Data-Driven Approaches for Modelling Train Control Models: Comparison and Case Studies

Jiateng Yin[1]*, Shuai Su[1]†, Jing Xun[1], Tao Tang[1], Ronghui Liu[2]

1. *State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, 100044, China*

2. *Institute for Transport Studies, University of Leeds, Leeds LS29JT, UK*

September 2, 2019

Abstract

In railway systems, the train dynamics are usually affected by the external environment (e.g., snow and wind) and wear-out of on-board equipment, leading to the performance degradation of automatic train control algorithms. In most existing studies, the train control models were derived from the mechanical analyzation of train motors and wheel-track frictions, which may require many times of field trials and high costs to validate the model parameters. To overcome this issue, we record the explicit train operation data in Beijing Metro within three years and develop three data-driven approaches, involving a linear regression-based model (LAM), a nonlinear regression-based model (NRM), and furthermore a deep neural network based (DNN) model, where the LAM and NRM can act as benchmarks for evaluating DNN. To improve the training efficiency of DNN model, we especially customize the input and output layers of DNN, batch normalization based layers and network parameter initialization techniques according to the unique characteristics of railway train models. From the model training and testing results with field data, we observe that DNN significantly enhances the predicting accuracy for the train control model by using our customized network structure compared with LAM and NRM models. These data-driven approaches are successfully applied to Beijing Metro for designing efficient train control algorithms.

Keywords: Train control models; Data-driven approaches; Artificial neural networks; Field test

1 Introduction

Due to the rapid development of communication, control and computer technologies in the last several decades, Automatic Train Operation (ATO) system has been developed and widely implemented in many urban rail transit lines throughout the world to replace manual driving. With ATO systems in urban railways, the driver no longer has to cautiously operate the train control handle, which remarkably reduces the manual labors for driving trains. Meanwhile, the rail line capacity is also enhanced with the aid of accurate monitoring and advanced control algorithms of ATO systems (Yin et al., 2017). More recently, the unattended train operation systems, in which there is no driving cabin but only one serving staff, were put into operation in Paris Metro Line 14 and Dubai Metro, which may represent the future direction of railway transportation systems.

In modern urban rail systems, the on-board ATO systems drive the trains automatically by tracking a pre-given position-velocity trajectory with designed control algorithms (e.g., feedback control and model predictive control

*E-mail address: jtyin@bjtu.edu.cn (J. Yin)
†shuaisu@bjtu.edu.cn (S. Su)

(Qiu et al., 2019)). However, a practically significant issue associated with ATO system is the inaccuracy of train control models under different environment, such as the equipment wear and tear, extreme weather (e.g., rain, snow) or the variance of track and air frictions, as shown by Figure 1. Especially, the initial train control models provided by the manufactory will become inaccurate with variant model parameters due to these influencing factors. Thus, the designed ATO algorithms may be unable to track the trajectory precisely which greatly reduces the performance of train operations, for example, low energy efficiency and impaired riding comfort (Cohen et al., 2015). Therefore, it is practically significant to construct an accurate train control model as the basis of designing ATO algorithms in real-world operations.

Some existing studies have addressed the train control modeling (TCM) issue with simulation tools and vehicle model parameter identifications (Andersen et al., 2012; Bham and Benekohal, 2002; Fadhloun et al., 2015; Wang and Rakha, 2018). However, these traditional techniques require a large amount of empirical knowledge and experiences from the perspective of mechanical engineering. Moreover, these parameters of TCM have to be tested and validated in field experiments repeatedly, which can be very expensive in practice. Meanwhile, we notice that the urban rail trains run periodically from the origin to the destination on the same line and a lot of historical data are recorded in a metro system. Thus, it is possible to directly apply data-driven techniques in TCM problem to learn the underlying features of train dynamics with the aid of historical train running data. In some relevant areas, such as quadrotor flying modelling (Bansal et al., 2016) road traffic prediction (Ma et al., 2015), data-driven approaches involving neural networks and deep learning have shown their superior capabilities in modelling a huge amount of high-dimensional data to predict the motion of robotics or vehicles, while these new technologies have not been fully investigated in railway train control fields.

(a) Trains in the depot (b) Trains running through curves (c) Trains running under extreme weather

Figure 1: The complex external environment of rail trains

As ATO systems being extensively promoted through urban rail transit systems and high-speed railway systems, it has become an emerging technology for railway engineers to construct an accurate, efficient and flexible longitude train control model (Lagay and Adell, 2018). In this paper, we will address this problem by using data-driven approaches based on the historical train running data for directly quantifying an accurate train control model in urban rail systems. Specifically, we record the explicit train operation data in Beijing Metro from 2015 to 2018, which are used to develop three data-driven approaches, involving a linear regression-based model (LAM), a nonlinear regression-based model (NRM), and furthermore a deep neural network based (DNN) model. In particular, the DNN model is constructed by a multi-layered feed-forward Artificial neural network (ANN) combined with batch normalization layers which provide the capability to model high-dimensional train operation data. We compare the DNN model with traditional train control models in Su et al. (2013) and Howlett and Pudney (1995) with field data in Beijing Metro, and our results demonstrate significant performance improvement by using DNN model. We also embed the developed data-driven models into a virtual train control platform, by which we design and validate automatic train control algorithms that have been tested in the field and then successfully implemented in Beijing

Metro since 2018.

1.1 Literature Review

The longitudinal TCM depicts the motions of railway rolling stock vehicles in the direction of the track (Cole, 2006). Due to the complexity of TCMs arising from the interactions of the wheel/rail system, track gradients and track bends, and traction/braking motors, a lot of studies have investigated the mathematical modeling of longitudinal TCMs, dating back to the birth of railways (Grag, 2012). Generally, the existing literature on identifying TCMs can be classified into two categories, i.e., physical-driven models and simulation-based tools. The former class of research employs principles of mechanics to describe the longitudinal movement of vehicles on the tracks, which typically involve the vehicle dynamics acceleration/deceleration estimation (Bham and Benekohal, 2002; Fadhloun et al., 2015; Oprea et al., 2013), track friction coefficient identification (Wu et al., 2014) and in-train forces by drag forces and gravitational components. For example, an efficient and comprehensive alternative model was defined in Oprea et al. (2013) based on set-valued friction with Coulomb's law in order to quantify the train braking dynamics. Wang and Rakha (2018) developed a new TCM through a series of highly nonlinear functions by simultaneously considering the train acceleration, deceleration and resistance frictions. The model was calibrated and tested through the field data of Portland light rail train fleet. A comprehensive survey of TCMs from the perspective of mechanical engineering can be found in Wu et al. (2016). In the aspect of simulation models, some advanced tools were developed to quantify the train energy consumption (Andersen et al., 2012; Gbologah et al., 2014; De Martinis and Corman, 2018), and simulate the macroscopic metro lines (Grube et al., 2011). For instance, OpenTrack developed by Swiss Federal Institute of Technology Zürich is able to simulate the rail system operations based on user-defined train, infrastructure, and timetable databases (Nash and Huerlimann, 2004). More recently, Serajian et al. (2019) developed a simulation technique to investigate the effects of train length on in-train longitudinal forces during the braking of a train. The train brake delay time was also taken into account in this study.

Currently, TCMs have been widely applied to a lot of theoretical research and practical implementations (Cao et al., 2019). For example, Howlett and Pudney (1995) proposed a general model for optimal train control where train position is set as the independent variable, and then developed a series of energy-efficient train control strategies to minimize the traction energy consumption (Albrecht et al., 2016). For designing automatic train speed control algorithms, relevant studies typically employed second-order continuous state-space formulations with nonlinear or uncertain influencing disturbances to model the longitudinal train movement. For example, Gao et al. (2013) designed an on-line approximation-based robust adaptive controller, in which a radial basis function was utilized as the approximator of TCM. Li et al. (2014) further extended the model by considering the interactive forces among the connected vehicles of a train and developed a robust sampled-data cruise control method for high-speed trains. Here we refer to Yin et al. (2017) that gives a literature review for the automatic train control algorithms and applications.

In recent years, a significant change in transportation engineering is that much more accurate data are collected in real-time from a variety of sources, which significantly contribute to the vast amount of research in the field of data-driven approaches for traffic flow prediction (Lv et al., 2015; Sekuła et al., 2018), travel demand estimation (Toole et al., 2015), and traffic speed prediction (Ma et al., 2015), etc. For instance, combining neural networks and a profiling method, Sekuła et al. (2018) proposed a data-driven approach for estimating historical traffic volumes between sparsely located traffic sensors in the Maryland highway network. As a powerful tool for depicting the deep features with a large amount of data, deep neural networks (DNN) are particularly favored by a lot of researchers, and the applications in image recognition, acoustic modeling, etc., have demonstrated great improvement compared with shallow networks (Hinton et al., 2012). In transportation fields, a deep learning model with autoencoders

(AEs) and denoising autoencoders (DAEs) was developed by Duan et al. (2016) for traffic data imputation. More recently, Wu et al. (2018) proposed a novel multi-layered hierarchical flow network representation to estimate travel demand using multiple data sources. A transportation-focused computational graph was developed to capture the deep-rooted structure of traffic demand flow estimation problem. Finally, the proposed computational graph was evaluated and validated under a demonstration network.

However, there is very few literature in the field of railway train operations to the best of our knowledge that apply data-driven approaches for modeling railway train dynamics. Chen and Gao (2012) developed neural network (NN) based models to estimate the train station parking error. However, this study only considers the braking process of trains and the constructed NNs are actually shallow networks with only two input nodes and one output node. Due to the complexity of train operations which involve accelerating, cruising, coasting and braking operations, these models cannot be directly applied to TCM problems.

In summary, most previous studies concentrate on TCM in the field of mechanical engineering, but very few of them consider data-driven modeling techniques. In order to improve the accuracy of TCMs in urban rail systems, this paper proposes three data-driven models, i.e., LAM, NRM and DNN, with the field data collected in Beijing Metro. According to the unique features of urban railway trains, we customize the input&output layers and batch normalization layers to enhance the performance of DNN. Moreover, we report the model training and testing results with field data, conduct numerical experiments to validate the model parameters and analyze the practical and implementation difficulties of the developed data-driven models.

The rest of this paper is organized as follows. In Section 2, we present the developed mathematical models involving LAM, NRM, and DNN. In particular, we explicitly present the construction of DNN, involving the design of input and output layers, batch normalization layers and DNN parameter selections according to the unique features of train control models. In Section 3, we describe the collection of field data and define six performance indicators to evaluate the accuracy of each model. In Section 4, we train and validate the developed models, and then, we present a comprehensive comparison of these three models to describe their practical implementations in practice. Section 5 concludes this paper.

2 Mathematical Models

2.1 Problem Statement

In the current literature (e.g., Howlett and Pudney (1995); Li et al. (2014)), the TCM in a railway system can be typically expressed with a state-space formulation, given as follows

$$F = M \times \frac{dv}{dt} - f(v, s) \tag{1}$$

$$v = \frac{ds}{dt} \tag{2}$$

where F denotes the train traction or braking force, M is the mass of the train, and s, v and t respectively represent the train position, speed and time. Function $f(v, s)$ denotes the auxiliary resistances caused by the friction resistances, gradients and track bends, etc. In the train driving process, the driver or ATO controller receives the real-time feedback data (i.e., train position s, speed v and speed limit v_{lim}) from on-board and track-side sensors. The speed controller or driver compares the feedback information with a recommended speed profile $\hat{v}$ to determine the best control command $u = \frac{F}{M}$. Afterward, the control command u drives the train motors to generate force F (traction force F^+ or braking force B^-) to move the train.

The closed-loop feedback control process of a railway train is illustrated in Figure 2. Here we can see that TCM is actually affected by two kinds of influencing factors, i.e., the transformation of control command u to the traction & braking motors and the resistances of trains during its movement. In practice, the explicit modeling of these two parts is very complex due to a lot of nonlinear properties and uncertain parameters that are difficult to be characterized mathematically (Yin et al., 2014, 2019). For example, the wheel-track and air frictions can both be expressed as highly nonlinear functions with respect to train speed v.

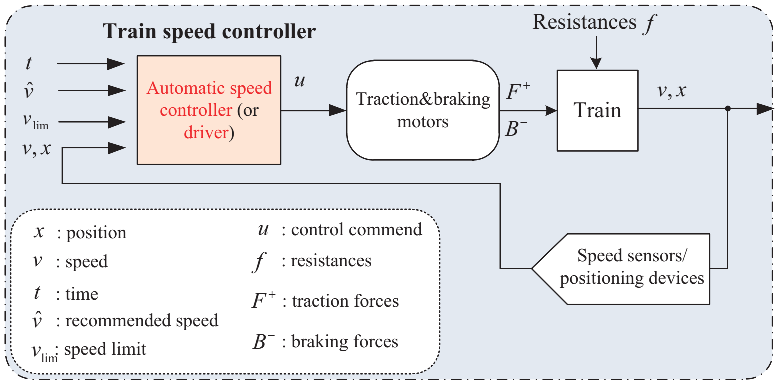


Figure 2: Closed-loop feedback control process of a railway train (Yin et al., 2017)

In the practical operations of urban rail systems, the trains run cycle by cycle through the first station to the destination and then travel back to the origin to prepare for the next cycle. The real-time train operation data are recorded by on-board ATO systems with a fixed time interval (for example each 0.2 second) in a discrete format. The recorded data typically includes the train position, speed, control command, train weight, line speed limit, line gradient, track bend, external weather, etc, at each time unit. Figure 3 demonstrates the recorded line speed limit, train speed and controller output of a train from one segment in Beijing Metro. We can see that, when the control command is positive, the train is in the traction phase and its speed is increased. When the control command is zero or negative, the train is in the coasting or braking phases, and the train speed is decreased. In this sense, the TCM can be essentially denoted by a mapping function from the current train state and input control command to the next train state. If we consider using speed and position as the state of a train, the mapping function is given by

$$\{s_{t+1}, v_{t+1}\} = \Phi\{s_t, v_t, u_t, \cdots s_0, v_0, u_0\}$$

where s_t and v_t are a set of train position and velocity vectors. Thus, it is very reasonable to directly approximate TCM using the historically recorded data in order to obtain the regression-based function Φ. In particular, if function Φ is accurate enough, we can use it for evaluating the performance of different ATO control algorithms in the simulation environment. For example, we can use the data-driven models to test if the train can arrive on time to the next station or if the ATO algorithm is energy-efficient.

Therefore, the aim of this study is to construct a data-driven model that uses the initial train state and control command u as input to *predict* the future train states. By training with historically recorded data, the model should be able to accurately predict the longitude movement of a train given arbitrary control sequences from the origin to the destination. For this purpose, this study constructs three data-driven approaches, which are a baseline model with linear regression, a nonlinear model and a DNN-based model, explicitly described in the following content.

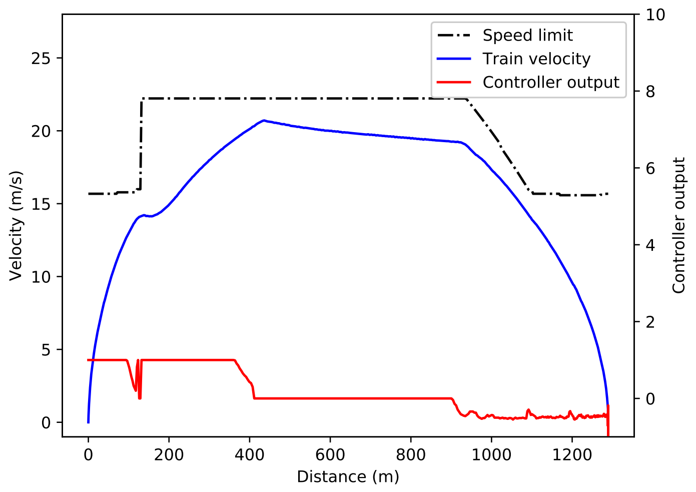


Figure 3: Illustration of historical recorded data

2.2 Model 1: Linear Regression-Based Model

A simple dynamic model of TCM can be constructed by selecting function $f(v, s)$ from the class of affine functions and fitting the linear coefficients to the observed data with least squares regression. This linear regression-based model (LAM) serves as our baseline model, and it has been shown to achieve good performance in many previous studies due to its concise structure (Su et al., 2013). In particular, LAM was demonstrated to be useful in urban rail transit where the train velocity is relatively low and the friction force has an approximately linear relation with train velocity.

In our work, LAM is basically defined as follows.

$$v_{t+1} = v_t + \alpha_1 u_t + \gamma_1 \tag{3}$$

$$s_{t+1} = s_t + \beta_1 v_t + \beta_2 u_t + \gamma_2 \tag{4}$$

Let $X_t = [v_t, s_t]^T$ denote the vector of *train state* defined on a set of discrete time units $t \in \{0, 1, \cdots, T\}$. The above discrete-time state-space model can be rewritten as follows.

$$X_{t+1} = AX_t + BU_t + W \tag{5}$$

where

$$A = \begin{bmatrix} 1 & 0 \\ \beta_1 & 1 \end{bmatrix}, \ B = \begin{bmatrix} \alpha_1 \\ \beta_2 \end{bmatrix}, \text{and} \ \ W = \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix}.$$

Then, the task for constructing LAM is to find the optimal coefficient vectors A, B and W given observed values of X and U, which is similar to system identification technique. In this model, we minimize mean squared prediction error over a set of historical train running data by solving

$$\text{LAM:} \min_{A,B,W} \sum_{0<t\leq T} \|X_t - \hat{X}_t\| \tag{6}$$

$$\text{s.t.} \quad X_{t+1} = AX_t + BU_t + W, \quad \forall t > 0 \tag{7}$$

where $\hat{X}_t$ is the observed value of X_t for each $t \in \{0, 1, \cdots, T\}$. The above problem results in a linear least squares problem with no constraints, which can be solved effectively using least-square regression methods.

2.3 Model 2: Nonlinear Regression-Based Model

The second model refers to a nonlinear regression-based model (NRM) that further extends LAM model by taking account into some nonlinear factors. Specifically, the effects of track resistances, air frictions, and track gradients are all nonlinear factors associated with train velocity and position, which can be involved in NRM. On the basis of the discrete state-space model in the above section, we first present a general formulation of NRM as follows.

$$v_{t+1} = v_t + \alpha_1 u_t + f_r(v_t) + f_g(s_t) \tag{8}$$

$$s_{t+1} = s_t + \beta_1 v_t + \beta_2 u_t + \gamma \tag{9}$$

where $f_r(v_t) = f_a + f_b v_t + f_c v_t^2$ is the Davis equation that refers to the track resistance and air friction, and f_a, f_b and f_c are the Davis parameters. $f_g(s_t) = g \cdot \sin(\theta(s_t))$ represents the line gradient where $\theta(s_t)$ is the line slop at position s_t and g is the standard gravity parameter. Thus, our aim is to obtain a set of model parameters involving α_1, β_1, β_2, γ, and David parameters f_a, f_b and f_c from historical train running data, which minimizes the expected cost between the predicted train running state and observed train running state. Define vector $X_t = [v_t, s_t]^T$ to represent the train position and velocity at time unit t. With a given initial state X_0, we can formulate TCM problem into an optimization model to find a set of train dynamic parameters from historical data samples $\hat{X}_t$.

$$\text{NRM:} \min_{f_a, f_b, f_c, \alpha_1, \beta_1, \beta_2, \gamma} \sum_{0<t\leq T} \|X_t - \hat{X}_t\| \tag{10}$$

$$\text{s.t.} \qquad X_{t+1} = X_t^T \begin{bmatrix} f_c & 0 \\ 0 & 0 \end{bmatrix} X_t + \begin{bmatrix} f_b + 1 & 0 \\ \beta_1 & 1 \end{bmatrix} X_t + \begin{bmatrix} \alpha_1 \\ \beta_2 \end{bmatrix} u_t + \begin{bmatrix} f_a + f_g(s_t) \\ \gamma \end{bmatrix}, \quad \forall t > 0 \tag{11}$$

The aim of the above model is to find a set of parameters involving $f_a, f_b, f_c, \alpha_1, \beta_1, \beta_2$ and γ in order to minimize the least square error between the predicted train states and observed train states from historical data. Nevertheless, the above model is a nonlinear model with a set of nonlinear equality constraints (11), which is actually difficult to be solved. Meanwhile, we see that the model can be equivalently reformulated into an unconstrained nonlinear optimization model by inverting the constraints into the objective function. Let nonlinear function G denote constraint (11) that maps a train state X_t and control command u_t to train state X_{t+1}, i.e., $X_{t+1} = G(X_t, u_t)$. Model (10) can be equivalency formulated into the following model

$$\text{NRM:} \min_{f_a, f_b, f_c, \alpha_1, \beta_1, \beta_2, \gamma} \sum_{0<t\leq T} \|G(X_{t-1}, u_{t-1}) - \hat{X}_t\| \tag{12}$$

As model (12) is a nonlinear optimization model with no constraints, it can be efficiently handled by a stochastic gradient descent (SGD) algorithm that enables to generate the optimal solution (or local-optimum) given a set of training data samples (Bottou, 2010).

2.4 Model 3: Deep Neural Network Based Regression Model

Since our research in this paper is motivated by a practical issue, we first only tried LAM and NRM models as described above. However, we found that they can only achieve good performance if we only consider one operation phase, for example, train braking phase. If we consider the whole train operation process on the segment which involves accelerating, cruising, coasting and braking phases, the performances of LAM and NRM can become very bad since these simple models are unable to approximate such complex vehicle dynamic systems.

With the exponential growth of computing resources and advanced parameter learning methods (e.g., SGD, ADAM), deep learning has shown its superior advantages in pattern recognition and function regression. Using a feed-forward artificial neural network with more than one layer of hidden units between its input and output, they have been successfully applied in image classification, speed recognition, traffic prediction, and computer gaming (Hinton et al., 2012; LeCun et al., 2015). Thanks to the advantages of DNN in modeling high-dimensional data classifications and approximations, our study constructs a customized DNN network in order to enhance the accuracy and robustness of LAM and NRM. Notice that the train dynamics have some unique characteristics that are different from other robotics (Bansal et al., 2016). For example, the input data (i.e., train velocity, position, and control command) are not on the same scale and the train states are usually affected by many other factors.

Figure 4 demonstrates the general framework of DNN network to approximate the real-world TCM with the help of historical data to accurately predict the running of trains. Specifically, the lower part of Figure 4 represents the close-loop control process of a physical TCM, which is composed of the following four parts: The traction&braking motor transfers the control command u to traction or braking forces to drive the train. Due to the movement of train, it suffers from a series of resistances, termed as f in this figure. Meanwhile, the speed sensors or other positioning devices monitor the real-time velocity and position of the train, and transmit the real-time data to train speed controller. The controller receives the real-time data and generate a new control command u by feedback control algorithms to drive the train. In our work, we recorded all the train running data from the physical TCM (i.e., the lower part of Figure 4) with the aim to train our DNN model. In particular, the input data of train speed control of physical TCM are used as input of DNN, while the output data of physical TCM are used as output (i.e., labels) to train DNN. More details about data collection can be found in Section 3. We can then use DNN model to predict the train running states if the DNN model is well trained with real-world data collected from physical TCM. In other words, the DNN model is trained to replace two blocks, i.e., traction&braking motors and train blocks, in real-world implementations of DND (see Section 4.3 for details).

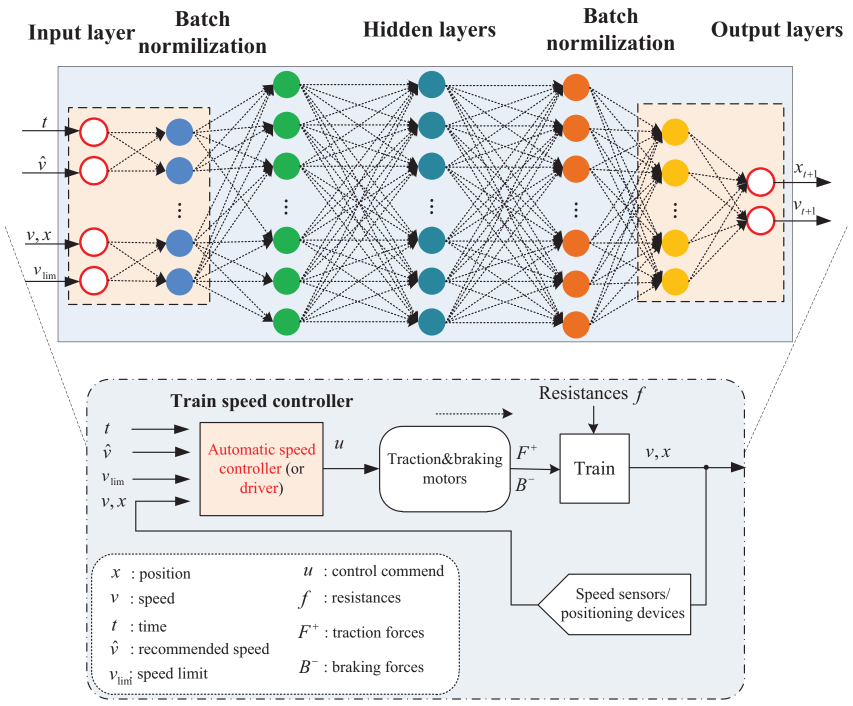


Figure 4: Illustration of DNN for predicting train dynamics

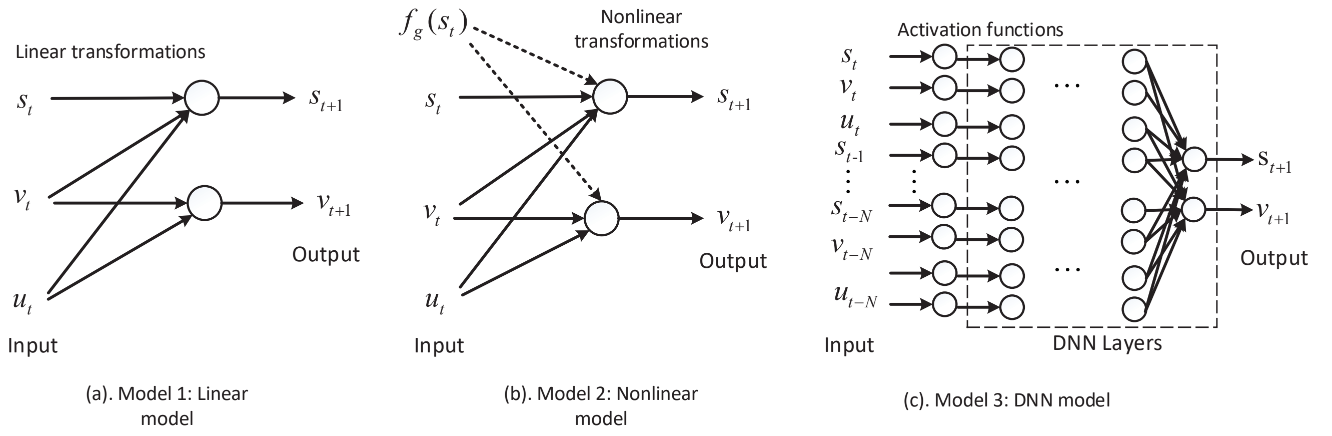


Figure 5: Comparison of input and output data for the constructed models

2.4.1 Input and output layer

Due to the limitation of linear or nonlinear regressions, the constructed benchmark LAM and NRM models only utilize the current train state X_t and controller command u_t as model input to predict the next train state X_{t+1}, which are essentially "one-step predictors". In fact, the train dynamic model is a very complicated system subject to many influencing factors from the train controller, traction motors and external resistances (see Figure 4). Further, there are typically *time-delays* in this closed-loop control process of a railway train (Yin et al., 2014). Here, the train time-delay refers to a short period of time (usually 1 to 3 seconds) to transform the control command u_t to the traction force F^+ (or braking force B^-) of a train. In other words, the next state X_{t+1} of a train is not only related to its former state X_t and u_t, but also related to a series of its former states X_{t-1}, X_{t-2}, X_{t-3}, u_{t-1}, u_{t-2}, u_{t-3}, etc. Since DNN has advantages in handling high-dimensional input data, we hereby employ the train running data in the former N steps as the input data for constructing our DNN model, which is actually a "multi-step predictor". A comparison of input data for these three models is explicitly presented in Figure 5. Thus, the input data sample and output data sample of DNN at time t are respectively expressed as follows.

Input data:

$$X_t = \{s_{t-N}, v_{t-N}, \cdots, s_{t-n}, v_{t-n}, \cdots, s_t, v_t | \forall 0 < n \leq N, N \in \mathbb{Z}^+\} \tag{13}$$

$$U_t = \{u_{t-N}, \cdots, u_{t-n}, \cdots, u_t | \forall 0 < n \leq N, N \in \mathbb{Z}^+\} \tag{14}$$

Output data:

$$X_{t+1} = \{s_{t+1}, v_{t+1}\}. \tag{15}$$

In the above equations, we call parameter N as a *feedforward timewindow*, and we set $t-n$ to 0 for any $0 < n \leq N$ if $t-n < 0$, indicating the start-up of a train at stations. In practice, the value of parameter N can be properly set according to the type of a train. For example, the time-delay of a transit light train is usually about 1 to 3 seconds. In this case, parameter N should at least cover this time period to keep the accuracy of TCM. Using Eq. (13), we actually convert the original input $[v_t, s_t]^T$ and u_t into a set of high-dimensional vectors of data samples, and each data sample has a dimension of $3N$, This treatment enables to capture the specific state of a train within a time period of $[t-N, t]$ rather than a time unit t. For simplicity, we define $\mathbf{x}_t$ to represent the input data vector $[X_t, U_t]$, and x_i for $i \in \mathcal{N}$ to represent each element of input data vector $\mathbf{x}_t$, where $\mathcal{N} = \{1, 2, \cdots, 3N\}$.

Example. Consider an example that the time-delay of a heavy-haul train is 3 seconds and the sampling time interval is 0.2 second in the recorded data. Then, the dimension of a DNN input layer should be at least $3/0.2 * 3 = 45$ to fully capture the dynamics of a train control model. We will further investigate the setting of parameter N for predicting accuracy in the experiments with field data. $\square$

2.4.2 Batch normalization

As described above, the inputs in our DNN model involve the train speed (unit: m/s), position (unit: m) and controller command (unit: m/s^2). It is clear that these three classes of input data are actually not in the same scales, which may cause the low efficiency for training DNN. Thus, training of DNN model requires the data preprocessing step, i.e., data normalization. Specifically, data normalization refers to normalizing the data dimensions so that they are of approximately the same scale. There are two common ways of achieving this normalization. One is to divide each dimension by its standard deviation, once it has been zero-centered. Another form of this preprocessing normalizes each dimension so that the min and max along the dimension is -1 and 1 respectively. Meanwhile, we note that training DNN is inherently complicated due to the fact that the distribution of each layer's inputs changes during the training process. Due to these above two reasons, we here employ the batch normalization technique to reduce the internal covariate shift and accelerate the training process of DNN (Ioffe and Szegedy, 2015). To clearly present the detailed process of batch normalization for DNN model, we take the first batch normalization layer, i.e., the layer between input layer and hidden layer for example. Define vector $\mathbf{y}_t^{(0)} = \{y_1, y_2, \cdots, y_{3N}\}$ to be the output of the first batch normalization layer, and each y_i for $i \in \mathcal{N}$ represents an element of vector $\mathbf{y}_t^{(0)}$. With batch normalization, the input data $\mathbf{x}_t$ is transformed to $\mathbf{y}_t^{(0)}$ by the following calculations.

$$\mu_\beta = \frac{1}{3N}\sum_{i=1}^{3N} x_i \tag{16}$$

$$\sigma_\beta^2 = \frac{1}{3N}\sum_{i=1}^{3N} (x_i - \mu_\beta)^2 \tag{17}$$

$$\bar{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}} \tag{18}$$

$$y_i = \gamma_i \bar{x}_i + \beta_i \tag{19}$$

Eqs. (16)-(18) represent the standard normalization process where ϵ is a constant added to the mini-batch variance for numerical stability (when $\sigma_\beta^2 = 0$). Eq. (19) indicates the scale and shift process, where γ_i and β_i are a pair of parameters, which aim to guarantee that the transformation inserted in the network represents the identity transform.

2.4.3 DNN network structure

Our developed DNN model is a feed-forward artificial neural network structured with an input layer at the bottom, stacked hidden layers and output layer, and each layer contains several neurons combined with a batch normalization procedure. With multiple hidden layers, the DNN can effectively characterize complex mapping functions between high-dimensional input feature vectors and observed data. To efficiently predict the dynamic state of a train, we implement a multi-layer neural network combined with batch normalization (Fang et al., 2017), described as follows.

Define $\mathbf{y}^{(0)}$ to be the input of the first hidden layer. For a DNN model with L hidden layers, the output of a hidden layer l is described as

$$\mathbf{x}^{(l)} = f(\mathbf{w}^{(l)}\mathbf{y}^{(l-1)} + \mathbf{b}^{(l)}) \tag{20}$$

where $l = 1, \cdots, L$, $\mathbf{w}^{(l)}$ and $\mathbf{b}^{(l)}$ respectively denote the neural weight matrix and bias of the l-th hidden layer, and $f(\cdot)$ is the activation function. The most commonly used activation function involves rectified linear unit (ReLU), Logistic function, and softmax, etc. Here, out study uses a softmax activation function combined with ReLU, which is given by

$$x_i^{(l)} = \frac{e^{\hat{y}_i^{(l-1)}}}{\sum_{i\in\mathcal{N}} e^{\hat{y}_i^{(l-1)}}}, \forall i \in \mathcal{N} \tag{21}$$

where $x_i^{(l)}$ denotes the ith element of hidden layer l, and $\hat{y}_i^{(l-1)}$ denotes the ith element of $\max\{\mathbf{w}^{(l)}\mathbf{y}^{(l-1)} + \mathbf{b}^{(l)}, 0\}$.

Given a set of training input data $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T\}$ and observed data $\hat{\mathbf{x}} = \{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \cdots, \hat{\mathbf{x}}_T\}$, we aim to train the DNN model such that the Euler distance between the input and output data is minimized. In this sense, we use a mean square error loss function, and formulate the lost function as

$$J(\mathbf{w}, \mathbf{b}) = \frac{1}{T} \sum_{0<t<T} \|\hat{\mathbf{x}}_t - \mathbf{y}_t^{(L)}\|_2^2 \tag{22}$$

where $\mathbf{y}_t^{(L)} = \{s_{t+1}, v_{t+1}\}$ is the output of layer L in DNN after Batch renormalization, in which these two terms s_{t+1} and v_{t+1} denotes the predicted train position and velocity at time $t+1$, respectively. We see that, the best train control model can be achieved by a set of DNN coefficients $\mathbf{w}^*$ and $\mathbf{b}^*$, which yield the minimal value of nonlinear loss function $J(\mathbf{w}, \mathbf{b})$. In other words, the considered problem is evenly transformed into a nonlinear optimization problem.

In this study, we adopt the back-propagation algorithm with the ADAM (an efficient algorithm for first-order gradient-based optimization of stochastic objective functions) to fine tune the model parameters $\mathbf{w}$ and $\mathbf{b}$. We can refer to Kingma and Ba (2014) for details about the implementing procedures of ADAM algorithm.

Remark 2.1: It is worth to clarify that the DNN structure developed in this paper actually differs from conventional deep learning structures (e.g., deep belief networks, convolutional neural networks) that are commonly employed in image processing, natural language processing, etc. The main reason is that the dimensions of input data are not so large as those in other machine learning fields. Thus, it is not required to employ complex network structures to model the train control models according to our computational results (see Section 4).

3 Data Description and Performance Indicators

3.1 Field data description

The proposed data-driven approaches were applied to the field data collected from on-board computers in Beijing Metro Yizhuang Line and Changping Line from 2015 to 2018. In the data collection and processing, multiple sensors (i.e., radars, infrared sensors, and acceleration sensors) were installed on the trains to track the trains' position, velocity, accelerations, gradients, etc (Yin et al., 2017). Data fusion algorithms were utilized to pre-process the data from multiple sensors. The train controller outputs were recorded by on-board ATO systems, which automatically operate the trains. In particular, the sampling time interval for the collected data is 0.2 second. Thus, we generate one data sample with train state information (speed, position, gradient) and controller output each $0.2s$. Considering all the collected data samples in these three years, the total data volume is about 30GB. For simplicity, we employ the data of a single train, which ran two cycles through a single day and each cycle includes a total of 26 segments (13 up-direction segments and 13 down-direction segments) in Yizhuang Line, as shown in Figure 6. Since the consumed time for one cycle is about one hour, we can actually obtain a total of $18,000$ data samples in one cycle and $36,000$ data samples in one single day. Figure 7 is a plot of recorded train speed and controller output over time (in 0.2-second unit).

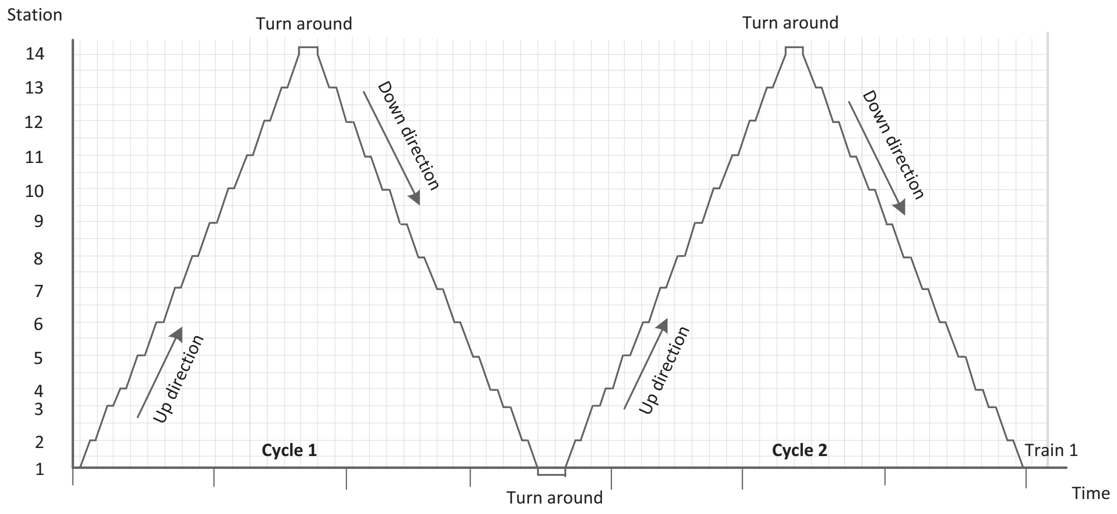


Figure 6: Illustration of space-time trajectory for the train with two cycles

Here it is worth to mention that, even though the trains are automatically controlled by the same feedback algorithm, the collected data, involving the control command u, train position s and train velocity v are actually different every day (or cycle) due to the varying environment and other influencing factors. In our study, we have picked out the data sets under the same conditions for the training of DNN. In particular, we use the train running data at the same weather (involving normal weather, rain, snow, fog, and smog) and the same departure time from the origin. The reason is that the trains at different time period have different weight. For example, the train weight on morning peak hours is higher than the trains on off-peak hours. In the numerical experiments, we will present the model training and testing results that are trained by the data set under normal weather conditions.

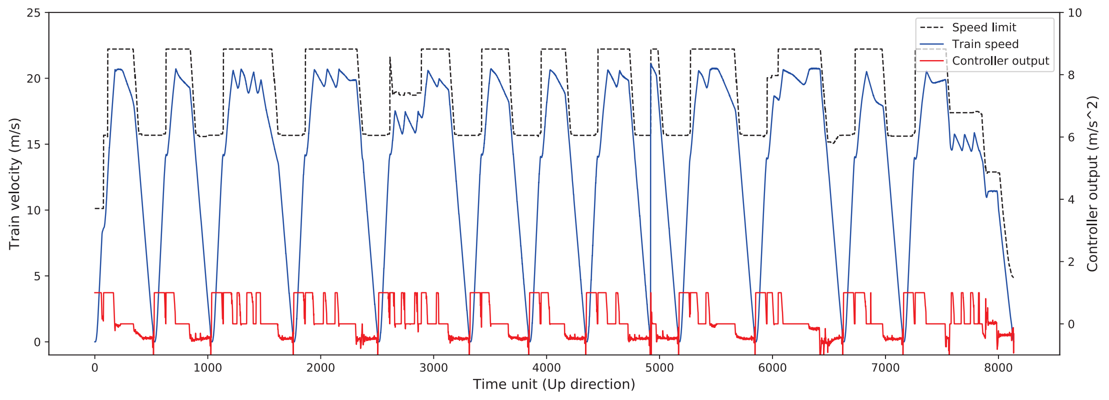
3.2 Performance indicators

In order to evaluate the performances of LAM, NRM and DNN, we adopt the mean absolute error (MAE), root mean square error (RMSE) and mean relative error (MRE) to verify the effectiveness of the developed models (Duan et al., 2016). Different from existing literature, one of the unique features in this study is that our proposed data-driven models predict both the positions and velocities of a running train. It is thus necessary to evaluate the performances with respect to the train position dimension and train velocity dimension separately. Therefore, a total of six performance indicators are defined, which are respectively given as follows

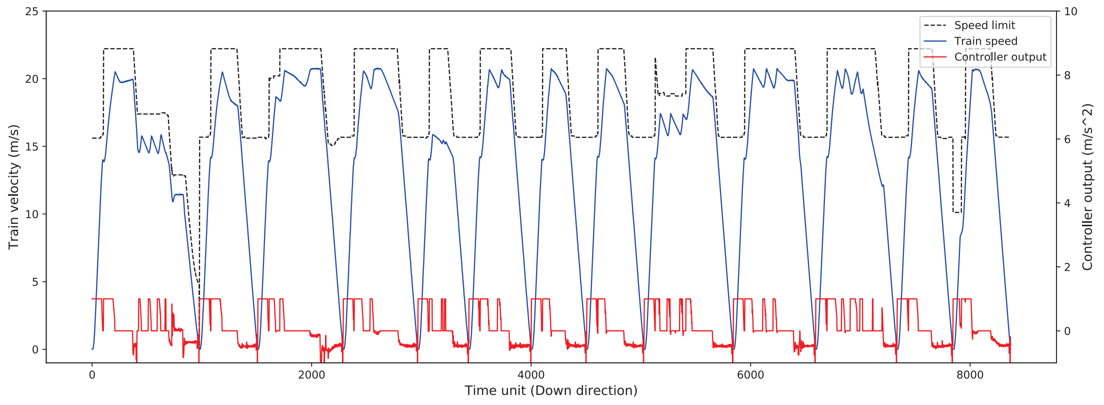
$$MAE_S = \frac{\sum_{t=1}^{T} |\hat{s}_t - s_t|}{T} \tag{23}$$

$$RMSE_S = \sqrt{\frac{\sum_{t=1}^{T} (\hat{s}_t - s_t)^2}{T}} \tag{24}$$

$$MRE_S = \frac{\sum_{t=1}^{T} \frac{|\hat{s}_t - s_t|}{\hat{s}_t}}{T} \tag{25}$$



(a) Up direction



(b) Down direction

Figure 7: Typical data recorded in a single cycle by a train

$$MAE_v = \frac{\sum_{t=1}^{T} |\hat{v}_t - v_t|}{T} \tag{26}$$

$$RMSE_v = \sqrt{\frac{\sum_{t=1}^{T} (\hat{v}_t - v_t)^2}{T}} \tag{27}$$

$$MRE_v = \frac{\sum_{t=1}^{T} \frac{|\hat{v}_t - v_t|}{\hat{v}_t}}{T} \tag{28}$$

where MAE_S, $RMSE_S$ and MRE_S represent the MAE, RMSE and MRE of train position predicting results, and MAE_v, $RMSE_v$ and MRE_v represent the predicting results of train velocities.

4 Experiment Results

In this section, we use the filed data in Beijing Metro to verify the effectiveness of developed data-driven models. Specifically, we first conduct sensitive analysis with respect to the key parameters of the DNN model, in order to summarize insightful experiences for efficiently designing DNN structures. Then the training and testing performances of LAM, NRM and DNN models are explicitly compared based on the performance indicators from Eqs. (23)-(28). Finally, we present the details for implementing the data-driven models in practical applications in Beijing Metro. The experiments are conducted using Python language (version 3.7) on a computer with I7-8700K CPU (8 cores and 16 threads), 16 GB memory and RTX 2070 GPU.

Using the field data collected in Beijing Metro, the model training and testing procedures are specified as follows. We first employ the data of a single train in one year from Yizhuang Line, which ran two cycles through a single day and each cycle includes a total of 26 segments (13 up-direction segments and 13 down-direction segments), as we have described above. According to the number of segments, the data are grouped into 26 data sets and each data set D_i represents the train running states on segment i. Using each data sets i, we train LAM, NRM and DNN models and save these well-trained models. For convenience's sake, we use the term *instance* i to indicate the training and testing of developed models with each data set D_i in the following content. We train LAM, NRM and DNN models using linear regression, SGD, and ADAM, respectively. The learning rates for SGD and ADAM are set as 0.01 and 0.005, respectively. We terminate the training process under the following two terminating conditions: 1) the running time exceeds 10 minutes; 2) the loss function keeps constant for 1000 iterations. Then we save the trained models and calculate the performance indexes using Eqs. (23)-(28).

Remark 4.1: Typically, the model training and testing procedures in machine learning fields are conducted by using two different groups of data sets respectively. Since our study focuses on the regression of train control models, the model testing step is conducted in a different manner. Specifically, we first use a group of data to train the proposed models on each segment i (model training step). In the model testing process, we choose a new group of train running data on the same segment from time $t = 0$ to $t = T$. We input the initial train state s_0 and v_0 and the control command u_t for $t = 0, 1, \cdots, T$ to the well trained models. The models can output a series of observed train states $s_1, v_1, \cdots, s_T, v_T$, as shown in Figure 8. Then, we compare the observed train sates with the field recorded data to test the predicting accuracy of data-driven models. In particular, if we consider the train running data on a whole segment, we can set $s_0 = 0$ and $v_0 = 0$ for simplicity, indicating that a train departs from the station at time $t = 0$.

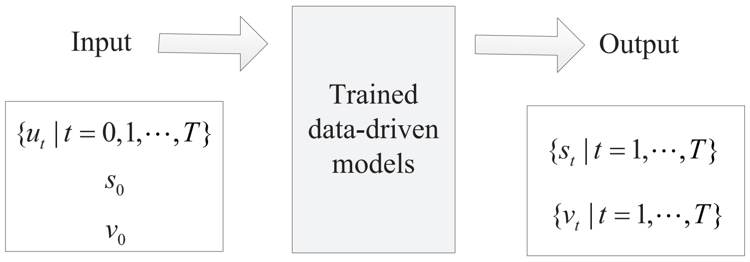
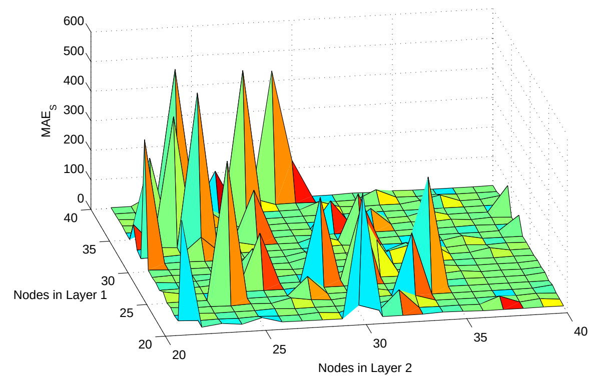

Input
$\{u_t \mid t=0,1,\cdots,T\}$
s_0
v_0
Trained data-driven models
Output
$\{s_t \mid t=1,\cdots,T\}$
$\{v_t \mid t=1,\cdots,T\}$

Figure 8: Input and output of model testing process

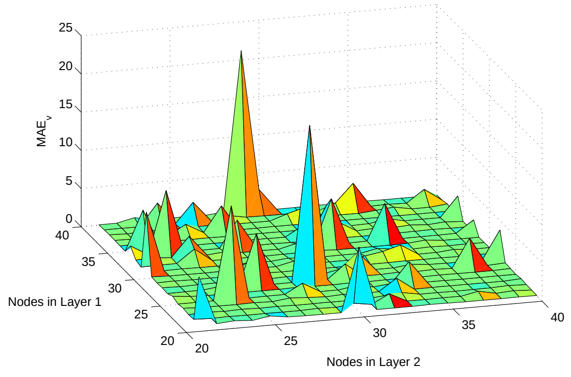
4.1 Parameter tuning of DNN model

In this section, we conduct a series of experiments to test the performance of DNN with different hyper-parameters and structures, which can be beneficial to determine the best parameter settings of DNN models.

First, the experiments are conducted to test the performance of DNN with different numbers of nodes in each hidden layers. To balance the predicting accuracy and computational effort, we build the DNN with one input layer, two hidden layers (i.e., Layer 1 and Layer 2) and one output layer, where each layer is combined with batch normalization. In the experiments, parameter N for DNN is initialized as 9 since the time-delays are about 1 second in urban rail trains, which indicates that the dimension of input layer for DNN is 27. We alter the number of nodes in each hidden layer from 20 to 40 in order to analyze the performance of DNN. We define integer values L_k to be the number of nodes on each hidden layer k. In this sense, a total of 400 (i.e., 20×20) experiments are conducted with different numbers of neuron nodes on each layer. We use data group D_1 to train the models (the maximum



(a) MAE_S



(b) MAE_v

Figure 9: Performance indicators with different of nodes on each layer

training time limit is 10 minutes), and then we test the models with field collected data and record the value of performance indicators MAE_S, $RMSE_S$, MRE_S, MAE_v, $RMSE_v$, MRE_v in each experiment.

As the general tendency is similar, we only present the performance indicators of MAE_S and MAE_v with different values of L_1 and L_2, which are shown in Figure 9a and Figure 9b, respectively. From the results, we see that MAE_S and MAE_v increase unexpectedly in some specific cases. This phenomenon is particularly noticeable when the nodes of Layer 1 are more than those of Layer 2. The possible reason is that DNN falls into a local optimal solution that terminates the iterations of DNN training process. Even though, we see that the performance of DNN is relatively stable in most experiments and DNN returns very small errors with respect to MAE_S and MAE_v, which indicates that most local optimal solutions are avoided. Moreover, we can see that DNN models are more stable as the numbers of nodes are larger. From this set of experiments, we draw the experiences in constructing a DNN model that local minima can be improved with a larger number of nodes on Layer 2. Therefore, our following experiments will set L_1 and L_2 as 20 and 35, respectively.

Next, we analyze the impact of parameter N on DNN predicting accuracy. As stated in Section 2.4.1, DNN employs high-dimensional input data, which acts as a *multi-step predictor*. Here, we alter the value of N from 1 to 20 to train and then test DNN models. Since the minimal discrete time unit is $0.2s$, the developed DNN traces the backward train states from $0.2s$ to $4s$, and the number of nodes in the input layer of DNN is increased from 3 to 60 as N varies from 1 to 20. After training DNN models with different values of N, we test the models as described above. Table 1 explicitly presents the testing performances of DNN with respect to these six indicators.

From Table 1, we can clearly see that DNN performs very bad when the value of N is smaller than 8. In particular, when N equals to 1, 2 or 3 (i.e., 3, 6 or 9 nodes in the input layer of DNN), the position and velocity predicting errors are very large and DNN models are actually not usable. This may indicate two conclusions: 1) DNN does not have unique advantages when the input data dimension is shallow; 2) The time-delay parameter of TCM has an important effect on the regression model that must be carefully considered in practical implementations. As we gradually increase N from 9 to 20, the number of nodes in the DNN input layer is added from 27 to 60. The obtained results in these experiments are much better and stabilized. And we find that adding more nodes in the input layer may not lead to evident improvement with respect to these performance indicators. Meanwhile, note that DNN models with more input nodes consume much longer time for the training of DNN. Therefore, it is very important to determine the best DNN parameters according to the rough ranges of time-delay and time-constant parameters of train control models. In our following experiments, we choose $N = 9$ (i.e., 27 nodes in the input

Table 1: Model test performance with different values of parameter N

Performance	MAE_S	$RMSE_S$	MRE_S	MAE_v	$RMSE_v$	MRE_v
N=1	716.989	777709	0.9299	22.3436	814.693	5.4334
N=2	472.023	403887	2.7372	13.1870	230.2840	1.0045
N=3	247.596	147956	2.5676	5.6872	71.5937	0.7228
N=4	9.73934	162.82	0.0470	0.2071	0.07637	0.0682
N=5	29.8879	4239.29	0.2742	1.2011	7.5334	0.1255
N=6	4.16205	26.79	0.0196	0.1171	0.0215	0.0306
N=7	70.1021	21542.3	0.7900	1.8251	20.7973	0.2292
N=8	8.17914	103.48	0.0210	0.2632	0.1373	0.1393
N=9	2.00332	7.57	0.0130	0.0692	0.0088	0.0186
N=10	9.96671	19.44	0.0311	0.2335	0.1104	0.0719
N=11	4.52388	45.07	0.0190	0.1064	0.0528	0.0176
N=12	9.46758	13.10	0.0240	0.2347	0.0892	0.0512
N=13	4.02104	36.21	0.0169	0.0997	0.0254	0.0173
N=14	2.39357	9.72	0.0090	0.0658	0.0081	0.0166
N=15	2.68499	13.97	0.0193	0.0904	0.0174	0.0186
N=16	8.50665	11.58	0.0163	0.2988	0.1776	0.0880
N=17	4.17673	7.19	0.0100	0.1353	0.0364	0.0390
N=18	2.24129	5.38	0.0080	0.0545	0.0057	0.0136
N=19	1.47835	4.60	0.0055	0.0468	0.0041	0.0120
N=20	3.85592	2.08	0.0123	0.0909	0.0186	0.0170

layer) to balance the DNN predicting accuracy and computational efforts.

4.2 Comparison of LAM, NRM and DNN models

In order to analyze the performance of DNN compared with LAM and NRM models, we then conduct a series of experiments to use the recorded train operation data to train and test the LAM, NRM and DNN models. We pick up four groups of data sets D_1, D_2, D_3 and D_4, corresponding to instance 1, instance 2, instance 3 and instance 4.

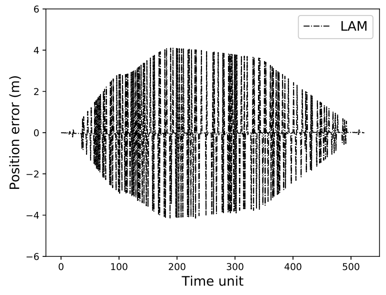
The model training results with training data sets D_1 - D_4 respectively by LAM, NRM and DNN are shown in Table 2. It can be seen that all these three developed models can realize a good predicting performance in the model training process. Specifically, MAE_S (i.e., the average train position predicting error) is at most 1.16 meters, and MAE_v (i.e., the average train velocity predicting error) is at most $0.24m/s$ in these experiments. In addition, we observe that the performance indicators of the nonlinear model NRM are only a little better than the benchmark linear model LAM. The possible reason is that the nonlinear characteristics of transit train model are caused by frictions or other influencing factors, which are not evident compared with the train traction&braking forces. More importantly, we can see that the improvement of DNN is also not very obvious compared with LAM and NRM. Even though DNN achieves the best results in Instance 1 and Instance 4, the improvements are only about 8% to 33%, and DNN even generates worse performances in Instance 2 and Instance 3 compared with LAM and NRM.

In addition, we also record the details of the predicted position and velocity errors on each segment. For example, the predicted position and velocity errors of instance 1 by LAM, NRM and DNN are shown in Figures 10 and 11. An interesting phenomenon is that the position predicting errors by LAM in Figure 10a and NRM in Figure 10b are very similar, which is possibly due to the reason that the basic parameters (i.e., α, β) for LAM and NRM are identical. From the velocity prediction errors in Figure 11a to Figure 11c, we see that the errors are much larger in the beginning and ending periods, i.e., when the time unit is less than 200 or larger than 500. In particular, these

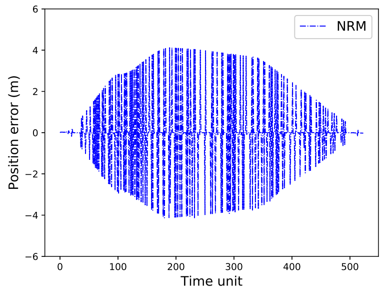
Table 2: Performance comparison with training data

Performance indicators		LAM	NRM	DNN
Instance 1	MAE_S	1.16	1.15	**1.04**
	$RMSE_S$	3.76	3.76	**2.10**
	MRE_S	0.01	0.01	0.01
	MAE_v	0.09	0.05	**0.04**
	$RMSE_v$	0.02	0.01	**0.00**
	MRE_v	0.03	0.04	**0.01**
Instance 2	MAE_S	0.03	**0.02**	0.22
	$RMSE_S$	0.01	0.01	0.11
	MRE_S	0.00	0.00	0.00
	MAE_v	0.13	0.03	**0.02**
	$RMSE_v$	0.03	0.00	0.00
	MRE_v	0.12	0.03	**0.01**
Instance 3	MAE_S	0.13	**0.12**	0.38
	$RMSE_S$	0.40	0.40	0.55
	MRE_S	0.01	0.00	0.00
	MAE_v	0.24	0.03	**0.02**
	$RMSE_v$	0.08	0.00	0.00
	MRE_v	0.12	0.03	**0.01**
Instance 4	MAE_S	**0.99**	1.00	1.04
	$RMSE_S$	3.16	3.16	**2.29**
	MRE_S	0.00	0.01	0.00
	MAE_v	0.07	0.06	**0.04**
	$RMSE_v$	0.01	0.01	**0.00**
	MRE_v	0.07	0.40	**0.01**

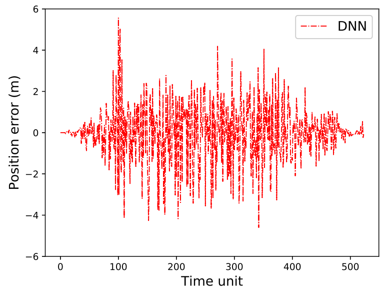
two periods correspond to the train traction and braking phases. This observation is actually consistent with the practical experiences that the train models are much more complex in traction&braking phases. In addition, we see that DNN model does not present evident improvement compared with LAM and NRM models in the training data sets.



(a) LAM

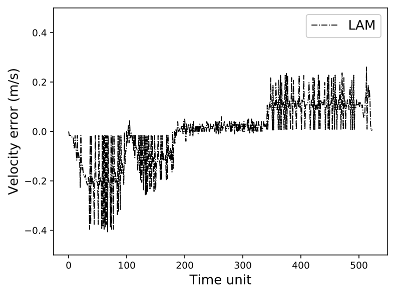


(b) NRM

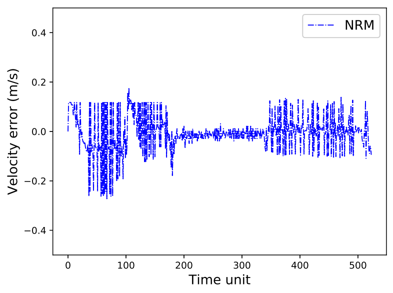


(c) DNN

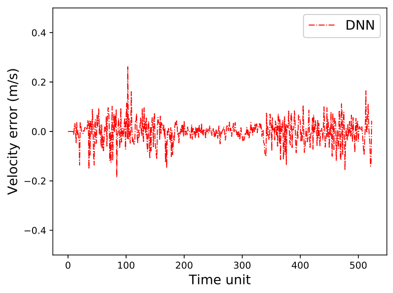
Figure 10: Position predicting error with training data for Instance 1



(a) LAM



(b) NRM

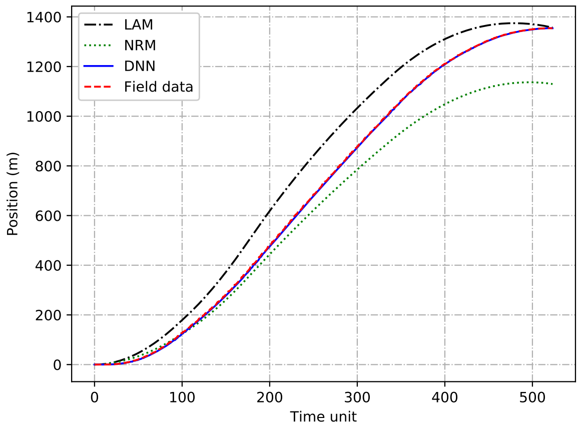


(c) DNN

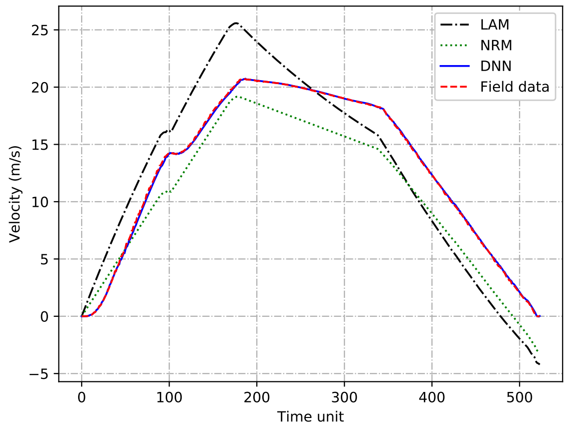
Figure 11: Velocity predicting error with training data for Instance 1

The former set of experiments presents the performance indicators by the training data set. However, the models trained by training data sets may have over-fitting problems, as stated in many existing research (Scheres and Chen, 2012; Yin et al., 2016). Next, we are particularly interested in the performance comparison when we test these three models after they are well trained. Specifically, we first initialize the trains' positions and velocities as zeros and input a vector of field train control command u_t for $t \in T$ in each experiment. We respectively use LAM, NRM and DNN models to predict the states of trains from one segment to the next. Finally, we can obtain a group of observed position and velocity curves respectively by these three models, which are then compared with the field observed position and velocity curves on the corresponding segment.

Figure 12 demonstrates the field data and the predicted train positions and velocities by LAM, NRM, and DNN, corresponding to Instance 1 in the model training experiments. The results are remarkable: (1) Even though LAM and NRM models have relatively good performance in the model training experiments, they cannot accurately predict the curves of train position and velocity from the model testing results, and the predicting errors are gradually accumulated as the time horizon moves. Finally, a large derivation between the predicted and observed train state is observed. The possible reason is that LAM and NRM are over-trained that impair the robustness and flexibility as they have fewer parameters. (2) We see that DNN evidently outperforms both LAM and NRM

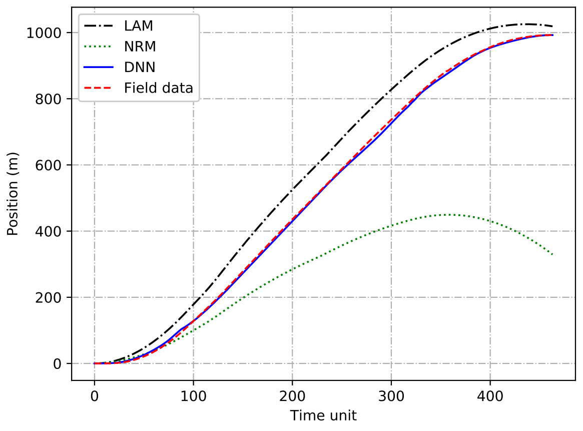


(a) Comparison of train position curves

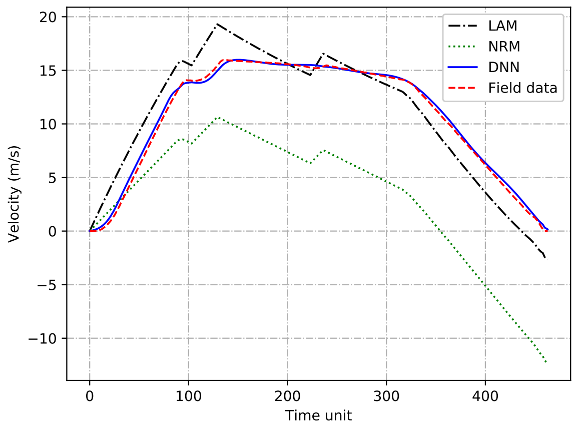


(b) Comparison of train velocity curves

Figure 12: Field data and predicted train states by LAM, NRM and DNN in Instance 1



(a) Comparison of train position curves



(b) Comparison of train velocity curves

Figure 13: Field data and predicted train states by LAM, NRM and DNN in Instance 2

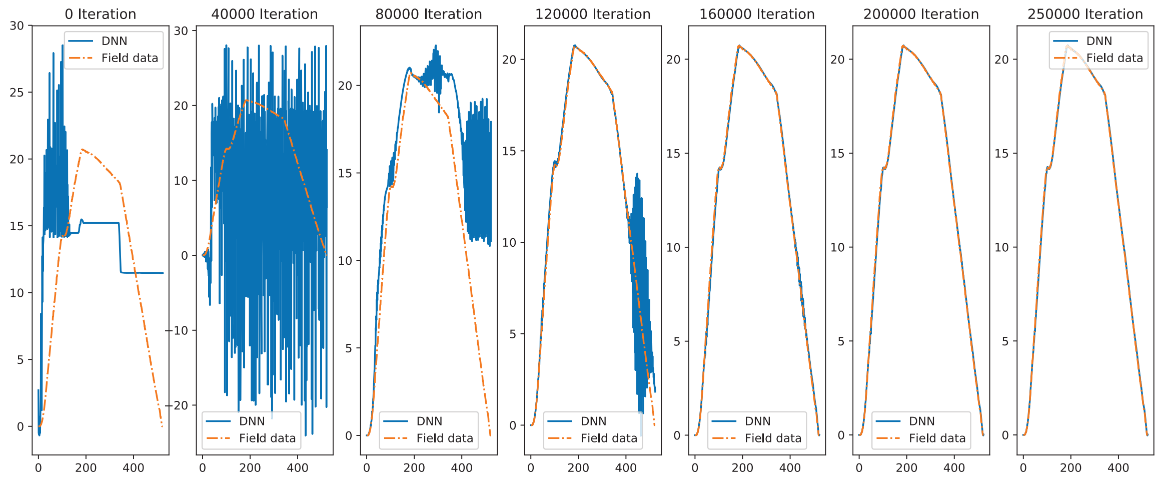
in the testing experiments. It achieves outstanding performance and the predicted position and velocity curves by DNN are nearly the same as those in the field data. The results by instance 2 in Figure 13 also reveal similar phenomenons, which could verify the effectiveness of DNN compared with LAM and NRM. Moreover, Table 3 reports the quantitative comparisons among LAM, NRM, and DNN in the testing experiments. The results are generally consistent with the phenomenons in Figure 12. In all these four instances, the predicted position and velocity errors by LAM and NRM are very large with respect to all the six performance indicators. The largest position predicting error even exceeds $240m$ in Instance 2. Meanwhile, DNN still maintains a very small predicting error in all these four instances, and its performance entirely outperforms LAM and NRM. In particular, we see that the largest velocity predicting error is only about $0.25m/s$ in Instance 3, which demonstrates its capability to model the train dynamics in practice.

Table 3: Performance comparison with testing data

Performance indicators		LAM	NRM	DNN
Instance 1	MAE_S	89.32	87.42	**2.00**
	$RMSE_S$	103.67	114.32	**7.57**
	MRE_S	0.64	0.55	**0.01**
	MAE_v	3.13	2.48	**0.07**
	$RMSE_v$	3.41	2.66	**0.01**
	MRE_v	1.13	0.77	**0.02**
Instance 2	MAE_S	63.13	242.12	**3.44**
	$RMSE_S$	68.81	318.86	**4.50**
	MRE_S	0.69	0.78	**0.09**
	MAE_v	1.84	7.60	**0.23**
	$RMSE_v$	2.12	8.38	**0.30**
	MRE_v	0.90	2.57	**0.11**
Instance 3	MAE_S	131.3	106.9	**8.19**
	$RMSE_S$	147.70	141.71	**9.63**
	MRE_S	0.75	0.59	**0.08**
	MAE_v	3.46	3.26	**0.25**
	$RMSE_v$	3.79	3.59	**0.30**
	MRE_v	1.06	1.00	**0.07**
Instance 4	MAE_S	75.69	121.75	**7.46**
	$RMSE_S$	89.11	161.36	**9.35**
	MRE_S	0.65	0.65	**0.10**
	MAE_v	3.05	3.73	**0.24**
	$RMSE_v$	3.31	4.08	**0.34**
	MRE_v	1.09	1.18	**0.08**

For clearly understanding the learning process of DNN models, we present the detailed predicted train velocity and position curves during different iterations of DNN in instance 1. Figure 14 demonstrates the predicted results and field data under iterations 0, 40000, 80000, 120000, 160000, 200000 and 250000. From the figures, we can see that DNN model performs very bad in the beginning phase with a set of initial parameters. From iterations 80000 to 120000, its performance becomes better and better as DNN model gradually learns the weights with ADAM. As we continue to train the model parameters of DNN, we see that it is finally capable to accurately predict train positions and velocities after about 200000 iterations.

Remark 4.2: In our experiments, we have also tried to use shallow NN (feed-forward NN with only one hidden layer) and adaptive-network-based fuzzy inference system for modeling train control dynamics. However, we have


(a) Predicted train velocity curves

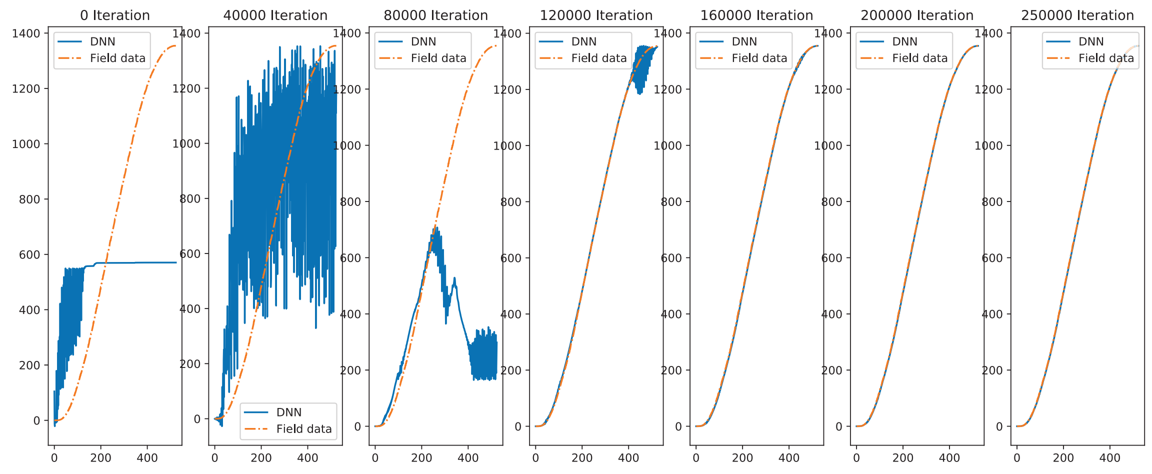

(b) Predicted train position curves

Figure 14: Predicted train velocity and position curves during different iterations

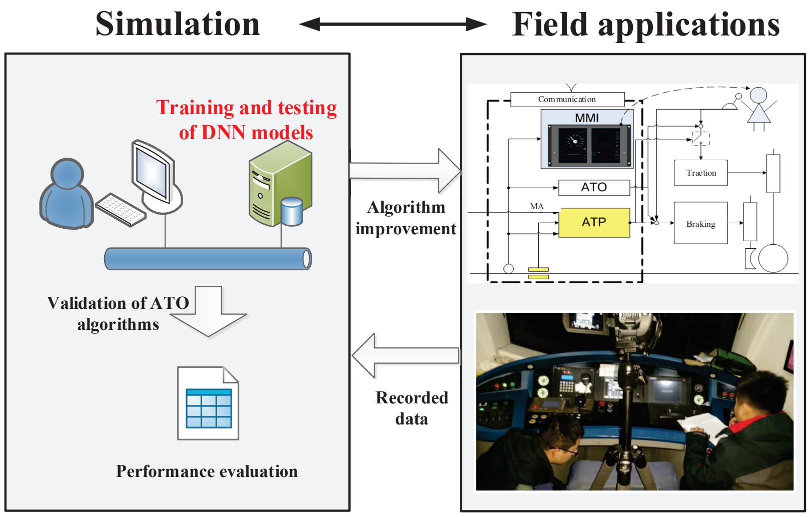


Figure 15: Simulation platform based on DNN models

found that the performances of these shallow networks are very sensitive to the initial parameter settings. In particular, when we generate the initial parameters randomly, we obtain good results in about two experiments within a total of ten experiments. In this sense, they cannot meet the practical requirements.

4.3 Real-world implementations in Beijing Metro

We implement our data-driven approaches through a collaboration project with Beijing Metro and a railway signalling company in order to improve the ATO algorithms from Oct. 2014. Some background information about the project can be found in Yang et al. (2016) and Yin et al. (2017). Before developing the data-driven models described in this study, we have to take a long trial-and-error process to field-test our designed ATO algorithms since the train parameters provided by vehicle companies are not accurate enough. As we have gradually collected a lot of historical data from 2014 in Beijing Metro, we began to develop TCMs in our lab by using the historical data. Initially, we trained linear regression based models, which are however not useable in practice since the identified train models are very different from the real-world case. Then, we turned our attention to the nonlinear train control models and moreover the DNN models. The huge amount of practical data enable to train and polish the NRM and DNN models in our lab, and the results are satisfying that they provide accurate and robust solutions to predict train moving dynamics. In particular, the high accuracy of the DNN model enables us to test our ATO algorithms directly through the in-lab environment, which greatly reduces the cost of field tests in the past. Until now, a virtual platform (as shown in Figure 15) has been successfully established based on the data-driven approaches and all the ATO algorithms can be pre-tested in this platform. Using this platform, we have embedded energy-efficient train control strategies in ATO systems in Beijing Metro since 2017.

As more and more data-driven (or artificial intelligence (AI) based) frameworks being developed in recent years, our experience in railway engineering is that although AI techniques have shown their advantages in some aspects (e.g., image processing, gaming), direct implementation of AI techniques in real-time railway train control is still impractical due to a lot of engineering limitations. For example, the computational capability of vehicle on-board computers still block the implementation of deep learning algorithms. On the other hand, the output of deep learning algorithms may be unforeseen as we have shown in Figure 14, and trial-and-error tricks are very common in AI field (such as reinforcement learning, stochastic gradient descent). In practical engineering areas, this is actually not allowed at all due to its unreliability and safety risks. Instead, our research demonstrates that data-driven and deep learning techniques can be very useful and effective in modeling real-world complex processes in an off-line environment since they have the capabilities in capturing a high-dimensional and huge amount of historical data. It is more reasonable to train (off-line) and validate data-driven models which can be useful for both testing in labs and field applications.

Remark 4.3: Our former experiments have shown the advantages of DNN compared with LAM and NRM approaches. Nevertheless, we do acknowledge that there are disadvantages of the developed DNN models in this paper. First, our experiments in the former sections have shown that "the better results are obtained when the DNN model is larger". However, the training time of DNN model is also increased dramatically. In this sense, how to realize the trade-off between model complexity and model predicting accuracy is a remained question. Second, our paper only implements DNN models for urban rail transit systems, where the external environment is much simpler compared with high-speed railways. It is undoubtedly that the DNN models cannot be directly applied in high-speed railways.

5 Conclusion

In this paper, we proposed a data-driven approach for modeling the train control models in urban rail systems with the aid of historical data. Three data-driven models, i.e., LAM based on linear regression, NRM based on nonlinear optimization and DNN based on deep neural network, were respectively developed and their parameter training algorithms were also specified. In particular, to improve the training efficiency of DNN, we designed the input and output layers of DNN and embedded batch normalization techniques according to the unique characteristics of railway train dynamics. To compare the performance of these three models, we defined six performance indicators to evaluate the train position and velocity predicting errors. The models were trained and tested with field data in Beijing Metro. The results demonstrate that DNN can significantly improve the accuracy of train models by as much as ten times compared with LAM and NRM. We also test the robustness of the DNN model associated with different model parameters, and moreover, we present our field experiences in implementing our data-driven models in practical train operations.

With the aid of advanced sensoring and communication technologies, a big amount of data have become available in recent years. Undoubtedly, deep learning is a very promising technique in the oncoming years that may tremendously change our life (Wang et al., 2018). Our study takes a first step toward applying these new techniques in practical applications of urban rail transit systems. Even though, some key issues still deserve further investigations. First, although our experiments have shown that DNN can be relatively stable with a large number of nodes on each hidden layer, it may still reach a local optimum under some particular parameter settings. In this sense, new training algorithms of DNN models should be explored to overcome these issues. Second, our paper only utilizes a fully-connected artificial neural network, and other deep learning structures such as recursive neural network and long short-term memory network should be addressed to further improve the flexibility and robustness of data-driven train control models.

Acknowledgement

This research was supported by the National Natural Science Foundation of China (Nos. U1734210, 71901016, 61790570, 61790573), the Beijing Jiaotong University Education Foundation (Nos. 9907006519) and by the China

Postdoctoral Science Foundation Funded Project (2019M650474). The collaboration with the UK partner is made possible thanks for a Newton Fund grant supported by the Royal Academy of Engineering (UK-CIAPP\286).

References

Albretcht A., Howlett P., Peter P., Vu X., Zhou P., 2016. The key principles of optimal train control-part 1: formulation of the model, strategies of optimal type, evolutionary lines, location of optimal switching points. Transportation Research Part B, 94, 482-508.

Albrecht T., 2014. Energy-efficient train operation. In: Hansen, I.A., Pachl, J. (Eds.), Railway Timetabling & Operations, Hamburg, pp. 91-116.

Andersen D.R., Booth G.F., Vithani A.R., Singh S.P., Prabhakaran, A. Stewart M.F., Punwani S., 2012. Train energy and dynamics simulator (teds)-a state-of-the-art longitudinal train dynamics simulator. In: Proceedings of the ASME 2012 Rail Transportation Division Fall Technical Conference (RTDF2012), Omaha (USA). American Society of Mechanical Engineers, pp. 16-17.

Bansal S., Akametalu A.K., Jiang F.J., Laine F., Tomlin C.J., 2016. Learning quadrotor dynamics using neural network for flight control. arXiv preprint arXiv:1610.05863.

Bham G., Benekohal R., 2002. Development, evaluation, and comparison of acceleration models. In: 81st Annual Meeting of the Transportation Research Board, Washington, DC.

Bordes A., Chopra S., Weston J., 2014. Question answering with subgraph embeddings. In Proc. Empirical Methods in Natural Language Processing, 2014.

Bottou L., 2010. Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT 2010, USA.

Cao Y., et al., 2019. Bio-inspired speed curve optimization and sliding mode tracking control for subway trains. IEEE Transactions on Vehicular Technology, 2019. (DOI:10.1109/TVT.2019.2914936)

Chen D., Gao C., 2012. Soft computing methods applied to train station parking in urban rail transit. Applied Soft Computing, 12, 759-767.

Cohen J.M., Barron A.S., Anderson R.J., Graham D.J., 2015. Impacts of unattended train operations on productivity and efficiency in metropolitan railways. Transportation Research Record: Journal of the Transportation Research Board, 2534, 75-83.

Cole C., 2006. Longitudinal train dynamics. Handbook of railway vehicle dynamics, 239-277.

De Martinis V, Corman F., 2018. Data-driven perspectives for energy efficient operations in railway systems: Current practices and future opportunities. Transportation Research Part C: Emerging Technologies, 95, 679-697.

Duan Y., Lv Y., Liu Y., Wang F-Y., 2016. An efficient realization of deep learning for traffic data imputation. Transportation Research Part C, 72, 168-181.

Fadhloun K., Rakha H., Loulizi A., Abdelkefi A., 2015. Vehicle dynamics model for estimating typical vehicle accelerations. Transportation Research Record, 2491, 61-71.

Fang S., Fei Y., Xu Z., Tsao Y., 2017. Learning transportation modes from smartphone sensors based on deep neural network. IEEE Sensors Journal, 17(18), 6111-6118.

Hannun A. Y., Rajpurkar P., Haghpanahi M., Tison G. H., Bourn C., Turakhia M. P., Ng A.Y., 2019. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nature medicine, 25(1), 65.

Gao S., Dong H., Chen Y., Ning B., Chen G., Yang X., 2013. Approximation-based robust adaptive automatic train control: an approach for actuator saturation. IEEE Transactions on Intelligent Transportation Systems, 14(4), 1733-1742.

Grag V., 2012. Dynamics of Railway Vehicle Systems. Elsevier.

Gbologah F., Xu Y., Rodgers M., Guensler R., 2014. Demonstrating a bottom-up framework for evaluating energy and emissions performance of electric rail transit options. Transportation Research Record, 2428, 10-17.

Grube P., Nunez F., Cipriano A., 2011. An event-driven simulator for multi-line metro systems and its application to Santiago de Chile metropolitan rail network. Simulation Modelling Practice and Theory, 19, 393-405.

Hinton et al., 2012. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine, 82-97.

Holett P.G., Pudney P.J., 1995. Energy-efficient train control. Springer-Verlag, Berlin.

Ioffe S., Szegedy C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

Kingma D.P., Ba J., 2014. ADAM: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Lagay R., Adell G.M., 2018. The Autonomous Train: a game changer for the railway industry. 2018 16th International Conference on Intelligent Transportation Systems (ITSC), 1-5, IEEE.

LeCun Y., Bengio Y., Hinton G., 2015. Deep learning. Nature, 521, 436-444.

Li S., Yang L., Li K., Gao Z., 2014. Robust sampled-data cruise control scheduling of high speed train. Transportation Research Part C, 46, 274-283.

Lv Y., Duan Y., Kang W., Li Z., Wang F.Y., 2015. Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems, 16(2), 865-873.

Ma et al., 2015. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C, 54, 187-197.

Nash A., Huerlimann D., 2004. Railroad simulation using OpenTrack. WIT Transactions on The Build Environment, 74, 2004.

Oprea R.A., Cruceanu C., Spiroiu M.A., 2013. Alternative friction models for braking train dynamics. Vehicle System Dynamics, 51(3), 460-480.

Qiu et al., 2019. Observer-based fuzzy adaptive event-triggered control for pure-feedback nonlinear systems with prescribed performance. IEEE Transactions on Fuzzy Systems, 2019.

Scheres S.H., Chen S., 2012. Prevention of overfitting in cryo-EM structure determination. Nature Methods, 9(9), 853.

Sekuła P., Marković N., Laan Z.V., Sadabadi K.F., 2018. Estimating historical hourly traffic volumes via machine learning and vehicle probe data: A Maryland case study. Transportation Research Part C, 97, 147-158.

Serajian R., Mohammadi S., Nasr A., 2019. Influence of train length on in-train longitudinal forces during brake application. Vehicle System Dynamics, 57(2), 192-206.

Silver D., Huang A., Maddison C.J., et al., 2016. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587), 484.

Su S., Li X., Tang T., Gao Z., 2013. A subway train timetable optimization approach based on energy-efficient operation strategy. IEEE Transactions on Intelligent Transportation Systems, 14(2), 883-893.

Toole J.L., Colak S., Sturt B., Alexander L.P., Evsukoff A., González M.C., 2015. The path most traveled: Travel demand estimation using big data resources. Transportation Research Part C, 58, 162-177.

Wang Y., Zhang D., Liu Y., Dai B., Lee L.H., 2018. Enhancing transportation systems via deep learning: A survey. Transportation Research Part C, in press.

Wang J., Rakha H.A., 2018. Longitudinal train dynamics model for a rail transit simulation system. Transportation Research Part C, 86, 111-123.

Wu Q., Cole C., Luo S., Spiryagin M., 2014. A review of dynamics modelling of friction draft gear. Vehicle System Dynamics, 52(6), 733-758.

Wu Q., Spiryagin M., Cole C., 2016. Longitudinal train dynamics: an overview. Vehicle System Dynamics, 54(12), 1688-1714.

Wu X., Guo J., Xian K., Zhou X., 2018. Hierarchical travel demand estimation using multiple data sources: A forward and backward propagation algorithmic framework on a layered computational graph. Transportation Research Part C, 96, 321-346.

Yang X., Li X., Ning B., Tang T., 2016. A survey on energy-efficient train operation for urban rail transit. IEEE Transactions on Intelligent Transportation Systems, 17(1), 2-13.

Ye H., Liu R., 2017. Ye H., Liu R., 2017. Nonlinear programming methods based on closed-form expressions for optimal train control. Transportation Research Part C, 82, 102-123.

Yin J., Chen D., Li L., 2014. Intelligent train operation algorithms for subway by expert system and reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, 15(6), 2561-2571.

Yin J., Zhao W., 2016. Fault diagnosis network design for vehicle on-board equipments of high-speed railway: A deep learning approach. Engineering Applications of Artificial Intelligence, 56, 250-259.

Yin J., Tang T., Yang L., Xun J., Huang Y., Gao Z., 2017. Research and development of automatic train operation for railway transportation systems: A survey. Transportation Research Part C, 85, 548-572.

Yin J., Yang L., Zhou X., Tang T., Gao Z., 2019. Balancing a one-way corridor capacity and safety-oriented reliability: a stochastic optimization approach for metro train timetabling. Naval Research Logistics (NRL), 66(4), 297-320.