An Efficient Two-stage Gradient Boosting Framework for Short-term Traffic State Estimation

Yichao Lu
Layer 6 AI
`yichao@layer6.ai`

Abstract

Real-time traffic state estimation is essential for intelligent transportation systems. The NeurIPS 2022 Traffic4cast challenge provides an excellent testbed for benchmarking short-term traffic state estimation approaches. This technical report describes our solution to this challenge. In particular, we present an efficient two-stage gradient boosting framework for short-term traffic state estimation. The first stage derives the month, day of the week, and time slot index based on the sparse loop counter data, and the second stage predicts the future traffic states based on the sparse loop counter data and the derived month, day of the week, and time slot index. Experimental results demonstrate that our two-stage gradient boosting framework achieves strong empirical performance, achieving third place in both the core and the extended challenges while remaining highly efficient. The source code for this technical report is available at `https://github.com/YichaoLu/Traffic4cast2022`.

1 Introduction

Short-term traffic state estimation is a crucial task in intelligent transportation systems with many practical downstream applications [39, 42, 25, 24, 17]. The NeurIPS 2022 Traffic4cast challenge [1] provides an excellent testbed for evaluating the performance of short-term traffic state estimation approaches. Given only one hour of sparse loop counter data, the task is to predict the traffic state for all road segments 15 minutes into the future. For the core challenge, the task is to predict the congestion classifications for all road segments. For the extended challenge, the task is to predict the expected time of arrival along super-segments.

Most state-of-the-art short-term traffic state estimation approaches employ graph neural networks to capture the complex spatio-temporal dependencies among traffic flows [13, 51, 4, 28, 21]. However, the complexity of urban road networks imposes enormous computational and resource challenges. Therefore, this technical report explores an efficient alternative to graph neural networks for short-term traffic state estimation. In particular, we present a two-stage gradient boosting framework. The advantage of gradient boosting decision trees over neural networks is its high efficiency and the ability to handle missing values without imputation [49, 50, 32, 33, 31, 37]. The first stage derives the month, day of the week, and time slot index based on the sparse loop counter data, and the second stage predicts the future traffic states based on the sparse loop counter data and the derived month, day of the week, and time slot index. The intuition behind the two-stage approach is as follows. We observe strong seasonality and time trends in traffic flows [27, 56]. Figure 1 shows the relation between average volume and the time slot index for a sample loop counter in London in the Traffic4cast 2022 training data. Dissecting the task of traffic state estimation into two stages not only harnesses the information within auxiliary labels but also facilitates the second stage model to capture the time patterns in traffic flows.

---

[1] `https://www.iarai.ac.at/traffic4cast/challenge/`

Figure 1: The relation between average volume and the time slot index for a sample loop counter in London in the Traffic4cast 2022 training data.
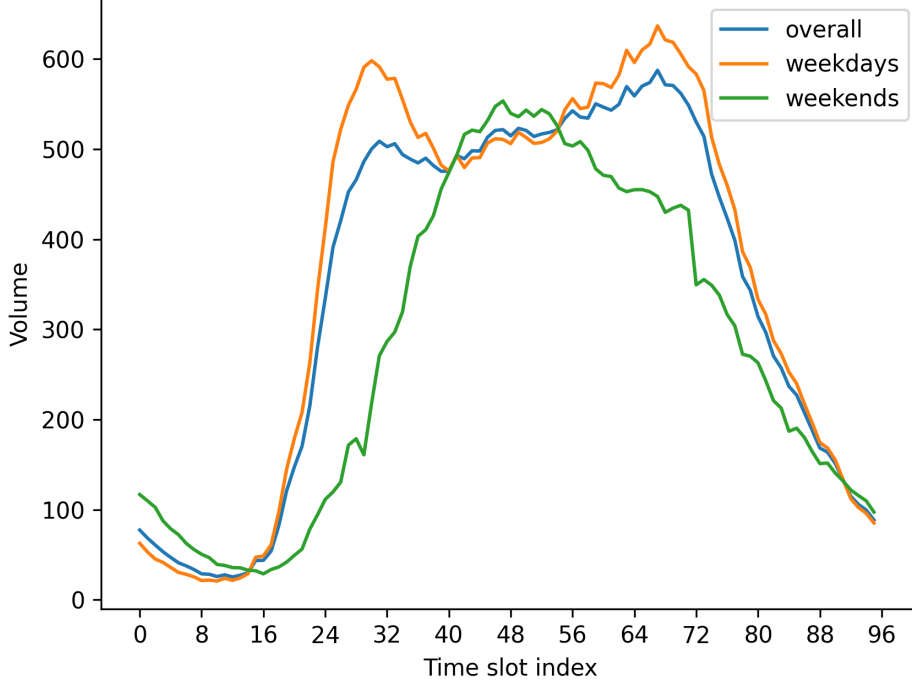
Experimental results on the NeurIPS 2022 Traffic4cast challenge leaderboards demonstrate that our two-stage gradient boosting framework achieves strong empirical performance, achieving third place in both the core and the extended challenges while remaining highly efficient. The whole pipeline can be trained in less than 3 hours on a single NVIDIA GeForce RTX 2080 Super Mobile GPU.

2 Approach

The presented two-stage gradient boosting framework is based on two highly efficient implementations of the gradient boosting decision trees algorithm: eXtreme Gradient Boosting (XGBoost) [11] and Light Gradient Boosting Machine (LightGBM) [22]; see Figure 2. The first stage derives the month, day of the week, and time slot index based on the sparse loop counter data. The second stage predicts the future traffic states based on the sparse loop counter data, the road graph attributes, and the predictions from the first stage. The advantage of using a two-stage framework is that it explicitly exploits the strong seasonality and time trends observed in traffic flows; see Figure 1. As such, the second stage model can better capture the time patterns related to traffic congestion and expected time of arrival.

2.1 First Stage

In the first stage, we use the volume counts for the nodes in the entire road graph as features. We predict the month, day of the week, and time slot index based on the sparse loop counter data. Models based on gradient boosting decision trees are particularly suitable for the first stage task since the loop counter data is sparse and there are many missing values. Tree-based methods can handle missing values by using surrogate splits, thus eliminating the need of imputation. In addition, decision trees are generally more robust to outliers.

2

Figure 2: The relation between average volume and the time slot index for a sample loop counter in London in the Traffic4cast 2022 training data.
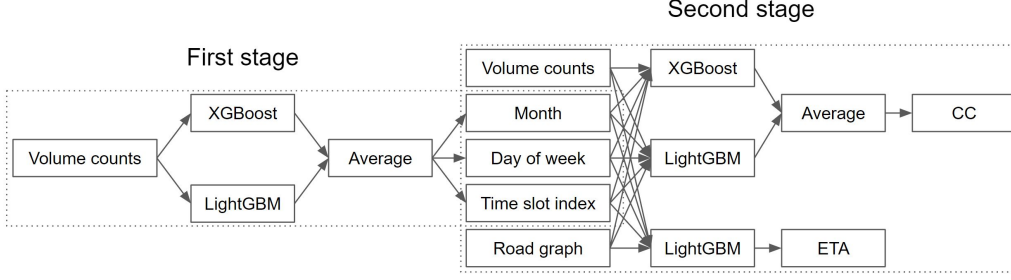
We use three separate models for the first stage, as gradient boosting decision trees do not directly support learning multiple targets. All three prediction tasks are modelled as a regression problem, where we optimize the model using L2 loss. The final prediction for the first stage is based on an ensemble of XGBoost and LightGBM models trained separately for each city. We average the predictions from XGBoost and LightGBM models and round the averaged prediction to the nearest integer.

2.2 Second Stage

In the second stage, we apply different models for each challenge since the applicability of gradient boosting decision trees in multi-task learning is limited as opposed to neural networks [7, 8, 36, 32]. At the core of the second stage is extensive feature engineering primarily based on target enconding. Target encoding (TE) calculates the conditional probabilities of the targets given sets of categorical features, which has been demonstrated to be effective in a wide range of machine learning tasks [34, 49, 50, 32, 44, 14, 33, 31, 30, 37, 26].

2.2.1 Core Challenge

For the core challenge, we engineered a number of features capturing the road network characteristics and traffic dynamics. The list of features we have used in the core challenge and their importance scores are presented in Table 1. We can see that the target encodings (TE) of different congestion classes (cc) are among the most important features. For the core challenge, target encoding refers to the fraction of each congestion class in the training set. We apply Bayesian smoothing to reduce overfitting, where the empirical means are the fraction of each congestion class for all road graph edges in the training set; see Equation 1.

$$TE_{cc}([Categories]) = \frac{Count([Categories]) * mean_{cc}([Categories]) + w * mean_{cc}}{Count([Categories]) + w} \quad (1)$$

In Equation 1, $Count([Categories]$ refers to the number of observations for $Categories$, $mean_{cc}$ is the empirical mean for the congestion class cc, and the pseudocount $w > 0$ is a smoothing parameter which we set to 20.

We train XGBoost and LightGBM using masked cross-entropy loss on congestion classes, which is the same as metric used in the challenge:

$$\ell(\hat{y}, y) = \sum_{n=1}^{N} \frac{1}{\Sigma_{n=1}^{N} w_{y_n} \cdot \mathbb{1}\{y_n \neq \text{ignore\_index}\}} l_n, \quad (2)$$

where

$$l_n = -w_{y_n} \log \frac{\hat{y}_{n,y_n} + \varepsilon}{\Sigma_{c=0}^{C-1} \hat{y}_{n,c} + \varepsilon} \cdot \mathbb{1}\{y_n \neq \text{ignore\_index}\} \quad (3)$$

3

Table 1: The list of features and their importance scores (core challenge).

#	Description	Score
1	TE of cc green given the time slot index and whether the date is a weekend	674.99
2	TE of cc red given the time slot index	432.00
3	TE of cc green given the time slot index	301.92
4	The importance of the highway within the road network	181.96
5	TE of cc red given the time slot index and whether the date is a weekend	166.05
6	TE of cc yellow given the time slot index and whether the date is a weekend	135.27
7	TE of cc red given whether the date is a weekend	134.05
8	Whether the date is a weekend	118.69
9	TE of cc yellow given the time slot index	114.36
10	TE of cc yellow given whether the date is a weekend	64.03
11	The time slot index	55.28
12	Month	52.22
13	TE of cc green given the time slot index and day of the week	46.05
14	TE of cc green given day of the week	45.41
15	TE of cc red given day of the week	41.98
16	Day of the week	41.39
17	Whether the road graph edge can only be used in one direction by vehicles	39.22
18	Numerical mapping of the OSM highway class	34.11
19	TE of cc red given the time slot index and day of the week	33.58
20	TE of cc red	31.34
21	TE of cc green given whether the date is a weekend	29.81
22	Whether the road graph edge runs in a tunnel	27.98
23	TE of cc yellow given day of the week	24.75
24	Edge speed (km per hour)	21.54
25	Maximum legal speed limit	20.31
26	TE of cc yellow	19.80
27	TE of cc green	18.83
28	Number of node hops to get to the closes vehicle counter in the graph	17.44
29	Number of traffic lanes on the road graph edge	14.22
30	Edge length in meters	13.69
31	Unique identifier of the source node	12.98
32	Unique identifier of the sink node	12.82
33	Unique identifier of the road graph edge	12.27
34	Number of incoming edges for the sink node	12.04
35	Number of outgoing edges for the source node	11.97
36	Number of incoming edges for the source node	11.78
37	Number of outgoing edges for the sink node	11.77
38	TE of cc yellow given the time slot index and day of the week	10.74
39	The volume count for the target node 15 minutes in the past	8.32
40	The volume count for the source node 15 minutes in the past	8.24
41	The volume count for the target node 45 minutes in the past	8.06
42	The volume count for the target node 60 minutes in the past	8.06
43	The volume count for the source node 60 minutes in the past	8.02
44	The volume count for the source node 45 minutes in the past	7.90
45	The volume count for the target node 30 minutes in the past	7.58
46	The volume count for the source node 30 minutes in the past	7.55

In Equation 2 and Equation 3, $\hat{y} \in \mathbb{R}^{N \times C}$ is the predicted probabilities for each congestion class, $y \in \{0, ..., C-1\}^N$ is the target, and $w_c = \frac{N}{|C| \cdot \Sigma_{i=1}^N \mathbb{1}\{y_i=c\}}$ refers to the class weight that is obtained by calculating the empirical mean for each congestion class. $C \in \mathbb{N}$ is the number of congestion classes, $N \in \mathbb{N}$ is the number of samples, $\varepsilon \in \mathbb{R}^+$ is a small constant to prevent overflow, and ignore_index $\in C \cup \{\bot\}$ specifies whether a target value is ignored or not.

The final prediction for the core challenge is made by an ensemble of XGBoost and LightGBM models, trained separately for each city.

Table 2: The list of features and their importance scores (extended challenge).

#	Description	Score
1	Smoothed TE given the time slot index and day of the week	228374728.6
2	Unique identifier of the supersegment	76275475.3
3	Smoothed TE given the time slot index	44789850.5
4	The time slot index	7016971.0
5	TE given the time slot index and day of the week	5319797.0
6	Month	3703889.9
7	TE given day of the week	3033482.3
8	TE given the time slot index	2704734.8
9	Number of nodes in the supersegment	2693478.1
10	TE for the supersegment	1884798.9
11	Day of the week	1871912.3
12	TE given whether the date is a weekend	1080182.0
13	Whether the date is a weekend	935208.7
14	Smoothed TE given the time slot index and whether the date is a weekend	804695.4
15	TE given the time slot index and whether the date is a weekend	510518.6

2.2.2 Extended Challenge

For the extended challenge, we only use LightGBM since XGBoost does not support the direct optimization of the L1 loss (as of v1.6.2). Unlike conventional gradient boosting methods that work as gradient descent in function space, XGBoost works as Newton-Raphson in function space, and uses a second order Taylor approximation in the loss function to connect to Newton Raphson method [11]. XGBoost therefore requires smooth objectives to compute the first- and second-order derivative statistics of the loss. On the other hand, L1 loss, also known as mean absolute error (MAE), is not continuously twice differentiable. Thus the direct optimization of the MAE metric is not possible in XGBoost. We experimented with approximating the L1 loss with Huber loss in XGBoost, but could not achieve good performance.

The list of features we have used in the extended challenge and their importance scores are presented in Table 2. For the extended challenge, target encoding (TE) refers to the average Expected Times of Arrival (ETA) for each supersegment:

$$TE_{eta}([Categories]) = \frac{1}{N} \sum_{i=1}^{N} ETA_i([Categories]) \tag{4}$$

We also use smoothed TE, where we use a weighted average of ETAs in the nearby time window to reduce overfitting:

$$Smoothed\_TE_{eta}(t) = \frac{TE_{eta}(t) + \sum_{i=1}^{4} [TE_{eta}(t-i) + TE_{eta}(t+i)] * (i+1)^4}{1 + \sum_{i=1}^{4} (i+1)^4 * 2} \tag{5}$$

The intuition behind smoothed TE is that, the ETAs should be similar within a short time window for any super-segment. Smoothing the target encoding of the ETAs can help prevent overfitting due to outliers. The feature importance scores in Table 2 also demonstrate that smoothed TE is more effective than the non-smoothed counterpart.

For the extended challenge, the task is to predict the ETAs along super-segments. The dynamic speed data obtained from GPS probes is used to derive travel times on the edges of the graph, which are then summed up to derive super-segment ETAs. We experimented with predicting the travel times on the edges of the graph, and then manually summing up the predicted travel times for all edges in the super-segment to estimate super-segment ETA. This results in degraded performance, presumably because using super-segments helps make the ETAs derived from the underlying speed data more robust to data outliers.

Table 3: Leaderboard results for the core challenge and the extended challenge.

Core challenge			Extended challenge		
Rank	Team	Score	Rank	Team	Score
1	ustc-gobbler	0.84310793876648	1	ustc-gobbler	58.4997215271
2	Bolt	0.84966790676117	2	TSE	59.782447814941
3	**oahciy (ours)**	**0.85041532913844**	**3**	**oahciy (ours)**	**61.22274017334**
4	GongLab	0.85603092114131	4	Bolt	61.254610697428
5	AP_DE	0.87350843350093	5	discovery	62.296744028727

2.3 Implementation Details

For each city, we randomly select two weeks as the validation set to mimic the distribution of the test set. During training, we use the leave-one-out strategy for target encoding, where the conditional probabilities of the targets are calculated ignoring the targets in the same day to prevent target leakage. During inference, we use all available target values for target encoding.

We use the same hyperparameters in both stages. For XGBoost, we set *max_depth* to be 5, *eta* (learning rate) to be 0.01, which limits the growth of the trees during training. We set *subsample* to be 0.5, *colsample_bytree* to be 0.9, and *colsample_bylevel* to be 0.9 to reduce the risk of overfitting. In addition, *tree_method* is set to be *gpu_hist* to use GPU acceleration. The rest of the hyperparameters are set to the default values for XGBoost. For LightGBM, we set *learning_rate* to be 0.1 and use the default values for the rest of the hyperparameters.

For both XGBoost and LightGBM, we train until the validation score does not improve for 1,000 rounds. This helps estimate the number of rounds that XGBoost and LightGBM require to achieve optimal performance on the held-out set. After that, we retrain the models using both the training set and the validation set for the estimated number of rounds to achieve the optimal performance.

3 Experiments

The final leaderboard results on the core and the extended challenges are presented in Table 3. Our two-stage gradient boosting framework achieves third place in both challenges, demonstrating its excellent accuracy, generalizability, and transferability.

We further examine the performance of our two-stage gradient boosting framework by conducting a comparative analysis against three alternative methods, namely the multilayer perceptron (MLP), graph neural network (GNN), and the single-stage alternative. We report the leaderboard performance and the training time for each approach in Table 4. We observe that approaches based on gradient boosting decision trees are much more efficient than neural networks based approaches and, at the same time, achieve comparable performance. GNN can outperform MLP while at the cost of much longer training time. Comparing the two-stage framework against the single-stage alternative, we can see that the two-stage framework significantly improves the performance and is only marginally slower than the single-stage model. This is because the first stage model is relatively lightweight and introduces little computational overhead.

To study the effect of the first stage errors on the second stage performance, we comapre the validation performance of the second stage model using the predictions from the first stage model against the same model using the ground truth date and time; see Table 5. We can see that when the ground truth date and time information is given, the second stage model can achieve significantly better performance, since it prevents errors from being propagated from the first stage to the second stage.

4 Discussion

In this challenge, we use the two-stage pipeline because the time of the entries in the test set is not provided as a feature. In a real-world production system, however, the date and the time for the prediction of interest are very easy to obtain. Therefore real-world production systems should directly use the second stage model to avoid the propagation of errors from the first stage to the second stage.

Table 4: Leaderboard results and training time for different approahces.

Core challenge			Extended challenge		
Approach	Score	Time (minutes)	Approach	Score	Time (minutes)
MLP	0.85685	197	MLP	61.39940	172
GNN	0.85204	1174	GNN	61.24305	679
Single-stage	0.85483	**95**	Single-stage	61.31830	**28**
Two-stage	**0.85041**	114	Two-stage	**61.22274**	47

Due to the limit of computational resources, in this challenge, we did not use (graph) neural network-based approaches. Graph neural networks (GNNs) have emerged as a promising short-term traffic state estimation technique due to their ability to model complex spatial dependencies and dynamics [28]. GNNs can capture the non-linear interactions between road segments and effectively handle spatio-temporal data, making them suitable for short-term traffic state estimation tasks [5]. The effectiveness of our engineered features when using XGBoost and LightGBM demonstrates that they successfully capture traffic dynamics and time patterns. To further boost the performance of traffic state estimation, we can try adding these engineered features into a deep learning-based framework and ensemble the prediction from deep learning models with the prediction from the gradient boosting approaches.

In our two-stage gradient boosting framework, the first and second stage models are trained separately. We have observed that the errors made by the first stage models can significantly impact the performance of the second stage models. Meanwhile, the predictions from the second stage models (the congestion class and the expected time of arrival) have a very strong dependence on the predictions from the first stage models (the month, day of the week, and time slot index). It is, therefore, possible to apply co-training strategies [2, 41] or coevolutionary algorithms [43, 18, 35] during the training of the first stage and second stage models, where the predictions from both stages are iteratively refined in a multi-phase fahsion.

5 Related Work

Short-term traffic state estimation is essential in transportation management as it provides valuable insights into traffic flow patterns and enables better decision-making for commuters and transportation agencies. In recent years, there has been an increasing interest in short-term traffic state estimation due to its potential to improve traffic safety, reduce travel time, and minimize environmental impacts [3]. Traditionally, short-term traffic state estimation models have relied on classic statistical learning methods such as autoregressive integrated moving average (ARIMA) [16], wavelet transform [20], and radial basis function network [54]. However, with the success of deep neural networks in capturing complex dependencies and non-linearities in large datasets, researchers have shifted their attention towards utilizing deep neural networks [39, 55, 58, 25, 59, 9, 24, 17].

To model the dynamics of a complex traffic system, a common approach is to represent it as a sequence of movie frames, where each pixel corresponds to the traffic intensity at a certain block of area, and each frame summarizes a discrete time bin, thus casting traffic forecasting as a video prediction task [58, 25, 59, 24, 17]. Early works focused on using convolutional neural networks (CNNs), which are primarily used for visual tasks such as image classification and object detection. To model temporal relationships, a common practice is to concatenate the sequence of frames along the channel dimension and apply 2-dimensional CNNs [57]. Another approach uses 3-dimensional CNNs, where convolution and pooling operations are performed spatio-temporally [60, 1].

Researchers have also explored the potential of recurrent neural networks (RNNs) to capture temporal dependencies [47]. However, vanilla RNNs suffer from gradient vanishing and exploding problems, and popular variants such as the Long Short-Term Memory (LSTM) [19] and the Gated Recurrent Unit (GRU) [12] are often used. State-of-the-art traffic forecasting models have also employed a hybrid of CNN and RNN layers as the underlying architecture, allowing the model to simultaneously exploit the ability of CNN units to model spatial relationships and the potential of RNN units to capture temporal dependencies [45, 46, 29, 58, 59].

Table 5: The effect of the first stage errors on the second stage performance.

	Core challenge		Extended challenge	
	Score	Δ	Score	Δ
Prediction	0.853078	-	61.24775	-
Ground truth	0.842203	1.27%	58.03751	5.24%

While CNNs and RNNs are effective in modeling data with an underlying Euclidean or grid-like structure, they fail to capture the complex graph structures in transportation systems such as the road network [23, 15, 51]. To address this limitation, graph neural networks (GNNs) have recently shown exceptional performance in a variety of traffic flow prediction tasks due to their ability to capture spatial dependencies presented in the form of non-Euclidean graph structures [10, 31, 28, 21, 26]. Additionally, it has been demonstrated that GNNs can effectively learn properties given by the underlying road network, which improves the generalization performance when making predictions on previously unseen cities [40].

Finally, Transformer-based models have emerged as a powerful tool for short-term traffic state estimation [6, 52, 53]. These models are designed to process sequential data, making them a natural fit for time-series forecasting problems [38, 37]. One of the key advantages of Transformer-based models is their ability to model long-range dependencies between time steps. This is achieved through the use of self-attention, which allows the model to focus on relevant parts of the input sequence while ignoring irrelevant information [48].

6   Conclusion

We present an efficient two-stage gradient boosting framework for short-term traffic state estimation. Experimental results on the NeurIPS 2022 Traffic4cast challenge demonstrate that our two-stage gradient boosting approach achieves excellent accuracy, generalizability, transferability, and computation efficiency.

References

[1] K. Bayoudh, F. Hamdaoui, and A. Mtibaa. Transfer learning based hybrid 2d-3d cnn for traffic sign recognition and semantic road detection applied in advanced driver assistance systems. *Applied Intelligence*, 51:124–142, 2021.

[2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[3] A. Boukerche and J. Wang. Machine learning-based traffic prediction models for intelligent transportation systems. *Computer Networks*, 181:107530, 2020.

[4] K.-H. N. Bui, J. Cho, and H. Yi. Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues. *Applied Intelligence*, pages 1–12, 2021.

[5] K.-H. N. Bui, J. Cho, and H. Yi. Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues. *Applied Intelligence*, 52(3):2763–2774, 2022.

[6] L. Cai, K. Janowicz, G. Mai, B. Yan, and R. Zhu. Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS*, 24(3):736–755, 2020.

[7] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Multi-task learning for boosting with application to web search ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1189–1198, 2010.

[8] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Boosted multi-task learning. *Machine learning*, 85:149–173, 2011.

[9] C. Chen, K. Li, S. G. Teo, X. Zou, K. Li, and Z. Zeng. Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(4):1–23, 2020.

[10] C. Chen, K. Li, S. G. Teo, X. Zou, K. Wang, J. Wang, and Z. Zeng. Gated residual recurrent graph neural networks for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 485–492, 2019.

[11] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[13] Z. Cui, K. Henrickson, R. Ke, and Y. Wang. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4883–4894, 2019.

[14] C. Deotte, B. Liu, B. Schifferer, and G. Titericz. Gpu accelerated boosted trees and deep neural networks for better recommender systems. In *Proceedings of the Recommender Systems Challenge 2021*, pages 7–14. 2021.

[15] F. Diehl, T. Brunner, M. T. Le, and A. Knoll. Graph neural networks for modelling traffic participant interaction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 695–701. IEEE, 2019.

[16] H. Dong, L. Jia, X. Sun, C. Li, and Y. Qin. Road traffic flow prediction with a time-oriented arima model. In *2009 Fifth International Joint Conference on INC, IMS and IDC*, pages 1649–1652. IEEE, 2009.

[17] C. Eichenberger, M. Neun, H. Martin, P. Herruzo, M. Spanring, Y. Lu, S. Choi, V. Konyakhin, N. Lukashina, A. Shpilman, et al. Traffic4cast at neurips 2021-temporal and spatial few-shot transfer learning in gridded geo-spatial processes. In *NeurIPS 2021 Competitions and Demonstrations Track*, pages 97–112. PMLR, 2022.

[18] N. García-Pedrajas, C. Hervás-Martínez, and J. Muñoz-Pérez. Covnet: a cooperative coevolutionary model for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 14(3):575–596, 2003.

[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[20] D.-R. Huang, J. Song, D.-C. Wang, J.-Q. Cao, and W. Li. Forecasting model of traffic flow based on arma and wavelet transform. *Jisuanji Gongcheng yu Yingyong(Computer Engineering and Applications)*, 42(36):191–194, 2006.

[21] W. Jiang and J. Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, page 117921, 2022.

[22] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[23] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[24] M. Kopp, D. Kreil, M. Neun, D. Jonietz, H. Martin, P. Herruzo, A. Gruca, A. Soleymani, F. Wu, Y. Liu, et al. Traffic4cast at neurips 2020? yet more on theunreasonable effectiveness of gridded geo-spatial processes. In *NeurIPS 2020 Competition and Demonstration Track*, pages 325–343. PMLR, 2021.

[25] D. P. Kreil, M. K. Kopp, D. Jonietz, M. Neun, A. Gruca, P. Herruzo, H. Martin, A. Soleymani, and S. Hochreiter. The surprising efficiency of framing geo-spatial time series forecasting as a video prediction task–insights from the iarai traffic4cast competition at neurips 2019. In *NeurIPS 2019 Competition and Demonstration Track*, pages 232–241. PMLR, 2020.

[26] M. Krenn, L. Buffoni, B. Coutinho, S. Eppel, J. G. Foster, A. Gritsevskiy, H. Lee, Y. Lu, J. P. Moutinho, N. Sanjabi, et al. Predicting the future of ai with ai: High-quality link prediction in an exponentially growing knowledge network. *arXiv preprint arXiv:2210.00881*, 2022.

[27] L. Li, X. Su, Y. Zhang, Y. Lin, and Z. Li. Trend modeling for traffic time series analysis: An integrated study. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3430–3439, 2015.

[28] M. Li and Z. Zhu. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4189–4196, 2021.

[29] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[30] Y. Lu. Learning to transfer for traffic forecasting via multi-task learning. *arXiv preprint arXiv:2111.15542*, 2021.

[31] Y. Lu. Predicting research trends in artificial intelligence with gradient boosting decision trees and time-aware graph neural networks. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 5809–5814. IEEE, 2021.

[32] Y. Lu, C. Chang, H. Rai, G. Yu, and M. Volkovs. Learning effective visual relationship detector on 1 gpu. *arXiv preprint arXiv:1912.06185*, 2019.

[33] Y. Lu, C. Chang, H. Rai, G. Yu, and M. Volkovs. Multi-view scene graph generation in videos. In *International Challenge on Activity Recognition (ActivityNet) CVPR 2021 Workshop*, volume 3, page 2, 2021.

[34] Y. Lu, R. Dong, and B. Smyth. Context-aware sentiment detection from ratings. In *Research and Development in Intelligent Systems XXXIII: Incorporating Applications and Innovations in Intelligent Systems XXIV 33*, pages 87–101. Springer, 2016.

[35] Y. Lu, R. Dong, and B. Smyth. Coevolutionary recommendation model: Mutual learning between ratings and reviews. In *Proceedings of the 2018 World Wide Web Conference*, pages 773–782, 2018.

[36] Y. Lu, R. Dong, and B. Smyth. Why i like it: multi-task learning for recommendation and explanation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 4–12, 2018.

[37] Y. Lu, Z. Gao, Z. Cheng, J. Sun, B. Brown, G. Yu, A. Wong, F. Pérez, and M. Volkovs. Session-based recommendation with transformers. In *Proceedings of the Recommender Systems Challenge 2022*, pages 29–33. 2022.

[38] Y. Lu, H. Rai, J. Chang, B. Knyazev, G. Yu, S. Shekhar, G. W. Taylor, and M. Volkovs. Context-aware scene graph generation with seq2seq transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15931–15941, 2021.

[39] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2014.

[40] H. Martin, D. Bucher, Y. Hong, R. Buffat, C. Rupprecht, and M. Raubal. Graph-resnets for short-term traffic forecasts in almost unknown cities. In *NeurIPS 2019 Competition and Demonstration Track*, pages 153–163. PMLR, 2020.

[41] X. Ning, X. Wang, S. Xu, W. Cai, L. Zhang, L. Yu, and W. Li. A review of research on co-training. *Concurrency and computation: practice and experience*, page e6276, 2021.

[42] N. G. Polson and V. O. Sokolov. Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79:1–17, 2017.

[43] M. A. Potter and K. A. De Jong. Evolving neural networks with collaborative species. In *Summer Computer Simulation Conference*, pages 340–345. SOCIETY FOR COMPUTER SIMULATION, ETC, 1995.

[44] B. Schifferer, G. Titericz, C. Deotte, C. Henkel, K. Onodera, J. Liu, B. Tunguz, E. Oldridge, G. De Souza Pereira Moreira, and A. Erdem. Gpu accelerated feature engineering and training for recommender systems. In *Proceedings of the Recommender Systems Challenge 2020*, pages 16–23. 2020.

[45] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

[46] X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. *Advances in neural information processing systems*, 30, 2017.

[47] C. Ulbricht. Multi-recurrent networks for traffic forecasting. In *AAAI*, pages 883–888, 1994.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[49] M. Volkovs, H. Rai, Z. Cheng, G. Wu, Y. Lu, and S. Sanner. Two-stage model for automatic playlist continuation at scale. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 1–6. 2018.

[50] M. Volkovs, A. Wong, Z. Cheng, F. Pérez, I. Stanevich, and Y. Lu. Robust contextual models for in-session personalization. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*, pages 1–5, 2019.

[51] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of the web conference 2020*, pages 1082–1092, 2020.

[52] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, and H. Xiong. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*, 2020.

[53] H. Yan, X. Ma, and Z. Pu. Learning dynamic and hierarchical traffic spatiotemporal features with transformer. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):22386–22399, 2021.

[54] W. Yang, D. Yang, Y. Zhao, and J. Gong. Traffic flow prediction based on wavelet transform and radial basis function network. In *2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM)*, volume 2, pages 969–972. IEEE, 2010.

[55] H. Yi, H. Jung, and S. Bae. Deep neural networks for traffic flow prediction. In *2017 IEEE international conference on big data and smart computing (BigComp)*, pages 328–331. IEEE, 2017.

[56] Y. Yin and P. Shang. Forecasting traffic time series with multivariate predicting method. *Applied Mathematics and Computation*, 291:266–278, 2016.

[57] B. Yu, H. Yin, and Z. Zhu. St-unet: A spatio-temporal u-network for graph-structured time series modeling. *arXiv preprint arXiv:1903.05631*, 2019.

[58] W. Yu, Y. Lu, S. Easterbrook, and S. Fidler. Crevnet: Conditionally reversible video prediction. *arXiv preprint arXiv:1910.11577*, 2019.

[59] W. Yu, Y. Lu, S. Easterbrook, and S. Fidler. Efficient and information-preserving future frame prediction and beyond. In *International Conference on Learning Representations*, 2020.

[60] S. Zhang, L. Zhou, X. Chen, L. Zhang, L. Li, and M. Li. Network-wide traffic speed forecasting: 3d convolutional neural network with ensemble empirical mode decomposition. *Computer-Aided Civil and Infrastructure Engineering*, 35(10):1132–1147, 2020.