RL-CNN: Reinforcement Learning-designed Convolutional Neural Network for Urban Traffic Flow Estimation

Bachelorarbeit
der Philosophisch-naturwissenschaftlichen
Fakultät der Universität Bern

Vorgelegt von

Alessandro Esposito
HS 20

Leiter der Arbeit

Professor Dr. Torsten Braun
Institut für Informatik und angewandte Mathematik

Abstract

Increasing availability of mobility data and technical advancements that leverage communication speed and computation efficiency enable the use of machine learning based methods for solving complex challenges regarding Intelligent Transportation Systems (ITS). This work employs Convolutional Neural Networks (CNNs) to estimate future urban traffic flows. CNNs are a popular form of deep learning networks that are able to identify and extract complex spatio-temporal data characteristics. Nevertheless, it is still an effortful task to find a highly-accurate CNN architecture due to the large set of available hyper-parameters. Hence, this thesis proposes RL-CNN, a framework to autonomously generate high-accuracy urban traffic flow predictors with no human intervention. The proposed framework employs Reinforcement Learning (RL) to autonomously search for optimized CNN architectures. These architectures are then used as highly-accurate urban traffic flow predictors. The proposed RL-CNN framework was trained and tested on a real-world, large-scale dataset, collected in the city of Porto, Portugal. The results of this thesis show that RL-CNN outperforms the compared state-of-the-art urban traffic flow predictors by **5% - 10%.** Additionally, the convergence time is improved by **20% - 33%** compared to another RL based approach tested in this paper.

RL-CNN: Reinforcement Learning-designed Convolutional Neural Network for Urban Traffic Flow Estimation

Mostafa Karimzadeh[1], Alessandro Esposito[1], Zhongliang Zhao[1 2], Torsten Braun[1], Susana Sargento[3]
Institute of Computer Science, University of Bern, Switzerland[1]
School of Electronic and Information Engineering, Beihang University, China[2]
Institute de Telecomunicações - Aveiro, Portugal[3]
Email : {mostafa.karimzadeh, torsten.braun}@inf.unibe.ch, alessandro.esposito@students.unibe.ch, zhaozl@buaa.edu.cn, susana@ua.pt

***Abstract*—Accurate prediction of urban traffic flows brings enormous advantages to big cities. Therefore, many urban traffic flow predictors have been developed in recent years. Urban traffic flow predictors aim to identify complex mobility patterns and capture urban traffic flow characteristics from large-scale historical datasets. Afterward, trained models are used to predict the future traffic volume in terms of the number of moving objects (e.g., vehicles). Convolutional Neural Networks (CNN) and other deep learning approaches are brilliant choices because of their ability to learn Spatio-temporal dependencies. Nevertheless, the extensive set of hyper-parameters tends to make these neural networks overly complex and challenging to design. In this work, we introduce RL-CNN, a framework based on Reinforcement Learning whose objective is to autonomously discover high-performance CNN architectures for the given traffic prediction task without human intervention. We examine the proposed RL-CNN model as a traffic flow estimator on a real-world and large-scale vehicular network dataset. We observe improvements of 5% - 10% in the average traffic flow prediction accuracy over the state-of-art approaches.**

***Index Terms*—Convolutional Neural Networks, Reinforcement Learning, Urban Traffic Estimation.**

I. Introduction

Comprehensive urban traffic information can benefit urban citizens' daily life and improve urban transportation efficiency. Accurate predictions of such traffic information are of great importance for route planning, navigation, and other mobility services. Urban traffic prediction generally applies traffic models to analyze various historical and real-time traffic data to predict traffic conditions in terms of the number of moving objects (e.g., vehicles) in the future. Thanks to the popularity of ubiquitous sensing and Intelligent Transportation Systems (ITS) in recent years, we can gather unprecedented mobility data by exploiting various mobile devices (e.g., smartphones and on-board GPS devices). Such emerging big data availability makes accurate traffic predictions viable.

To leverage such benefits, conventional machine learning models (e.g., SVM, Random Forest, etc.) attempt to predict urban traffic flows. However, traffic flows in large cities can be influenced by various factors in practice, for example, transport regulations, weather conditions, etc. The classical machine learning models have shown to be inadequate to describe traffic characteristics comprehensively and can not achieve accurate predictions [1].

Recently, Convolutional Neural networks (CNN) and other deep learning-based approaches show outstanding ability to learn and predict future states of urban traffic. However, the broad set of hyper-parameters makes defining a high-performance architecture for neural networks a complex task [2]. Hyper-parameters are typically the variables that determine the architecture of neural networks (e.g., number of hidden layers, number of units in each hidden layer, etc.).

This work proposes a framework to predict urban traffic flows in large cities based on historical data. We use a recently proposed model to remove human interventions in CNN architecture design [3], Zoph et al. utilized a Reinforcement learning (RL) based method to find the most appropriate neural network architecture for a given dataset. Thus, RL provides a self-learning process based on positively rewarding *high-performance architectures* and penalizing *low-performance ones*. Within this process, the objective is to adapt the architecture description of a neural network (e.g., the number of hidden layers, the number of units and the activation functions in each hidden layer, etc.) towards the most efficient architecture a provided dataset.

Results show that our proposed RL-CNN framework achieves 5% - 10% average prediction accuracy improvements over the state-of-art approaches tested in this paper.

We organized the other sections of this work as follows. The related work and existing researches for urban traffic flow prediction are reviewed in section II. The problem statement, addressing the challenges of future traffic flow estimation in urban areas, is presented in section III. Section IV describes the proposed traffic flow predictor using Reinforcement Learning and CNN. The results and the evaluation methodology are presented in section V. Finally, section VI concludes the paper.

II. Related Work

In recent years neural network-based predictors and statistical models are introduced as two efficient approaches for

predicting urban traffic flows. For instance, as a statistical model, the k-nearest neighbors (KNNs) have been proposed to estimate the urban traffic [4]. Besides, Support vector machines (SVMs) based predictors [5], [6], [7] delivered promising results in estimating the future states of urban traffic. Deep learning-based predictors are also employed in the traffic flow prediction problem. The authors in [8] integrated Deep Neural Networks with Markov based algorithm to estimate the number of moving objects on the urban trajectories. Besides, in [9], the authors suggested integrating Artificial Neural Networks (ANNs) with statistical models to estimate traffic flow in urban areas. Due to the poor generalization ability of ANNs caused by their shallow architecture, several efficient deep learning-based predictors were proposed to predict traffic flows in large cities. In [10], the authors applied the deep learning-based predictors to estimate the future state of urban traffic in city environments. In [11], Huang et al. benefited from Deep Belief Networks (DBNs) to traffic flow forecasting. Besides, in [12] authors suggested to combine combines deep restricted Boltzmann machines (RBMs) with a Recurrent Neural Network (RNN, which inherited the advantages of the both predictors. The work in [13] proposed to use a Stack Auto-Encoder (SAE) to capture Spatio-temporal features of urban traffic; the extracted features are fed to a deep learning-based predictor to estimate urban traffic. In [14] authors presented promising results by applying CNNs to estimate urban traffic flows. In general, neural networks-based predictors delivered better results compared with traditional statistical models because they have much deeper architecture and are therefore able to learn complex data characteristics. Regardless of the success of the presented deep learning-based traffic flow predictors, developing neural networks' architecture is still an exhausting procedure. In this paper, we automate the development process of a high-performance CNN based traffic flow predictor with no human intervention.

III. Problem Statement

This work addresses the challenges of future traffic flow estimation in urban areas. Estimating future traffic flows is an important part of ITS, city traffic management systems, and general traffic flow optimization, by allocating and de-allocating (optimizing) available resources based on future traffic flow estimation.

Our research focuses on getting highly-accurate CNN-based traffic flow predictors and the automation process for developing them. We define a traffic flow as the aggregated number of moving objects (e.g., vehicles, etc.) that passed a base station (e.g., RSU - Road Side Unit, etc.) in a defined time period. Time period T is defined as the tuple of two date-time values $T = [t_1, t_2]$. The difference between two date-time values is defined as granularity $G = t_2 - t_1$ and is measured in minutes. Therefore, based on the provided data, we attempt to estimate the traffic flow within a time period of granularity $G = 30minutes$ and over the whole day. The number of moving objects N passing RSU R during time period $T \in P$ can therefore be noted as $R_T = N$. Thus, we further defined the traffic flow at time T over the whole city as $C_T = [R_T^1, ..., R_T^k]$, where k is the total amount of RSUs in the city. Figure 1 displays the traffic flow density in the city of Porto, Portugal, which is, in fact, the intensity of traffic congestion in a chosen time period [15]. The plot illustrates that, in the city center of Porto, Portugal, the main roads and the bridges have much higher traffic flow density than the farther parts of the city and, therefore, indicate that these areas are more congested than others, noticeable by the orange and red coloring of the areas, respectively. The blue and light green areas indicate less traffic flow density and less traffic congestion, although also covering areas in the city center. Hence, this work aims to create highly-accurate traffic flow density predictors so that traffic density can be estimated, and consequently, traffic congestion over the whole city can be reduced.

Furthermore, developing the architectures for deep learning models such as CNN is a tedious and challenging process: many hyper-parameters have to be fine-tuned, and CNN structures have increased depth and complexity, resulting in a broad set of possible CNN architectures of almost infinite size. Therefore, we attempt to automate human CNN development processes using RL-based methods, anticipating faster developed, and highly-accurate CNN predictors. This paper focused on an RL-based approach for developing optimized CNN architectures without human intervention. Afterward, optimized CNN acts as a highly-accurate predictor that attempts to forecast future traffic flows regarding the number of aggregated moving objects that passed the RSU during a defined time period.

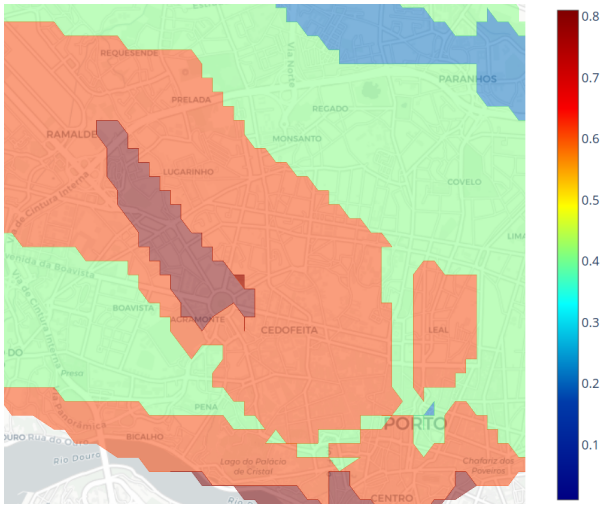


Fig. 1: Traffic flow density in the city of Porto, Portugal

IV. Urban Traffic Flow Prediction

In this section, we introduce a Reinforcement Learning based framework called RL-CNN. The objective of RL-CNN is to autonomously choose a high-performance CNN predictor for a given learning task. Our learning task focuses on finding high-quality urban traffic flow predictors. We use the Neural

Architecture Search (NAS) framework as the primary search method in this work. [3]. In this framework, the RL-based controller creates architectures for an RNN. The architecture of a neural network generally describes the number of hidden layers and their composition and connectivity. In this framework, the learning algorithm is trained to make predictions on a validation dataset. The RL-based controller is updated with the algorithm's outputs to generate improved architectures in the future. Our proposed RL-CNN framework consists of three essential units: (i) CNN as a child predictor whose objective is to reach an acceptable traffic flow prediction accuracy; (ii) the search space that the controller attempts to explore to define high-performance CNNs; and (iii) the controller based on Q-learning to propose improved child CNN architectures so that the expected prediction results are maximized. The following subsections and Figure 2, representing the system model, are explaining the details of each unit and how they are integrated to predict traffic flow in urban environments.

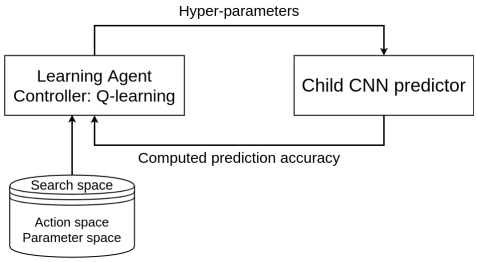


Fig. 2: RL-CNN System Architecture

A. Deep Learning and Convolutional Neural Networks

Although various forms of deep learning models share a common architecture, which contains an input layer, an output layer, and from one to more than a thousand hidden layers in between, raw data initializes the input layer's values, while the output layer emits the desired predictions. All hidden layers are responsible for transforming states of the input layer into the output layer's expected predictions by capturing the high-level abstraction of the input data. Each layer in the network contains units, and the number of units can vary among different layers. Links exist between units of any two neighboring layers, and each link is associated with a weight.

The CNN model is a popular deep learning algorithm composed of an input and an output layer and multiple hidden layers, which can be the convolutional, pooling, or fully connected layers. The convolutional layer adopts filters that apply certain transformations on the input data to capture their properties. Hence, convoluted data in the form of a feature map of the transformed input data is output by convolutional layers. Besides, convolutional operations preserve the spatial relationship in the dataset using small portions of neighboring input data. Convolutional layers are often followed by pooling layers that employ $MaxPooling$ or $MinPooling$ filters to learn more abstract representations of the data. $MaxPooling$ filters apply maximization functions, and $MinPooling$ filters apply minimization functions while running over convoluted data. Thus, pooling layers simplify data by summarizing regions, therefore downsizing convoluted data and reducing the dataset's complexity. Finally, the last layer is the fully connected layer, fed with the output of the last convolutional or pooling layer to complete the prediction task.

B. Search Space

To have high accurate CNN as a traffic flow predictor, the learning agent (controller in Section IV-C) aims to explore the defined search space, which includes:

- Parameter space is defined as a set of all relevant parameters that the learning agent can choose to suggest a description of architecture to child CNN. We define the depth of CNN model as the maximum number of layers (convolutional, pooling and flatten layers) as an integer number selected from $(3, 100]$. For each convolutional layer, the size of filters is selected from $\{5, 100, 150, 200, 250\}$ and the size of pooling layer is chosen from $\{5, 10, 15, 20, 25\}$;
- Action space forces the controller to take certain actions. For instance, $MaxPooling$ layers often deliver better results in comparison with $MinPooling$ layers [16]. Therefore, we constrain our learning agent to always add a $MaxPooling$ layer after each convolutional layer. Also, we allow the controller to terminate the architecture search process if the child CNN delivers an acceptable traffic flow prediction accuracy (e.g., 90%).

Additionally, the learning agent is forced to use Softmax [17] and Rectified Linear Unit (ReLU) [18] as activation functions on output layers and hidden convolutional layers, respectively. ReLU's ability to increase non-linearities while also decreasing training time makes it particularly useful when stacking hidden layers (e.g., using deep neural networks). In contrast, the Softmax activation function is primarily used in the output layer to output a vector representing the probability distribution of potential outcomes (e.g., list of future traffic flow in terms of moving objects, etc.). Note that the framework itself is not restricting the search space. We purposely defined it for faster convergence. Otherwise, the architecture search process will terminate when the learning agent can reach the convergence level (e.g., reaching a highly accurate child CNN architecture).

C. CNN Architecture Search Strategy

Different approaches have been developed to explore the search space to define neural network architectures. These approaches include random search, Bayesian Optimization, RL, and other gradient-based methods used in works of Liashchynskyi et al. [19], Wu et al. [20], and Baker et al. [21], respectively. This research focuses on RL-based NAS, where a Q-learning-based controller generates a neural network architecture, trains the generated neural architecture on training data, and computes an accuracy metric on validation data. The controller receives the prediction accuracy in the form of a reward. Positive reward signals the increase in

the prediction accuracy, and a high positive reward value is equivalent to a high accuracy increase. Therefore, a negative reward indicates a decrease in the prediction accuracy, and large negative reward values signal a poor prediction accuracy.

The agent's ultimate goal is to learn to generate neural network architectures that result in a high validation accuracy by maximizing the controller's reward. We are now summarizing the theoretical notation of Q-learning, as used in our research. For that, we setup a discrete and finite *Search space*, consisting of *Parameter space* S and *Action space* A (see Section IV-B). This leaves the agent with a finite but huge set of possible neural network architectures to choose from, hence the use of RL. At each iteration $t \in \{0, 1, 2, \ldots\}$, the agent in state $s \in S$ will take an action $a \subseteq A(s)$ to pass into the next state s'. At each iteration t, the computed accuracy is given to the agent as a reward signal ($r_t \in \mathbb{R}$), which depends on the transition from state s to the next state s'. The learning agent's goal is to find the most accurate network architecture. To do so, the agent needs to maximize the total accumulated reward over all possible iterations. Next follows the definition of the recursive reward maximization problem. We define $Q'(s, a)$, for any state $s \in S$ and action $a \in A(s)$, as the maximum total expected reward. Furthermore, $Q'(s, a)$ is also called Q-value and $Q'(\cdot)$ is known as the action-value function. We define $r(s, a)$ as the computed reward value for choosing action a in state s and $\mathbb{E}_{s'|s,a}$ as the expected probability of selecting action a in state s to translate into state s'. Therefore, we can note the Bellman's Equation, which is the recursive maximization equation, as follows:

$$Q'(s, a) = \mathbb{E}_{s'|s,a}\left[r(s, a) + \gamma \max_{a' \in \mathcal{A}(s')} Q'(s', a')\right] \quad (1)$$

The learning agent has no *a-priori* knowledge about the consequences of choosing the suggested architectures. The only known information consists of the search space, including all available actions and parameters. Therefore, an iterative approach of the Bellman's Equation is needed (see Equation 2) so that the learning agent can adapt based on the reward for choosing the suggested architecture.

$$Q_{t+1}(s, a) = (1 - \alpha) Q_t(s, a) + \alpha\left[r_t + \gamma \max_{a' \in \mathcal{R}(s')} Q_t(s', a')\right] \quad (2)$$

The priority of newly acquired information over old information is determined by the Q-learning rate $\alpha \in (0, 1]$, whereas the importance of future rewards is determined by the discount factor $\gamma \in (0, 1]$. This research uses the ϵ-greedy exploration/exploitation strategy [22]. Our approach is to take a random action with probability ϵ, where we set the ϵ to decay over time so that the controller learns to explore the search space early and gradually exploits the suggested architectures to the child CNN to achieve the maximum reward.

V. Performance Analysis

We validate the proposed RL-CNN framework to generate a high-performance CNN algorithm as an urban traffic flow estimator by conducting extensive experiments on a real-world dataset. The experiment details are explained in subsection V-A, the metrics are depicted in subsection V-B, and the results in subsection V-C.

A. Experiment Setup

We require mobility trace datasets to train and test the proposed RL-CNN framework properly. For this, a real-world VANET testbed deployed in the city of Porto, Portugal, is used in this work [15]. More than 600 networked vehicles (OBUs) were tracked by the over 70 RSUs (Road-Side Units) distributed all over the city of Porto. One part of the collected large-scale dataset consists of RSUs records (RSU ID, RSU location, traffic volume, etc.), and the other part includes information about OBUs (OBU ID, connection times, connected RSU Ids, etc.). However, to train our RL-CNN and make traffic flow estimations, only RSU IDs, OBU IDs, and timestamps are required. We mainly focused on prediction accuracy as our evaluation metric. Our proposed framework performs the search for a highly accurate network during the exploration phase. For that, we split the dataset into two subsets, of which 70% is used for training the algorithm, and the remaining 30% is used for testing the suggested estimators. During the exploitation phase, which starts after convergence, the discovered CNN estimator is trained for 300 iterations over the whole dataset, the so-called epochs. Too many epochs can lead to performance problems, resulting in longer training times and over-fitting. Because of that, a method named *Early Stopping* is used in this research. The computed accuracy is monitored by Early Stopping, and training is halted if the accuracy change over the patience epochs (the number of epochs to wait before Early Stopping triggers, e.g., $P_{epochs} = 5$) is less than $\Delta_{min} = 0.5$. The other parameters, batch size, an initial learning rate of the CNN, are set to 200 and 0.003, respectively. We want to give a higher weight to rewards in the far future. Therefore, the discount factor (γ) is set to 1, and the Q-learning rate (α) is set to 0.01. We used a High-Performance Computing Cluster at the University of Bern in Switzerland (HPC Cluster - UBELIX [1]) with Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz for training and evaluation of our suggested traffic flow predictors.

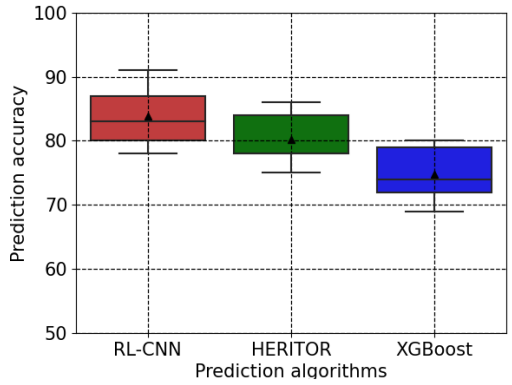
B. Evaluation Metrics

To evaluate the proposed RL-CNN algorithms traffic flow prediction quality, we use the accuracy [3], which measures how often the predicted result is correct compared to the overall number of predictions.

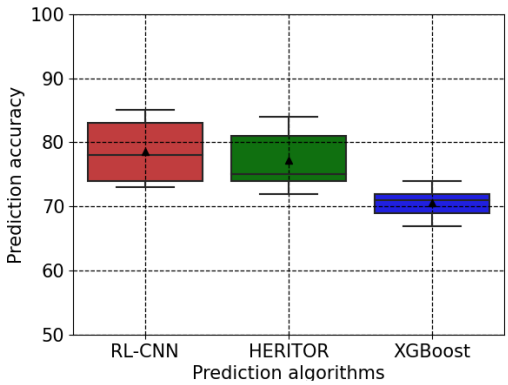
$$Accuracy = \frac{\#predictions_{correct}}{\#predictions_{correct} + \#predictions_{incorrect}} \quad (3)$$

In this work, a correct prediction occurs if the RL-CNN based estimation, optimized CNNs, equals to the effective number of aggregated moving objects; an incorrect prediction occurs if the estimation does not equal the effective number of aggregated moving objects.

[1] https://docs.id.unibe.ch/ubelix

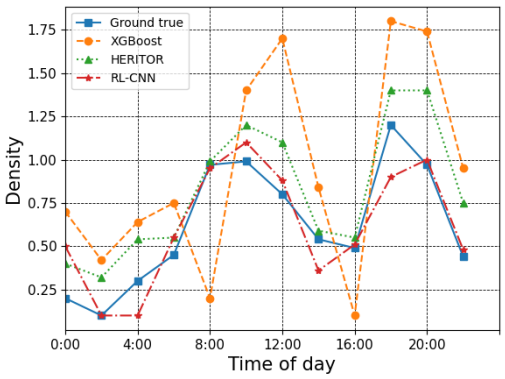


(a) Business days

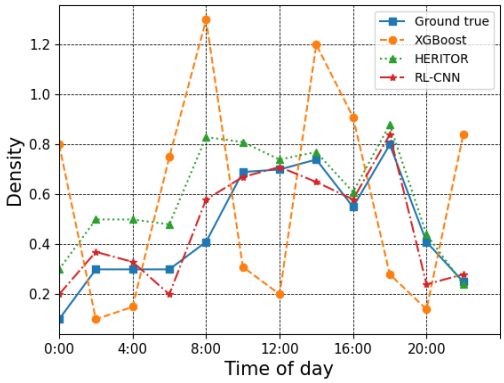


(b) Weekends

Fig. 3: *T*raffic flow prediction results.



(a) Business days



(b) Weekends

Fig. 4: *D*ensity prediction results

C. Evaluation Results

This subsection details the results of the traffic flow prediction accuracy of the proposed RL-CNN model, and the accumulated training time needed for the convergence of finding the RL-CNN based density predictor. We run experiments using the Porto dataset to compare the accuracy of the RL-CNN with HERITOR, proposed in [23], and XGBoost, proposed in [24].

Figure 3 shows the traffic flow prediction accuracy of the RL-CNN, HERITOR, and XGBoost in the business days (Monday to Friday) and weekends (Saturday and Sunday). The results clearly show that the RL-CNN outperforms both HERITOR and XGBoost on both business days and weekends. RL-CNN delivers an average prediction accuracy of 85% and 79% for business days and weekends, respectively. Meanwhile, HERITOR delivers an average traffic flow prediction accuracy of 80% for business days and 77% for weekends. XGBoost is the one with the lowest performance compared to the other prediction algorithms, with an average traffic flow prediction accuracy of 74% for business days and 70% for weekends. The proposed RL-CNN architecture achieves the highest maximum prediction accuracy for both business days and weekends, providing the best prediction accuracy in simple and more complex situations. This is the result of hyper-parameter tuning done autonomously, combined with CNN predictors' excellent ability to capture Spatio-temporal features. Especially on business days, the RL-CNN density predictor achieves a prediction accuracy of 91%. On weekends, the proposed RL-CNN density estimator reached below 80% prediction accuracy, and the RL-CNN density predictor reached a maximum of 85% prediction accuracy. However, this can be due to the quality of input data, which favored business days over weekends (more data on business days).
We compared the urban traffic flow prediction results of RL-CNN, HERITOR, and XGBoost with the ground truth and presented the average density over the city throughout the day in Figure 4. Our results demonstrate the efficiency of RL-CNN. During both business days and weekends, the traffic flow prediction provided by RL-CNN is very much like the real traffic flow. The density estimations of RL-CNN-based predictors (colored in red) are following the ground truth data (colored in blue) closely. Beyond the figure's results, we also measured improvements of $5\% - 10\%$ over the state-of-the-art methods tested in this paper. Therefore, this shows that our proposed RL-CNN framework accurately estimates the traffic flow density and captures the upward and downward traffic

flow movements precisely.

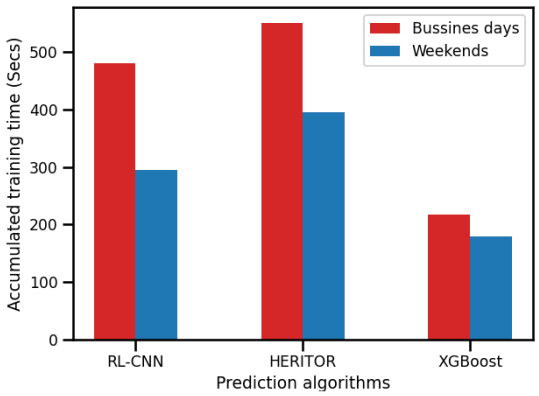


Fig. 5: Convergence times for prediction algorithms

Furthermore, Figure 5 shows the comparison of elapsed accumulated training time until the respective algorithm converges. The two RL based algorithms, the proposed RL-CNN and HERITOR, are outperformed by XGBoost. This is not surprising because XGBoost is designed to be highly efficient and extremely fast. XGBoost does not need to find an optimized neural network architecture before making the density prediction. Once found, the RL-based neural network density predictors are also rapid in estimating future traffic flows. Nevertheless, comparing both RL based approaches, RL-CNN outperforms HERITOR by 20% on business days and 33% on weekends. This makes our proposed RL-CNN not only superior on prediction performance but also in resource usage.

VI. CONCLUSIONS

This paper proposed the RL-CNN framework, which employs reinforcement learning to discover high-accurate CNNs without human interventions. We examine the proposed RL-CNN model as a traffic flow estimator on a real-world and large-scale dataset, namely the Porto dataset. We observe improvements of 5% - 10% in average traffic flow prediction accuracy over the state-of-art while maintaining reasonable and improved convergence time compared to state-of-the-art RL frameworks. The combination of prediction results and convergence time evaluation indicates that our proposed RL-CNN framework can be of great use in assisting traffic flow forecasting tasks. Future work will address this applicability to traffic management platforms.

REFERENCES

[1] L. Yisheng, Y. Duan, W. Kang, and Z. Li, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 865–873, 01 2014.

[2] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Graph convolutional recurrent neural network: Data-driven traffic forecasting," *CoRR*, vol. abs/1707.01926, 2017. [Online]. Available: http://arxiv.org/abs/1707.01926

[3] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, 2016. [Online]. Available: http://arxiv.org/abs/1611.01578

[4] D. Xia, B. Wang, H. Li, Y. Li, and Z. Zhang, "A distributed spatial-temporal weighted model on mapreduce for short-term traffic flow forecasting," *Neurocomput.*, vol. 179, no. C, p. 246–263, Feb. 2016. [Online]. Available: https://doi.org/10.1016/j.neucom.2015.12.013

[5] Z. Ma, G. Luo, and D. Huang, "Short term traffic flow prediction based on on-line sequential extreme learning machine," in *Proceedings of the 8th International Conference on Advanced Computational Intelligence, ICACI 2016*. Institute of Electrical and Electronics Engineers Inc., Apr. 2016, pp. 143–149, 8th International Conference on Advanced Computational Intelligence, ICACI 2016 ; Conference date: 14-02-2016 Through 16-02-2016.

[6] Chun-Hsin Wu, Jan-Ming Ho, and D. T. Lee, "Travel-time prediction with support vector regression," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 276–281, 2004.

[7] W.-C. Hong, "Traffic flow forecasting by seasonal svr with chaotic simulated annealing algorithm," *Neurocomput.*, vol. 74, no. 12–13, p. 2096–2107, Jun. 2011. [Online]. Available: https://doi.org/10.1016/j.neucom.2010.12.032

[8] W. Zheng and D.-H. Lee, "Short-term freeway traffic flow prediction: Bayesian combined neural network approach," *Journal of Transportation Engineering-asce - J TRANSP ENG-ASCE*, vol. 132, 02 2006.

[9] J. Cheng, "A neural network approach to ordinal regression," *CoRR*, vol. abs/0704.1028, 2007. [Online]. Available: http://arxiv.org/abs/0704.1028

[10] N. Polson and V. Sokolov, "Deep learning predictors for traffic flows," 04 2016.

[11] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191–2201, 2014.

[12] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PloS one*, vol. 10, p. e0119044, 03 2015.

[13] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," 09 2016.

[14] ——, "Deep spatio-temporal residual networks for citywide crowd flows prediction," *CoRR*, vol. abs/1610.00081, 2016. [Online]. Available: http://arxiv.org/abs/1610.00081

[15] P. Santos, J. Rodrigues, T. Lourenço, P. D'Orey, Y. Luís, S. Sargento, A. Aguiar, and J. Barros, "Portolivinglab: an iot-based sensing platform for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 523–532, April 2018.

[16] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2017. [Online]. Available: https://arxiv.org/abs/1611.01578

[17] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 11 2018.

[18] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 807–814. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104425

[19] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for nas," 2019.

[20] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimizationb," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26 – 40, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1674862X19300047

[21] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *CoRR*, vol. abs/1611.02167, 2016. [Online]. Available: http://arxiv.org/abs/1611.02167

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, 2015.

[23] M. Karimzadeh, R. D. Aebi, A. M. de Souza, Z. Zhao, T. I. Braun, S. Sargento, and L. Villas, "Reinforcement learning-designed LSTM for trajectory and traffic flow prediction," in *2020 IEEE Global Communications Conference: Selected Areas in Communications: Machine Learning for Communications (Globecom2020 SAC MLC)*, Taipei, Taiwan, Dec. 2020.

[24] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016. [Online]. Available: http://arxiv.org/abs/1603.02754