On the Data-Driven Prediction of Arrival Times for Freight Trains on U.S. Railroads

William Barbour[1], Chinmaya Samal[2], Shankara Kuppa[3], Abhishek Dubey[4], Daniel B. Work[5]

***Abstract*— The high capacity utilization and the predominantly single-track network topology of freight railroads in the United States causes large variability and unpredictability of train arrival times. Predicting accurate *estimated times of arrival* (ETAs) is an important step for railroads to increase efficiency and automation, reduce costs, and enhance customer service. We propose using machine learning algorithms trained on historical railroad operational data to generate ETAs in real time. The machine learning framework is able to utilize the many data points produced by individual trains traversing a network track segment and generate periodic ETA predictions with a single model. In this work we compare the predictive performance of linear and non-linear support vector regression, random forest regression, and deep neural network models, tested on a section of the railroad in Tennessee, USA using over two years of historical data. Support vector regression and deep neural network models show similar results with maximum ETA error reduction of 26% over a statistical baseline predictor. The random forest models show over 60% error reduction compared to baseline at some points and average error reduction of 42%.**

I. Introduction

Freight trains on US railroads can have unpredictable runtimes due to infrastructure capacity constraints resulting from the topology of the networks, which are primarily composed of single track with sidings [1]. Single track network segments require complex dispatching by human dispatchers to coordinate meet and pass movements [2] and have been shown to be a primary driver of train delay [3]. This is in contrast to multi track routes which run with much more regularity and predictability and consequently allow more trains to achieve their minimum runtimes [4]. Additionally US networks carry a heterogeneous train mix with respect to train type and physical characteristics. Freight trains vary widely in length and priority and some locations serve both freight and passenger trains which can also be quite long; all of these factors are known to negatively impact predictability of the network and delay experienced by trains [5]. Accurate predictions of *estimated times of arrival* (ETAs) for train made hours in advance are necessary for railroad operations because they are used for calling the next crew, assembling trains with mixed cargo, allocating locomotives, and providing customer delivery estimates [6].

The research question addressed is the prediction of ETAs for freight trains on US railroads. In our prior work [7], we considered the prediction problem for US freight railroads using individualized machine learning models in which a distinct model is used for each origin-destination pair. This was shown to allow the models to learn location-specific feature weights to implicitly account for the fact that some factors (e.g., tonnage) have a more important role in subsets of the network (i.e., when traversing in terrain with a steep grade). A variety of linear and non-linear support vector regression models were able to demonstrate a 21% improvement over a baseline statistical method.

In the present work, we expand the comparison of methods to include a deep learning based method and a random forest predictor in addition to the support vector regression methods considered in [7]. To address the larger data requirements encountered by the deep learning based method, we encode the origin-destination information in the feature space and present a single model for all origin destination pairs on the network. We find that the quality of the ETA can vary greatly between the various machine learning methods, with the random forest significantly outperforming all other methods considered in this work. Consequently, the main research contribution of this article is a comparison of existing, well-studied machine learning algorithms for suitability in the real-time generation of accurate arrival

[1]William Barbour is with the Department of Civil and Environmental Engineering and the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37240

[2]Chinmaya Samal is with the Department of Computer Science and Computer Engineering and the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37240

[3]Shankara Kuppa is with CSX Transportation, Jacksonville, FL 32202

[4]Abhishek Dubey is with the Department of Computer Science and Computer Engineering and the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37240

[5]Daniel B. Work is with the Department of Civil and Environmental Engineering and the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37240

times for freight trains on congested US rail networks using a single unified model that makes predictions from multiple locations.

The remainder of this article is organized as follows. In Section II we examine some prior work related to the research question of predicting and updating ETAs for freight trains. In Section III, we present the machine learning problem framework, details of models used, and data features used to predict ETAs. In Section IV, the implementation and experimental specifics are outlined and results are presented. Sections V and VI compare and discuss model performance, conclude the article, and suggested avenues of further investigation.

II. Related work

There are a variety of techniques by which ETAs can be generated, including analytical methods, simulation tools, and data-driven methods. Analytical methods have been widely used to produce schedules and arrival estimates for passenger and freight trains based on the network topology and the traffic across particular routes. These scheduling methods are often solved using optimization and can handle multiple objectives such as delay minimization or passenger cost [8]; see [9], [10] for an extensive survey of many of these methods. Heuristic techniques are often employed to solve these optimization problems [11]. Scheduling has been extended to be robust to small perturbations produced by delays [12] and applicable for mixed passenger and freight traffic [13]. This is important because small deviations from a predetermined schedule can easily compound into larger deviations and delay, which is referred to as *knock-on delay* [14]. Additionally, rescheduling can be performed regularly or as new information becomes available [2], [15]. Simulation methods, likewise, can predict train trajectories and ETAs and have been shown in some areas to closely match reality; see [1], [11] for examples and a more thorough comparison of analytical and simulation literature.

Analytical models and simulation methods have the notable caveat of requiring explicit declaration of rules, such as train dispatching rules, track speed limits, train dynamics, and physical constraints [14]. The arrival times of freight traffic on US railroads are due to many factors of the heterogeneous train lineup and network congestion factors regarding the available capacity and delay that can be expected from meets and passes on single track sections [5]. Additionally, the effects from these many variables are not absolute. Analytical models and scheduling methods are adept at finding a feasible train plan (end even an optimal one) but it is difficult to account for the full range of intricacies that governs dynamic freight rail operations in practice and the changes in these dynamics between locations and over time [16]. For these reasons, we address the problem with a data-driven approach, which allows a prediction model to capture complex relationships between variables implicitly through the training process without enumerating the rules and relationships explicitly [16].

Machine learning methods, in general, are adept at this task of learning linear and non-linear relationships between input variables and outputs, but deep neural networks in particular can capture a high degree of interdependent and non-linear complexity [17]. However, the most notable limitation of data-driven methods is the requirement of abundant high-quality data. Increased availability of data streams in the rail industry has given rise to many modern applications in operations, maintenance, and safety [6].

Data-driven prediction has been studied primarily in the context of passenger rail. Kecman and Goverde [18] illustrate the importance of route conflicts in their prediction of train event times; they also estimate running and dwell times using track occupancy data and multiple predictive models, including random forests [19]. Wang and Work [16] created historical and online regression models to predict Amtrak arrival times and noted significant advantages of using data from a train's current performance as well as that of its neighbors in the online regression model. Marković et al. [20] applied support vector regression and artificial neural network models to the delay prediction problem on Serbian passenger rail. Yaghini et al. [21] also used a neural network model trained only on time and route information in predicting discretized passenger train delay in Iran. Liu et al. [22] used a neural network model in predicting runtime between stations and dwell time at stations in urban rail transit. Oneto et al. [23] explored neural network architecture and performance for the Italian rail network using historical and real-time data. Compared to freight rail in the United States, passenger rail has delay values typically on the order of minutes, compared to the hours of delay that can occur for US freight trains.

In terms of freight rail prediction, Gorman [3] calculated freight train running time by estimating free run time and congestion-related delay by geographic area and performed an econometric analysis of individual delay variables. Bonsra and Harbolovic [24] performed estimation of freight train arrival times in an offline manner using resulting route conflicts between trains and also noted the importance of these congestion-related factors as well as train priority.

III. Methodology

This section discusses general supervised machine learning framework used to predict ETAs, the raw data and the processed training data, and the specifics of the deep learning model and comparison models used to address the problem of ETA prediction.

A. Supervised machine learning ETA prediction

In the context of predicting ETAs, we consider a machine learning formulation in which a set of training data $x_{tr} = \{x(1), x(2), ..., x(m)\}$, where $x(i) \in \mathbb{R}^n$, and corresponding known outputs $y_{tr} = \{y(0), y(1), ..., y(m)\}$ are used to learn a model $f(\cdot)$ that maps input vectors $x(i)$ to output values $y(i)$. The dimension n refers to the number of input *features* that are used to describe each sample. The model should minimize some measure of prediction error $(f(x(i)) - y(i))$, which is referred to as a *loss*. We test the resulting model using test input data $x_{ts} = \{x(m+1), x(m+2), ..., x(m+p)\}$ and known output data $y_{ts} = \{y(m+1), y(m+2), ..., y(m+p)\}$ that does not overlap with training data. It is desirable to learn a model that generalizes new data that it has not seen before. Failing to do so is referred to as overfitting and occurs when a model is allowed to become overly complex to produce high accuracy on only observations that have been used directly to train the model.

In our case, the training samples $x(i)$ are each a large feature vector containing information about the freight train, its crew, and the network state at the time that a prediction is being made. The output $y(i)$ refers to the true runtime between the train's current position at the time the feature vector was computed and the train's destination. These points in time when ETAs are generated correspond to discrete locations on the network, referred to as *stations* or *mileposts*, and typically coincide with control points at the ends of sidings. In this manner, many ETA updates are generated for a train along a route.

Consequently, the features change as time elapses. Not only does the network state change, but the characteristics of the train and its crew are also subject to change. Therefore, each feature vector $x(i)$ contains real-time data from the exact moment that the ETA should be generated at the corresponding milepost. Next we briefly describe the models compared in this work.

B. Statistical model

We benchmark all machine learning model against a statistical predictor that uses historical data. Simple statistical predictors are known to be in use in some ETA prediction systems on the railroad. For this reason, we include a statistical model that predicts the median runtime $\tilde{y}$ from a route station to the train's destination as the ETA for each train. The median is used in lieu of the mean because the distribution of train runtime has a long tail of high outlier values. Due to the confidentiality of the raw dataset, rather than presenting absolute runtimes (which are on the order of hours, and can have delays on the order of 10s of minutes to hours), we present all results normalized to this baseline model.

C. Support vector regression

The *support vector regression* (SVR) model considered in this work uses the standard ε-SVR formulation described in [25]. The predictor $f(x) = w^T x(i) + b$, where $w \in \mathbb{R}^n$ is the vector of feature weights and $b \in \mathbb{R}^1$ is the regression intercept, is derived from the convex optimization problem:

$$\begin{aligned} \underset{w,b,\xi,\xi^*}{\text{minimize}} \quad & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m}(\xi(i) + \xi^*(i)) \\ \text{subject to} \quad & y(i) - w^T x(i) - b \leq \varepsilon + \xi(i), \quad \forall i \\ & w^T x(i) + b - y(i) \leq \varepsilon + \xi^*(i), \quad \forall i \\ & \xi(i),\ \xi^*(i) \geq 0, \quad \forall i \end{aligned} \tag{1}$$

The model applies a 2-norm to the feature weights w and uses a hinge loss with a tolerance ε to errors where the loss is zero. Deviations $y(i) - f(x(i))$ for each observation i in excess of ε are denoted $\xi(i)$ and $\xi^*(i)$ and penalized by a scalar factor C relative to feature weights. The coefficient C and ε are hyperparameters that are tuned in the model training phase.

Support vector regression also allows the use of a non-linear kernel function that transforms the input data $x(i)$ into a much higher dimensional space $\Phi(x(i))$. The higher dimensional transformation is not defined explicitly but is the result of the inner product $\Phi^T\Phi$ in the regression, known as the kernel trick [26]. In this work we present results using a polynomial kernel and radial basis function kernel.

D. Random forest regression

Random forest regression is an ensemble algorithm that constructs a series of regression trees using randomly sampled subsets of the training data for each tree and a subset of available data features for splitting within trees [27], [28]. Regression trees are constructed by splitting data samples at each node in the tree according to values of input features. Each node resulting from the split more effectively isolates data samples with similar output values. The best split is determined by a minimization of resulting prediction error. Nodes

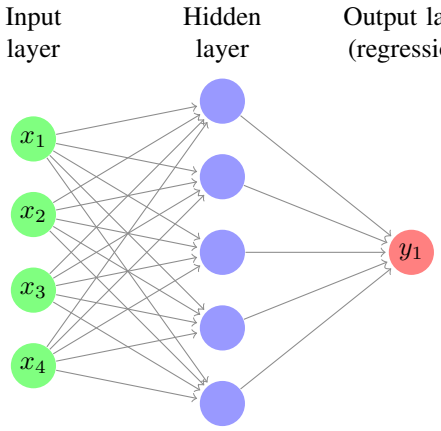

Fig. 1: Simple Feed-forward neural network with one hidden layer.

are no longer split when the number of samples in the node falls below the minimum or the decrease in prediction error falls below a defined threshold. The predicted output value for each terminal node in the tree is calculated from the corresponding training samples that terminated in the node. The predictions made by individual trees are averaged to arrive at the ensemble prediction. Combining many weak learner regression trees in the random forest predictor has shown to be an effective methodology and helps avoid overfitting [29], [30].

E. Deep feed-forward neural network model

A neural network consists of multiple neurons organized in layers, with individually weighted connections between neurons in adjacent layers. At minimum, a feed-forward neural network consists of three distinct layers: the input layer, one hidden layer, and an output layer consisting of one node for regression, as shown in Figure 1, or multiple nodes for classification. A deep feed-forward neural network has multiple hidden layers. The depth of the model is determined by the number of hidden layers. After being processed at the hidden layer nodes, their outputs are forwarded to the output layer which then makes a prediction, according to its activation function. This feed-forward process is characteristic of the neural network and is used for making predictions, given an input vector. The training phase of a neural network is comprised of selecting the optimal weights for each of the connections between the neurons. More specifically, given the input vector at the input layer, and the known output that actually occurred, the problem is defined as choosing the weights for the connections between neurons so as to minimize the error between the prediction and actual observation. Gradient-based optimization is used for training the neural network and choosing weights.

F. Description of data

Machine learning algorithms have the notable limitation of requiring a large amount of data to complete the training process. This work is made possible by access to over two years of historical train data from CSX Transportation that includes information about physical train characteristics, train crew information, and can be used to compile a full network state at any time during the data collection period.

Specifically, we build a feature set with the quantities described in Table I. The feature set is a mix of categorical, binary, and continuous quantities. Network state is calculated based on trains occupying network segments between stations, along with their relevant properties. The prediction location is included as a one-hot encoding of the route locations.

The resulting feature space has 184 dimensions and the number of labeled data records is over 170,000, each of which represents a feature vector captured at a timing point for the train and the corresponding runtime label. The training and testing data is min-max normalized before being used in the models.

IV. Numerical Experiments

We now describe the exact implementation of the models and the experimental details.

The network segment where these methods are applied is between Nashville, TN, and Chattanooga, TN. The segment is 140 miles long and is almost entirely single-track. Other CSX network segments connect at the endpoints of the route, but none intersect the middle of the route. The mileposts at which ETAs are updated are spaced about 4 miles apart on average. This is known to be one of the more unpredictable routes on the network during the period on which the data was collected, as it has high volume relative to capacity, a heterogeneous train mix, and significant topography, which is difficult to navigate for long and heavy trains.

The neural network models were implemented using Keras [31] with TensorFlow [32] backend. Support vector regression, random forest, and statistical models were built using Scikit-Learn [33]. All models were tested on a computer with 16-core 3.4 GHz processor, 64 GB of RAM, and Nvidia GTX 1080 GPU. Note that neural network models were run on the GPU, while other models were run on the CPU.

Model performance was evaluated using five-fold shuffled cross validation. As previously mentioned, in

TABLE I: Summary and dimension of implemented features.

Feature	Dim.	Description
Train length, tonnage and horsepower per ton	3	The physical characteristics of the train.
Train priority	1	Priority ranking on a 1-20 scale; mapping given by CSX.
Crew time remaining	1	Amount of time remaining that the current train crew can legally work.
On duty time to departure	1	Time between crew on duty time and train departure.
Traffic counts	7	Count of other trains between origin and destination; includes total count and counts subdivided by direction and priority relative to train being predicted.
Available sidings	1	Count of sidings on route with length greater than that of the train being predicted.
Track segment occupancy	30	Binary variables denoting whether a segment on the origin-destination route is occupied by another train.
Occupying train direction, priority, and relative priority	90	Denotes the direction (same or opposite), priority (1-20 scale), and relative priority (compared to train being predicted) of a train occupying a track segment on the origin-destination route.
Track segment occupancy around destination point	15	Indicates track segment occupancy for segments around the destination point, but not included in the primary route.
Location	35	Binary indication of station corresponding to this feature vector.

order to maintain confidentiality of the operational practices on this route segment, the machine learning models are compared to the statistical predictor as a baseline and results are given in percent improvement in mean average error over the baseline predictor. Mean average error would be in units of minutes and is defined as

$$MAE = \frac{1}{m_{te}} \sum_{i=1}^{m_{te}} |f(x(i)) - y(i)|. \quad (2)$$

The SVR model was tuned using an exponential grid space for the hyperparameters C and ε. Kernel hyperparameters (γ for RBF kernel and degree for polynomial kernel) were also tuned in the same grid space. Optimal values were selected from all combinations of hyperparameter values in the grid space and found to be $C = 10$. and $\varepsilon = 0.075$ for all SVR models. The random forest

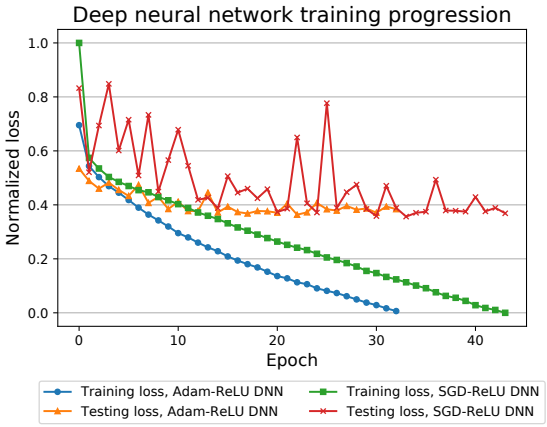

Fig. 2: Learning curve for deep neural network showing normalized training and testing loss values.

regression model was tuned by exploring a grid space that included the hyperparameters: number of estimators, maximum features considered in split, and minimum samples required for node split. Values explored in the grid space were chosen based on the dimensionality and characteristics of the data and hyperparameter values were chosen to achieve high predictive performance and minimize overfitting.

The deep neural network model architecture is the results of extensive tuning both in the configuration of hidden layers (from three to ten hidden layers were tested), activation function (ReLU and tanh were tested) and optimization function (Adam and SGD were tested) used in the model. Ultimately, eight hidden layers are used with 200, 200, 150, 100, 70, 40, 20, and 10 nodes in each, respectively. The rectified linear unit [34] is chosen for the activation function of neurons; the Adam optimizer [35] is found to perform best for training the neural network. Early stopping criteria are employed to avoid overfitting. The learning curve for the resulting Adam-ReLU model is shown in Figure 2 and compared to the same architecture using stochastic gradient descent optimization. The Adam optimizer not only converges faster (in 32 epochs), but also shows far less variability in validation loss on the way to convergence.

The predictions resulting from the testing data samples are grouped by the station to which they correspond. Each of these samples serves as an ETA prediction made for a particular train at that station, traveling towards the destination following station 35. The results for all six models are shown in Figure 3, in terms of percent improvement in MAE compared to the baseline

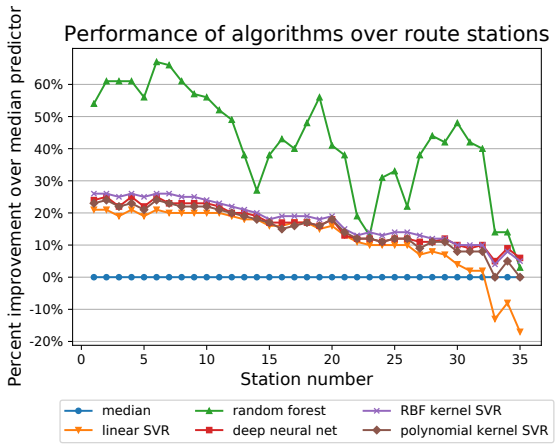

Fig. 3: Relative improvement of arrival time estimates at each station.

TABLE II: Summary of model performance over full route

Model	Mean % Improvement	Maximum % Improvement
Median	0.0%	0.0%
Linear SVR	12.2%	21.0%
Polynomial kernel SVR	15.3%	23.6%
RBF kernel SVR	17.6%	26.4%
Random Forest	42.1%	67.1%
Deep Neural Net	16.3%	24.9%

statistical predictor. The results across the route are averaged and shown in Table II. Model training times were also monitored and are shown in Table III. SVR models are constrained to single-threaded computation in this implementation. Conversely, random forest model can be trained in parallel across CPU cores and the DNN model can use the GPU for computation.

V. Discussion

There is a noticeable grouping of the SVR models and DNN that maintains over 20% improvement over baseline at stations far from the destination and decreases in relative performance approaching the destination. Linear SVR conspicuously drops below the baseline at the three stations closest to the destination. Meanwhile, the random forest model outperforms all other models at nearly every station. Its performance varies more widely

[1]Random forest implementation runs in parallel on CPU.
[2]DNN was implemented on GPU.

TABLE III: Mean model training time.

Model	Mean training time (seconds)
Median	0.1
Linear SVR	20
Polynomial kernel SVR	11360
RBF kernel SVR	5560
Random Forest[1]	25
Deep Neural Net[2]	250

across the route, but achieves improvements exceeding 60% relative to the baseline at stations far from the destination. Predictions at this point are of particular interest for the railroad because of the difficulty of prediction and the increased decision-making potential for hours in the future. The random forest model also see frequent prediction improvements over 50% and average 42% improvement over baseline across the route.

As predictions are made closer to the destination, the mean runtime and expected mean average error (in absolute terms) decrease. But mean average error relative to baseline also decreases as predictions were made closer to the destination. This is likely due to the fact that runtimes are also less variable close to a train's destination and the factors that drive the residual variability are difficult to quantify with available data. For example trains can be held outside of the yard due to personnel constraints or space constraints such as lack of availability of a specific track needed (e.g., for refueling or classification). We hope to construct features that quantify this destination yard state in future work.

Fluctuating performance of all models, but particularly the random forest model is notable. This can be explained in part by the nonlinear dynamics of the route. Train and route features differ in their predictive impact by location [7]. For example, route topography plays a role in the predictive impact of train length and tonnage. At locations with a significant hill on the route, long and heavy trains will have a statistically higher runtime than others; but after the hill is traversed, the statistical difference in remaining runtime will diminish. We see a performance variation at approximately the route midpoint that is likely due to a mountain that must be traversed producing an effect of this sort. However, the dramatic performance variability of the random forest model is likely caused by the nuanced relationships that tree-based regressors can extract from categorical and binary data such as the network state used in this work. It is possible that predictions made by the random forest

model at some of the low-performing stations depend highly on additional variables not present in the feature space, such as availability of helper locomotives that supplement train power when traversing hills that are present on the route. The training error of the random forest model is up to 10% lower than the testing error (in absolute terms), but the testing results are consistent through cross validation.

VI. Conclusion

In this article we discuss the advantages and difficulties of producing freight train arrival time estimates using data-driven methods. We discuss the raw data and input features used in the research. Six models including one statistical model, three SVR models, a deep neural network model, and a random forest regression model are implemented. Performance of the models is analyzed at locations across the study area and found to vary, particularly for the random forest model.

The random forest model achieves the best performance yet realized on this data set, with an average 42% improvement in MAE relative to the baseline statistical predictor. The average improvement of the random forest model and the maximum predictive improvements of over 60% are actionable for freight rail operational decision making.

Areas of future research include enhancing the feature space, which may include route topography, data quantifying the state of rail yards, and externalities such as weather and scheduled maintenance. We also know that the delay encountered by a train on the line of road is closely coupled to the meet and pass movements that the train will make on its route. For this reason, we are investigating trajectory prediction from the dispatching perspective. A dispatching model that acts similarly to a human dispatcher could be a highly capable ETA predictor. Additionally, we hope to investigate synthetic data generation to supplement training data for the deep neural network model, which may allow for a larger, more complex network to be built.

Acknowledgment

The authors acknowledge funding by the Roadway Safety Institute, the University Transportation Center for USDOT Region 5, which Includes Minnesota, Illinois, Indiana, Michigan, Ohio, and Wisconsin. Financial support was provided by the United States Department of Transportation's Office of the Assistant Secretary for Research and Technology (OST-R). Financial support was also provided by the Federal Highway Administration's Office of Innovative Program Delivery (OIPD), via the Dwight David Eisenhower Transportation Fellowship Program.

References

[1] Pavankumar Murali, Maged Dessouky, Fernando Ordóñez, and Kurt Palmer. A delay estimation technique for single and double-track railroads. *Transportation Research Part E: Logistics and Transportation Review*, 46(4):483–495, 2010.

[2] Andrea Dariano, Dario Pacciarelli, and Marco Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, 2007.

[3] Michael F Gorman. Statistical estimation of railroad congestion delay. *Transportation Research Part E: Logistics and Transportation Review*, 45(3):446–456, 2009.

[4] Samuel Sogin, Yung-Cheng Lai, C Dick, and Christopher Barkan. Comparison of capacity of single-and double-track rail lines. *Transportation Research Record: Journal of the Transportation Research Board*, 2374:111–118, 2013.

[5] Mark Dingler, Yung-Cheng Lai, and Christopher Barkan. Impact of train type heterogeneity on single-track railway capacity. *Transportation Research Record: Journal of the Transportation Research Board*, 2117:41–49, 2009.

[6] Faeze Ghofrani, Qing He, Rob MP Goverde, and Xiang Liu. Recent applications of big data analytics in railway transportation systems: A survey. *Transportation Research Part C: Emerging Technologies*, 90:226–246, 2018.

[7] William Barbour, Juan Carlos Martiniz Mori, Shankara Kuppa, and Daniel B. Work. Prediction of arrival times of freight traffic on us railroads using support vector regression. *Transportation Research Part C: Emerging Technologies (expected)*, 2018.

[8] Yuting Zhu, Baohua Mao, Yun Bai, and Shaokuan Chen. A bi-level model for single-line rail timetable design with consideration of demand and capacity. *Transportation Research Part C: Emerging Technologies*, 85:211–233, 2017.

[9] Jean-Francois Cordeau, Paolo Toth, and Daniele Vigo. A survey of optimization models for train routing and scheduling. *Transportation science*, 32(4):380–404, 1998.

[10] Alberto Caprara, Leo Kroon, Michele Monaci, Marc Peeters, and Paolo Toth. Passenger railway optimization. *Handbooks in operations research and management science*, 14:129–187, 2007.

[11] Quan Lu, Maged Dessouky, and Robert C Leachman. Modeling train movements through complex rail networks. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 14(1):48–75, 2004.

[12] Fahimeh Khoshniyat and Anders Peterson. Improving train service reliability by applying an effective timetable robustness strategy. *Journal of Intelligent Transportation Systems*, pages 1–19, 2017.

[13] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. Scheduling extra freight trains on railway networks. *Transportation Research Part B: Methodological*, 44(2):215–231, 2010.

[14] Malachy Carey and Andrzej Kwieciński. Stochastic approximation to the effects of headways on knock-on delays of trains. *Transportation Research Part B: Methodological*, 28(4):251–267, 1994.

[15] Francesco Corman and Egidio Quaglietta. Closing the loop in real-time railway control: Framework design and impacts on operations. *Transportation Research Part C: Emerging Technologies*, 54:15–39, 2015.

[16] Ren Wang and Daniel B Work. Data driven approaches for passenger train delay estimation. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 535–540. IEEE, 2015.

[17] Donald F Specht. A general regression neural network. *IEEE transactions on neural networks*, 2(6):568–576, 1991.

[18] Pavle Kecman and Rob MP Goverde. Online data-driven adaptive prediction of train event times. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):465–474, 2015.

[19] Pavle Kecman and Rob MP Goverde. Predictive modelling of running and dwell times in railway traffic. *Public Transport*, 7(3):295–319, 2015.

[20] Nikola Marković, Sanjin Milinković, Konstantin S Tikhonov, and Paul Schonfeld. Analyzing passenger train arrival delays with support vector regression. *Transportation Research Part C: Emerging Technologies*, 56:251–262, 2015.

[21] Masoud Yaghini, Mohammad M. Khoshraftar, and Masoud Seyedabadi. Railway passenger train delay prediction via neural network model. *Journal of Advanced Transportation*, 47(3):355–368, 2012.

[22] Yafei Liu, Tao Tang, and Jing Xun. Prediction algorithms for train arrival time in urban rail transit. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pages 1–6. IEEE, 2017.

[23] Luca Oneto, Emanuele Fumeo, Giorgio Clerico, Renzo Canepa, Federico Papa, Carlo Dambra, Nadia Mazzino, and Davide Anguita. Dynamic delay predictions for large-scale railway networks: Deep and shallow extreme learning machines tuned via thresholdout. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(10):2754–2767, 2017.

[24] Kunal Kunal Baldev Bonsra and Joseph Harbolovic. *Estimation of run times in a freight rail transportation network*. PhD thesis, Massachusetts Institute of Technology, 2012.

[25] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[26] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[27] Leo Breiman. *Classification and regression trees*. Routledge, 1984.

[28] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[29] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[30] Aparna Oruganti, Fangzhou Sun, Hiba Baroud, and Abhishek Dubey. Delayradar: A multivariate predictive model for transit systems. In *BigData*, pages 1799–1806, 2016.

[31] François Chollet et al. Keras. `https://keras.io`, 2015.

[32] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.