Applied Soft Computing 13 (2013) 2375–2389

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

Nonlinear identification of a gasoline HCCI engine using neural networks coupled with principal component analysis

Vijay Manikandan Janakiraman[a,*], XuanLong Nguyen[b], Dennis Assanis[c]

[a] Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA
[b] Department of Statistics, University of Michigan, Ann Arbor, MI, USA
[c] Stony Brook University, NY, USA

ARTICLE INFO

Article history:
Received 1 July 2012
Received in revised form 29 October 2012
Accepted 6 January 2013
Available online 30 January 2013

Keywords:
Neural networks
Nonlinear system identification
HCCI engine modeling
Multi-layer perceptron
Radial basis network
Principal component analysis

ABSTRACT

Homogeneous charge compression ignition (HCCI) is a futuristic combustion technology that operates with high efficiency and reduced emissions. HCCI combustion is characterized by complex nonlinear dynamics which necessitates the use of a predictive model in controller design. Developing a physics based model for HCCI involves significant development times and associated costs arising from developing simulation models and calibration. In this paper, a neural networks (NN) based methodology is reported where black box type models are developed to predict HCCI combustion behavior during transient operation. The NN based approach can be considered a low cost and quick alternative to the traditional physics based modeling. A multi-input single-output model was developed each for indicated net mean effective pressure, combustion phasing, maximum in-cylinder pressure rise rate and equivalent air–fuel ratio. The two popular architectures namely multi-layer perceptron (MLP) and radial basis network (RBN) models were compared with respect to design, prediction performance and overall applicability to the transient HCCI modeling problem. A principal component analysis (PCA) is done as a pre-processing step to reduce input dimension thereby reducing memory requirements of the models. Also, PCA reduces the cross-validation time required to identify optimal model hyper-parameters. On comparing the model predictions with the experimental data, it was shown that neural networks can be a powerful approach for non-linear identification of a complex combustion system like the HCCI engine.

1. Introduction

In recent years, the requirements on automotive performance, emissions and safety have become stringent. In spite of advanced concepts entering the industry, achieving fuel economy, emission and cost targets simultaneously still remain an arduous task. HCCI engines shifted the spotlight from traditional spark ignited (SI) and compression ignited (CI) engines owing to its ability to reduce emissions and fuel consumption significantly [1–3]. The fuel lean mixtures allow HCCI to operate with a larger compression ratio similar to diesel engines resulting in high thermal efficiency [4]. Also, absence of throttle improves volumetric efficiency. The fuel and oxidizer are premixed which results in clean combustion and hence reduced emissions [4]. A characteristic feature of HCCI combustion is that the peak in-cylinder temperatures are low resulting in low nitrogen oxides (NO_X) emissions [5]. The homogeneous mixtures result in reduced soot emissions [4]. In spite of its known advantages, HCCI combustion poses several challenges for implementation. These include the absence of a direct trigger for combustion, narrow operating range and high sensitivity to ambient conditions amongst others.

Control of HCCI combustion is a challenging problem and control decisions are often made using a predictive model of the engine [6–8]. The quality of the predictive model in terms of its accuracy of prediction and computational requirement for on-line application are some of the important criteria that directly affects the control performance for HCCI engines. HCCI combustion is characterized by complex nonlinear chemical kinetics and thermal dynamics and high-fidelity behavior can be assessed using numerical simulations [9–12]. Such models demand a large computational time and effort and are not directly suitable for control. Control-oriented reduced order models [13,8] can be developed by using simplifying assumptions but require significant development time and associated costs. To accelerate HCCI implementation on automotive applications, a key requirement is to develop predictive dynamic models quickly that can capture the required dynamics for control purposes and has the potential to be implemented on-board having limited computation and memory resources. This forms the

* Corresponding author at: 2032 AL, 1231 Beal Ave, Ann Arbor, MI 48109-2133, USA. Tel.: +1 734 358 6633.
E-mail address: vijai@umich.edu (V.M. Janakiraman).

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.01.006

main motivation for the interdisciplinary research considered in this work.

For the HCCI identification problem, neural networks (NN) were selected for their fast operation and sufficient approximation capabilities to fit nonlinear systems [14–19]. Also, when NN is trained on real-world data, it represents the real system and makes no simplifying assumptions of the underlying phenomena. The dynamics of sensors, actuators and other complex processes which are usually overlooked/hard to model using physics can be captured using the identification method. In addition, for a system like the combustion engine, prototype hardware is typically available and extensive experimental data can be collected making the identification approach more attractive. Several implementations of NN for nonlinear system modeling and control have been reported in the literature [19–21]. A diesel locomotive engine was identified using NN [22]. Recurrent neural network models were applied as virtual sensors to predict specific quantities like NO_X [23], air–fuel ratio [24]. Recently the authors have shown the applicability of support vector machines [25,26] for modeling combustion dynamics for HCCI. The identification of HCCI combustion is not common owing to the complex behavior and narrow operating range of HCCI [27,28]. Apart from the authors, the only other report on HCCI identification [7] involved a subspace based identification where linear and polynomial kernel models were developed for model predictive control.

In this paper, a framework for nonlinear system identification using neural networks and PCA is developed for gasoline HCCI combustion. For the training process, transient engine data is required but data collection from a HCCI engine is very challenging during transients as certain excitations can drive the engine unstable or even misfire. Hence a methodology for design of experiments is described considering the steady state experiments and filtering unstable excitations. PCA is used to reduce the input (feature) dimension for the regression problem which makes neural network training efficient and fast. A multi-input single-output model was developed for the primary indicators used to evaluate combustion quality and performance of HCCI such as net mean effective pressure (NMEP), combustion phasing indicated by crank angle at 50% mass fraction burned (CA50), maximum in-cylinder pressure rise rate (R_{max}) and equivalent air–fuel ratio (EAFR) in the cylinder. A comparison study is conducted to evaluate two different neural network architectures – Multi-Layer Perceptron model and the radial basis network models for suitability of application to the HCCI identification problem.

2. Identification using neural networks

A generic nonlinear identification model using the nonlinear auto regressive model with exogenous input (NARX) is considered as follows

$$y(k) = f[u(k-1), \ldots, u(k-n_u), y(k-1), \ldots, y(k-n_y)] \quad (1)$$

where $u(k) \in \mathbb{R}^{u_d}$ and $y(k) \in \mathbb{R}^{y_d}$ represent the inputs and outputs of the system, respectively, k represents the discrete time index, $f(.)$ represents the nonlinear function mapping specified by the model, n_u, n_y represent the number of past input and output samples required (order of the system) for prediction while u_d and y_d represent the dimension of inputs and outputs, respectively. Let x represent the augmented input vector obtained by appending the input and output measurements from the system.

$$x = [u(k-1), \ldots, u(k-n_u), y(k-1), \ldots, y(k-n_y)]^T \quad (2)$$

The measurement sequence can be converted to the form of training data as required by neural networks

$$\{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathcal{X}, \mathcal{Y}) \quad (3)$$

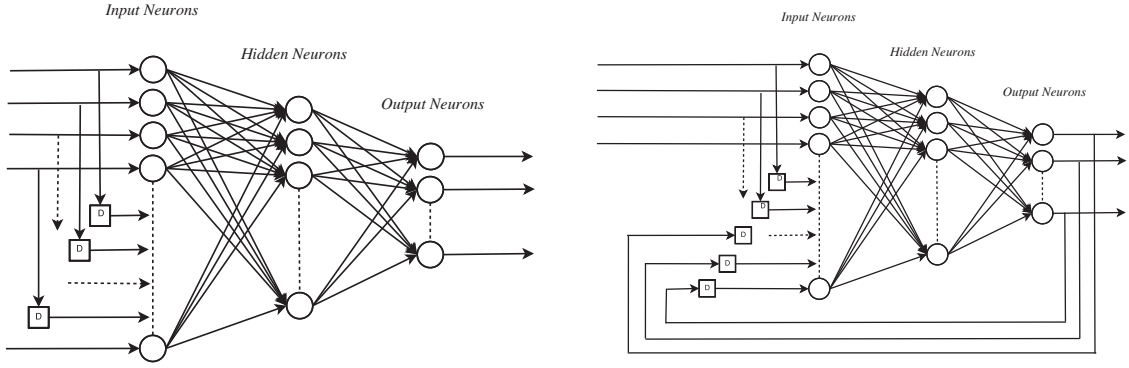
where $\mathcal{X}$ denotes the space of the input features ($\mathcal{X} = \mathbb{R}^{u_d n_u + y_d n_y}$ and $\mathcal{Y} = \mathbb{R}$). The goal of neural networks is to approximate the underlying input–output function mapping $f(.)$ by minimizing a risk function with respect to the model parameters

$$R(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i(x, w))^2 \quad (4)$$

where w represents the model parameters and $\hat{y}(x, w)$ represents the network predictions. It has to be noted from Eq. (1) that the identification model represents a dynamic system and requires previous samples of data for predictions. By presenting the past samples of data to the neural models, a notion of memory is incorporated into the networks and hence the networks are able to capture the dynamics of the system. Such an architecture is commonly referred to as a recurrent network. In this study, an externally recurrent network where the network inputs take the form as defined in (2). Such a selection enables a simple architecture (series-parallel architecture shown using Fig. 1(a)) and trained can be done using available training algorithms. A series-parallel network can only be used for a one-step-ahead prediction and can be converted to a parallel architecture (shown using Fig. 1(b)) by feeding back the output of the series-parallel network instead of the actual system output. The parallel network [15] is capable of performing multi-step-ahead predictions in a recurrent manner required for this work. More complicated memory networks and training procedures for recurrent networks are available for further reference [29–31].

2.1. Multi-Layer Perceptron model

The Multi-Layer Perceptron (MLP) model is constructed using several layers of interconnected nodes having one or more hidden



(a) Series-Parallel Architecture (b) Parallel Architecture

Fig. 1. Recurrent neural network architecture.

layers and the nodes are connected in a feed-forward manner between the input and the output layer. MLPs are known as universal function approximators as they can approximate any smooth function to an arbitrary accuracy [14,32]. The network considered in this study consists of an input layer, an output layer and a hidden layer. The predicted output from the network can be expressed as

$$\hat{y}_{MLP} = f_2[b_2 + W_2^T f_1\{b_1 + W_1^T(x)\}] \quad (5)$$

$$f_1(z) = \frac{2}{1+e^{-2z}} - 1 \quad (6)$$

where f_1 and f_2 represent the activation functions of the hidden and output layers, respectively. The functions f_1 and f_2 returns outputs of the same dimension as that of their inputs. A tangential sigmoid function is used to represent f_1 while a linear function is used to represent f_2. The parameters b_1 and b_2 represent the bias terms of the hidden and output layers while W_1 and W_2 represent the weights of the connection between input-hidden layers and hidden-output layers, respectively. The number of input and output neurons corresponds to the dimensions of $\mathcal{X}$ and $\mathcal{Y}$, respectively.

The MLP is trained using standard back-propagation via the Levenberg–Marquardt (LM) algorithm [33] which is more efficient compared to other available gradient based training algorithms such as the conjugate gradient and the variable learning rate back-propagation [34]. The sensitivity of the cost function (4) with respect to the network weights are determined and weights updated so that the cost (4) is minimized. Batch training is done where the entire training data is repeatedly presented to the network. Batch training is more efficient compared to incremental training but requires all the data to be available for an off-line training. The activation functions (sigmoid and linear) take finite values for the entire range of network operation and hence the MLP model requires a relatively less number of parameters to map complex functions. However, training could become complicated involving heavy computation making the process slow for large data sets. The MLP is a nonlinear regression problem and training with an iterative algorithm like the LM, usually finds a local minima. Global optimization methods such as simulated annealing [35–37], and genetic algorithms [38,35,39] can be used in training but convergence is usually slow and may not suit problems with large input dimensions.

2.2. Radial Basis Network model

The Radial Basis Network (RBN) model is a two layered network where the connections are feed-forward between the input and the output layers similar to the MLP model. The nodes are connected directly between the input and the hidden layers (i.e., with unit weights). The hidden layer neurons have a Gaussian activation function f_3 that determines the excitation level of the neurons depending on how close the input data is located with respect to the neuron's center (μ_h) of the activation functions. Hence a notion of location and similarity is introduced in the RBN model resulting in local learning. The output layer is linear and hence the output is a weighted linear combination of the activation levels of the hidden neurons. Hence when the centers of the activation functions are fixed, the training reduces to determining the hidden-output layer weights using linear least squares. This usually makes training in RBN models fast compared to MLP and is an important distinction between the two models. The predicted output and the activation function of the RBN model can be expressed as

$$\hat{y}_{RBN} = b_4 + W_4^T f_3\{x\} \quad (7)$$

$$f_{3_h}(z) = exp\left(-\frac{\|z-\mu_h\|^2}{2\sigma_h^2}\right) \quad (8)$$

where $\mu_h \in \mathbb{R}^{u_d n_u + y_d n_y}$ and $\sigma_h \in \mathbb{R}$ are the center and the spread of the Gaussian function for a given hidden neuron h. The spread σ_h is constant for all hidden neurons and is considered as a hyper-parameter and is not updated in the training process.

RBN models are also considered universal approximators [40–42] and are popular in regression and classification applications. The training of RBN model involves two tasks – identifying the neuron parameters (μ_h) and determining the hidden to output layer weights (W_4). Several means of finding the neuron parameters include randomly selecting locations from the input data, unsupervised learning of the input structure using clustering [43–49], orthogonal least squares [50–52] among others. The hidden-output layer weights are determined using linear least squares. In this paper, the RBN model is trained using the Matlab subroutine where at every training pass, a new hidden neuron is created at the location of the input vector that results in reducing the network error the most. The above process is repeated until the error reaches a specified goal or the limit for maximum number of hidden neurons is reached. Such a simple training method is found sufficient for the considered problem.

3. HCCI combustion system

3.1. HCCI background and model variables

A major difference between HCCI and traditional SI or CI combustion is that the HCCI combustion do not have a direct trigger for ignition. Hence the properties of the gas mixtures before combustion dictates the combustion behavior. Several complex phenomena such as gas transport, chemical kinetics, heat transfer, and gas mixing makes the combustion very sensitive to the mixture properties and physical conditions at which the engine operates [12]. Fundamental HCCI research has proved that the temperature and concentrations of mixture components at intake valve closing (IVC) play a major role in determining the auto-ignition phenomenon in HCCI combustion [1,53,54]. It is practically not feasible to measure mixture concentrations and in-cylinder temperatures dynamically. Hence these quantities are typically modeled using simplified physics and experimental correlations [13]. Also there exists a cycle-to-cycle coupling in residual affected HCCI as the exhaust residuals from the previous cycle are reused [55,13,56,53]. Hence temperature and concentrations of the combustion products from the previous cycle affect the combustion behavior during the present cycle. The knowledge of combustion reactions, heat release, heat transfer, flow dynamics are required to model the quantity, temperature and concentrations of residuals. All the above mentioned phenomena are extremely complex and require complex modeling, extensive validations, significant development time and associated costs. Also, the resulting models are built from simplified physical and chemical relations with several assumptions and approximations [57].

Identification based on sensor measurements is equally challenging as direct measurements of key quantities are not possible. Dynamically measuring in-cylinder temperatures and mixture concentrations are infeasible or very expensive in the time scale as required for transient engine operation [58]. Also, the system is highly nonlinear and has a narrow region of stable operation [27,28,59–61]. The sensitive nature of HCCI combustion coupled with a narrow region of stable operation makes it extremely challenging to obtain dynamic data that contains rich information about the system for identification. The measurements are very noisy and with high variability which increases further complication in selecting a robust identification method. However, if the identification process is made systematic for the HCCI system and benefits proved, it could be a powerful alternative for the physics based modeling approach which is one of the objectives of this work. For

Table 1
Specifications of the experimental HCCI engine.

Engine type	4-Stroke In-line
Fuel	Gasoline
Displacement	2.0 L
Bore/stroke	86/86
Compression ratio	11.25:1
Injection type	Direct injection
Valvetrain	Variable valve timing with hydraulic cam phaser having 119° constant duration defined at 0.25 mm lift, 3.5 mm peak lift and 50° crank angle phasing authority
HCCI strategy	Exhaust recompression using negative valve overlap

the HCCI identification process, the following measurable quantities are considered as precursors or indicators for the previously mentioned key quantities that cannot be measured directly.

1 The temperature (T_{in}), pressure (P_{in}) and flow rate ($\dot{m}_{in}$) at intake manifold
2 The exhaust gas temperature (T_{ex})
3 Engine coolant temperature (T_c)
4 Controllable quantities such as injected fuel mass (FM in mg/cyc), crank angle at intake valve opening (IVO), crank angle at exhaust valve closing (EVC), crank angle at start of fuel injection (SOI). The valve events are measured in degrees after exhaust top dead center (deg eTDC) while SOI is measured in degrees after combustion top dead center (deg cTDC).
5 Equivalent air to fuel ratio (EAFR) defined as

$$\text{EAFR} = \frac{(A/F)}{(A/F)_s} \tag{9}$$

where A/F = mass of air per cycle/mass of fuel per cycle and $(A/F)_s = (A/F)$ at stoichiometric condition.
6 Indicators of combustion behavior such as combustion phasing indicated by the crank angle at 50% mass fraction burned (CA50), combustion work output indicated by net mean effective pressure (NMEP), combustion roughness indicated by maximum rate of pressure rise (R_{max}).

3.2. Experiment design

The data for this study is collected on a gasoline HCCI engine with variable valve timing whose specifications are listed in Table 1. HCCI is achieved by auto-ignition (without spark initiation) of the gas mixture in the cylinder. A large fraction of internal exhaust gas recirculation (EGR) is trapped in the cylinder to maintain low combustion temperatures for reduced emissions of oxides of nitrogen. Variable valve timing allows adjustment of the timing of the closure of the exhaust valve and opening of the intake valve so as to create a negative valve overlap (NVO) to trap the desired quantity of EGR in the cylinder. The fuel injection also happens during the NVO period. The EGR and fuel injected directly influences the temperature and concentration of the gas mixture entering the next combustion cycle. The pressure trace during one combustion cycle along with valve events and fuel injection events are shown in Fig. 2. An amplitude modulated pseudo-random binary sequence (A-PRBS) has been used to design input signals. The data is sampled using the AVL Indiset acquisition system where in-cylinder pressure is sensed every crank angle while NMEP, CA50 and R_{max} are determined on a per-combustion cycle basis. The other quantities like T_{in}, P_{in}, $\dot{m}_{in}$, T_{ex}, T_c, EAFR, etc. are measured using the on-board engine control unit.

As mentioned earlier, HCCI has a narrow operating region and a large input excitation close to unstable regions tend to knock, misfire the engine or operate on limit cycles. Hence it becomes very challenging to design excitation signals for HCCI combustion to acquire data for identification. For this purpose, prior knowledge

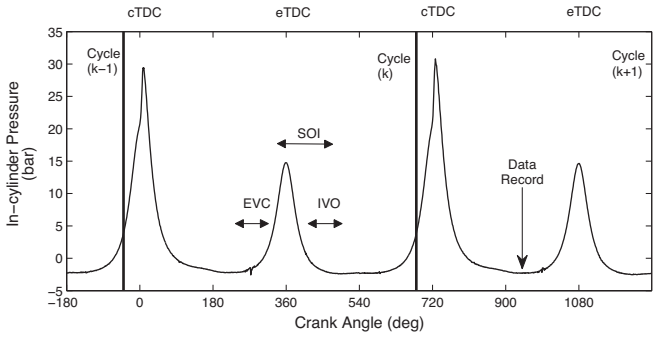


Fig. 2. HCCI engine pressure trace showing cycle definition, actuator ranges of intake valve opening (IVO), exhaust valve closing (EVC), start of injection (SOI). The crank angle at data recording is also shown.

Table 2
Actuator extremes for stable HCCI (from the steady state DOE model).

Input	Min limit	Max limit
Fuel mass (mg/cyc)	7.0437	12.9161
IVO (deg eTDC)	78	128
EVC (deg eTDC)	−119	−69
SOI (deg cTDC)	270	380

about the system is used to eliminate excitation signals that lead to instability. As a first check, a design of experiments model (DOE) [62] using a set of steady state experiments is used to eliminate the unstable input combinations. The feasible limits of inputs at a speed of 3000 RPM as given by the DOE model are shown in Table 2.

The A-PRBS sequence is designed to excite the engine within this stable HCCI region defined by the DOE model. It should be noted that the DOE filtered input limits are valid only for steady state conditions and a large step near the boundary of stable HCCI can lead to instabilities. Hence as a means of precaution against running the engine in an unstable manner and to avoid restarting the measurements, a simple feedback was created, which attempts a particular input combination and if found to be unstable, quickly skips to the next combination in the A-PRBS sequence. As a first attempt, the CA50 was considered the feedback signal. During a small time window, any input combination that resulted in a CA50 above 11 (found by observing the CA50 during misfires) is immediately skipped, and the engine is run on the next combination in the sequence [25]. A subset of the input signals and the recorded outputs from the engine are shown in Fig. 3. Nearly 20000 cycles of data were collected at a constant speed of 3000 RPM, which corresponds to about 25 min of engine testing. About 25% of the data were found to be arising from unstable operation and were removed (Section 4.1).

4. Neural network design

This section describes the neural network framework that is developed in this paper. The framework includes appropriate data pre-processing, model selection and training.

4.1. Data preprocessing

Data preprocessing is necessary for efficient machine learning. This includes normalization of all data to lie between −1 and +1 which ensures the model parameters to be of the same order and improves numerical stability. For the HCCI system, additional preprocessing needs to be done before the data becomes meaningful for learning. HCCI experiments discussed in Section 3.2 involves running the engine at occasional misfires. The data obtained during misfires and some post-misfire data must be removed so that the models only capture the desired HCCI behavior. The misfire data

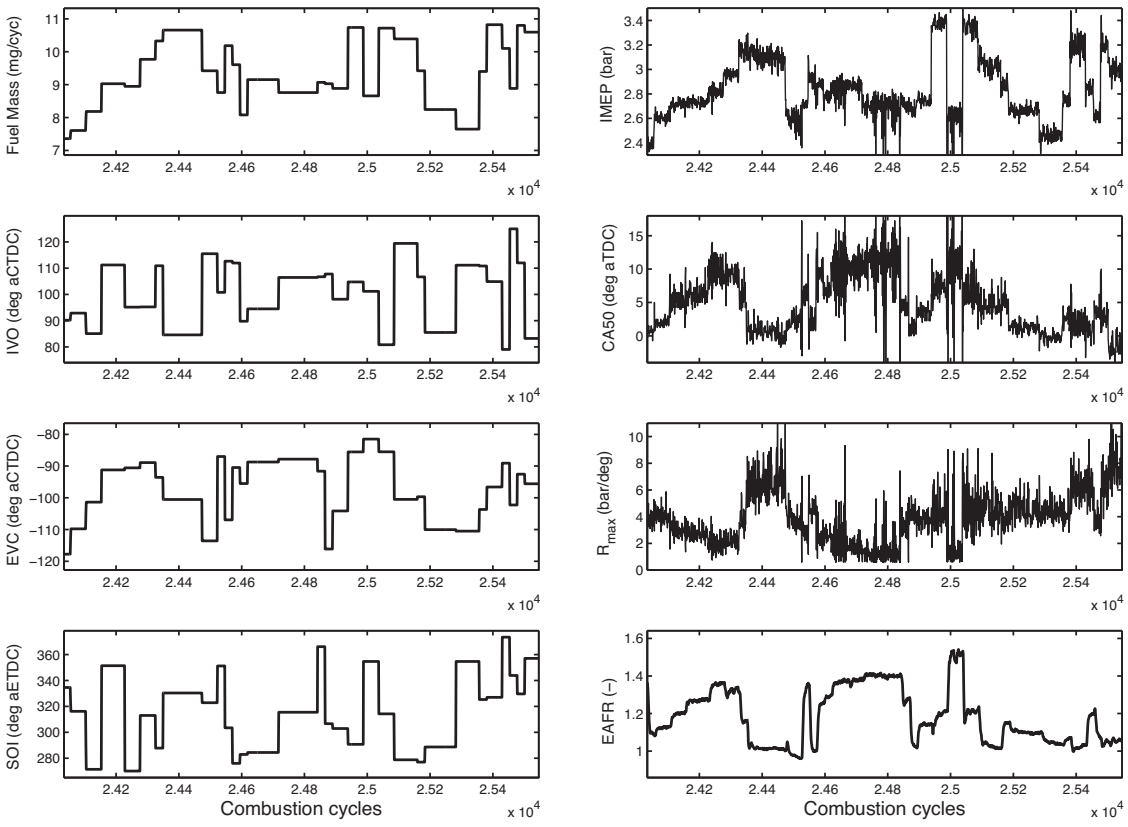


Fig. 3. A subset of experimental data showing the A-PRBS inputs and the measured engine outputs.

can be identified and removed by observing the allowable range for actuators and combustion variables like EAFR, work output and combustion phasing [26].

4.2. Model selection

For efficient neural network design, the models are required to capture the underlying phenomena with a simple structure and minimum number of parameters. Also, the regressors need to be appropriately selected so that the number of parameters and storage requirements of the models are minimized. Assuming no priori knowledge about the system, the model hyper-parameters such as number of hidden neurons, regularization coefficient (λ in MLP and σ_h in RBN) and system order (n_u and n_y) are selected based on cross-validation [25]. The time sequence data set $\{(u_1, y_1), \ldots, (u_n, y_n)\}$ is first converted to a regression data set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. Among the 20,000 observations, about 8,000 observations were randomly selected and used for training. If the data was not downsized randomly, then continuous time-sequence data would be used for training. Such a data set will have several steady state observations which is redundant and slows down training process. The random sampling of data for training makes the training fast and efficient as a small sample can be used to represent both steady state and transient data. The training data set is further divided into a validation-training (4000 observations) and a validation-testing data sets (4000 observations). The models are first trained using the validation-training data set and tested on the validation-testing set for several combination of hyper-parameters (a full grid search) and the combination that resulted in the minimum validation-testing error was considered the optimal hyper-parameters. In some cases, if the minimum testing error is not very sensitive to the hyper-parameters, the simplest model is picked.

When the system order is increased, the number of input dimensions for the network (n_i) increases making the models computationally expensive both during training and prediction. Also, the input data itself may be correlated adding redundancy in input features which is undesirable from a training perspective. Hence a systematic method of principal component analysis (PCA) was performed to obtain a smaller set of uncorrelated feature components. In the following subsection, input feature extraction is performed using PCA where in the most significant subspace of transformed features are identified.

4.2.1. Principal component analysis

PCA is a common unsupervised learning technique used to convert a set of possibly correlated variables to a set of uncorrelated variables. PCA has been used as a pre-processing step during neural network modeling to reduce dimensionality of training inputs [63,64]. Consider the matrix of input features $X = [x_1 x_2 \cdots x_n]$ where $X \in \mathbb{R}^{m \times n}$ scaled to zero mean and unit co-variance. Let the scaled feature matrix be $\tilde{X} = [\tilde{x}_1 \tilde{x}_2 \cdots \tilde{x}_n]$. The covariance matrix (Σ) of the scaled feature matrix ($\tilde{X}$) is given by

$$\Sigma = \tilde{X}\tilde{X}^T \in \mathbb{R}^{m \times m} \tag{10}$$

Let the eigen values and eigen vectors of the co-variance matrix (Σ) be represented by

$$E_\Lambda = \{\Lambda_1, \Lambda_2, \Lambda_3, \ldots, \Lambda_m\} \tag{11}$$

$$E_v = \begin{bmatrix} v_1 & v_2 & v_3 & \cdots & v_m \end{bmatrix} \tag{12}$$

where $\Lambda_1 > \Lambda_2 > \Lambda_3 > \cdots > \Lambda_m$ represent the eigen values and v_i is the eigen vector corresponding to the eigen value Λ_i, $v_i \in \mathbb{R}^m$ and $E(x)$ is the expected value of x. Each eigen vector represents a component of the input data while the corresponding eigen value represents the variance. To capture the maximum variation within

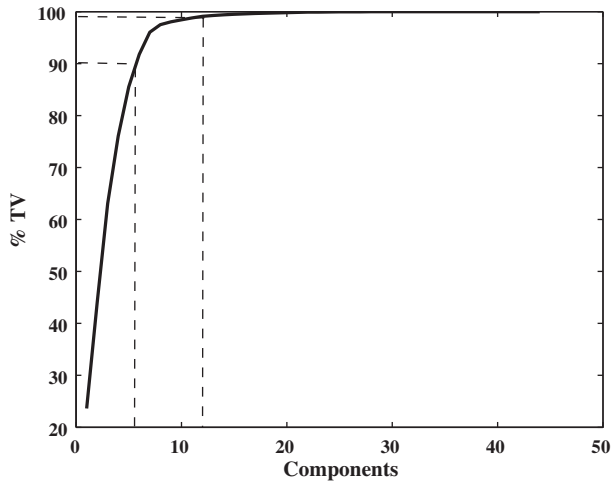


Fig. 4. Plot showing the percentage total variation with number of components for EAFR model with $n_u = n_y = 4$. The first 6 components capture 90% of TV while first 12 components capture 99% of TV where the total number of components is about 44.

the input data, the first l components are selected that retains 99% of the total variations. The percentage of total variation can be expressed as

$$\text{TV} = \left(\frac{\Lambda_1 + \Lambda_2 + \cdots + \Lambda_l}{\Lambda_1 + \Lambda_2 + \cdots + \Lambda_m}\right) \times 100 \tag{13}$$

Fig. 4 shows the percentage total variation with addition of components for the EAFR model with $n_u = n_y = 4$. It can be seen that the first 12 components captures majority of the variance. Precisely, the first 6 components capture 90% of TV while first 12 components capture 99% of TV where the total number of components is about 44. Similar observations can be made for each model and for varying system order. A constant TV cutoff of 99% was used to obtain principal components for both RBN and MLP training for all models. It should be noted that the cutoff was chosen arbitrarily high and if the cutoff is varied, the number of principal components obtained would be different.

In order to quantify the benefits of PCA, separate models were built with and without PCA and performance evaluated. Figs. 11–19 in the appendix shows the results of the full grid search for optimal hyper-parameters and the optimal combination that gives minimum validation error is marked with red. Table 3 lists the optimal hyper-parameters for the MLP and RBN models with and without PCA (the optimal hyper-parameter for the linear regression model is also included). It can be seen from Table 3 that pre-processing the input data using PCA is beneficial for both the MLP and RBN models. Similar orders of validation errors were obtained in both cases (with and without PCA) for all four models of NMEP, CA50, R_{max} and EAFR. However, the input layer dimensions (n_i) are small for the models with PCA indicating that a smaller subspace of input data is sufficient to map the underlying phenomena. Also, the training time required on a workstation having 3.47 GHz processor with 15 GB of RAM are compared for models with and without PCA. The time consumed for training the CA50 MLP model without PCA is about 2 times higher compared to training with PCA resulting in the same accuracy. For the NMEP RBN model, time consumed for training without PCA is 1.2 times higher compared to training with PCA yielding same accuracy levels. Also, the number of model parameters (defined in Section 5) are about 2–3 times lesser for the models with PCA indicating a storage benefit for these models.

5. Prediction results and discussion

The models with the selected hyper-parameters is freshly trained (along with PCA) using the complete training data set and evaluated against the unseen testing data set. The trained networks are simulated on the test data set in the same structure as used for training (series parallel architecture). This would evaluate the model for one-step-ahead prediction quality. The performance of the models are measured using mean squared error (MSE) given

Table 3
The optimal values of number of hidden neurons (n_h), regularization coefficient (λ in MLP and σ_h in RBN) and system order (n_u and n_y) determined using cross-validation. Here Err_{val} represents the minimum validation error in the grid search and n_i represents input layer dimension and n_p the number of parameters of the neural network model.

Architecture		Hyper-parameter	NMEP	CA50	R_{max}	EAFR
Linear regression	With PCA	$n_u = n_y$	5	5	5	1
	No PCA	$n_u = n_y$	5	5	5	5
MLP		n_h	10	10	10	10
		λ	0.01	0.1	1	0.0001
	With PCA	$n_u = n_y$	2	3	3	2
		Err_{val}	0.0011	0.0621	0.0502	0.0007
		n_i	9	11	9	9
		n_p	111	131	111	111
		n_h	8	10	10	8
		λ	0.0001	1	0.01	0.01
	No PCA	$n_u = n_y$	2	3	2	4
		Err_{val}	0.001	0.0621	0.0493	0.0001
		n_i	22	33	22	44
		n_p	193	351	262	369
RBN		n_h	200	160	200	200
		σ_h	1	10	1	10
	With PCA	$n_u = n_y$	2	3	2	2
		Err_{val}	0.0012	0.063	0.0518	0.0006
		n_i	9	11	12	9
		n_p	2001	1921	2601	2001
		n_h	200	200	200	120
		σ_h	10	1	1	10
	No PCA	$n_u = n_y$	3	2	2	4
		Err_{val}	0.0012	0.0617	0.0519	0.0001
		n_i	33	22	22	44
		n_p	6801	4601	4601	5401

Table 4
Prediction performance of MLP and RBN models with the PCA pre-processing. The results of linear regression model is also included as a benchmark. Here n_m and n_p represents the number of memory units and number of model parameters required for prediction. The training and testing errors are for one-step ahead prediction while MSAP error indicates the mean squared error for multiple-step ahead prediction. The minimum error among linear, MLP and RBN models is highlighted in bold.

		NMEP	CA50	Rmax	EAFR
MLP model	Training error	0.0011	0.0545	0.0473	0.0013
	Testing error	**0.0014**	**0.0608**	**0.0529**	0.0014
	nm	22	33	33	22
	np	131	161	171	131
	MSAP error	**0.0045**	**0.0786**	**0.0696**	0.0055
RBN model	Training error	0.0012	0.0564	0.0430	0.0009
	Testing error	0.0018	0.0651	0.0563	**0.0013**
	nm	22	33	22	22
	np	2401	1921	2401	2401
	MSAP error	0.0172	0.1535	0.0946	**0.0050**
Linear regression	Training error	0.0016	0.0730	0.0587	0.0048
	Testing error	0.0017	0.0772	0.0561	0.0047
	nm	55	55	55	11
	np	19	18	21	9
	MSAP error	0.0228	0.1157	0.0807	0.0189

by Eq. (14). The MSE for the training and testing phases for all the models are compared in Table 4.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{y_d}(y_j^i - \hat{y}_j^i)^2 \tag{14}$$

It can be observed from Table 4 that both MLP and RBF networks are able to learn the HCCI combustion dynamics to a good accuracy for one-step-ahead prediction. As a baseline, a linear regression model is selected and trained using the data set after performing cross-validation based hyper-parameter selection similar to the neural models. All three model structures – MLP, RBN and Linear regression have a similar order of accuracy for the considered engine variables. It can be seen that the minimum testing error is consistently achieved by the MLP models. Also, the MLP model results in small system orders compared to the other two architectures. The number of parameters are significantly less compared to RBN because RBN models store the centers of the radial basis neurons. It should be noted that the linear regression surprisingly performs well for one-step-ahead prediction for the considered engine variables. However, a large system order is required for the models indicating that even with PCA, the data has to remain in a high dimension feature space in order for linear models to capture the underlying behavior. More crucially, the linear models did not perform well under multi-step-ahead predictions (see MSAP error in Table 4 and Figs. 7–10) and hence linear models are found to be unsatisfactory for identifying the combustion behavior of HCCI.

Even though the two networks are very different in construction and training, both MLP and RBN networks are able to capture the underlying phenomena equally well. The order of training and testing errors also prove that the networks can generalize and can predict well for unseen inputs. The combined effect of PCA and cross-validation has resulted in models that are efficiently trained and have good performance in terms of prediction MSE. The

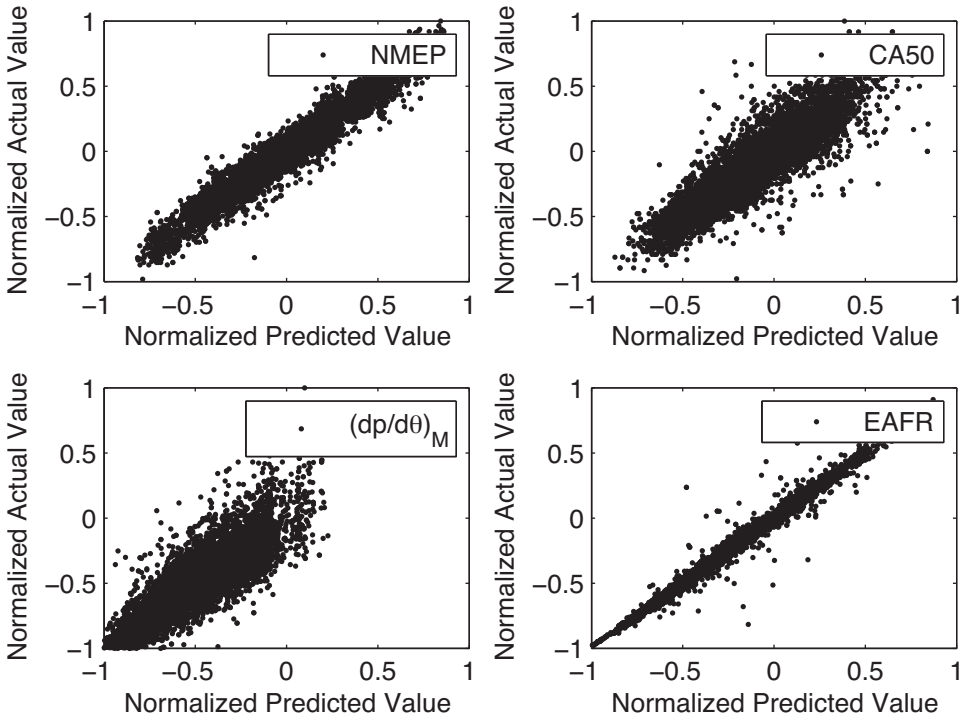


Fig. 5. Correlation between model predictions and true values (MLP model).

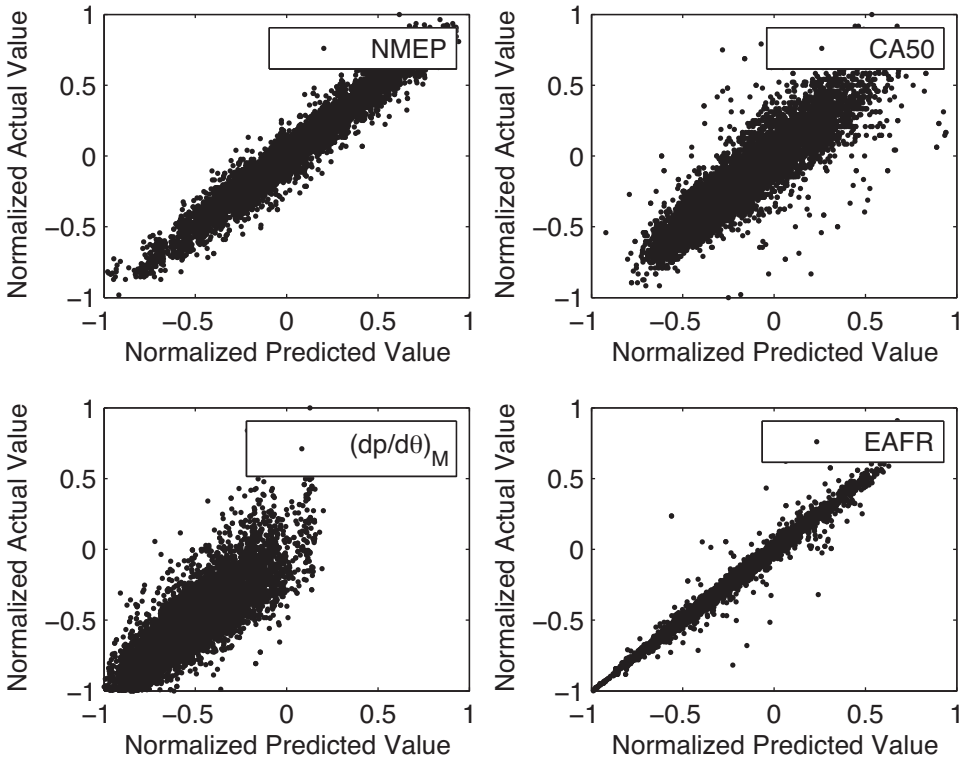


Fig. 6. Correlation between model predictions and true values (RBN model).

correlations between model predictions and the actual data for all models are shown using Figs. 5 and 6 indicating that the models perform well to unseen inputs over the entire range of network operation. It should be noted that the quantities NMEP, CA50 and R_{max} are unfiltered sensor signals and has a large variance while EAFR is a filtered signal obtained from the engine control unit and thus having smaller variance.

An important comparison between MLP and RBN models can be made with respect to the memory required (n_m) and the total number of parameters (n_p) used to fit the data. Memory in this context is referred to as the space required to store the measurement variables (like FM, SOI, NMEP, CA50, etc.) for prediction on-board in the engine control unit. The total number of parameters can be referred to as the parameters used to fit the data. This quantifies to the number of parameters determined by the training algorithm (a function of number of hidden neurons n_h and input feature dimension m). The memory requirement is given by $n_m = u_d n_u + y_d n_y$ while the number of parameters is given by $n_p = n_i n_h + n_h + n_h y_d + y_d$ for MLP network and $n_p = n_i n_h + n_h y_d + y_d$ for RBN models. However, it should be noted that the number of

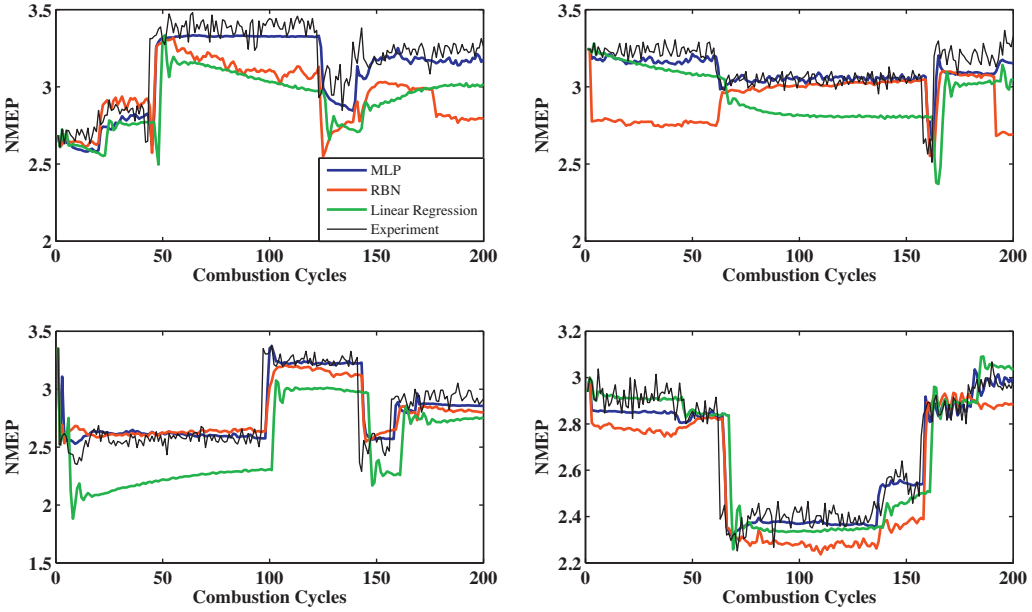


Fig. 7. Comparison of 200 step ahead prediction for NMEP by MLP, RBN and linear models with actual engine data. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

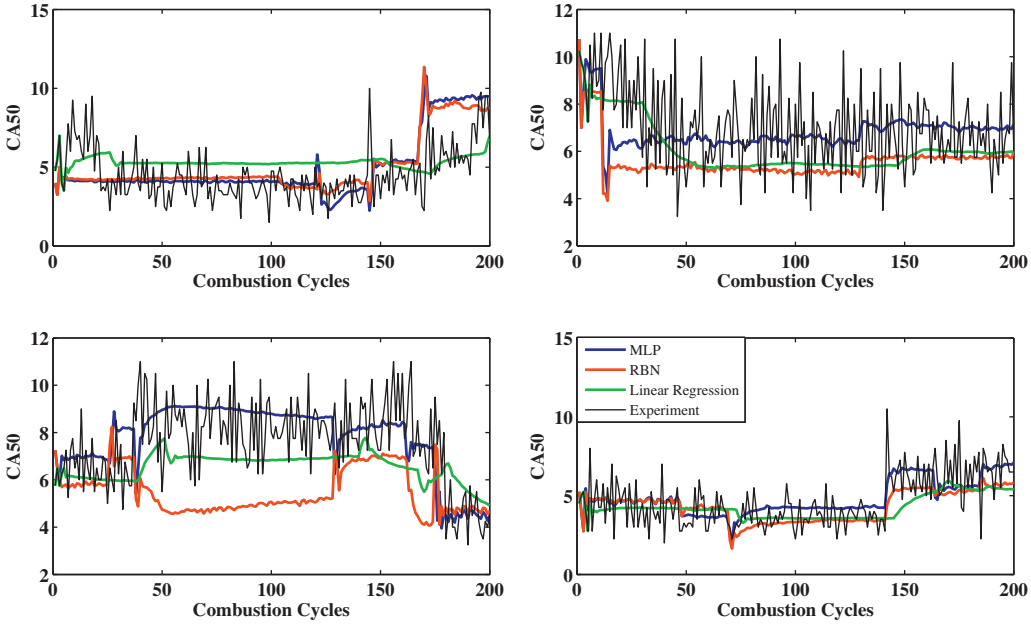


Fig. 8. Comparison of 200 step ahead prediction for CA50 by MLP, RBN and linear models with actual engine data. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

hidden neurons, n_h is very large in case of RBN, i.e., to achieve similar performance levels of MLP network, RBN requires extremely large number of parameters to fit the data. With further increase in the dimension of the signals and the number of observations, the size of the RBN model can increase significantly which can limit the RBN models from being implemented on an electronic control unit to perform real time predictions. It should be noted that the RBN model is relatively faster to train. However, the training time is not considered as a metric for comparison as training can be afforded to be done off-line for the application considered in this paper. Hence the MLP model is considered suitable for the HCCI identification problem both from prediction accuracy and storage perspectives.

5.1. Multi-step-ahead prediction

In order to observe the multi-step-ahead prediction capability of the models, a completely separate data set is used where the NN models are simulated with input of the form

$$[u(k-1), \ldots, u(k-n_u), \hat{y}(k-1), \ldots, \hat{y}(k-n_y)] \quad (15)$$

where k indicates present time index. An output feedback is made in the network to create a parallel architecture as shown in Fig. 1(b). Figs. 7–10 compares the 200-cycle predictions of the MLP and RBN models for unseen input trajectories. In each plot, four different input trajectories are presented and the networks' predictions summarized. The output quantities are plotted every combustion cycle

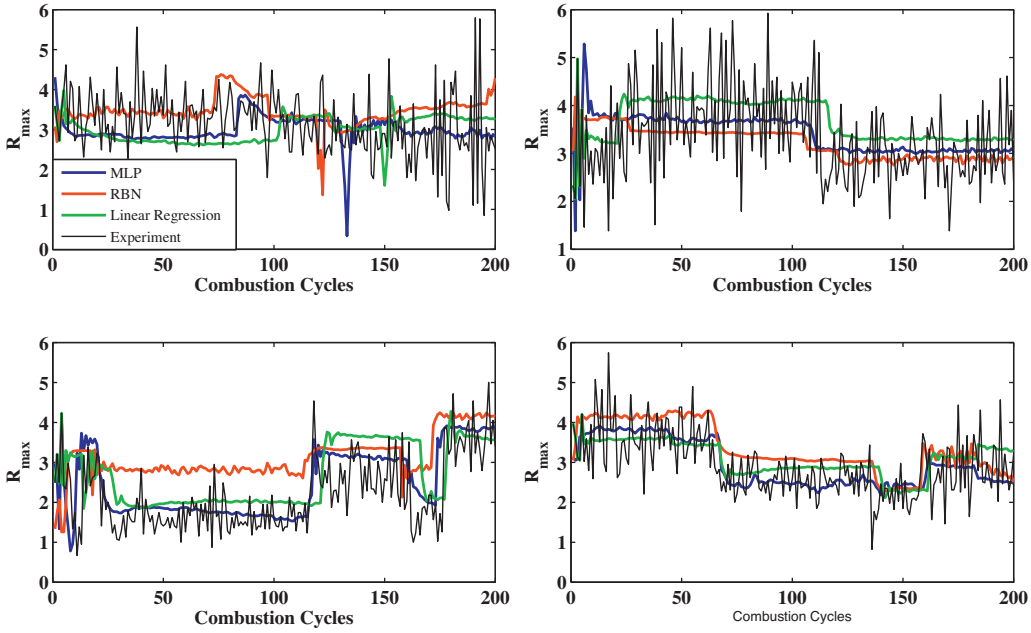


Fig. 9. Comparison of 200 step ahead prediction for maximum pressure rise rate by MLP, RBN and linear models with actual engine data. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

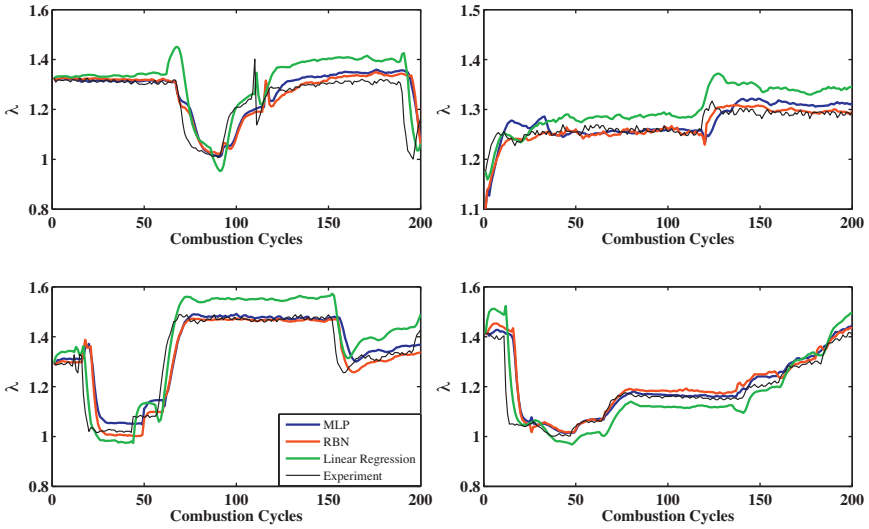


Fig. 10. Comparison of 200 step ahead prediction for EAFR by MLP, RBN and linear models with actual engine data. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

and each plot shows the predictions for about 8 seconds of engine data. The MSE for the multi-step ahead predictions are included in Table 4.

It can be seen from Figs. 7–10 and Table 4 that the linear models do not perform as well as they did for one-step-ahead predictions. The reason could be that the nonlinear maps are approximated by linear relationships that creates large bias errors at several operating conditions. Also, the model output is fed back for subsequent predictions in a multi-step-ahead prediction. It is important that the model possesses sufficient robustness and any poor predictions not affect the future predictions in a significant manner. The linear models appear to be lacking this feature compared to the nonlinear models. One possible reason could be over-fitting as the linear models fits the data with a high order when low order models [7,8] were found sufficient for the HCCI variables. It is also surprising to observe that the RBN models do not perform well at several operating conditions including Fig. 7 (top left subplot between 100 and 200 cycles, top right subplot between 0 and 50 cycles), Fig. 8 (bottom left subplot between 50 and 120 cycles), etc. A possible reason could be the local nature of the approximation captured by RBN. The local nature of RBNs might result in over-fitting (inability to generalize) if there are not enough training data in certain regions of interest. MLP models on the other hand perform reasonably well compared to the other two architectures. Both the steady-state and transient behavior is well captured. The good performance of MLP models may be attributed to the global nature of the sigmoidal activation function. Hence it is concluded that MLP models are sufficient for the considered HCCI variables for both one-step-ahead and multi-step-ahead predictions. The developed network mimics a dynamic model of the HCCI engine and can be used for further analysis and control purposes.

6. Conclusions

Developing accurate dynamic models can accelerate practical implementation of an advanced combustion technology like the HCCI engine. Neural network based identification offers an attractive alternative to physics based modeling in terms of cost and development times. The engine variables used for controller development such as NMEP, CA50, R_{max} and EAFR are modeled using available measurements from the engine. It is shown than using PCA, efficient recurrent neural network models can be built based on both MLP and RBN architectures. An unbiased comparison between the two networks along with a linear regression baseline show that all models are able to capture the one-step-ahead behavior of HCCI engine dynamics to a good accuracy but the MLP model outperforms the RBN and linear models in multi-step-ahead prediction which is the ultimate goal of this study. The MLP model also results in less memory and storage and has the potential to be implemented on-board the engine ECU. Future work would focus on capturing the effects of disturbance quantities such as engine coolant temperature, ambient temperature, pressure and humidity on HCCI combustion via an on-line learning framework using neural networks and controls development.

Role of the funding source

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Acknowledgements

This material is based upon work supported by the Department of Energy [National Energy Technology Laboratory] under Award Number(s) DE-EE0003533. This work is performed as a part of the ACCESS project consortium (Robert Bosch LLC, AVL Inc., Emitec Inc.) under the direction of PI Hakan Yilmaz, Robert Bosch, LLC.

Appendix A.

For all the plots from Fig. 11, the data for cross-validation is a subset of the training data which is divided into validation training and validation testing. The shaded surface represents validation training error while the unshaded surface represents validation testing error (see Figs. 12–19).

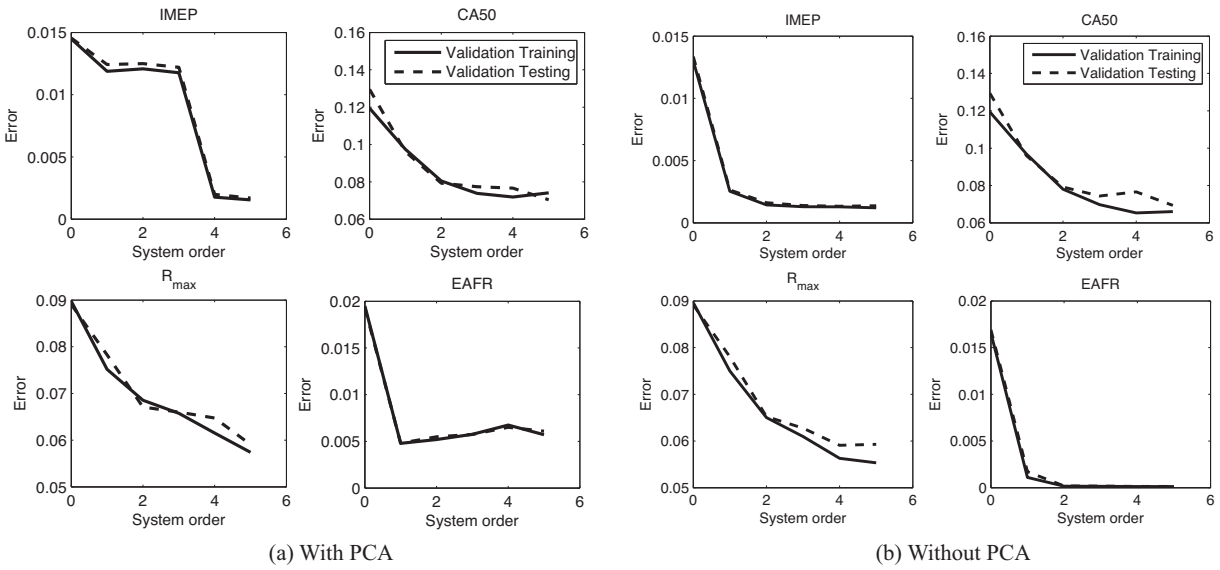


Fig. 11. Cross-validation for linear regression models using grid search showing training and validation error curves for different system order n_o.

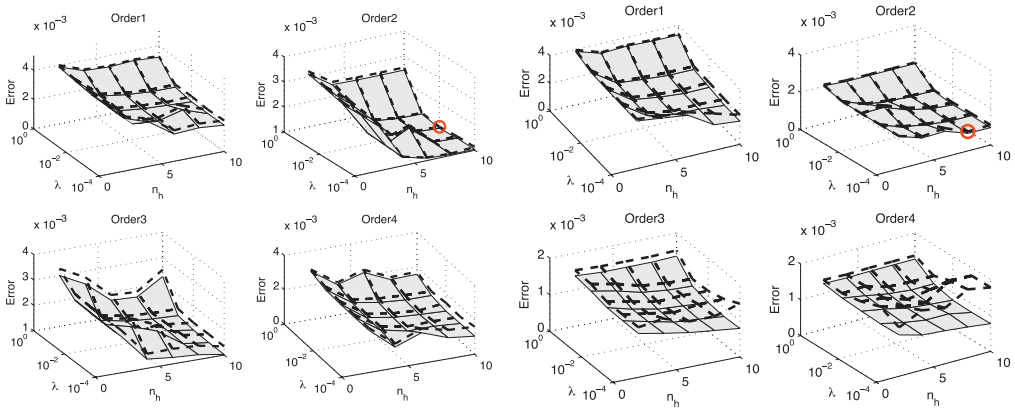


Fig. 12. Cross-validation for NMEP (MLP) model grid search showing training and validation error surfaces for different combinations of system order n_o, number of hidden neurons n_h and regularization coefficient λ. The optimal hyper-parameter combination is marked in red. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

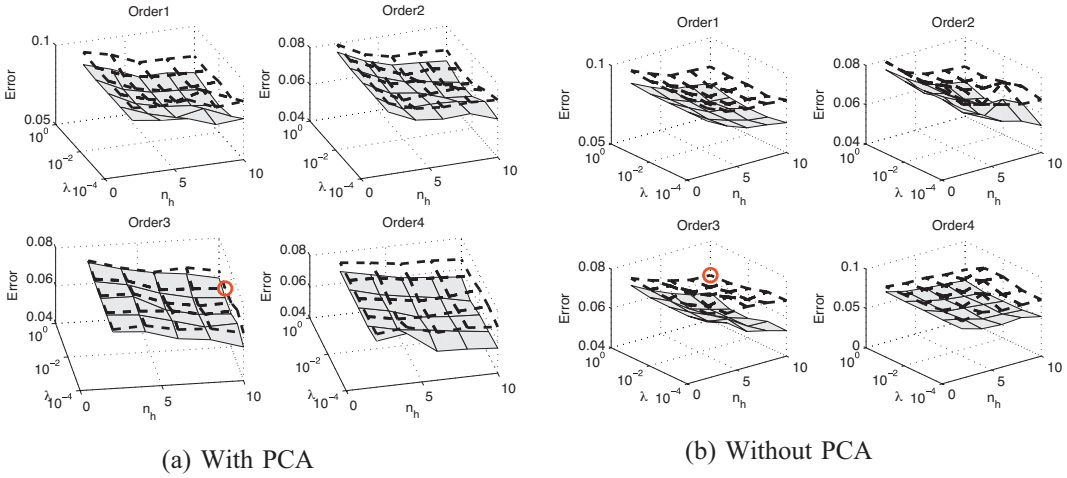


Fig. 13. Cross-validation for CA50 (MLP) model grid search showing training and validation error surfaces for different combinations of system order n_o, number of hidden neurons n_h and regularization coefficient λ. The optimal hyper-parameter combination is marked in red. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

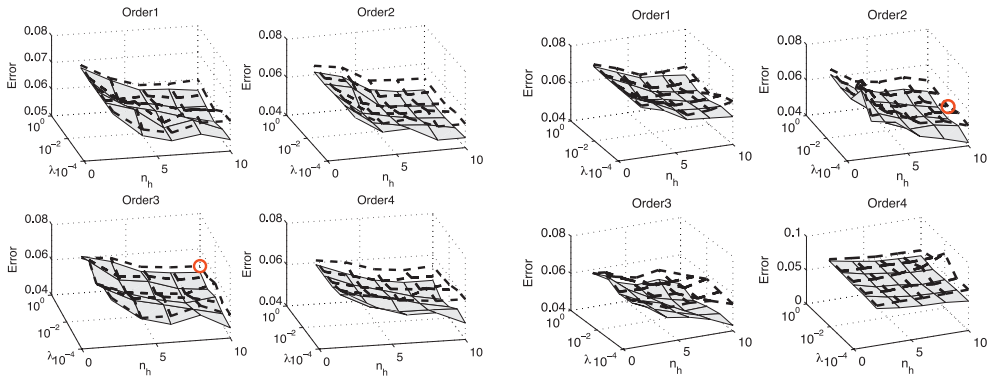


Fig. 14. Cross-validation for R_{max} (MLP) model grid search showing training and validation error surfaces for different combinations of system order n_o, number of hidden neurons n_h and regularization coefficient λ. The optimal hyper-parameter combination is marked in red. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

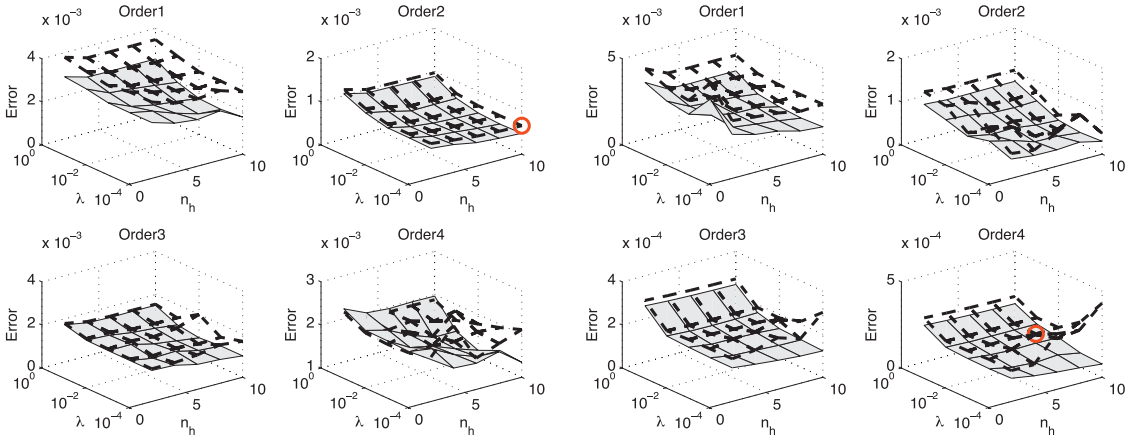


Fig. 15. Cross-validation for EAFR (MLP) model grid search showing training and validation error surfaces for different combinations of system order n_o, number of hidden neurons n_h and regularization coefficient λ. The optimal hyper-parameter combination is marked in red. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

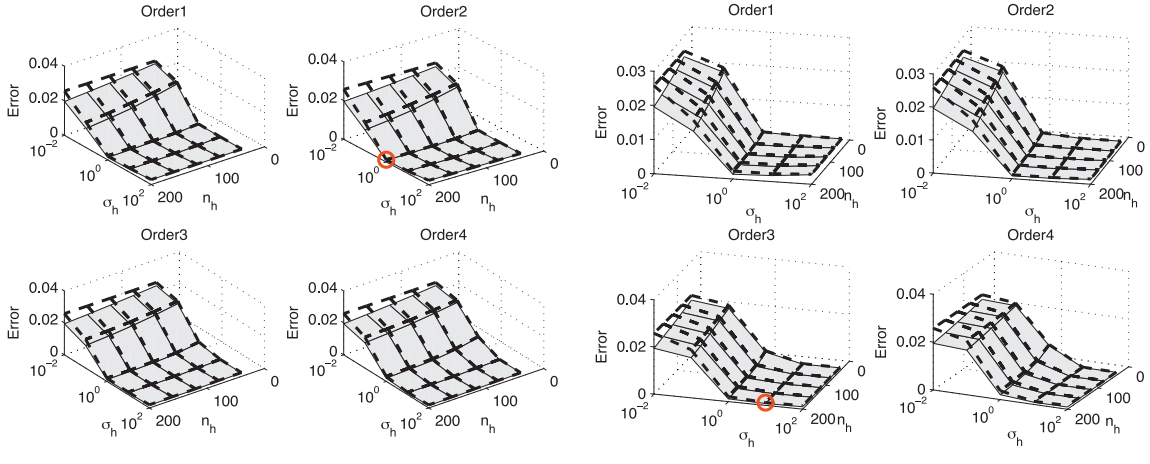


Fig. 16. Cross-validation for NMEP (RBN) model grid search showing training and validation error surfaces for different combinations of system order n_o, number of hidden neurons n_h and spread parameter σ_h. The optimal hyper-parameter combination is marked in red. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

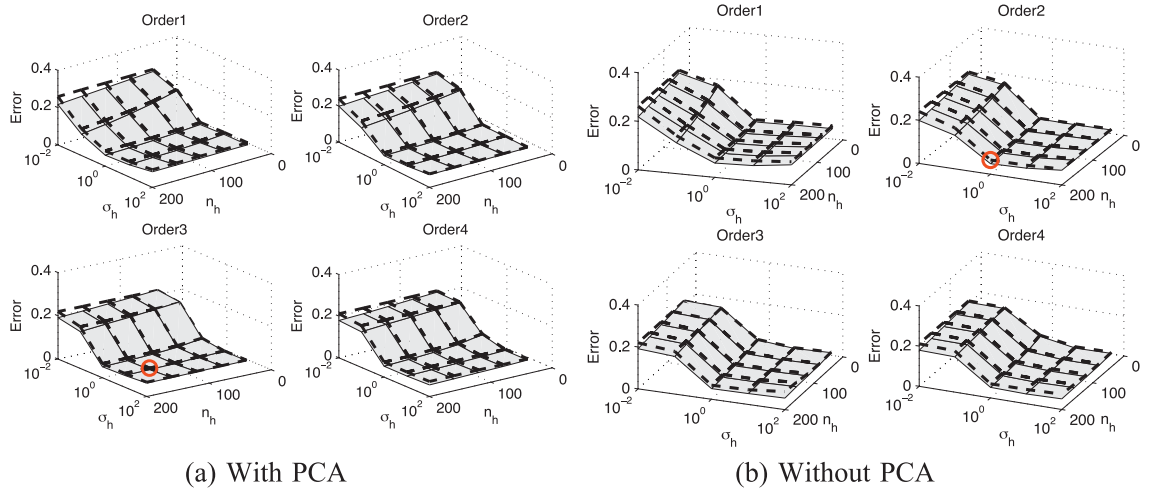


Fig. 17. Cross-validation for CA50 (RBN) model grid search showing training and validation error surfaces for different combinations of system order n_o, number of hidden neurons n_h and spread parameter σ_h. The optimal hyper-parameter combination is marked in red. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

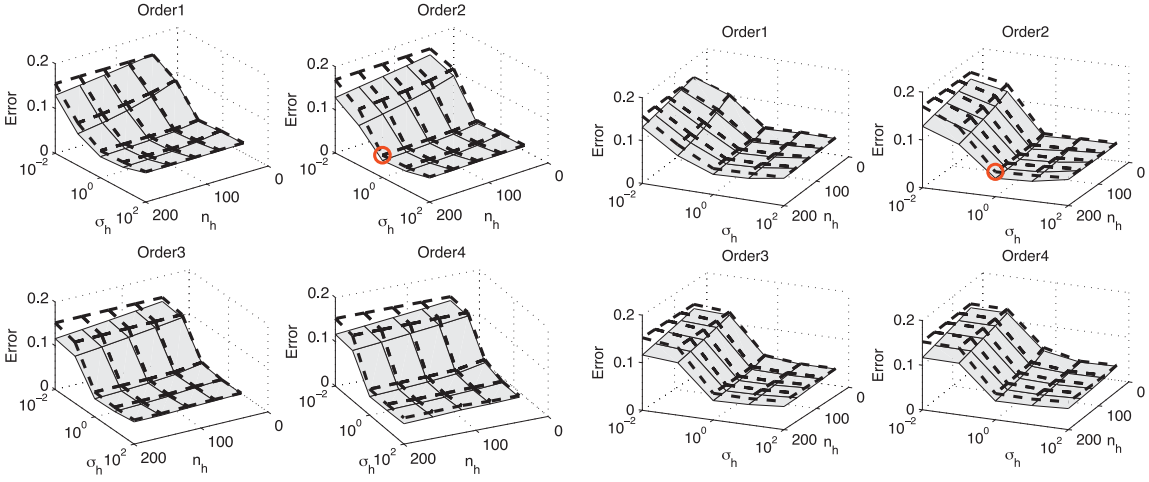


Fig. 18. Cross-validation for R_{max} (RBN) model grid search showing training and validation error surfaces for different combinations of system order n_o, number of hidden neurons n_h and spread parameter σ_h. The optimal hyper-parameter combination is marked in red. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

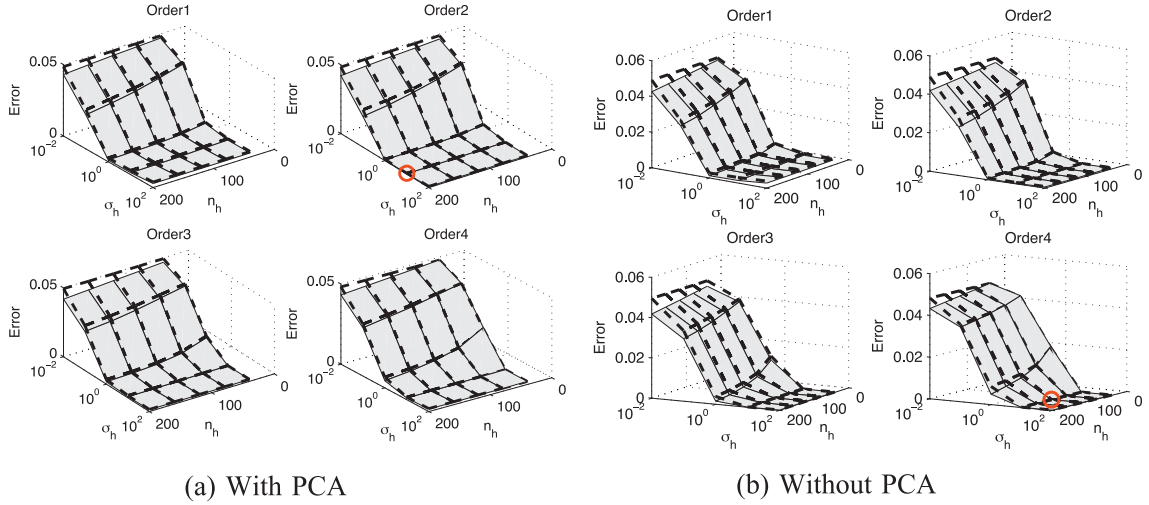


Fig. 19. Cross-validation for EAFR (RBN) model grid search showing training and validation error surfaces for different combinations of system order n_o, number of hidden neurons n_h and spread parameter σ_h. The optimal hyper-parameter combination is marked in red. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

References

[1] R. Thring, Homogeneous-charge compression-ignition engines, SAE Paper 892068, 1989.
[2] B. Johansson, M. Christensen, P. Einewall, Homogeneous charge compression ignition using iso-octane, ethanol and natural gas – a comparison to spark ignition operation, in: International Fuels & Lubricants Meeting & Exposition, Tulsa, OK, USA, SAE Paper 972874, 1997.
[3] J. Mizuta, Y. Sato, T. Aoyama, Y. Hattori, An experimental study on premixed-charge compression ignition gasoline engine, in: International Congress & Exposition, Detroit, MI, USA, SAE Paper 960081, 1996.
[4] R. Bechtold, K. Epping, S. Aceves, J. Dec, The potential of HCCI combustion for high efficiency and low emissions, in: SAE Powertrain & Fluid Systems Conference & Exhibition, SAE Technical Paper 2002-01-1923, San Diego, CA, 2002.
[5] G.H. Abd-Alla, Using exhaust gas recirculation in internal combustion engines: a review, Energy Conversion and Management 43 (2002) 1027–1042.
[6] C. Chiang, C. Chen, Constrained control of homogeneous charge compression ignition (HCCI) engines, in: 5th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2010.
[7] R. Johansson, P. Tunestal, J. Bengtsson, P. Strandh, B. Johansson, Model predictive control of homogeneous charge compression ignition (HCCI) engine dynamics, in: 2006 IEEE International Conference on Control Applications, 2006.
[8] H.H. Liao, A.F. Jungkunz, C.F. Chang, S. Park, N. Ravi, M.J. Roelle, J.C. Gerdes, Model-based control of HCCI engines using exhaust recompression, in: IEEE Transactions on Control Systems Technology, 2010.
[9] S.-C. Kong, A study of natural gas/DME combustion in HCCI engines using CFD with detailed chemical kinetics, Fuel 86 (2007) 1483–1489.
[10] Z. Zheng, M. Yao, Charge stratification to control HCCI: experiments and CFD modeling with n-heptane as fuel, Fuel 88 (2009) 354–365.
[11] Z. Wang, S.-J. Shuai, J.-X. Wang, G.-H. Tian, A computational study of direct injection gasoline HCCI engine with secondary injection, Fuel 85 (2006) 1831–1841.
[12] M. Yao, Z. Zheng, H. Liu, Progress and recent trends in homogeneous charge compression ignition (HCCI) engines, Progress in Energy and Combustion Science 35 (2009) 398–437.
[13] G.M. Shaver, J.C. Gerdes, M.J. Roelle, Physics-based modeling and control of residual-affected HCCI engines, Journal of Dynamic Systems, Measurement, and Control 131 (2009) 021002.
[14] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.
[15] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical system using neural networks, in: IEEE Transactions on Neural Networks, vol. 1, no. 1, 1990, pp. 4–27.
[16] S. Mukhopadhyay, K.S. Narendra, Adaptive control using neural networks and approximate models, in: IEEE Transactions on Neural Networks, vol. 8, no. 3, 1997.
[17] Y. Nakamura, K.I. Funahashi, Approximation of dynamical systems by continuous time recurrent neural networks, in: Neural Networks, vol. 6, 1993, pp. 801–806.
[18] K. Warwick, A. Delgado, C. Kabhainpati, Dynamical recurrent neural network for system identification and control, in: Proceedings of Control Theory Application, IEEE, vol. 142, no. 4, 1995, pp. 307–314.
[19] G. Feng Chak, C. Kwong, J. Ma, On the approximation capability of neural networks – dynamic system modeling and control, Asian Journal of Control (2001) 122–130.
[20] D.L. Michael, Y. Beham, Modeling a variable valve timing spark ignition engine using different neural networks, Journal of Automobile Engineering 218 (2004) 1159–1171.
[21] M. Saif Tan, Yonghong, Nonlinear dynamic modeling of automotive engines using neural networks, in: IEEE International Conference on Control Applications, 1997, pp. 408–410.
[22] J. Zhen-hua, N. Sheng-fang, L. Biao, L. Qing-chun, System identification of locomotive diesel engines with autoregressive neural network, in: 4th IEEE Conference on Industrial Electronics and Applications, 2009, pp. 3417–3421.
[23] M. Sorrentino, I. Arsie, C. Pianese, Development of recurrent neural networks for virtual sensing of nox emissions in internal combustion engines, SAE International Journal of Fuels and Lubrication 2 (2010) 354–361.
[24] C. Pianese, G. Rizzo, M. Sorrentino, I. Arsie, S. Di Iorio, Recurrent neural networks for air–fuel ratio estimation and control in spark-ignited engines, in: IFAC World Congress, Elsevier, 2008.
[25] D. Assanis, V. Janakiraman, J. Sterniak, Support vector machines for identification of HCCI combustion dynamics, in: 9th International Conference on Informatics in Control, Automation and Robotics (ICINCO).
[26] V.M. Janakiraman, X. Nguyen, J. Sterniak, D. Assanis, A system identification framework for modeling complex combustion dynamics using support vector machines, in: Lecture Notes in Electrical Engineering – Informatics in Control, Automation and Robotics, Springer, Berlin, Heidelberg, 2013.
[27] Y. Urata, M. Awasaka, J. Takanashi, T. Kakinuma, T. Hakozaki, A. Umemoto, A Study of Gasoline-Fuelled HCCI Engine Equipped with an Electromagnetic Valve Train, SAE Technical Paper 2004-01-1898, 2004, http://dx.doi.org/10.4271/2004-01-1898.
[28] R. Scaringe, C. Wildman, W.K. Cheng, On the high load limit of boosted gasoline HCCI engine operating in NVO mode, SAE International Journal of Engines 3 (2010) 35–45.
[29] C.L. Giles, T. Lin, B.G. Horne, Remembering the past: the role of embedded memory in recurrent neural network architectures, in: IEEE Workshop On Neural Networks For Signal Processing, IEEE Press, 1997, pp. 34–43.
[30] R.J. Williams, D. Zipser, Experimental analysis of the real-time recurrent learning algorithm, Connection Science 1 (1989) 87–111.
[31] J.L. Elman, Finding structure in time, Cognitive Science 14 (1990) 179–211.
[32] V. Kurková, Kolmogorov's theorem and multilayer neural networks, Neural Networks 5 (1992) 501–506.
[33] A. Ranganathan, The Levenberg–Marquardt Algorithm, 2004 http://www.sciencedirect.com/science/article/pii/S0968090X12000575
[34] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the marquardt algorithm, IEEE Transactions on Neural Networks 5 (1994) 989–993.
[35] R.S. Sexton, R.E. Dorsey, J.D. Johnson, Optimization of neural networks: a comparative analysis of the genetic algorithm and simulated annealing, European Journal of Operational Research 114 (1999) 589–601.
[36] N.M. Barnes, A.W. O'Neill, D. Wood, Rapid, supervised training of a two-layer, opto-electronic neural network using simulated annealing, Optics Communications 87 (1992) 203–206.
[37] B. Cohen, D. Saad, E. Marom, Efficient training of recurrent neural network with time delays, Neural Networks 10 (1997) 51–59.
[38] A. Blanco, M. Delgado, M.C. Pegalajar, A real-coded genetic algorithm for training recurrent neural networks, Neural Networks 14 (2001) 93–105.
[39] R.S. Sexton, J.N.D. Gupta, Comparative evaluation of genetic algorithm and backpropagation for training neural networks, Information Sciences 129 (2000) 45–59.
[40] J. Park, I.W. Sandberg, Universal approximation using radial-basis-function networks, Neural Computation 3 (1991) 246–257.
[41] E. Hartman, J.D. Keeler, J.M. Kowalski, Layered neural networks with Gaussian hidden units as universal approximations, Neural Computation 2 (1990) 210–215.
[42] T. Chen, R. Chen, Approximation capability to functions of several variables, nonlinear functionals and operators by radial basis function neural networks, IEEE Transactions on Neural Networks 6 (4) (1995) 904–910.
[43] A.D. Niros, G.E. Tsekouras, A novel training algorithm for rbf neural network using a hybrid fuzzy clustering approach, Fuzzy Sets and Systems 193 (2012) 62–84.
[44] A. Alexandridis, H. Sarimveis, G. Bafas, A new algorithm for online structure and parameter adaptation of RBF networks, Neural Networks 16 (2003) 1003–1017.
[45] A. Staiano, R. Tagliaferri, W. Pedrycz, Improving rbf networks performance in regression tasks by means of a supervised fuzzy clustering, Neurocomputing 69 (2006) 1570–1581.
[46] A. Guillén, J. González, I. Rojas, H. Pomares, L.J. Herrera, O. Valenzuela, A. Prieto, Using fuzzy logic to improve a clustering technique for function approximation, Neurocomputing 70 (2007) 2853–2860.
[47] H.M. Liu, S.P. Wang, P.C. Ouyang, Fault diagnosis in a hydraulic position servo system using RBF neural network, Chinese Journal of Aeronautics 19 (2006) 346–353.
[48] A. Taghavipour, M.S. Foumani, M. Boroushaki, Implementation of an optimal control strategy for a hydraulic hybrid vehicle using CMAC and RBF networks, Scientia Iranica 19 (2) (2012) 327–334.
[49] D. David Casasent, X.W. Chen, New training strategies for RBF neural networks for X-ray agricultural product inspection, Pattern Recognition 36 (2003) 535–547.
[50] J.B. Gomm, D.L. Yu, Selecting radial basis function network centers with recursive orthogonal least squares training, IEEE Transactions on Neural Networks 11 (2000) 306–314.
[51] S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, IEEE Transactions on Neural Networks 2 (1991) 302–309.
[52] D.-S. Huang, W.-B. Zhao, Determining the centers of radial basis probabilistic neural networks by recursive orthogonal least square algorithms, Applied Mathematics and Computation 162 (2005) 461–473.
[53] X. Lu, W. Chen, Z. Huang, A fundamental study on the control of the HCCI combustion and emissions by fuel design concept combined with controllable EGR. Part 2: Effect of operating conditions and EGR on HCCI combustion, Fuel 84 (2005) 1084–1092.
[54] J.E. Dec, M. Sjberg, Isolating the effects of fuel chemistry on combustion phasing in an HCCI engine and the potential of fuel stratification for ignition control, 2004.
[55] C.J. Chiang, A.G. Stefanopoulou, Dynamics of homogeneous charge compression ignition (HCCI) engines with high dilution, in: American Control Conference, 2007, pp. 2979–2984.
[56] G.M. Shaver, Stability analysis of residual-affected HCCI using convex optimization, Control Engineering Practice 17 (2009) 1454–1460.
[57] R.P. Fitzgerald, R. Steeper, J. Snyder, R. Ronald Hanson, R. Hessel, Determination of cycle temperatures and residual gas fraction for HCCI negative valve overlap operation, SAE International Journal of Engines 3 (2010) 124–141.
[58] M.C. Weikl, F. Beyrau, A. Leipertz, Simultaneous temperature and exhaust-gas recirculation-measurements in a homogeneous charge-compression ignition engine by use of pure rotational coherent anti-stokes Raman spectroscopy, Applied Optics 45 (2006) 3646–3651.
[59] M.M. Andreae, W.K. Cheng, T. Kenney, J. Yang, On HCCI Engine Knock, SAE Technical Paper 2007-01-1858, 2007, http://dx.doi.org/10.4271/2007-01-1858.
[60] T. Johansson, B. Johansson, P. Tunestal, H. Aulin, HCCI operating range in a turbocharged multi cylinder engine with vvt and spray-guided di, SAE 2009-01-04, 2009.

[61] J. Hyvnen, G. Haraldsson, B. Johansson, Supercharging HCCI to Extend the Operating Range in a Multi-Cylinder VCR-HCCI Engine, SAE Technical Paper 2003-01-3214, 2003, http://dx.doi.org/10.4271/2003-01-3214.
[62] T. Lang, T. Kruse, S. Kurz, Modern statistical modelling and evolutionary optimisation methods for the broad use in ECU calibration, in: 6th IFAC Symposium Advances in Automotive Control, 2010.
[63] A. Bucinski, T. Baczek, J. Krysinski, R. Szoszkiewicz, J. Zaluski, Clinical data analysis using artificial neural networks (ANN) and principal component analysis (PCA) of patients with breast cancer after mastectomy, Reports of Practical Oncology & Radiotherapy 12 (2007) 9–17.
[64] Y. Xiao, Y. He, A novel approach for analog fault diagnosis based on neural networks and improved kernel PCA, Neurocomputing 74 (2011) 1102–1115.