IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 30, 2018, accepted November 28, 2018, date of current version May 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2885199

Aero-Engine On-Board Model Based on Batch Normalize Deep Neural Network

QIANGANG ZHENG, JUAN FANG, ZHONGZHI HU, AND HAIBO ZHANG

Nanjing University of Aeronautics and Astronautics, Jiangsu Province Key Laboratory of Aerospace Power System, Nanjing 210016, China

Corresponding author: Qiangang Zheng (zhqg@nuaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 51576096, in part by Q. Lan and the 333 Project, in part by the Research Funds for Central Universities under Grant NF2018003, and in part by the Six Talents Peak Project of Jiangsu Province.

ABSTRACT A new on-board turbo-fan engine modeling method based on a batch normalize (BN) mini-batch gradient descent (MGD) deep neural network (NN) is proposed. This new method adopts BN algorithm, which accelerates the network training speed and overcomes the gradient vanish problem. Hence, using the BN algorithm, the neural network adopts the deeper structure, which means the network has a stronger representation capacity. This mini-batch gradient descent (MGD-NN) algorithm that consumes much less time to update the NN parameters is adopted. Therefore, it is more suitable for training big dataset and establishing a high-accuracy engine model in a large flight envelope. Finally, to verify whether the proposed method could be applied to larger flight envelope, the conventional NN also adopts MGD (called MGD-NN). The turbo-fan engine models based on these two modeling methods are both conducted within a sub-sonic cruise envelope. The simulation results show that the proposed modeling method has much higher accuracy than the MGD-NN. Moreover, the proposed method has the characteristics of less data storage, low computation complexity, and good real-time performance, which are the most importance indices for model realize on-board.

INDEX TERMS Aero-engine model, batch normalize, deep neural network, turbo-fan on-board model, mini-batch gradient descent, data storage.

I. INTRODUCTION

Aero-engine model always plays a key role in engine control system design. If the necessary simulation is conducted in the mathematic engine model, instead of real one, there will be many advantages, such as reducing the costs of research and development, decreasing the accident risks, moreover shortening the development period [1]. For fully utilizing the performance of the controlled plat, the modern aero-engine control systems are always adopting model-based one as reported in NASA Intelligent Engine Control (IEC) [2]–[4]. The modern advanced engine control methods such as Performance Seeking Control (PSC) [5], Life Extending Control (LEC) [6] and Fault-Tolerant Control (FTC) [8], always require an on-board engine model with high accuracy and real time performance to track unmeasured parameters. Hence, how to establish a high accuracy and real-time model for Full Authority Digital Electronic Controller (FADEC) is the key technology to realize modern control methods [9], [10].

The most popular on board aero-engine modeling method is piecewise linear modeling. The main advantage of this method is that it has well real-time performance [9]–[12]. However, cumulative errors that are caused by piecewise process inevitably exist. Hence, a Support Vector Regression (SVR) modeling method was adopted to establish engine models [13], [14].Unfortunately, the real time performance of SVR will be increase rapidly along with the increase of training data. For a large fight envelope, it always sets up many sub SVR models that will increase its data storage [14]–[20].

The Neural Network (NN), due to the ability of approximating nonlinear function and good real-time performance, had attracted a lot of interests in engine modeling [21], [22]. However, the optimization method of conventional NN always adopts Batch Gradient Descent (BGD) method [23], which computes all the gradients of the entire training data when updating the NN parameters. This method consumes a lot of time and limits its application to huge training data. Usually, the traditional NN is three layered neural network. In order to increase accuracy of the predictive model, the conventional NN should increase the node of hidden layer, which will cause the NN overfitting. The researchers realize that the increase of the hidden layers will greatly improve the model fitting capacity and model precious [24].

In addition, the multi-layers hidden layers extract the attributive feature from low-level to high-level, which realize feature extract automatically. However, how to train the deep neural network is a main obstacle for its application.

Fortunately, Hinton in 2006 proposed Deep Belief Networks (DBNs), which uses the unsupervised learning procedure for restricted Boltzmann machines to pre-train one hidden layer at a time. It is a novel and effective way to train deep neural networks [25]. It reignites the research interest in NN field. After that, a lot of breakthrough about Deep Learning has sprung up. Such as, the error of object recognition had greatly decreased by using deep convolutional neural networks [26], [27]. Document [28] applied deep neural networks to acoustic modeling in speech recognition, which is the first major industrial application of deep learning. Sutskever *et al.* [29] proposed sequence-to-sequence learning with neural networks and get a state-of-the-art machine translation results. Faster R-CNN is proposed [30], which realizes real-time object detection with region proposal networks. The BN (Batch Normalization), which is treated a major breakthrough of the deep learning, was proposed [31]. This method could avoid gradient disappeared and gradient overflow problem through normalizing the nodes of network layer by layer. Moreover, the BN could accelerate neural networks training speed about five to twenty times and could play as regularization technology, which will improve the generalization of the network.

For the optimization technology, a Stochastic Gradient Descent (SGD) just computes the gradient information for representative training point at each iteration [32]. It has much faster convergence speed than BGD and might be suitable for huge data training. However, the SGD might be sensitivity to the noise data and might not be the best decent direction for only selecting one training. Hence, Mini-batch Gradient Descent (MGD) which is a compromised between BGD and SGD was proposed [32]–[34]. It cost less time for training NN with big data than BGD and has better descent direction than SGD.

Therefore, a new aero-engine on-board modeling method, which adopts BN-MGD-DNN, is proposed here. Simulations of MGD-NN proposed by Khan and Sahai [35] before and the proposed modeling method are both carried out in a subsonic cruise envelope. Compared to MGD-NN, the results show that the proposed method has high precision and better generalization.

II. THE PRINCIPLE OF BN-MGD-DNN

In this section, the details of the loss function, the structure, and the back forward algorithm of BN-MGD-DNN are described.

A. THE LOSS FUNCTION OF BN-MGD-DNN

The Back Propagation (BP) algorithm is always used to train NNs. NNs have variant loss functions. The most suitable loss function is Summed Squared Error (SSE). For explaining the advantages of BN-MGD-DNN, the cost functions of BGD and SGD are also given as follows.

The loss function of BGD, which is commonly used by the conventional NN, can be described as follows:

$$J(\mathbf{W},\mathbf{b};\mathbf{x},\mathbf{y}) = \min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{N} \frac{1}{2}\left\|\mathbf{h}(\mathbf{x}_i)-\mathbf{y}_i\right\|^2 \tag{1}$$

where $\mathbf{x}$ is an input vector, $\mathbf{y}$ is an output vector, N is the size of training set, $\mathbf{h}(\mathbf{x}_i)$ is the output of the neural network of the n-th hidden layer, $\mathbf{W}$ is weight matrix and $\mathbf{b}$ is offset vector. From Eq. (1), it can be inferred that the SSE of BGD needs to compute the whole training set to update NN parameters, which is why the conventional NN has much computation complexity and consumes longer time to train.

For the SGD NN, the loss function is defined as:

$$J(\mathbf{W},\mathbf{b};\mathbf{x},\mathbf{y}) = \min_{\mathbf{W},\mathbf{b}} \frac{1}{2}\left\|\mathbf{h}(\mathbf{x}_i)-\mathbf{y}_i\right\|^2 \tag{2}$$

It can be inferred that the SGD calculates SSE only through one training point. Hence, it cost less time to train NN. However, its training result of NN is sensitive to noise data.

Hence, the MGD method is proposed [33], [34]. The training set of MGD is randomly divided into M batches with same size N_b. The loss function of MGD can be defined as following:

$$J_1(\mathbf{W},\mathbf{b};\mathbf{x},\mathbf{y}) = \min_{\mathbf{W},\mathbf{b}} \sum_{j=1}^{N_b} \frac{1}{2}\left\|\mathbf{h}(\mathbf{x}_{b_i,j})-\mathbf{y}_{b_i,j}\right\|^2 \tag{3}$$

where $\mathbf{x}_{b_i,j}$ is the jth input vector of batch b_i, $\mathbf{y}_{b_i,j}$ is the jth output vector of batch b_i, $\sum b_i = N$, $i = 1, 2, \cdots M$, $\bigcup_i^M \bigcup_j^{b_i} \mathbf{x}_{b_i,j} = \bigcup_i^N \mathbf{x}_i$, $\bigcup_i^M \bigcup_j^{b_i} \mathbf{y}_{b_i,j} = \bigcup_i^N \mathbf{y}_i$. The loss function of MGD is calculated by using the sub-training sets, instead of the entire training set or one training point. That is why the MGD costs less time to train NN than BGD, and has much higher accuracy than SGD.

B. THE STRUCTURE OF BN-MGD-DNN

The BN-MGD-DNN is a non-linear mapping for multi-input $\mathbf{x}_i$ and multi-output $\mathbf{y}_i$ system, where $i = 1, 2, \cdots N$, N is the size of the training set. The multi-layer BN-MGD-DNN has more than three hidden layer, and each hidden layer can be described as:

$$\mathbf{a}^{l+1} = \mathbf{W}^l\mathbf{h}^l + \mathbf{b}^l \tag{4}$$

$$\bar{\mathbf{a}}^{l+1} = \mathbf{g}(\mathbf{a}^{l+1}) \tag{5}$$

$$\mathbf{h}^{l+1} = \sigma(\bar{\mathbf{a}}^{l+1}) \tag{6}$$

where $\mathbf{W}^l$ is weight matrix and $\mathbf{b}^l$ is bias (or offset) vector, $\boldsymbol{\sigma}$ is activation function, $\mathbf{h}_i^0 = \mathbf{x}_i$ is the input of the neural net, $\mathbf{h}_i^l$ (for $l > 0$) is the output of the k-th hidden layer, $l = 1, 2, \cdots n_{layer}$, n_{layer} is the number of layers and $\mathbf{g}$ is Batch Normalizing Transform.

The structure of BN-MGD-DNN and conventional DNN are shown in Figure 1 and Figure 2. Compared with conventional DNN, the proposed BN-MGD-DNN adds a BN layer

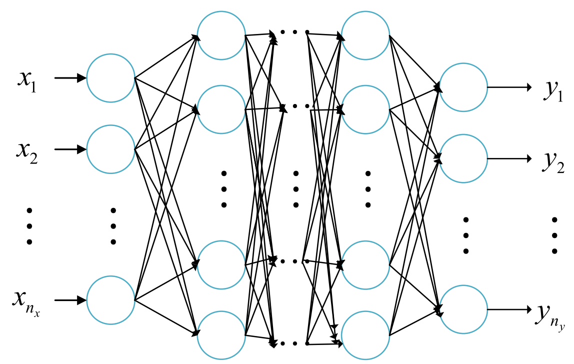

FIGURE 1. The structure of DNN.

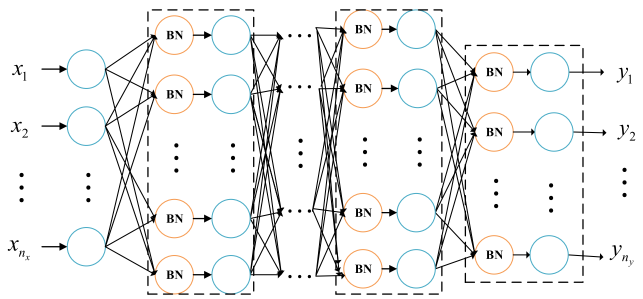

FIGURE 2. The structure of BN-DNN.

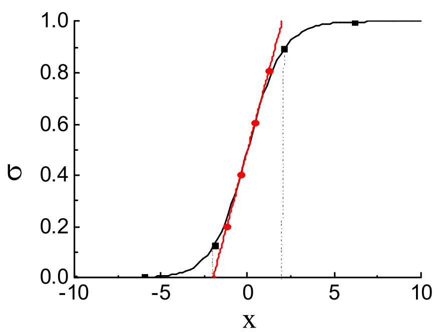

FIGURE 3. Sigmoid curve.

before the input of hidden layer note. The $\mathbf{g}$ is as follows:

$$\hat{a}_{i,j}^l = \frac{a_{i,j}^l - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{7}$$

where ε is a very small positive number, and

$$\mu_B = \frac{1}{N_b}\sum_{j=1}^{N_b} a_{i,j}^l \tag{8}$$

$$\sigma_B^2 = \frac{1}{N_b}\sum_{j=1}^{N_b} (a_{i,j}^l - \mu_B)^2 \tag{9}$$

Simply normalizing each input of a layer note may change the network representation capacity. For instance, as shown in Figure 3, if the activation function is sigmoid function, then the normalizing will constraint the input to the linear regime of the nonlinearity. To address this, for each activation, a pair of parameters γ, β is introduced. These two parameters scale and shift the normalized value:

$$\bar{a}_{i,j}^l = \gamma \hat{a}_{i,j}^l + \beta \tag{10}$$

where γ, β are learned as the original model parameters. In addition, they could restore the representation power of the network.

C. BACK-PROPAGATION ALGORITHM

The training method of neural network usually uses genetic algorithm, particle swarm optimization [35], gradient descent method, conjugate gradient method, or quasi-Newton method. Among them, the most popular one is gradient descent method [36]. At each iteration for updating the parameters, $\mathbf{b}$, γ, β can be described as:

$$W_{ij}^l \leftarrow W_{ij}^l + \eta \nabla W_{ij}^l \tag{11}$$

$$b_i^l \leftarrow b_i^l + \eta \nabla b_i^l \tag{12}$$

$$\gamma_i^l \leftarrow \gamma_i^l + \eta \nabla \gamma_i^l \tag{13}$$

$$\beta_i^l \leftarrow \beta_i^l + \eta \nabla \beta_i^l \tag{14}$$

where η is the learning rate. Back-propagation is applied to solve the gradient of network parameters. The principle of back-propagation is shown in Figure 4. Supposing $\bar{\boldsymbol{\delta}}^{n_{net}}$ as:

$$\begin{aligned}\bar{\boldsymbol{\delta}}^{n_{net}} &= \frac{\partial J(\mathbf{W}, \mathbf{b}; \mathbf{x}, \mathbf{y})}{\partial \bar{\mathbf{a}}^{n_{net}}} = \frac{\partial J(\mathbf{W}, \mathbf{b}; \mathbf{x}, \mathbf{y})}{\partial \mathbf{h}^{n_{net}}} \otimes \frac{\partial \mathbf{h}^{n_{net}}}{\partial \bar{\mathbf{a}}^{n_{net}}} \\ &= \frac{\partial J(\mathbf{W}, \mathbf{b}; \mathbf{x}, \mathbf{y})}{\partial \mathbf{h}^{n_{net}}} \otimes \frac{\partial \mathbf{h}^{n_{net}}}{\partial \bar{\mathbf{a}}^{n_{net}}}\end{aligned} \tag{15}$$

where n_{net} is the number of network layer, $\otimes$ is hadamard product, that is $\mathbf{x} \otimes \mathbf{y} = [x_1y_1, x_2y_2, \cdots x_ny_n]^T$.

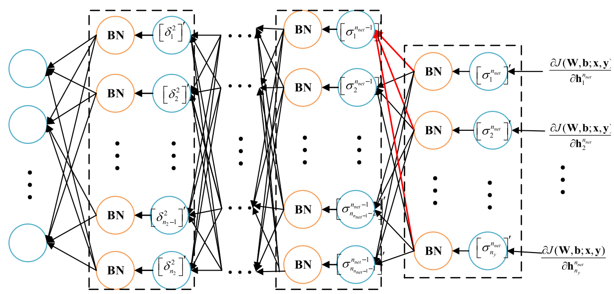

FIGURE 4. The principle of Back-propagation algorithm.

Supposing $\boldsymbol{\delta}^l$:

$$\boldsymbol{\delta}^l = \frac{\partial J}{\partial a_{i,j}^l} = \frac{\partial J}{\partial \hat{a}_{i,j}^l}\frac{1}{\sqrt{\sigma_B^2 + \varepsilon}} + \frac{\partial J}{\partial \sigma_B^2}\frac{2(a_{i,j}^l - \mu_B)}{N_b} + \frac{\partial J}{\partial \mu_B}\frac{1}{N_b} \tag{16}$$

where $l = n_{net}, n_{net} - 1, \cdots, 2$, and

$$\frac{\partial J}{\partial \hat{a}_{i,j}^l} = \left[\bar{\delta}_{i,j}^l\right]' \gamma_i^l$$

$$\frac{\partial J}{\partial \sigma_B^2} = \sum\nolimits_{j \in \chi_k} \frac{\partial J}{\partial \hat{a}_{i,j}^l}(a_{i,j}^l - \mu_B)\frac{-1}{2}(\sigma_B^2 + \varepsilon)^{-3/2}$$

TABLE 1. The change range of model input.

	H /km	Ma	W_{fb}	A_8	α_f	α_c
Minimum	9	0.65	$W_{fb,pla=30}$	$A_{8,ds}$	-4°	-4°
Maximum	13	0.9	120% $W_{fb,pla=70}$	130% $A_{8,ds}$	4°	4°

$$\frac{\partial J}{\partial \mu_B} = \left(\sum\nolimits_{j\in\chi_k} \frac{\partial J}{\partial \hat{a}_{i,j}^l}\frac{-1}{\sqrt{\sigma_B^2+\varepsilon}}\right) + \frac{\partial J}{\partial \sigma_B^2}\frac{\sum_{j\in\chi_k} -2(a_{i,j}^l-\mu_B)}{N_b} \tag{17}$$

Then for $l = n_{net}-1, n_{net}-2, \cdots, 2$, $\bar{\boldsymbol{\delta}}^l$ is available.

$$\bar{\boldsymbol{\delta}}^l = \left[\mathbf{W}^l\right]^T \boldsymbol{\delta}^{l+1} \otimes \left[\boldsymbol{\sigma}^l\right]' \tag{18}$$

Computing the desired partial derivatives which are given as:

$$\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},\mathbf{y})}{\partial W_{ij}^l} = \mathbf{h}_j^l \boldsymbol{\delta}_j^{l+1} \tag{19}$$

$$\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},\mathbf{y})}{\partial b_j^l} = \boldsymbol{\delta}_j^{l+1} \tag{20}$$

$$\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},\mathbf{y})}{\partial \gamma_i^l} = \sum\nolimits_{j\in\chi_k} \left[\bar{\delta}_{i,j}^l\right]' \hat{a}_{i,j}^l \tag{21}$$

$$\frac{\partial J(\mathbf{W},\mathbf{b};\mathbf{x},\mathbf{y})}{\partial \beta_i^l} = \sum\nolimits_{j\in\chi_k} \left[\bar{\delta}_{i,j}^l\right]' \tag{22}$$

III. ON-BOARD REAL-TIME ENGINE MODEL

Through the above discussion, the nonlinear mapping model of BN-MGD-DNN could be establish as follow:

$$\mathbf{y} = f_{BN-MGD-DNN}(\mathbf{x})$$

where $\mathbf{x}$ is model input and $\mathbf{y}$ model output.

For different engine control method, such as PSC, model predictive control, this on-board model input and output will be different. In this paper, the model is applied to performance seeking control in the follow-up research. Hence, it selects flight height H and flight Mach number engine and control variables such as Ma, fuel flow W_{fb}, exhaust nozzle throat area A_8, the variable inlet guide vane of fan α_f, the variable inlet guide vane of compressor α_c as the input. The output of engine model chooses specific fuel consumption S_{fc}, engine thrust F, fan rotor speed N_f, compressor rotor speed N_c, fan surge margin S_{mf}, compressor surge margin S_{mc} and high turbine inlet temperature T_4

IV. BN-MGD-DNN MODELING AND SIMULATIONS

To verify the effectiveness of the proposed method, an on-board engine model with a large flight envelope is set up and validated. For comparison, the same simulation of the popular modeling method MGD-NN [37], which could be

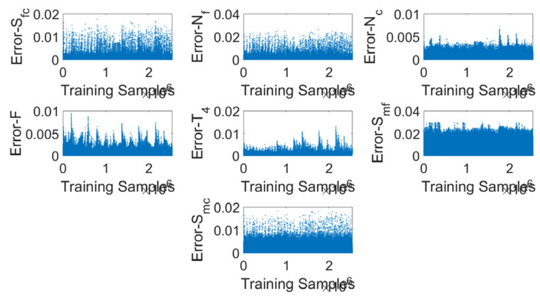

FIGURE 5. The training relative errors of BN-MGD-DNN.

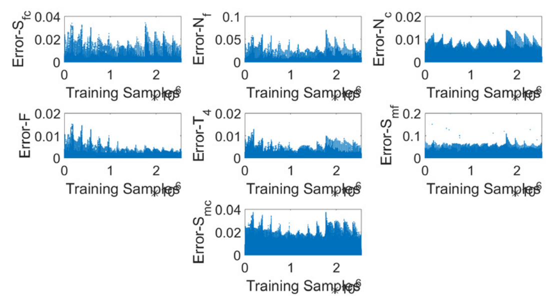

FIGURE 6. The training relative errors of MGD-NN.

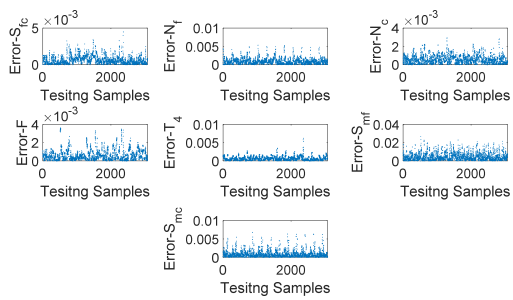

FIGURE 7. The testing relative errors of BN-MGD-DNN.

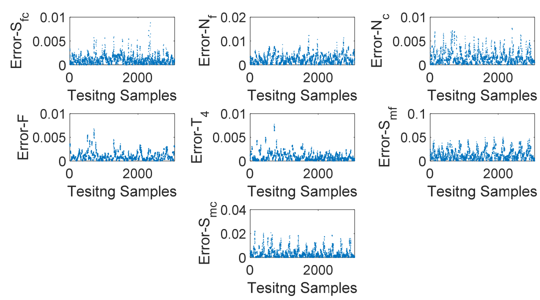

FIGURE 8. The testing relative errors of MGD-NN.

applied to big training data set, will also be utilized with a same data samples set herein.

The engine model input ranges are shown in Table 1, where $W_{fb,pla=30}$ is the fuel flow when power level angle $Pla = 30°$, $W_{fb,pla=70}$ is the fuel flow when $Pla = 70$. $A_{8,ds}$ is the engine design point exhaust nozzle throat area. For ensuring the predictive precision of the model, the CLM are fully simulated in the large input set. The number of training set

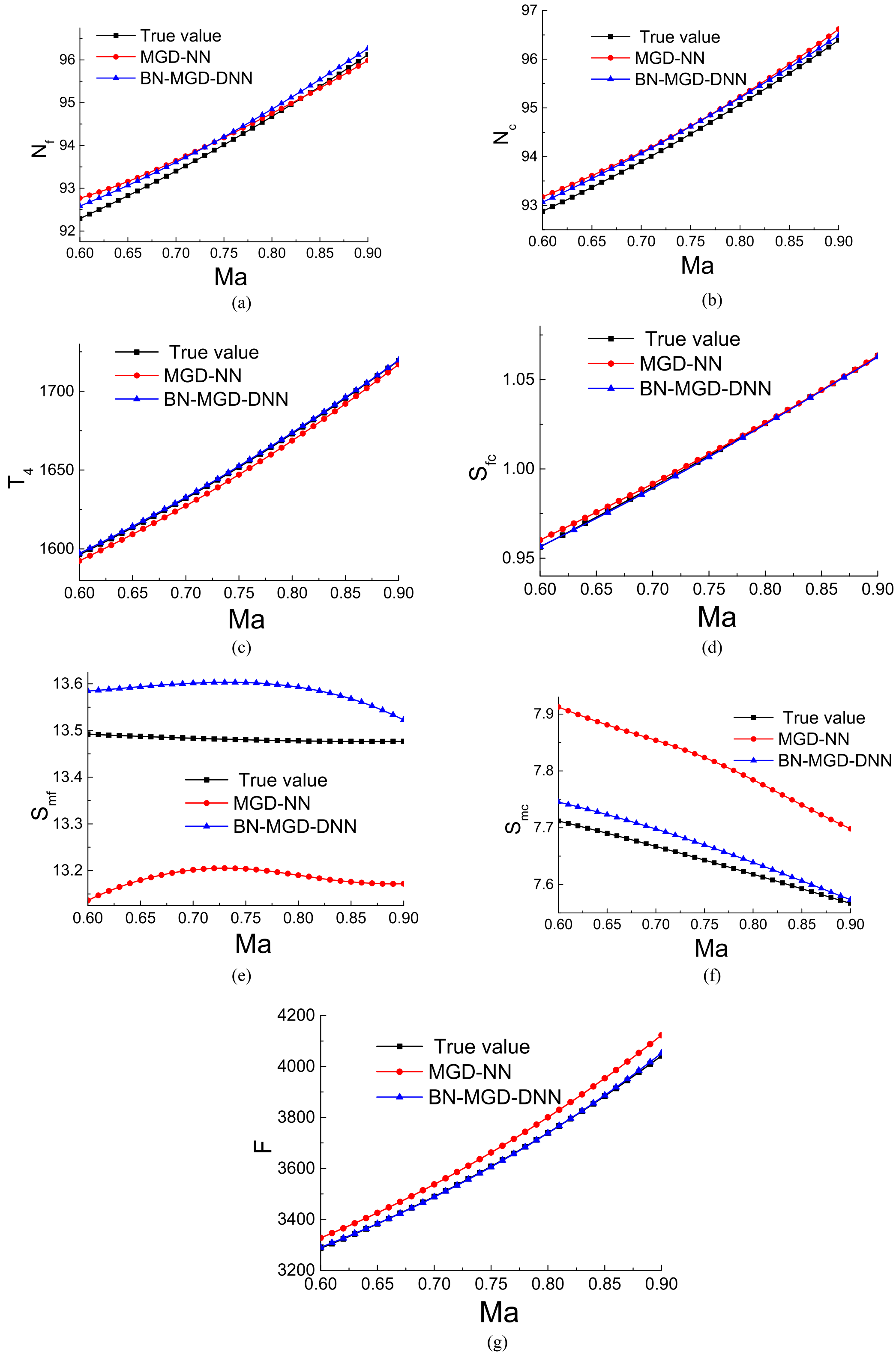

FIGURE 9. The curves of engine parameters when $Pla = 70°$. (a) The curve of N_f. (b) The curve of N_c. (c) The curve of T_4. (d) The curve of S_{fc}. (e) The curve of S_{mf}. (f) The curve of S_{mc}. (g) The curve of F.

is 2,548,260, which means impossible training for support vector machine. Moreover, the number of testing set is 3072.

Though debugging, the structures of BN-MGD-DNN and MGD-NN are chosen as [6,8,10,10,8,7] and [6,40,7] respectively. The mini-batch number is chosen as 3000. The regulation parameter is $\lambda = 10^{-8}$.

Figure 5 and Figure 6 give the training relative errors of BN-MGD-DNN and MGD NN respectively. The relative error is defined as:

$$error = \left| \frac{\tilde{y} - y}{y} \right| \tag{23}$$

where y is the real value and $\tilde{y}$ is the predictive value of model. It can be inferred from Figure 5 and Figure 6 that the training relative errors of BN-MGD-DNN are within 3% and meet required precision. What's more, the precisions of BN-MGD-DNN are much higher than the precision of MGD-NN. Especially the precisions of S_{fc}, N_f, S_{mf} and S_{mc} are twice higher than the ones of MGD-NN. The testing relative errors of BN-MGD-DNN and MGD-NN are shown in Figure 7 and Figure 8. The errors of BN-MGD-DNN are within 1% except the error of S_{mf}, which within 3%. Compared with MGD-NN, the errors of BN-MGD-DNN are twice higher except T_4. Figure 9 shows the predictive values of engine parameters along with the change of Mach number when $Pla = 70°$. It also shows that the predictive of BN-MGD-DNN model are much better than MGD-NN.

Table 2 gives mean squared error (MSE) of MGD-NN and BN-MGD-DNN. It can be inferred that the BN-MGD-DNN has better training accuracy and testing accuracy than MGD-NN. Compared with MGD-NN, the training MSEs of BN-MGD-DNN decrease by 1.4, 2.17, 2.0, 1.3, 1.13, 2.4 and 2.8 time. The testing MSEs of BN-MGD-DNN are decrease by 1.75, 2.0, 2.3, 1.3, 1.3, 2.3 and 3.3 time.

TABLE 2. The mean squared error (MSE) of MGD-NN and BN-MGD-DNN(%).

	S_{fc}	N_f	N_c	F	T_{41}	S_{mf}	S_{mc}
Test (DNN)	0.05	0.06	0.069	0.08	0.097	0.25	0.13
Test (NN)	0.07	0.13	0.14	0.1	0.11	0.59	0.37
Train (DNN)	0.04	0.06	0.06	0.07	0.08	0.25	0.1
Train (NN)	0.07	0.12	0.14	0.09	0.1	0.57	0.33

TABLE 3. Comparison for BN-MGD-DNN and MGD NN.

	Data storage (double)	Computation complexity	Average testing time
MGD-NN	567	614	0.067ms
BN-MGD-DNN	579	622	0.103ms

Table 3 gives the data storage, computation complexity and average testing time of MGD-NN and the proposed method.

The data storage of MGD-NN is 567 (weights 520($6\times40+40\times7$) + bias 47 ($40+7$)).

The data storage of the proposed method is 579 (weights 364 ($6\times8+8\times10+10\times10+10\times8+8\times7$) + bias 43($8+10+10+8+7$) + μ_B43($8+10+10+8+7$) + $\sigma_B^2$43($8+10+10+8+7$) + γ43($8+10+10+8+7$) + β43($8+10+10+8+7$)).

The computation complexity of MGD-NN is 614 (multiplication operation 520($6\times40+40\times7$) + addition operation 47 ($40+7$) + active function 47 ($40+7$)).

TABLE 4. Nomenclature.

Symbol		Explanation
BN	=	Batch normalize
MGD	=	Mini-batch gradient descent
DNN	=	Deep neural network
H	=	Flight height
Ma	=	Flight Mach number
W_{fb}	=	Fuel flow
A_8	=	Exhaust nozzle throat area
α_f	=	Variable inlet guide vane of fan
α_c	=	Variable inlet guide vane of compressor
S_{fc}	=	Specific fuel consumption
F	=	Engine thrust
T_4	=	High pressure turbine inlet temperature
N_f	=	Fan rotor speed
N_c	=	Compressor rotor speed
S_{mf}	=	Fan surge margin
S_{mc}	=	Compressor surge marge

And the calculate amount of BN-MGD-DNN is 622 (multiplication operation 407 ($6\times8+8\times10+10\times10+10\times8+8\times7+8+10+10+8+7$) + division operation 43 ($8+10+10+8+7$) + addition operation 86($8+10+10+8+7+8+10+10+8+7$) + subtraction operation 43($8+10+10+8+7$) + active function 43 ($8+10+10+8+7$)).

These two programs have the same testing running environments: Windows 7 Ultimate with Service Pack 1 (×64); Matlab 2016a; Intel i5-4590; the RAM is 8G. The testing times of the MGD-NN and the proposed model modeling method are 0.067 millisecond and 0.103 millisecond respectively.

Therefore, compared to the conventional neural network – MGD NN, the proposed modeling method has much higher testing and training accuracy while maintain the characteristics of low data storage, low computation complexity and good real time performance. All of these performance indexes are the most importance index to decide whether it can be applied to be an on-board engine model modeling method. Hence, the proposed method can be applied to establish an on-board real-time engine model.

V. CONCLUSIONS

Through the simulation tests for on-board engine model with the BN-MGD-DNN and MGD-NN, some conclusions can be summarized. Through the introduction of mini-batch gradient descent and L_2 regulation, these two methods can be applied to big training data and can be applied as modeling method in larger flight envelop. The proposed method has better generalization performance than the conventional neural network MGD-NN. The main reason is that the BN-MGD-DNN has deeper layer and has stronger representation capacity. As shown in the simulation results, the proposed modeling method is more suitable for establishing an on-board real time engine model.

APPENDIX

See Table 4.

REFERENCES

[1] S. Jianguo, "Prospects of the aero-engine control development in the early time of the 21st century," *J. Aerosp. Power*, vol. 16, no. 2, pp. 97–102, 2001.
[2] S. Garg, "Propulsion controls and diagnostics research in support of NASA aeronautics and exploration mission programs," NASA Glenn Res. Center, Cleveland, OH, USA, Tech. Rep. NASA TM-216939, 2011.
[3] H. Richter, *Advanced Control of Turbofan Engines*. New York, NY, USA: Springer Science & Business Media, 2011.
[4] S. Garg, "Propulsion controls and health management research at NASA Glenn research center," NASA Glenn Res. Center, Cleveland, OH, USA, Tech. Rep. NASA/TM-211590, Apr. 2002.
[5] F. Sun, L. Miao, and H. Zhang, "A study on the installed performance seeking control for aero-propulsion under supersonic state," *Int. J. Turbo Jet-Eng.*,vol. 33, no. 4,pp. 341–351, 2016.
[6] Q. Zheng, H. Zhang, L. Miao, and F. Sun, "On-board real-time optimization control for turbo-fan engine life extending," *Int. J. Turbo Jet-Engines*, vol. 34, no. 4, pp. 321–332, 2017.
[7] D. L. Simon and J. B. Armstrong, "An integrated approach for aircraft engine performance estimation and fault diagnostics," ASME, New York, NY, USA, Tech. Rep. GT-69905, 2012.
[8] S. Garg, "Aircraft turbine engine control research at NASA Glenn research center," NASA Glenn Res. Center, Cleveland, OH, USA, Tech. Rep. NASA/TM-217821, 2013.
[9] T. Brotherton, A. Volponi, R. Luppold, and D. L. Simon, "eSTORM: Enhanced self tuning on-board real-time engine model," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2003, pp. 3075–3086.
[10] A Volponi and D. L. Simon, "Enhanced self tuning on-board real-time model (eSTORM) for aircraft engine performance health tracking," Pratt Whitney Aircraft, East Hartford, CT, USA, Tech. Rep. NASA/CR-215272, 2008.
[11] J. S. Litt, "An optimal orthogonal decomposition method for Kalman filter-based turbofan engine thrust estimation," *J. Eng. Gas Turbines Power*, vol. 130, no. 1, pp. 1–12, 2008.
[12] T. U. J. Grönstedt, "Identifiability in multi-point gas turbine parameter estimation problems," in *ASME Turbo Expo, Power Land, Sea, Air*. New York, NY, USA: The American Society of Mechanical Engineers, 2002, pp. 9–17.
[13] C.-W. Fei and G.-C. Bai, "Wavelet correlation feature scale entropy and fuzzy support vector machine approach for aeroengine whole-body vibration fault diagnosis," *Shock Vib.*, vol. 20, no. 2, pp. 341–349, 2013.
[14] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, 1997, pp. 155–161.
[15] Q. Zheng, L. Miao, H. Zhang, and Z. Ye, "On-board real-time optimization control for turbofan engine thrust under flight emergency condition," *Proc. Inst. Mech. Eng. I, J. Syst. Control Eng.*, vol. 231, no. 7, pp. 554–566, 2017.
[16] Q. Zheng and H. Zhang, "A global optimization control for turbo-fan engine acceleration schedule design," *Proc. Inst. Mech. Eng. G, J. Aerosp. Eng.*, vol. 232, no. 2, pp. 308–316, 2018.
[17] Q. Zheng, Z. Xu, H. Zhang, and Z. Zhu, "A turboshaft engine NMPC scheme for helicopter autorotation recovery maneuver," *Aerosp. Sci. Technol.*, vol. 76, pp. 421–432, May 2018.
[18] W. Jiankang, Z. Haibo, Y. Changkai, D. Shujing, and H. Xianghua, "An adaptive turbo-shaft engine modeling method based on PS and MRR-LSSVR algorithms," *Chin. J. Aeronaut.*, vol. 26, no. 1, pp. 94–103, 2013.
[19] B. Gu, V. S. Sheng, Z. Wang, D. Ho, S. Osman, and S. Li, "Incremental learning for v-support vector regression," *Neural Netw.*, vol. 67, pp. 140–150, Jul. 2015.
[20] Z. Qiangang and Z. H. Yongjin, "Research on simplex B-splines algorithm in on-board steady state modeling of turbofan engines," *J. Propuls. Technol.*, vol. 12, no. 36, pp. 1887–1894, 2015.
[21] S. Hussain, M. Mokhtar, and J. M. Howe, "Sensor failure detection, identification, and accommodation using fully connected cascade neural network," *IEEE Trans. Ind. Electron.*, vol. 62, no. 3, pp. 1683–1692, 2015.
[22] K. Xu, M. Xie, L. C. Tang, and S. L. Ho, "Application of neural networks in forecasting engine systems reliability," *Appl. Soft Comput.*, vol. 2, no. 4, pp. 255–268, 2003.
[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[24] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003,
[25] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[26] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, 2014.
[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*. 2012, pp. 1097–1105.
[28] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
[29] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[31] S. Ioffe and C. Szegedy. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [Online]. Available: https://arxiv.org/abs/1502.03167
[32] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging SGD," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 685–693.
[33] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science), vol. 7700, G. Montavon, G. B. Orr, and K. R. Müller, Eds. Berlin, Germany: Springer, 2012, doi: 10.1007/978-3-642-35289-8_25.
[34] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of The Trade*. Berlin, Germany: Springer, 2012, pp. 9–48.
[35] K. Khan and A. Sahai, "A comparison of BA, GA, PSO, BP and LM for training feed forward neural networks in e-learning context," *Int. J. Intell. Syst. Appl.*, vol. 4, no. 7, p. 23, 2012.
[36] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation method," *Biol. Cybern.*, vol. 59, nos. 4–5, pp. 257–263, 1988.
[37] Q. Zheng, H. B. Zhang, Y. Li, and Z. Hu, "Aero-engine on-board dynamic adaptive MGD neural network model within a large flight envelope," *IEEE Access*, vol. 6, pp. 45755–45761, 2018.

QIANGANG ZHENG received the Ph.D. degree from the Nanjing University of Aeronautics and Astronautics (NUAA), in 2018. He is currently a Teacher with the College of Energy and Power Engineering, NUAA. His research interests include modeling, fault diagnosis, optimization control, and model predictive control of aero-engines.

JUAN FANG is currently pursuing the M.S. degree in aerospace propulsion theory and engineering with the College of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics. Her research interests include on-board adaptive aero-engine model and performance seeking control.

ZHONGZHI HU received the B.S. degree in automatic control from the Beijing Institute of Technology, China, and the M.S. and Ph.D. degrees in electrical engineering from The University of British Columbia, Canada. He has almost 30 years of extensive experience in modeling and simulation, controls and PHM for gas turbine, wind turbine, and other industry applications, as a Researcher and an Engineer, including 15 years of research and development and engineering positions with General Electric. He is currently a Professor with the Nanjing University of Aeronautics and Astronautics, Nanjing, China. He has published over 40 refereed conference and journal papers and holds five U.S. patents. His research interests are in gas turbine modeling, simulation, controls, and PHM.

HAIBO ZHANG received the Ph.D. degree from the Nanjing University of Aeronautics and Astronautics (NUAA), in 2005. He is currently a Professor with the College of Energy and Power Engineering, NUAA. His main research interests include modeling, fault diagnosis, and the control of aero-engines.