefit of preview control for torque response, fuel consumption, and NOx emission was examined within the scope of the existing control framework; off-line numerical optimization was not considered in this phase. Alternatively, a pragmatic approach was followed at the cost of optimality.

In order to realize preview control, the reference signals for the air-path control system for the non-preview scenario are shifted in time using an exhaustive search method in a simulation environment.

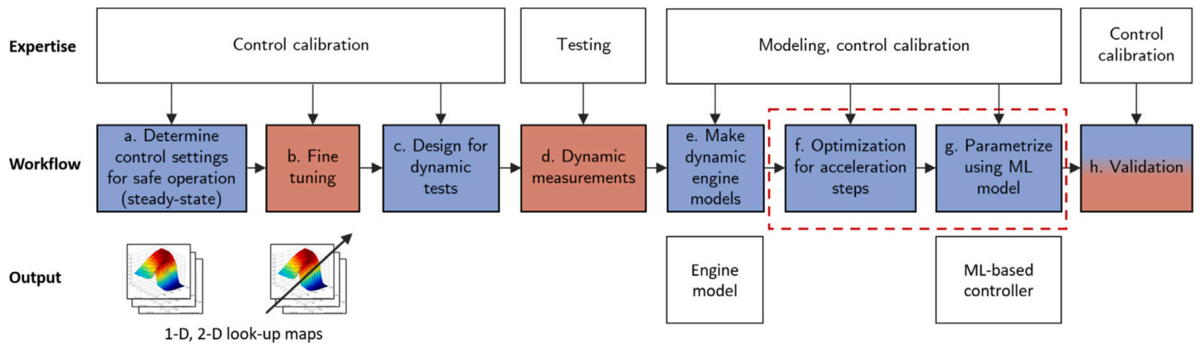
Expertise
Control calibration
Testing
Modeling, control calibration
Control calibration
Workflow
a. Determine control settings for safe operation (steady-state)
b. Fine tuning
c. Design for dynamic tests
d. Dynamic measurements
e. Make dynamic engine models
f. Optimization for acceleration steps
g. Parametrize using ML model
h. Validation
Output
1-D, 2-D look-up maps
Engine model
ML-based controller

Fig. 5. Workflow of the proposed calibration process for steady-state and transient engine operation. Dashed block indicates the scope of this work. Steps in blue are performed in off-line environment (i.e., at desk), steps in red are performed in on-line environment (i.e., at test bench) and steps in blue-red indicates both off-line and on-line environment. Arrow indicates that the parameters in the map are updated. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

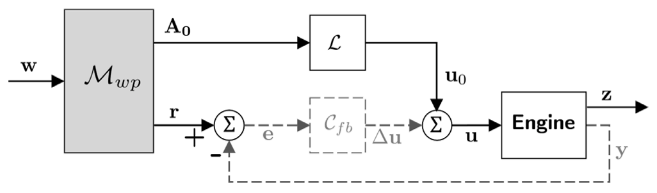

Fig. 6. Schematic of ML-based air-path control system. Dashed blocks and lines represent feedback control loop, which is made passive for this study.

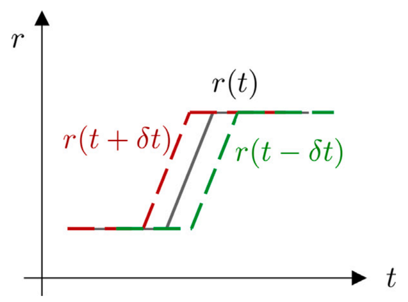

Fig. 7. Illustration of the time shifts in the reference trajectory r. Signal retarding is shown by green dashed lines and advancing is shown in red dashed lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

To determine the best reference signals with preview, the original reference signals are either advanced or retarded by δt (in seconds), as illustrated in Fig. 7. Various signal shifts are studied for three different engine torque steps and two different engine speeds. Details can be found in Appendix A. It is concluded that the best system performance is achieved by advancing the reference signals r_{p_2}, r_{dp} by 2 s and retarding the $r_{\Psi_{O_2},2}$ signal by 2 s for a medium torque demand. For the purpose of method development this signal shift strategy is adopted for all the acceleration and deceleration ramps in Section 4. The resulting reference signals and feedforward control actions are denoted by $\mathbf{r}'$ and $\mathbf{u}_0'$, respectively. This study demonstrates a proof of concept for improved engine performance with preview control. Accumulation of performance gains in individual transient over a driving cycle promises a significant real-world improvement in the system performance.

4. Multi-objective calibration of LSTM controller

This section discusses the followed calibration approach for the proposed LSTM neural network. Two different LSTM-based controllers are designed for non-preview and preview control. For non-preview control, the LSTM neural network provides an efficient parametrization of the control signals compared to the physics-based, computationally expensive benchmark feedforward controller. With preview control, the LSTM neural network can further improve the control system performance.

The main challenge in the calibration process is to find the best trade-off between model accuracy, system performance, calibration effort and computational requirements. This is not straightforward, since these are multiple competing objectives. In this section, a systematic method is presented to determine the optimal settings for the air path control parameters. First, the multi-objective optimization problem is presented, followed by a step-wise approach for hyper-parameter selection.

4.1. Problem definition

For the calibration of the proposed LSTM-based controllers, the multi-objective optimization problem is defined as,

$$\begin{aligned}&\min_{\Omega,\Theta} f(\mathbf{J},\mathbf{d}),\\ &s.t.\ \ q_i(\Omega,\Theta)\le 0,\ \ i=1,2,\ldots,q,\\ &\qquad s_j(\Omega,\Theta)=0,\ \ j=1,2,\ldots,r,\\ &\text{and}\ \ \Omega\in\mathcal{S}_1,\ \ \boldsymbol{\Theta}\in\mathcal{S}_2.\end{aligned}\tag{1}$$

where, $f(\mathbf{J},\mathbf{d})$ is defined as,

$$\begin{aligned}f(\mathbf{J},\mathbf{d}) = {}& d_1 J_v(\boldsymbol{\Omega},\boldsymbol{\Theta}) + d_2 J_{ce}(\boldsymbol{\Omega}) + d_3 J_p(\boldsymbol{\Omega},\boldsymbol{\Theta}) +\\ & d_4 J_{mc}(\boldsymbol{\Omega}) + d_5 J_{it}(\boldsymbol{\Omega}),\end{aligned}\tag{2}$$

Herein, $\mathbf{J}$ is a multi-objective, linearly scaled cost function and $\mathbf{d}$ is the vector of weights d_i used for scaling the cost terms such that, $d_i \in \mathbb{R}^+, \forall i = \{1,2,\ldots,5\}$. Scaling is necessary as the units and scales of the cost terms vary. The design variable $\boldsymbol{\Omega}$ is a vector consisting of hyperparameters, i.e., parameters that define the model capacity, training framework and dataset processing for optimization. Functional constraints are denoted by q, s and $\mathcal{S}_1$, $\mathcal{S}_2$ are the set constraints. The best values for $\boldsymbol{\Omega}$ and corresponding LSTM model parameters Θ i.e., weights and biases, are determined by solving Eq. (1).

Model accuracy. J_v is a neural network loss function given by,

$$J_v(\boldsymbol{\Omega},\boldsymbol{\Theta}) = \frac{1}{n_s}\sum_{i=1}^{n_s} L_i(\hat{\mathbf{y}},\mathbf{y})\tag{3}$$

where $\hat{\mathbf{y}}$ are the ML model predictions and $\mathbf{y}$ are the true output values. L_i is the mean square error for n_o number of outputs and ith data sample expressed as,

$$L_i(\hat{\mathbf{y}},\mathbf{y}) = \frac{1}{n_o}\sum_{k=1}^{n_o} c_k(\hat{y}_{k,i}-y_{k,i})^2\tag{4}$$

$J_v(\boldsymbol{\Omega},\boldsymbol{\Theta})$ is the average value of L_i for n_{val} number of validation samples. c_k is the weight of kth output in the total loss calculation. Here,

$c_k = 1$ is used, which means equal weightage is given to individual outputs.

Calibration effort. J_{ce} is the calibration effort, which is a summation of development time required at each step of the process shown in Fig. 5. It is calculated by aggregating expert effort for different tasks in the calibration process t_{expert}, time required for experimentation on the test benches $t_{\text{experiment}}$, computation time for engine modeling and ML model training on the personal computer $t_{\text{computation}}$, and validation in simulation and on test benches. This work focuses on two steps that takes the major fraction of the calibration effort in Fig. 5: (f) Optimization for acceleration steps and (g) Parametrize using RNN-LSTM neural network. Therefore, for these steps, a comparison is made between the benchmark and the LSTM-based controller. It is expressed as,

$$J_{ce} = (t_{\text{expert}} + t_{\text{experiment}} + t_{\text{computation}})|_{\text{steps - f,g}} \tag{5}$$

System performance. J_p is the system performance metrics, which is defined as,

$$J_p(\boldsymbol{\Omega}, \boldsymbol{\Theta}) = l_1 m_{fuel} + l_2 m_{NO_x}, \tag{6}$$

where $l_1, l_2 \in \mathbb{R}^+$ are scaling weights. m_{fuel} is the cumulative fuel consumption (in g), m_{NO_x} is the cumulative engine out nitrogen oxide emission (in g) over a specific driving cycle.

Computational requirements. In this work, we focus on memory complexity and inference time Inference time is chosen as it has an impact on control system performance, for example, a delayed output from C_{ff} would deteriorate transient performance. A neural network with a large memory complexity would require larger read-only-memory and incur more costs for implementation. J_{mc} is the memory complexity of the controller. For the benchmark controller, it is the number of calibration parameters such as, values in look-up tables and physical constants. For a ML model, it is the number of model parameters i.e., weights and biases. It is calculated as,

$$J_{mc} = |\Theta| \tag{7}$$

where Θ represents the calibration parameters in the controller and $|\Theta|$ is number of elements in Θ. J_{it} is the inference time required by the controller to make output prediction for a given input vector at a time instant on the ECU. For the benchmark controller, it is 8 ms. For the ML model, it is measured on the laptop by taking an average of inference time values for each data sample of the validation dataset.[1] It is expressed as,

$$J_{it,ML} = \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \Delta t_i \tag{8}$$

where Δt_i is the inference time for ith data sample.

4.2. ML model calibration process

In this section, the calibration of the ML model-based controllers for non-preview and preview transient control is described in detail. The main steps of the process are shown in Fig. 8. The focus is on the open loop preview control of the air-path system. To focus on methodology development and save time in creating design of experiments for large dimensional parameters space, engine speed demand $N_{e,dem}$ is kept constant at 1700 rpm and just engine torque demand is varied.

[1] All computations are performed on a standard laptop with 16 GB of RAM, Intel i7-9750 processor running at 2.60 GHz. The ML model is optimized in Python 3.8.8 using keras version 2.7 with Tensorflow 2 backend.

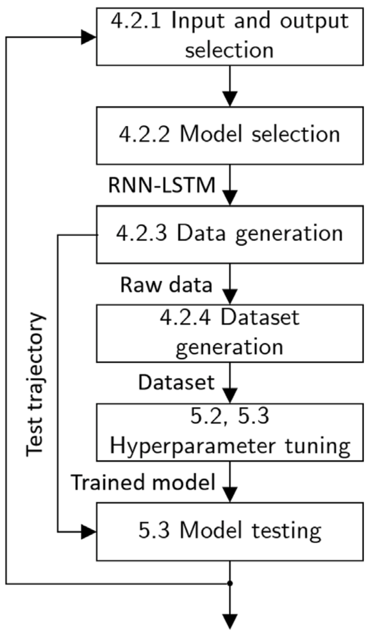

Fig. 8. Main steps in calibration of a ML model-based controller. Numbers refer to the subsection numbers, where each step is presented in detail.

4.2.1. Input and output selection

The proposed control scheme is shown in Fig. 6. The benefit of this control design is that a single model can be used to model nominal control settings for reference and feedforward control actions. The prediction model $\mathcal{M}_{wp}$ is defined as,

$$\mathcal{M}_{wp} : \mathbf{w}(t_{now} - t_p, t_{now} + t_f) \rightarrow \begin{bmatrix} \mathbf{A}_0(t_{now+1}) \\ \mathbf{r}(t_{now+1}) \end{bmatrix} \tag{9}$$

where,

$$\mathbf{w} = \begin{bmatrix} M_{e,dem} & \nabla M_{e,dem} \end{bmatrix}^\top \tag{10}$$

$\mathbf{w}$ is a time sequence consisting of values t_p seconds in past, t_f seconds in future and t_{now} is the current time instant. $\nabla M_{e,dem}$ is the rate of change in the engine torque demand. This choice of inputs allows to capture all steady-state and transient torque demands at a constant engine speed. The model outputs are the nominal opening areas of the wastegate valve $A_{wg,0}$ and the EGR valve $A_{egr,0}$, and references $\mathbf{r} = \begin{bmatrix} r_{p_2} & r_{dp} & r_{\Psi_{O_2},2} \end{bmatrix}^\top$ for next time instant t_{now+1}. One step ahead prediction is made because it is sufficient to implement this controller in the control system. The valve opening areas $\mathbf{A}_0 = \begin{bmatrix} A_{wg,0} & A_{egr,0} \end{bmatrix}^\top$ are translated using a linearization map $\mathcal{L}$ to determine the valve opening positions $\mathbf{u}_0$ between 0 and 100%. Based on the control mode, the reference r_{p_2} or r_{dp} are input to the control system for feedback control. It is assumed that $t_f = 2$ seconds as no benefit of increasing preview length on system performance was observed in the study presented in Section 3. $t_p = 2$ seconds is chosen such that it minimizes computation time at a small cost in model prediction accuracy from a sensitivity study, see Appendix C for results.

4.2.2. Model selection

LSTM neural network is chosen to realize the prediction model $\mathcal{M}_{wp}$ as discussed earlier in Section 3.2. The output of a single LSTM cell i at time instant t is described by the forward propagation equations (Goodfellow et al., 2016):

$$h_i(t) = \tanh\Big(s_i(t)\Big) q_i(t) \tag{11}$$

with,

$$s_i(t) = f_i(t)\, s_i(t-1) + g_i(t) \tanh\left[b_i + \sum_{m=1}^{d} u_{i,m} w_m(t) + \sum_{j=1}^{n} v_{i,j} h_j(t-1) \right] \quad (12)$$

$$q_i(t) = \sigma\left[b_i^q + \sum_{m=1}^{d} u_{i,m}^q w_m(t) + \sum_{j=1}^{n} v_{i,j}^q h_j(t-1) \right] \quad (13)$$

$$g_i(t) = \sigma\left[b_i^g + \sum_{m=1}^{d} u_{i,m}^g w_m(t) + \sum_{j=1}^{n} v_{i,j}^g h_j(t-1) \right] \quad (14)$$

$$f_i(t) = \sigma\left[b_i^f + \sum_{m=1}^{d} u_{i,m}^f w_m(t) + \sum_{j=1}^{n} v_{i,j}^f h_j(t-1) \right] \quad (15)$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (16)$$

where σ is a sigmoid activation function; $j = \{1, 2, \dots, n\}$ are the indexes of memory cells in the layer; m is the input dimension; f, g and q are outputs of the forget, input and output gates, respectively. The model parameter vector, which is expressed as,

$$\Theta = [\mathbf{b}\, U\, \mathcal{V}]^T \quad (17)$$

consists of bias vector $\mathbf{b} = \left[b^f \;\; b^g \;\; b^q \right]^\top$; weights matrix $U = \left[\boldsymbol{u}^f \;\; \boldsymbol{u}^g \;\; \boldsymbol{u}^q \right]^\top$ on connection from inputs to the forget, input and output gates; weights matrix $\mathcal{V} = \left[\boldsymbol{V}^f \;\; \boldsymbol{V}^g \;\; \boldsymbol{V}^q \right]^\top$ on connection from hidden state at $(t-1)$ to the forget, input and output gates; and recurring self-connections weights vector $\boldsymbol{u}$ and $\boldsymbol{V}$.

Without preview information: Given the input at current time instant t_{now} and past information for t_p seconds i.e., $\boldsymbol{w}(t_{now} - t_p, t_{now})$, the prediction of control actions and references are made for next time instant i.e., $\boldsymbol{y}(t_{now+1})$. The kth model output $y_k(t_{now+1})$ is given by,

$$\begin{aligned} \hat{y}_k(t_{now+1}) &= \mathcal{M}_{wp}(\mathbf{w}(t_{now} - t_p, t_{now}); \boldsymbol{\Theta}, \boldsymbol{\Omega}) \\ &= b_k^d\; h_0 + \sum_{i=1}^{n_{cells}} v_{k,i}^{d\,\top} h_i(t_{now}) \end{aligned} \quad (18)$$

where $\mathbf{w}$ is the vector of inputs, $\boldsymbol{\Theta}$ represents all model parameters i.e., weights and biases, and $\boldsymbol{\Omega}$ is the hyperparameters setting .

With preview information: Given the input preview for future t_f seconds and past information for t_p seconds i.e., $\boldsymbol{w}(t_{now} - t_p, t_{now} + t_f)$, the prediction of control actions and references are made for next time instant i.e., $\boldsymbol{y}(t_{now+1})$. The kth model output $y_k(t_{now+1})$ is given by,

$$\begin{aligned} \hat{y}_k(t_{now+1}) &= \mathcal{M}_{wp}(\mathbf{w}(t_{now} - t_p, t_{now} + t_f); \boldsymbol{\Theta}, \boldsymbol{\Omega}) \\ &= b_k^d\; h_0 + \sum_{i=1}^{n_{cells}} v_{k,i}^{d\,\top} h_i(t_{now} + t_f) \end{aligned} \quad (19)$$

4.2.3. Data generation

To build a ML model, data is generated from the Simulink simulation environment, which consists of calibrated 0D mean value engine model and the control system. To generate dynamic data, varying torque demand ramps at a constant engine speed are input to the simulation model. The torque ramps are designed by varying initial torque demand $M_{e,dem,0}$, duration of torque demand ramp Δs and the change in torque demand $\Delta M_{e,dem}$ as illustrated in Fig. 9. Fig. 10 shows the range of design parameters which results in total of 84 torque demand ramps. The torque transient are designed for dp mode of air-path control system, which limits the maximum torque demand to 210 Nm. $M_{e,dem,0}$ is varied between 0 Nm and 200 Nm with $\Delta M_{e,dem} \in \{10, 50, 100, 150\}$ Nm. Different transient are designed by varying $\Delta s \in \{0.6, 1, 2, 3, 4, 5, 6\}$ seconds. These values are derived from the engine torque demand of a Worldwide harmonized Light vehicles Test Procedure (WLTP). In order to design a fast transient, the minimum value of $\Delta s = 0.6$ seconds is chosen, which is a minimum value that is input to the air-path control system on the real engine (Moergastel et al., 2019). The maximum value of $\Delta s = 6$ seconds is selected by analyzing the reference tracking performance of the air-path control system. It was seen that the controller tracks the reference closely for $\Delta s \geq 6$ seconds, where the engine is assumed to be in quasi-stationary state.

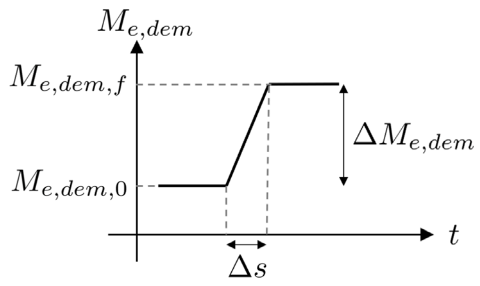

Fig. 9. Illustration of a torque demand ramp.

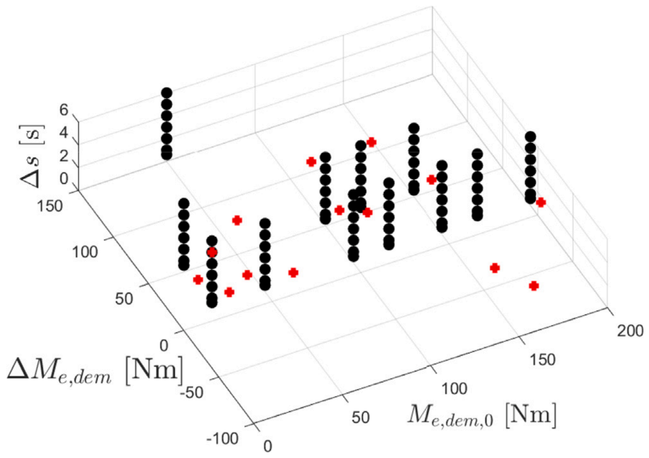

Fig. 10. Design space of parameters for design of experiments of different torque demand ramps. Black circles are for generating training data and red squares for the test cycle. The value of parameters in red square are determined such that a continuous transient test cycle is generated as shown in Fig. 11.

4.2.4. Dataset generation

The torque ramps generated in the previous subsection are processed to generate the input dataset for multi-input LSTM recurrent network, which is a three dimensional (3D) tensor where the first dimension is the window length, second dimension is number of windows or data samples and third dimension is the number of inputs as shown in Fig. 11. A single window is a 2D tensor consisting of data points in the window for individual input. Multiple data samples are generated from each torque ramp by a moving window at a sampling time $T_{s,2}$, which determines the total number of windows. Small value of $T_{s,2}$ will increase the dataset size, which would increase the model training and computation time due to increased number of computations. The sampling time $T_{s,1}$ determines the number of data points in a window, therefore, a smaller value increases the length of the window. The value of $T_{s,1}$ is chosen based on largest sampling time of the output signal which is 0.02s for $r_{\Psi_{O_2,2}}$. $T_{s,2} = 0.1$ seconds is determined from a sensitivity study described in Appendix B. The resulting dataset size for above mentioned sampling times is 6372 data samples. The window length is 201 for $t_p = 2$ and $t_f = 2$ seconds sampled at $T_{s,1} = 0.02$ seconds. Similarly, the output dataset is generated by sampling data point at time instant t_{now+1} for individual output As a result, the output dataset is a 2D tensor where first dimension is outputs and the second dimension is data sample index illustrated in Fig. 11.

The dataset generated here is used in the neural network model training process. Before model training, the dataset is divided into two different sets with 70% of the dataset for training and 30% for validation.[2] Training set is used to learn the model parameters via

[2] Typical dataset splits are 70%–80% for training and 30%–20% for validation (Goodfellow et al., 2016).

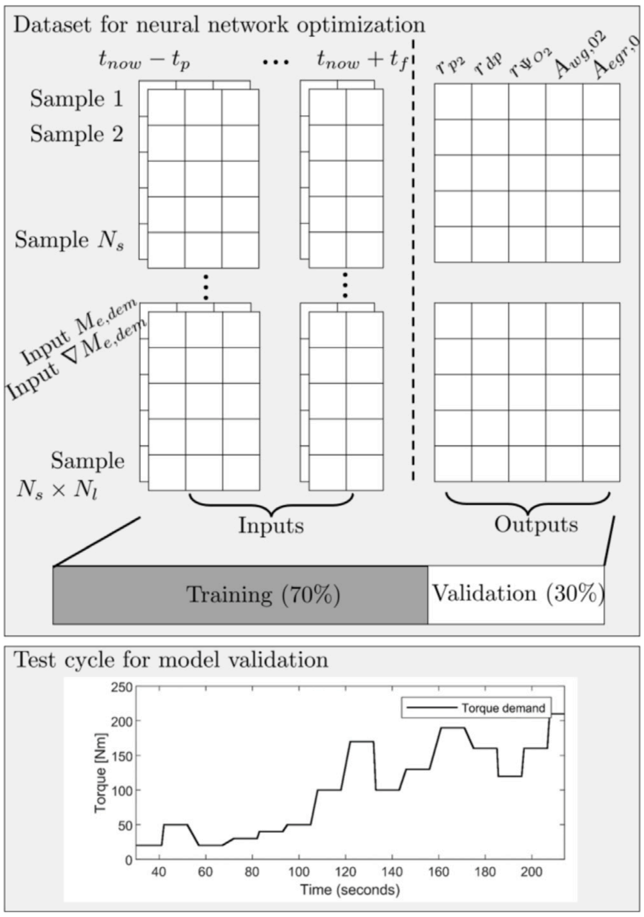

Fig. 11. Overview of datasets for neural network training and testing. Inputs and outputs datasets are 3D and 2D tensor, respectively. N_s is the number of windows per torque demand ramp and $N_l = 84$ is the number of ramps generated from DOE points in black circles shown in Fig. 10. The total number of data samples $N_s \times N_l$. Figure below shows the test cycle generated using DOE points in red squares shown in Fig. 10.

Table 3
Hyperparameter settings for dataset generation.

Category	Hyperparameter	Value
Dataset	Dataset split	Training (70%) and validation (30%)
	Training data points size	4460
	Validation data points size	1912
	Input scaling	Scaled to $[0,1]$
	Output scaling	Scaled to $[0,1]$

optimization while validation set guides the selection of the hyperparameters. This is done to prevent model overfitting to the data i.e., the model has smaller error on the training data while larger error on the test data, also called as poor model generalizability. Furthermore, the scales and units of the inputs and outputs are significantly different, therefore, the input and output datasets are scaled to $[0,1]$, also called as min–max scaling such that equal weightage is allotted to individual output in optimization. The resulting hyperparameter settings for dataset are summarized in Table 3.

5. Calibration results

In the previous section, the multi-objective optimization problem has been defined, followed by the ML model structure and data processing. This section defines the cost function and constraints of the multi-objective optimization problem. This optimization problem is solved by bi-level optimization approach, where a subset of hyperparameters $\Omega_1 \subset \boldsymbol{\Omega}$ are determined first using a exhaustive search method. Ω_1 are then fixed and remaining hyperparameters $\Omega_2 = \boldsymbol{\Omega} \setminus \Omega_1$ are then determined using similar method. A multi-objective analysis is presented in order to chose the best LSTM model capacity for ECU implementation.

5.1. Bi-level optimization for hyperparameter tuning

For LSTM neural network design, $\boldsymbol{\Omega} = \left[\eta\ n_e\ n_{mb}\ n_p\ n_{runs}\ n_l\ n_c\right]^\top$, where η is the learning rate, n_e is the number of epochs beyond which ML model training is halted, n_{mb} is the size of a minibatch i.e., a subset of the total training dataset, n_p is the patience value defined as the number of epochs of the neural network optimization to be monitored after a local minima in the validation loss curve, n_{runs} is the number of optimization runs of model training with different initial weights, n_l is the number of LSTM layers in the neural network and n_c is the number of LSTM cells in each LSTM layer. The multi-objective optimization problem is a mixed-integer programming problem defined as,

$$\begin{aligned} \min_{\boldsymbol{\Omega},\boldsymbol{\Theta}} \ & d_1 J_v(\boldsymbol{\Omega},\boldsymbol{\Theta}) + d_2 J_{ce}(\boldsymbol{\Omega}) + d_3 J_p(\boldsymbol{\Omega},\boldsymbol{\Theta}) + \\ & d_4 J_{mc}(\boldsymbol{\Omega}) + d_5 J_{it}(\boldsymbol{\Omega}), \\ s.t.\ \ & \boldsymbol{d} = \left[10^5\ \ 1\ \ 10^{-1}\ \ 10^{-3}\ \ 10\right]^\top \\ & 1 \le n_e,\ n_{mb},\ n_r \\ & n_{mb} - n_e \le 0, \\ & n_p \in \{10, 20, \dots, 50\}, \\ & n_l \in \{1, 2, 3, 4\}, \\ & n_c \in \{2^0, 2^1, \dots, 2^5\}, \end{aligned} \tag{20}$$

and $\Omega \setminus \{\eta\} \in \mathcal{S}_1 \subseteq \mathbb{Z}^+,\ \boldsymbol{\Theta} \in \mathbb{R},\ \eta \in \mathcal{S}_2 \subseteq \mathbb{R}^+$.

The weights $d_1, d_2, \dots, d_5$ are chosen such that each cost term has order of magnitude one. The cost terms in J_p are scaled with weights $l_1 = 1$ and $l_2 = 10^2$. J_p is expressed as,

$$J_p(\boldsymbol{\Omega},\boldsymbol{\Theta}) = m_{fuel} + 10^2 m_{NO_x}, \tag{21}$$

In order to determine the suitable LSTM model capacity, hyperparameters that define the network training framework are fixed. This makes this control problem a bi-level optimization problem. More precisely, $\boldsymbol{\Omega}$ is partitioned into two sets as,

$$\begin{aligned} \boldsymbol{\Omega} &= \left[\ \Omega_1 \ \middle|\ \Omega_2\ \right]^\top \\ &= \left[\ \eta \quad n_e \quad n_{mb} \quad n_p \quad n_{runs} \ \middle|\ n_l \quad n_c\ \right]^\top \end{aligned} \tag{22}$$

Ω_1 impacts the J_v and J_{ce}, therefore, the level-1 optimization is defined as,

$$\begin{aligned} \min_{\Omega_1} \ & J_v(\Omega_1, \Omega_2, \boldsymbol{\Theta}) + J_{ce}(\Omega_1, \Omega_2) \\ s.t.\ & \text{constraints in (20)} \end{aligned} \tag{23}$$

Ω_2 define the LSTM model capacity and impacts the J_p, J_{mc} and J_{it}, therefore it is determined in level-2 optimization defined as,

$$\begin{aligned} \min_{\Omega_2,\boldsymbol{\Theta}} \ & J_v(\Omega_1, \Omega_2, \boldsymbol{\Theta}) + J_{ce}(\Omega_1, \Omega_2) + \\ & J_p(\Omega_2, \boldsymbol{\Theta}) + J_{mc}(\Omega_2) + J_{it}(\Omega_2) \\ s.t.\ & \text{constraints in (20)} \end{aligned} \tag{24}$$

This bi-level optimization problem to determine Ω_1, Ω_2 is solved using an exhaustive search approach in a manual manner. In the level-1 optimization, Ω_1 is determined, which is then kept fixed. Thereafter, Ω_2 is determined. In order to perform an exhaustive search, a range of values of Ω_1 and Ω_2 are studied and the best values that minimize cost functions in Eqs. (23) and (24) respectively are chosen. Alternative approach is to use automatic hyperparameter selection algorithms, however, they are computationally much costly (Goodfellow et al., 2016).

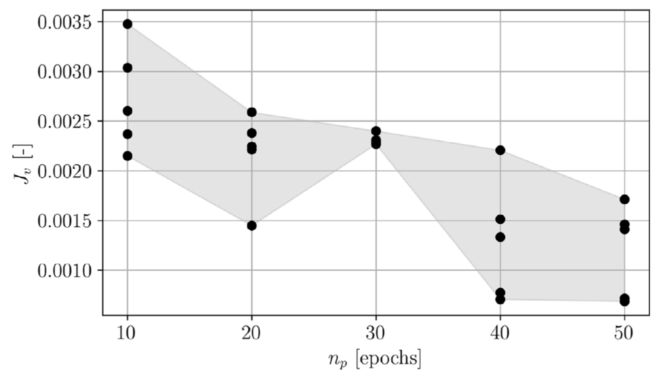

Fig. 12. Sensitivity study to determine n_p. Black dots correspond to 5 training runs with varying initialization. Other parameters fixed at $n_e = 500$, $n_l = 1$, $n_c = 64$, $n_{mb} = 100$, Adam optimizer with adaptive η.

5.2. Level-1 optimization results

The neural network training begins with initialization of its parameters Θ.[3] However, the choice of these initial weights largely affect the convergence of loss learning curve in case of deep neural network such as LSTM (see Goodfellow et al., 2016, Ch. 8). Moreover, numerical difficulties or non-convergence is observed with some choices of initial values within acceptable training time. Therefore, to avoid these challenges, multiple training runs with different initial weights are made. The effect of different initial weights on model loss is evidently seen in Fig. 12. Varying losses are seen for each run for constant n_e, n_p, n_l, n_c.

Early-stopping (ES) of the neural network model training process is a form of regularization that limits the effective model capacity by preventing overtraining of the network (Bishop, 1995; Sjöberg & Ljung, 1995). In ES, n_p number of epochs are searched after an existing local minima to find a better minima and training is halted if the existing minima is the best. The existence of noise in learning curves using minibatch optimization makes the training with ES prone to early halt or non-regularized based on n_p. Therefore, tuning of n_p is important as early halt of training occurs for small n_p or the training runs non-regularized. n_p is determined from a sensitivity study by studying its impact on J_v for fixed settings for other hyperparameters as shown in Fig. 12. Large J_v is seen for $n_p < 40$ while for $n_p = 40$ and 50, similar J_v is seen. However, J_{ce} is 6% larger with $n_p = 50$ compared to $n_p = 40$. Another approach for regularization is to limit number of epochs n_e for which training is performed. Model overfitting and underfitting occur for large and small values of both n_e, n_p, respectively. Therefore, following two cases are studied (see Appendix E for results):

Case 1: $n_r = 5, n_e = 500, n_p = 40, n_l = 1, n_c = \{2^5, 2^6\}$, Adam optimizer with adaptive η;

Case 2: $n_r = 1, n_e = 6000, n_p = 0, n_l = 1, n_c = \{2^5, 2^6\}$, Adam optimizer with adaptive η.

It is seen that there is marginal decrease in J_v at the cost of large training time. Therefore, in order to keep training time minimal and study wide range of model capacities, $n_r = 5, n_p = 40, n_e = 500$ are chosen. For large training datasets as in this work, minibatch stochastic method generally results in faster convergence than batch gradient methods that use the complete dataset to compute gradient (see Goodfellow et al., 2016, Ch. 8). This method uses minibatches of size n_{mb} from the total dataset for gradient calculation. For example, $n_{mb} = 1$ uses one sample at a time for gradient calculation and often achieves the best J_v at the cost of significantly large J_{ce}. Moreover, training with minibatches introduce noise in the learning curves due to gradient calculation over a small dataset, which is not necessarily in the direction of the global minima.

[3] Initial values of weights $U, \mathcal{V}, \boldsymbol{u}, \boldsymbol{V}$ and bias $\boldsymbol{b}$ are drawn from uniform distribution described in Glorot and Bengio (2010).

Table 4
Chosen values of Ω_1.

Parameter	Value
η	Adaptive (Adam optimizer)
n_e	500
n_{mb}	100
n_p	40
n_r	5

Gradient-based optimization methods are usually less sensitive to noise introduced due to minibatch optimization and can handle small batch sizes such as 100 (see Goodfellow et al., 2016, Ch. 8). In order to reduce the training time, a minibatch of size 100 is used in this work. Learning rate η has a significant impact on J_v, J_{ce}. Convergence of J_v is slow for large η, therefore increases J_v, J_{ce}. Smaller values of η show good convergence and reduction in J_v at cost of large J_{ce}. Therefore, in order to efficiently optimize neural networks, three commonly used optimization algorithms for neural networks training are compared, see Appendix D for results. It is found that adaptive learning rate algorithms achieve smaller loss faster than algorithms with fixed learning rate, therefore Adam with a default initial learning rate of 0.001 is used in this study. The chosen settings for Ω_1 are listed in Table 4.

5.3. Level-2 optimization results

Hyperparameters $\Omega_2 = \begin{bmatrix} n_l & n_c \end{bmatrix}^\top$ are determined in level-2 optimization by fixing Ω_1. Adding cells increase the number of input combinations, which increases the model capacity to capture complex non-linear patterns in a time-series input. n_l, n_c are varied as in Eq. (20). During neural network training, learning curves of two losses are of importance: (1) Training loss J_t and (2) Validation loss J_v. Firstly, neural networks with $n_l \in \{1, 2, 3, 4\}$ are trained and it is seen that adding layers bring limited benefit in J_v as seen in Fig. 13 A possible explanation for this is that the capacity of a single layer LSTM cells networks is sufficient to capture the non-linear patterns from inputs to outputs, therefore, adding layers bring limited benefit. This finding is consistent with those of other studies on LSTM recurrent network based controller modeling for diesel engine air-path (Peng et al., 2022) and rainfall-runoff modeling (Boulmaiz et al., 2020). Therefore, $n_l > 4$ are not studied in order to reduce J_{ce}. Secondly, n_c is increased by a factor of two starting with $n_c = 1$. It is seen that adding cells per layer reduces the both J_t, J_v while adding layers show limited effect. This similarity of trends in J_t, J_v could be the result of using validation dataset for early stopping that affects the model structure (Sjöberg & Ljung, 1995). The impact of adding LSTM cells on the prediction accuracy of individual model outputs was studied by visualizing individual output loss as shown in Fig. 14. The individual output's share towards the total validation loss can also be calculated from this figure. A small decrease in validation loss for outputs $r_{\Psi_{O_2,2}}$, $A_{wg,0}$, $A_{egr,0}$ is seen for number of cells larger than 2^2. Moreover, the change in losses for individual output is dissimilar and not necessarily decreasing (increase in loss for $r_{\Psi_{O_2,2}}$ for increasing cells from 2^1 to 2^2) for increasing number of cells. A possible explanation for these results is that the parameter update in gradient-based optimization is in direction of negative gradient of J_v and not the individual outputs loss function. Therefore, loss do not have to decrease for all outputs in order to reduce the overall cost function J_v.

The model architecture with minimal J_v is LSTM recurrent network with $n_l = 1, n_c = 32$, where $J_t = 8 \times 10^{-4}$ and $J_v = 7 \times 10^{-4}$. The mean square error (MSE) and mean absolute error (MAE) of individual

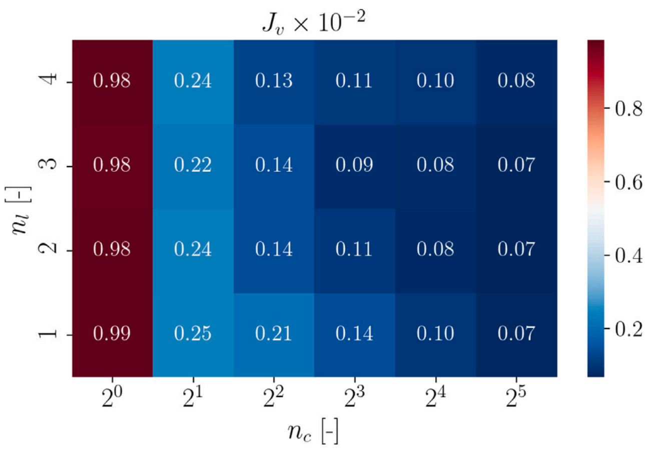

Fig. 13. Heatmap of the J_v for varying n_l, n_c. Best J_v among 5 training runs is shown here.

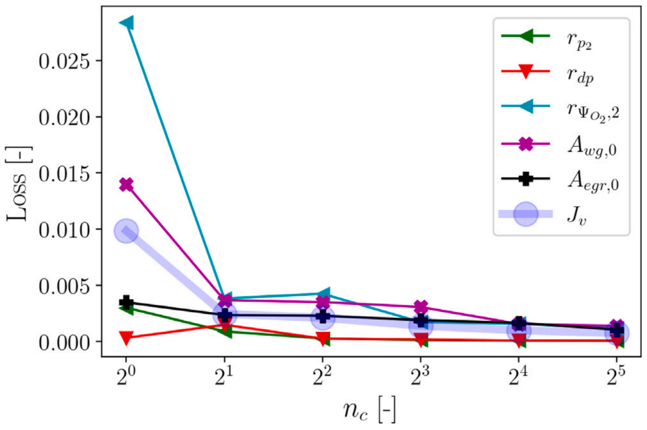

Fig. 14. Loss of individual output over validation dataset for $n_l = 1, n_c \in \{2^0, 2^1, \ldots, 2^5\}$.

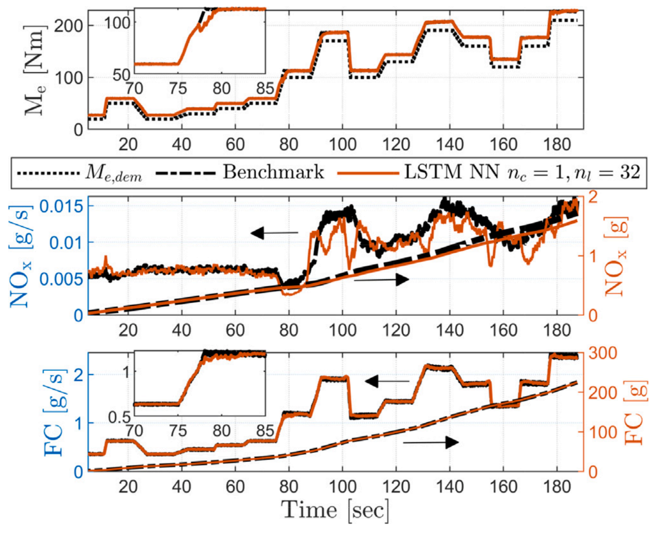

Fig. 15. J_p for benchmark without preview and LSTM NN with preview on the test cycle.

outputs of this model capacity for the validation dataset are shown in Table 5. It is seen that model outputs have varying prediction accuracy even though equal weightage i.e., $c_k = 1$ is given to each output in

Table 5
Individual output mean square error (MSE) and mean absolute error (MAE) over validation dataset for $n_l = 1$, $n_c = 32$. Scaled refers to loss calculated when the outputs are scaled to [0, 1]. Absolute scale is when the outputs are scaled back to original units.

Output	MSE (scaled)	MSE (absolute scale)	MAE
r_{p_2}	8.65×10^{-5}	0.77	0.64 [kPa]
r_{dp}	6.78×10^{-5}	0.03	0.13 [kPa]
$r_{\Psi_{O_2},2}$	1.17×10^{-3}	0.04	0.07 [%]
$A_{wg,0}$	1.4×10^{-3}	0.01	0.05 [cm^2]
$A_{egr,0}$	1.06×10^{-3}	6.74	1.59 [%]

Table 6
J_{ce} for the benchmark and LSTM-based calibration approaches.

	Benchmark	LSTM-based
Experimentation	30	–
Expert effort	16	4.5
Simulations on PC	1	9.3
Total [working days]	47	13.8

the optimization loss function in Eq. (4). A possible explanation is that outputs $r_{\Psi_{O_2},2}$, $A_{wg,0}$ and $A_{egr,0}$ are a combination of steady-state models and transient compensation maps and are more complex to model compared to r_{p_2} and r_{dp}. This is also evident from larger MSE and MAE of these outputs.

Next, the effect of $\Omega_2 = \begin{bmatrix} n_l & n_c \end{bmatrix}^\top$ is studied on cost terms J_{ce}, J_p, J_{mc}, J_{it}. Table 6 shows the comparison of J_{ce} between LSTM-based preview controller and the benchmark controller. It is seen that J_{ce} with LSTM model is significantly smaller than the benchmark. In the benchmark process, reference setpoints are stored in maps and feedforward controller comprises of physics-based (PB) models of components such as turbocharger, intercooler, EGR valve and air-filter. For calibration of these PB models, large number of experiments and expert effort are required. Whereas the proposed approach requires no component testing as the data required for training LSTM-based controllers can be generated using the dynamic engine models. For preview control, the strategy proposed discussed in Section 3 utilizes the control signals for non-preview control. This significantly reduces the J_{ce} otherwise required in formulation of complex MPC problem for non-linear engine process. Using the design of acceleration and deceleration steps in Section 4.2.3, calibration can be performed for both steady-state and transient operations. Furthermore, expert tasks associated with the proposed approach consists of data generation, hyperparameter tuning and validation. These tasks require approximately 71% less effort than those associated with the benchmark steps f. and g. shown in Fig. 4. This reduction is significant as the steps f and g constitute as large as 58% of the total effort in benchmark calibration process. With regard to simulations, significant time of 9.3 working days is required to design a ML-based controller in steps 5.2 and 5.3 of the process shown earlier in Fig. 8.

J_p using the LSTM-based controller for non-preview and preview control are validated on a synthetically generated transient test cycle shown in Fig. 11. This cycle is generated using randomly chosen combinations of $M_{e,dem,0}$, Δs and $\Delta M_{e,dem}$ within the parameter design space in Fig. 10. In addition, negative ramps in engine torque demand i.e., negative values of $\Delta M_{e,dem}$ are chosen to evaluate the model's capability to generalize for engine operation outside the training region. In order to determine impact of using LSTM-based controller on J_p, the open-loop performance is compared with the benchmark. This prevents the PID-based feedback controller from compensating the error between reference and measured values. J_p is analyzed for the model architecture with the minimum error on the validation set (30% of the dataset) i.e., $n_l = 1$, $n_c = 32$ over the driving cycle. Fig. 15 and Table 7 shows the comparison of J_p between the benchmark controller and LSTM-based controller with and without preview. For *LSTM-based*

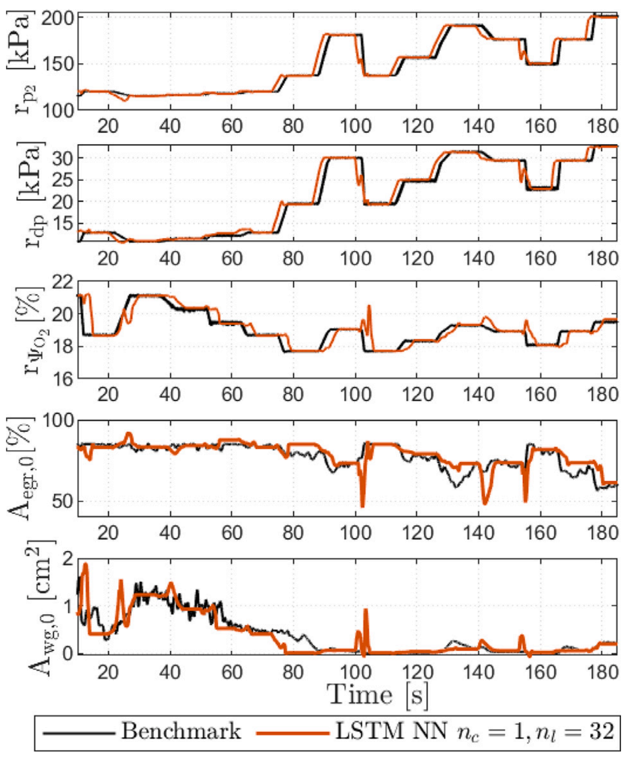

Fig. 16. Comparison of control signals between benchmark without preview and LSTM NN with preview on the test cycle.

Table 7
Comparison of control system performance.

	Cumulative NOx [g]	Cumulative FC [g]
Benchmark controller without preview	1.71	223.5
LSTM NN without preview, $n_l = 1, n_c = 32$	1.72	223.3
LSTM NN with preview, $n_l = 1, n_c = 32$	1.63	223.1

controller with preview, the torque output and fuel consumption performance is similar to the benchmark controller in majority of the driving cycle except for transient between time intervals $[70, 85]$ and $[90, 110]$ seconds. Between $[70, 85]$ seconds, the engine torque response is slower, which is attributed to large prediction error in $A_{wg,0}$ and $A_{egr,0}$ in this time interval as shown in Fig. 16. The cause of this prediction error is interpolation error in the LSTM model, which arises due to scarcity of data points for these transient in the training and validation dataset. As can be seen from Fig. 10, the design space for generating training data does not include $M_{e,dem} = 50$, $\nabla M_{e,dem} = 50$, which explains the above observation. For time interval $[90, 110]$ seconds, it is seen that the test ramp lies outside the design space of training data and are attributed to extrapolation errors. Moreover, peaks in NO$_x$ emissions are seen during deceleration steps in time intervals such as, $[100, 110]$ and $[150, 160]$ seconds. This highlights a major limitation of data-driven models for data outside their training region. This error is not seen in the LSTM-based controller without preview, which was trained using both positive and negative torque ramps. With regard to NO$_x$ emissions, a significant reduction of 5% is seen in cumulative values with LSTM-based preview controller compared to the benchmark. The reductions are observed during the positive acceleration steps, which is expected with the proposed preview strategy in Section 3, where the retard in $r_{\Psi_{O_2},2}$ signal prevents peaks of NO$_x$ emissions. For the *LSTM-based controller without preview*, it is seen that J_p is similar to benchmark as seen from cumulative values in Table 7.

For the benchmark controller, $J_{mc} = 3 \times 10^4$ parameters and $J_{it} = 8$ milliseconds in the Simulink environment. For LSTM controllers, J_{mc} shows positive correlation with both n_l, n_c, while J_{it} remains consistent for fixed n_l and increases significantly with n_l, see Appendix F for details. Table 8 shows the value of cost terms for the benchmark controller without preview and LSTM models with three different model capacities for preview control. A trade-off is seen between J_v, J_{mc}, J_{it} as illustrated in Fig. 17. Despite the large number of simulations, J_{ce} with the proposed process is approximately 71% lower than the benchmark with comparable performance with non-preview controller and improved performance with preview controller as shown in Fig. 15. With LSTM model capacity $n_l = 1$, $n_c = 32$ for control without preview, a lower $\mathbf{J}$ is seen due to significantly smaller J_{ce} and J_{mc} despite larger J_v. For preview control, it is seen that all three model capacities achieve smaller J_p compared to benchmark controller without preview. LSTM neural network with $n_l = 1$, $n_c = 2$ achieves the smallest J_p, however, at cost of significantly large J_v. In order to have explainability of control system behavior, it is necessary to have accurate control signals. LSTM neural network with $n_l = 1$, $n_c = 32$ achieves the smallest $\mathbf{J}$ owing to smaller J_v despite slightly larger J_{mc}, J_p compared to network with $n_l = 1$, $n_c = 2$. LSTM neural network with $n_l = 4$, $n_c = 32$ has a large J_{it} due to large number of model parameters, which increases the number of computations for inference. The neural network with $n_l = 1$, $n_c = 32$ exhibits the best balance for these criteria among studied model capacities. This analysis provides a systematic approach towards selection of ML model capacity for its implementation on ECU.

Table 8
Cost function values for benchmark without preview and LSTM models for without and with preview control.

	Model capacity	$d_1 J_v$	$d_2 J_{ce}$	$d_3 J_p$	$d_4 J_{mc}$	$d_5 J_{it}$	$\mathbf{J}$
Benchmark without preview	–	0	47	39.45	30	80	196.5
LSTM NN without preview	$n_l = 1, n_c = 32$	70	13.8	39.55	4.65	70	198
LSTM NN with preview	$n_l = 1, n_c = 2$	990	13.8	38.11	0.055	70	1111.9
	$n_l = 1, n_c = 32$	70	13.8	38.63	4.65	70	197.1
	$n_l = 4, n_c = 32$	80	13.8	38.6	29.6	240	402

6. Conclusions

An efficient calibration approach for transient control is needed for the control development of complex future automotive powertrains. Preview control further improves performance by compensating for system delays given future information about changing references or disturbances. In this paper, an efficient calibration approach to realize non-preview and preview control is proposed. This approach requires no component testing for feedforward controller tuning, which reduces the experimentation cost, efficiently parametrizes the reference trajectory and feedforward controller signals using a ML model and further improves controller performance in transient by efficiently realizing preview control. The practical implication of realizing this calibration approach is the need of experts that can combine system knowledge and ML to develop ML-based controllers. The preview control signals are parametrized using a LSTM neural network, which is designed by solving a multi-objective optimization problem for maximization of system performance and minimization of calibration effort, model inaccuracy and computational requirements. This problem formulation allows efficient calibration of the LSTM-based controller and facilitates a systematic choice of neural network model capacity for real-time implementation. A significant reduction of 71% in control calibration effort is achieved compared to the benchmark model-based calibration process. LSTM model-based non-preview controller achieves similar system performance, computational requirement, smaller calibration effort at the cost of small inaccuracies in the model predictions. Whereas with LSTM model-based preview controller, a significant reduction of 5%

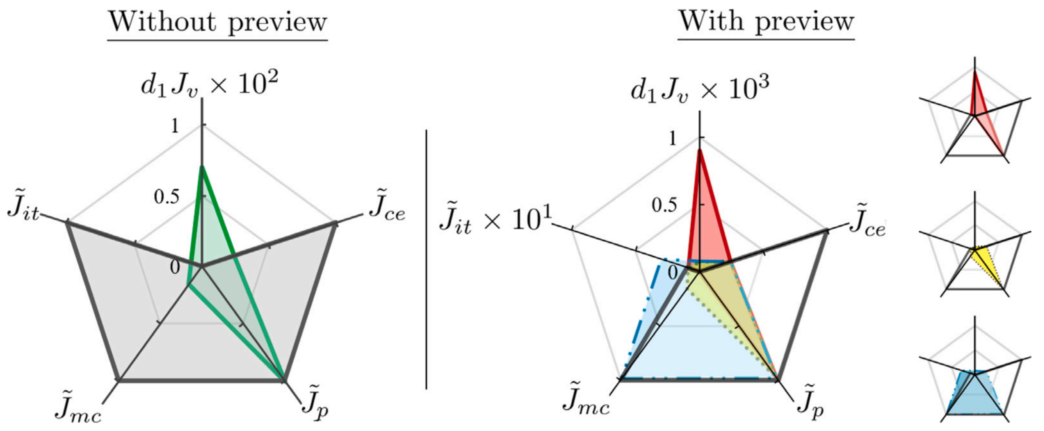

Fig. 17. Radar chart showing relationship between the multiple cost terms for without preview control on left and with preview control on right for different LSTM models and benchmark controller without preview. Normalized values $\tilde{J} = J/J_{bm}$ are shown here for cost terms $J_{ce}, J_p, J_{mc}, J_{it}$. Benchmark without preview (—), LSTM $n_c = 1, n_l = 32$ without preview (—), LSTM $n_c = 1, n_l = 2$ with preview (—), LSTM $n_c = 1, n_l = 32$ with preview (• • •) and LSTM $n_c = 4, n_l = 32$ with preview (– · –). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table A.9
Comparison of engine performance between benchmark and preview controller for individual torque steps.

Torque step	Shifting strategy	Engine speed [rpm]	Initial engine torque [N m]	Final engine torque [N m]	Benchmark controller			Preview controller		
					Torque rise time [s]	BSNOx [g/kWh]	BSFC [g/kWh]	Torque rise time [s]	BSNOx [g/kWh]	BSFC [g/kWh]
1	r_{p_2}, r_{dp}: Advance 2 s; $r_{\Psi_{O_2,2}}$: advance 0.5 s	1500	10	100	0.03	0.98	222	0.03	0.93	222
2	r_{p_2}, r_{dp}: Advance 2 s; $r_{\Psi_{O_2,2}}$: retard 2 s	1700	150	250	0.1	1.4	219	0.1	1.3	218
3	r_{p_2}, r_{dp}: Advance 2 s	1700	150	350	4.6	1.76	228	4.6	1.78	228

in cumulative NO_x emissions, similar fuel consumption with smaller calibration effort are achieved over a transient driving cycle.

Future steps include implementation of the LSTM recurrent networks on the engine rapid control prototyping system and studying the impact of noisy inputs on the model's prediction accuracy and the control system performance.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors would like to thank DENSO Japan for the financial support and the Powertrain R&D team at DENSO Aachen Engineering Centre (AEC), Germany for constructive discussions.

Appendix A. Study to determine best signal shift for preview control

Future vehicles are expected to be equipped with a Global positioning system (GPS) and global information system (GIS), which can provide information on vehicle speed limit, road inclination and traffic flow. This can be used to predict engine torque demand and engine speed demand for the route ahead. In this study, three different engine torque steps and two different engine speeds are studied as described in Table A.9. These torque steps includes engine operation region as follow: (1) low to medium torque demand, (2) medium torque demand and (3) medium to high torque demand regions. Engine speeds demand are chosen to be 1500 for torque step 1 and 1700 rpm for torque steps 2, 3 from a range of frequent speeds for a light-duty diesel engine. The range of studied δt is determined based on the rise time[4] of p_2 and $\Psi_{O_2,2}$. The rise times are determined for multiple step inputs of engine torque at increasing engine speeds. The range of response time for p_2 is $(0,6]$ seconds and for $\Psi_{O_2,2}$ is $(0,2]$ seconds. The signals r_{p_2}, r_{dp} are advanced by $\delta t \in \{0.5, 1, \dots, 6\}$ seconds to increase fresh air-mass flow rate and engine torque response before the start of the transient. The signal $r_{\Psi_{O_2,2}}$ is either retarded or advanced by $\delta t \in \{0.5, 1, 1.5, 2\}$ seconds in order to affect oxygen concentration in the intake manifold and NO_x emissions. Brake specific NO_x ($BSNO_x$) emissions, brake specific fuel consumption (BSFC) and engine torque rise time are compared with the benchmark controller without preview for individual torque step.[5]

For torque step 1, there is similar torque response compared to the benchmark controller, therefore, references are manipulated such that NO_x emissions can be reduced. For torque step 2, the best system performance is attained by advancing the reference signals r_{p_2}, r_{dp} by 2 s and retarding the $r_{\Psi_{O_2,2}}$ signal by 2 s. The wastegate and EGR valves are further closed during the transient which increases the EGR mass flow rate and significantly reduces the NO_x emissions. Reduction of 7.1% in $BSNO_x$ and 0.45% reduction in BSFC is attained with preview control. For torque step 3, advancing references signals r_{p_2}, r_{dp} by

[4] Rise time is defined as the time required for system to reach within ±5% of the new setpoint.

[5] Brake specific quantity is calculated as,

$$\text{BSNOx} = \frac{\int_{t_0}^{t_f} \dot{m}_{NO_x}(t)\, dt}{\int_{t_0}^{t_f} P_b(t)\, dt} \tag{A.1}$$

where t_0 and t_f are initial and the final time instances of a torque step; $\dot{m}_{NO_x}(t)$ is the instantaneous mass flow rate of NO_x emissions; $P_b(t)$ is the instantaneous brake power.

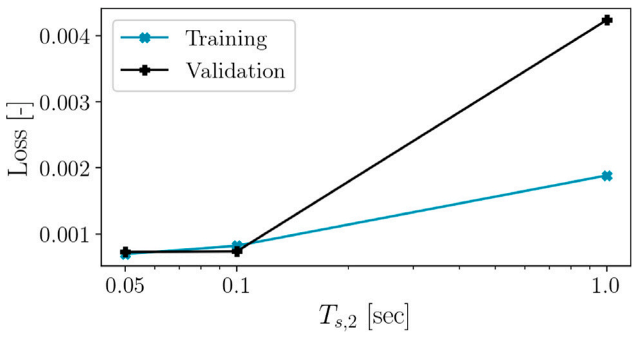

Fig. B.18. Sensitivity of training and validation loss with varying $T_{s,2}$.

2 s shows little benefit in the initial torque response with similar performance for other outputs.

It is seen that the preview control varies with engine operating conditions. Among the studied torque steps, the significant gain is seen for a mid-range torque transient i.e., torque step 2 with little gain in torque step 1. For the purpose of method development, the strategy *advancing the reference signals* r_{p_2}, r_{dp} *by 2 seconds and retarding the* $r_{\Psi_{O_2},2}$ *signal by 2 seconds* is adopted for all the acceleration and deceleration ramps in Section 4. The resulting reference signals and feedforward control actions are denoted by $\mathbf{r}'$ and $\mathbf{u}'_0$, respectively. This study demonstrates a proof of concept for improved engine performance with preview control. Accumulation of performance gains in individual transient over a driving cycle promises a significant real-world improvement in the system performance.

Appendix B. Sensitivity study to determine $T_{s,2}$

The value of $T_{s,2}$ is determined by a sensitivity study of the LSTM model with 1 layer and 32 cells as shown in Fig. B.18. The decrease in training and validation loss for $T_{s,2} \leq 0.1$s is marginal, therefore, $T_{s,2}$ is chosen to be 0.1s in order to minimize the training and computation time.

Appendix C. Sensitivity study to determine t_p

The impact of t_p on training and validation error of the LSTM model of fixed complexity i.e., one layer and 32 cells is studied as shown in Fig. C.19. A significant decrease in both training and validation loss is observed by increasing t_p to 2 s compared to 1 s. Further, only a small reduction in losses is observed by increasing it beyond 2 s, however at the cost of larger training time due to larger input length. The training time increases by a factor of $\sim$ 1.7 times by increasing t_p from 2 s to 4 s. This factor is significant as the cumulative training time to train varying model capacities (i.e., hyperparameters $\Omega_2 = \begin{bmatrix} n_l & n_c \end{bmatrix}^\top$) will increase significantly.

Appendix D. Comparison of neural network optimization algorithms

Two algorithms with adaptive learning rates i.e., Adam and RMSProp are compared to stochastic gradient descent with fixed learning rate value of 0.001 and momentum value of 0.9. Table D.10 shows the J_v for different algorithms with a upper bound on maximum number of epochs. It can be seen that ADAM and RMSProp, which have adaptive learning rates achieve smaller losses faster than SGDm algorithm.

Appendix E. Impact of n_r, n_p, n_e on J_v and $t_{\text{computation}}$

Figs. E.20 and E.21 show the J_v and $t_{\text{computation}}$, respectively for Case 1 and Case 2, defined in Section 5.2 . It is seen that the smallest J_v with Case 1 is close to Case 2 and requires approximately 74% and 68% less $t_{\text{computation}}$ for $n_l = 1$, $n_c = 2^5$ and $n_l = 1$, $n_c = 2^6$, respectively.

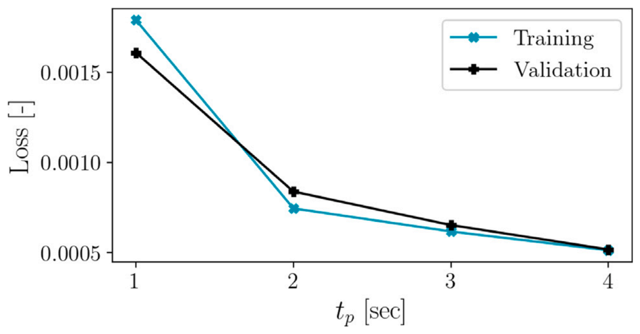

Fig. C.19. Training and validation loss with varying length of the t_p.

Table D.10
J_v with different optimization algorithms. Underlined values refer to the minimum loss values among algorithms. Other parameters fixed at $n_e = 500$, $n_l = 1$, $n_c = 32$, $n_{mb} = 100$.

	Adam	RMSProp	SGDm
J_v	$\underline{7.5 \times 10^{-4}}$	7.7×10^{-4}	4.1×10^{-3}

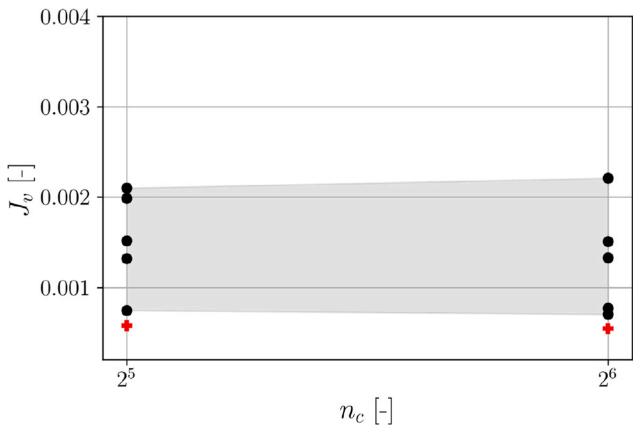

Fig. E.20. J_v for Case 1 represented by black circles and Case 2 represented by red plus markers. The shaded region show the range of J_v for $n_r = 5$.

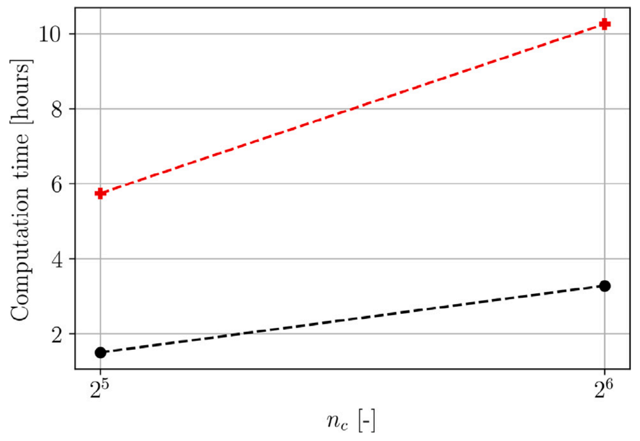

Fig. E.21. $t_{\text{computation}}$ for Case 1 represented by black circles and Case 2 represented by red plus markers.

Appendix F. Impact of n_l and n_c on J_{it} and J_{mc}

J_{it} and J_{mc} for LSTM models are shown in Tables F.11 and F.12, respectively.

Table F.11
J_{it} in milliseconds for different capacities of LSTM models.

		n_c					
		2^0	2^1	2^2	2^3	2^4	2^5
n_l	4	23	24	20	22	22	24
	3	19	21	20	18	19	19
	2	12	13	13	13	13	13
	1	8	7	7	7	7	7

Table F.12
$J_{mc} \times 10^4$ for different capacities of LSTM models.

		n_c					
		2^0	2^1	2^2	2^3	2^4	2^5
n_l	4	0.0062	0.0175	0.0569	0.203	0.764	2.96
	3	0.005	0.0135	0.0425	0.149	0.553	2.13
	2	0.0038	0.0095	0.0281	0.0941	0.341	1.30
	1	0.0026	0.005	0.0137	0.0397	0.13	0.465

References

Alfieri, E., Amstutz, A., & Guzzella, L. (2009). Gain-scheduled model-based feedback control of the air/fuel ratio in diesel engines. *Control Engineering Practice*, *17*(12), 1417–1425.

Aliramezani, M., Koch, C. R., & Shahbakhti, M. (2022). Modeling, diagnostics, optimization, and control of internal combustion engines via modern machine learning techniques: A review and future directions. *Progress in Energy and Combustion Science*, *88*, Article 100967.

Atkinson, C. (2014). *Fuel efficiency optimization using rapid transient engine calibration*: *SAE world congress, SAE technical paper 2014-01-2359*, Society of Automotive Engineers (SAE).

Bishop, C. M. (1995). Regularization and complexity control in feed-forward networks. In *Proceedings international conference on artificial neural networks, vol. 1* (pp. 141–148).

Boulmaiz, T., Guermoui, M., & Boutaghane, H. (2020). Impact of training data size on the LSTM performances for rainfall–runoff modeling. *Modeling Earth Systems and Environment*, *6*(4), 2153–2164.

Garg, P., Silvas, E., & Willems, F. (2021). Potential of machine learning methods for robust performance and efficient engine control development. *IFAC-PapersOnLine*, *54*(10), 189–195.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *JMLR workshop and conference proceedings*, *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Isermann, R. (2014). *Engine modeling and control*. Springer Berlin Heidelberg.

Kanzaki, S., Okada, K., & Inaba, M. (2005). Bracing behavior in humanoid through preview control of impact disturbance. In *5th IEEE-RAS international conference on humanoid robots* (pp. 301–305). IEEE.

Karlsson, M., Ekholm, K., Strandh, P., Johansson, R., & Tunestål, P. (2010). Multiple-input multiple-output model predictive control of a diesel engine. *IFAC Proceedings Volumes*, *43*(7), 131–136.

Liu, Z., Dizqah, A. M., Herreros, J. M., Schaub, J., & Haas, O. (2021). Simultaneous control of NOx, soot and fuel economy of a diesel engine with dual-loop EGR and VNT using economic MPC. *Control Engineering Practice*, *108*, Article 104701.

Mentink, P., Willems, F., Kupper, F., & Van den Eijnden, E. (2013). *Experimental demonstration of a model-based control design and calibration method for cost optimal Euro-VI engine-aftertreatment operation*: *SAE world congress, SAE technical paper 2013-01-1061*, Society of Automotive Engineers (SAE).

Moergastel, B. v., Visser, S., Norgel, K., & Herrmann, O. (2019). New approaches for virtualized diesel powertrain system development. In *8th int. symposium on development methodology* (pp. 1–10).

Moriyasu, R., Nojiri, S., Matsunaga, A., Nakamura, T., & Jimbo, T. (2019). Diesel engine air path control based on neural approximation of nonlinear MPC. *Control Engineering Practice*, *91*, Article 104114.

Norouzi, A., Heidarifar, H., Shahbakhti, M., Koch, C. R., & Borhan, H. (2021). Model predictive control of internal combustion engines: A review and future directions. *Energies*, *14*(19), 6251.

Norouzi, A., Shahpouri, S., Gordon, D., Winkler, A., Nuss, E., Abel, D., Andert, J., Shahbakhti, M., & Koch, C. R. (2022). Deep learning based model predictive control for compression ignition engines. *Control Engineering Practice*, *127*, Article 105299.

Peng, Q., Huo, D., & Hall, C. M. (2022). Neural network-based air handling control for modern diesel engines. *Proceedings of the Institution of Mechanical Engineers, Part D (Journal of Automobile Engineering)*.

Peng, Q., Huo, D., & Hall, C. M. (2023). Neural network-based air handling control for modern diesel engines. *Proceedings of the Institution of Mechanical Engineers, Part D (Journal of Automobile Engineering)*, *237*(5), 1113–1130.

Sastry V M, S., Garg, P., Silvas, E., & Willems, F. (2022). Systematic comparison of supervised learning methods to reduce calibration effort in engine control development. *IFAC-PapersOnLine*, *55*(20), 37–42.

Sequenz, H. (2013). *Emission modelling and model-based optimisation of the engine control, vol. 8*. VDI Verlag.

Sjöberg, J., & Ljung, L. (1995). Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, *62*(6), 1391–1407.

Stewart, G., Borrelli, F., Pekar, J., Germann, D., Pachner, D., & Kihas, D. (2010). Toward a systematic design for turbocharged engine control. In *Automotive model predictive control* (pp. 211–230). Springer.

Yu, K., Yang, H., Kawabe, T., & Tan, X. (2015). Model predictive control of a power-split hybrid electric vehicle system with slope preview. *Artificial Life and Robotics*, *20*(4), 305–314.

Zhao, D., Liu, C., Stobart, R., Deng, J., Winward, E., & Dong, G. (2013). An explicit model predictive control framework for turbocharged diesel engines. *IEEE Transactions on Industrial Electronics*, *61*(7), 3540–3552.