Predicting Internet Path Dynamics and Performance with Machine Learning

Zhenghui Wang
Shanghai Jiao Tong University
Shanghai, China
felixwzh@sjtu.edu.cn

Hao Wang
Shanghai Jiao Tong University
Shanghai, China
wrystal@sjtu.edu.cn

Yuheng Zhi
Shanghai Jiao Tong University
Shanghai, China
zyh1996@sjtu.edu.cn

Shukai Liu
Shanghai Jiao Tong University
Shanghai, China
83558399@qq.com

ABSTRACT

We study the problem of predicting Internet path dynamics and performance with `traceroute` measurement and machine learning models. In this work, we first point out the drawbacks of the data processing in one previous work [12], and reprocess the data in several more rational ways. Then data analysis and experiments are conducted on the reprocessed data with random forest model before evident temporal locality is discovered in the behavior of the paths. Inspired by this property, we further leverage Long Short-Term Memory (LSTM) model, one kind of recurrent neural networks (RNN), and get promising results. We make our reprocessed data and experiment code publicly available[1].

ACM Reference Format:
Zhenghui Wang, Hao Wang, Yuheng Zhi, and Shukai Liu. 2017. Predicting Internet Path Dynamics and Performance with Machine Learning. In *Proceedings of SJTU Computer Network Workshop (CNW).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Internet paths change frequently due to inter/intra-domain routing changes, load balancing, and even misconfigurations and failures[7]. Some of these changes can seriously disrupt performance, causing longer round-trip times, congestion, or even loss of connectivity[4]. Thus, it is of great significant to predict Internet path dynamics and performance.

In the original paper[12], the authors focus on the problem of predicting Internet path changes and path performance using `traceroute` measurements. They use the recent route information of paths (route age for paths, route changes in the past, average RTT, etc) to do some predictions of paths. Their are three predict targets: (i) the remaining life time of a route (i.e., the time before a route changes), (ii) the number of route changes in a future timeslot, and (iii) the average RTT that the path P will experience in the next `traceroute` measurement.

To achieve these goals, they introduce a $NETPerfTrace$ system, which relies on a standard random forest model for prediction. Moreover, they use extensive evaluation on the impact of different input features by studying the correlations between the inputs and the prediction targets, as well as target selection techniques.

The dataset the authors provide with is a full week of Paris-`traceroute` measurements performed through the M-Lab open Internet measurement initiative[2]. They observe more than 450,000 different paths sampled through Paris-`traceroute` measurements from more than 180 geo-distributed servers. However, most of the paths are not periodically sampled during this week. And only 2,346 paths have at least 100 `traceroute` measurements during the analyzed week. So the dataset contains corresponding information of these 2,346 paths.

We find some drawbacks in their data process.(see section 3) And in our own work, firstly we reprocess the provided data in more reasonable ways and use the same method (random forest model) mentioned by the original paper[12] to compare the results. Secondly, another machine learning method, LSTM model, is applied to get better performance.

2 RELATED WORKS

In the original paper[12], the problem they solve is to use random forest model to predict three targets for one path: (i) the remaining life time of a route for one path at time t (i.e., the time before a route changes), namely $R_r(t)$, (ii) the number of route changes for one path P in a future timeslot T, namely $rcP_T(t)$ and (iii) the average RTT that the path P will experience in the next `traceroute` measurement, namely $avgRTT_P(t)$.

And to achieve these three targets, 69 input features are used to describe the statistical properties of route dynamics and path latency.

For the first route dynamic target, the first group of 11 features,referred to as F_A, is chosen. It describes the statistical properties (average, minimum, maximum, and percentiles) of the route duration observed for each path.

[1] https://github.com/felixwzh/CN-group-project

[2] https://www.measurementlab.net/

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
CNW, December 2017, Shanghai, China
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

And for the second target, the second group of 14 features, as F_B, are relevant to the prediction of number of route changes. F_B features take into account the statistical properties of route changes. In addition to that, F_B contains information about the number of route changes observed for a path so far and a binary feature indicating whether a route change occurred for a path in the current time slot.

And the last group of 44 features ,referred to as F_C describes statistical properties of path latency, which is relavent to the prediction of the next `traceroute` measurement. They calculate the statistical properties of four RTT metrics (average, minimum, maximum and standard deviation) reported from each `traceroute` measurement.

Then they study the correlation among the input features and the targets and apply feature selection techniques to select the best features for prediction. When using as input the full set of 69 input features $F_A \cup F_B \cup F_C$, and perform wrapper-based feature selection on top of this full set. The result shows that the top important features for three targets are not necessarily the ones in corresponding F_A, F_B or F_C.

And the authors use three different ways of selecting input features: (i) use all of the 69 features to predict each target, (ii) when predicting target X, use corresponding feature set F_X, (iii) after wrapper-based feature selection, use the top features for each target. And finally, what they get is the third way of choosing input features can achieve better results (though there is only minor difference in output results).

Other Applications of MLCN. The last 2 decades has seen emerging applications of machine learning techniques used in computer networks [5]. Classic methods, such as SVM, RBM, PCA and LDA, and deep learning methods including DBN, DNN and RNN, are both applied to areas like anomaly detection, web security, traffic prediction, etc. A usual procedure is to use autoencoders, both deep and classic ones, for dimensionality reduction and depend on SVM and DBN as a classifier.

Here we mention a few representative works. Alom et al. [1] focus on intrusion detection system. Tao et al. [10] search to build a data fusion algorithm with awareness of network security situation obtained by PCA, LDA and Fisher score. Lv et al. [6] first develop deep learning methods in traffic flow prediction. Hou et al. [3] aims to learn filtering rules for malicious contents automatically. Thottan et al. [11] are among the first to introduce self-learning methods to anomaly detection and a series of botnet detection methods follows up [2, 8, 9, 13].

Random Forest. Random Forest is a classical model in machine learning. It basically integrates a bunch of decision trees trained on different parts of the dataset, thus eliminating the risk of overfitting when a single tree is trained. A Random Forest makes predictions by averaging that of all the decision trees in it or by taking the majority vote in classification scenario. In addition, recent years have seen the development of Deep Random Forest, which employs the superior descriptive ability of deep models in the prediction process as well as maintaining the explanatory of decision tree-based models.

LSTM. LSTM is an improved version of the original RNN, which largely resolved the problem of gradient vanishing and explosion in original RNN and is capable of capturing multi-scale dependencies in sequential data. The main modification it made to the original RNN unit is to add multiplicative **gate** unit that control the proportions of information to forget/pass on to the next time step. Even more modern version of LSTM like BLSTM equip the model with the ability to learn from the future, but here we only use the basic LSTM to explore the time dependencies in the dataset.

3 DATA ANALYSIS

With careful review of the data used in the original paper, here we: (i) show the authors had used the data incorrectly in the context of making predictions, and (ii) reveal the temporal properties that the original method failed to employ but perfectly fits LSTM model (long short term memory neural networks), which we will later use to improve the performance.

3.1 Justification of Data Processing

The original paper [12] made a major mistake on data employment: they implicitly used data from the future to predict events in the future, which is not possible. More specifically, all of their tests on the performance of *NetPerfTrace* depend on statistics calculated out of data of all time, including the data they were to predict. Features like percentiles of route life, for example, take the route life of all routes during the whole week into account, including route r_i. However, these features still take part in predicting the residual life of r_i. It resembles guessing something you already knew.

In fact, a principle of making predictions is to use the data already observed to guess the unobserved. Thus, in all the experiments of this paper, we made it a hard constraint that all the features used to predict any event at, say, time t, only come from the raw data collected before t. First, we re-validated the performance of classic machine learning method on two tasks(task 1 and task 3) according to our justified data processing manner, (The reason why we miss the task 2 is that it is not possible to reprocess data with respect to task 2 according to the raw data given by the original paper) in the following three ways: (i) use the global statistical features (average, minimum, maximum, and percentiles, etc) of the former $k\%$ data to train the models and predict the targets of the latter $(1-k)\%$ data, (ii) use the global features to train the models and update them during the predicting procedure and (iii) use data in a certain recent time-slot to calculate the global features. Then we train LSTMs on these data in two different ways: (i) input only a scalar, the prediction target, to the network and (ii) input the prediction target and the global features as a vector.

3.2 Temporal Properties of the Data

We believe temporally adjacent routes and `traceroute` records of a path share some common network nodes, thus their behavior contain implicit dependencies in temporal domain. While the data are organized in the order of time, the original method ignored this property but regard them as independent instances, i.e., the prediction for the next time step only depends on the features of the present, but does not use information given by more previous time steps. This manner fails the original method to capture even the most naive temporal dependencies. Here we do not provide

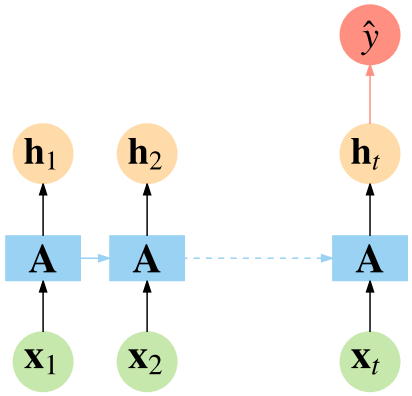


Figure 1: Illustration for LSTM.

strict proofs of the temporal dependencies that lie in these routes but simply throw some light on the possibility of their existence.

Applying simple differential analysis to the raw data, we found the prediction targets indeed show local temporal patterns which we can hopefully capture with LSTMs. **Fig.2**, **Fig.3** and **Fig.4** exhibit differentiated results of the three prediction targets in different paths. We can see that these paths behave in somehow consistent temporal patterns along their own ways but differently from each other.

No more solid evidence could be provided to prove the periodicity of these patterns because pattern itself is hard to define. However, these results are already enough to make us hopeful about the performance of LSTM, which are expert in capturing temporal dependencies, no-matter long-term ones or short-term ones.

4 METHODOLOGY

4.1 Data Processing

We reprocess the data in several more rational ways, which in other words, predicting targets using the previous `traceroute` data only. And to be specific, we divide the data into two sets, namely training set and test set. We calculate statistical properties (global features) of paths using three different manners: (*i*) use the global features (max, min, average, percentiles, etc.) in the training set to get the model and predict the targets of the latter $(1-k)\%$ data, (*ii*) use the global feature of the former $k\%$ to train the models and update these features during the predicting procedure(i.e.,update the global features everytime there is a new `traceroute` measurement of a path), and (*iii*) only use recent `traceroute` data of certain length (namely *window size*) in the training set to calculate the global features, and when using the trained model to predict targets, we only consider features calculated in recent `traceroute` measurements of the same *window size* . For convenience, in the following article, we use $data-i$ to represent reprocessing the original data using the ith processing manner (i can be either 1,2 or 3). And for the third processing manner, we use $data-3-ws\#$ to indicate the *window size* of #.

Target	$R_r(t)$	$avgRTT_P(t)$
data-1	0.4016	0.8124
data-2	0.3985	0.8261
data-3-ws10	0.7132	0.8018
data-3-ws20	0.6314	0.7910
data-3-ws50	0.5359	0.7790

Table 1: Experiment results(*ABC*) of random forest models.

4.2 Long Short-Term Memory

In order to leverage the prediction targets' character of temporal locality, we further use Long Short-Term Memory (LSTM) to do time series modeling. More formally, we take $t-1$ most recent `traceroute` measurements features $(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{t-2}}, \mathbf{x_{t-1}})$and the current `traceroute` measurement feature $\mathbf{x_t}$ together as the input, i.e., $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{t-1}}, \mathbf{x_t})$. And the output is a single scalar $\hat{y}$ which is supposed to be the prediction of the target y.

Let $G_\theta(\mathbf{x_i}, \mathbf{h_i})$ the θ parameterized LSTM, we have:

$$\mathbf{h_{i+1}} = G_\theta(\mathbf{x_i}, \mathbf{h_i}), \tag{1}$$

iteratively. and when we have the $\mathbf{h_t}$, we calculate our final prediction $\hat{y}$ by a fully connected layer with parameter $\mathbf{W}$:

$$\hat{y} = \mathbf{W} \cdot \mathbf{h_t}, \tag{2}$$

and the loss function is:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2. \tag{3}$$

The training objective is to minimize $\mathcal{L}$. The LSTM we adopt is illustrated in **Fig.1**

5 EXPERIMENT

5.1 Random Forest Model

In our experiment, firstly, we use the standard random forest model (the same as the original paper) with the reprocessed data. In practice, we set $k = 70, 80$ respectively. And in the third processing manner, the *window size* is set to 10, 20 and 50 recent `traceroute` measurements, respectively. We find that there is little difference when setting k either to 70 or 80. Below are the results when $k = 70$. And we also define an evaluation metric *ABC*, which indicates the area below the curve ($ABC \in [0, 1]$). The closer *ABC* is to 1, the better prediction result it has. These figures in **Fig.5** and **Fig.6** illustrate the relationship between sample percentage and relative prediction error(i.e., $|\ Real - Predict\ |\ /Real$).

Tab.1 describes the *ABC* of two tasks in different data processing manner. From **Tab.1**, it can be inferred that for target $avgRTT_P(t)$: there is minor difference in processing the data in different manners . They all achieve relatively good results, which also indicates that for average RTT reported from `traceroute` measurement, it is not sensitive to the temporal locality. However, for the first predicting target, $R_r(t)$, it is very sensitive to the temporal locality. It changes rapidly as time changes. And to achieve better prediction result, using too much previous data will result in noise and decrease the performance of the model. Because from **Tab.1**, we can clearly observe that $data-3-ws10$ achieves the largest *ABC* compared with other processing manners. It considers only the recent data

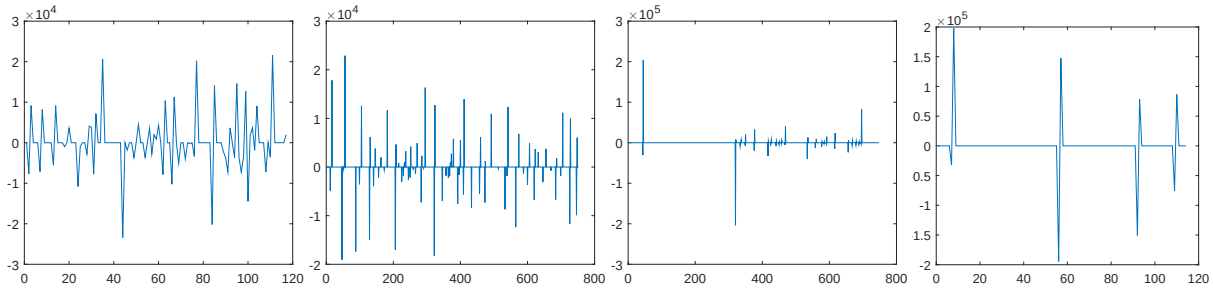


Figure 2: Temporal patterns of route life among 4 different paths.

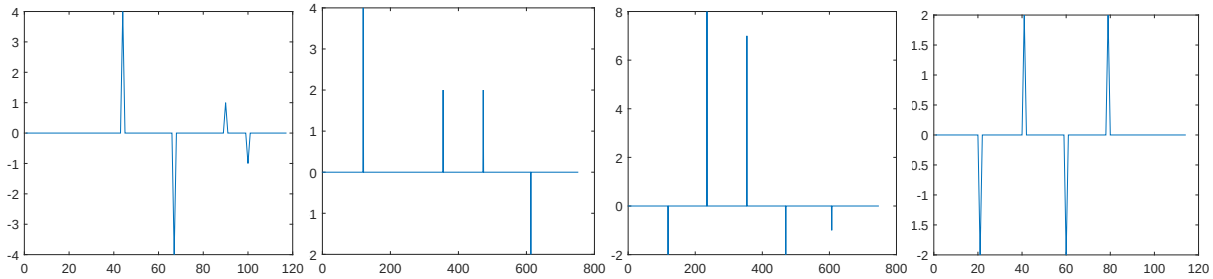


Figure 3: Temporal patterns of the number of route changes among 4 different paths.

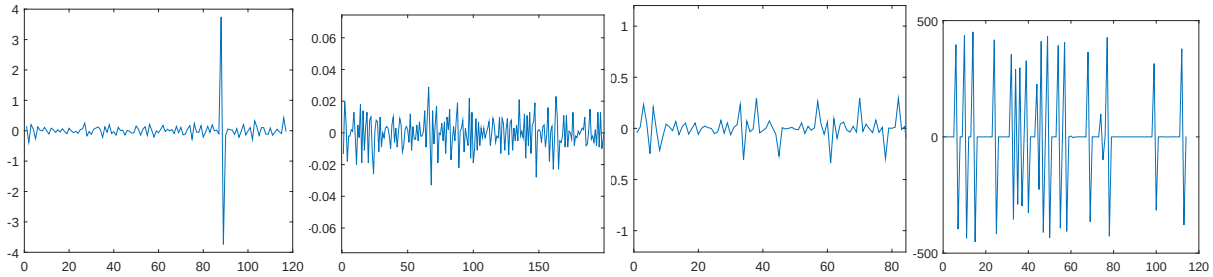
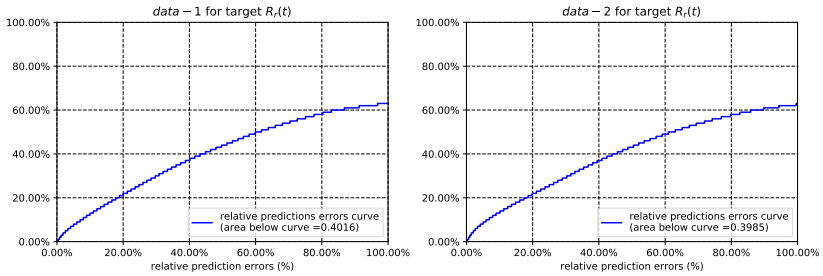


Figure 4: Temporal patterns of round trip time among 4 different paths.



in a relatively short time slot. Indeed, it is reasonable to get such conclusion, for the remaining life time of a route is always closely related to their route age (which can be indicated from the recent previous `traceroute` data).

5.2 LSTM Model

As we observed above, among the 3 different *window size* (10, 20, 50), *window size* of 10 has the best performance. Thus, we set the timesteps for our LSTM model to be 10, either.

We adopt two kinds of input for the LSTM at each timestep: (*i*) scalar and (*ii*) vector.

5.2.1 Scalar Input. Since we have observed some local temporal patterns with the prediction targets, The most naive approach is to

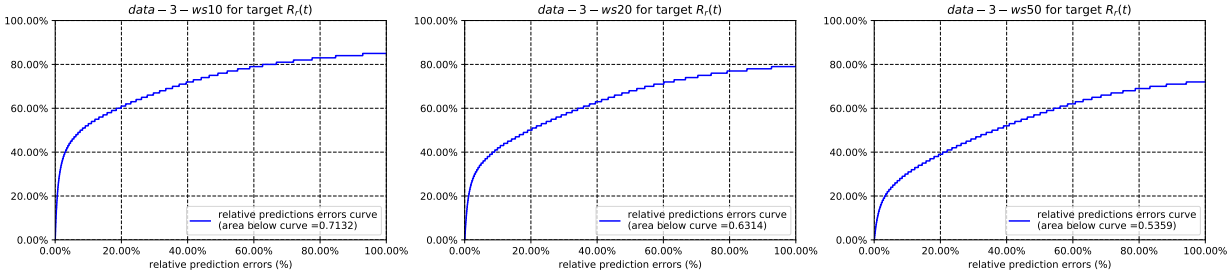


Figure 5: Relative prediction errors for $R_r(t)$ of random forest models.

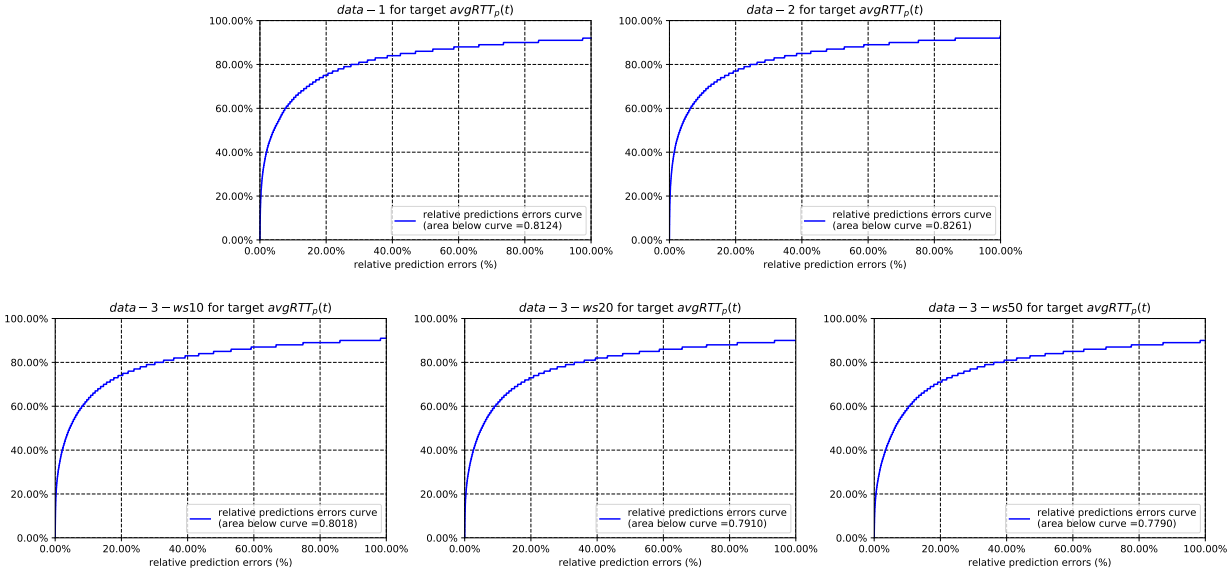


Figure 6: Relative prediction errors for $avgRTT_P(t)$ of random forest models.

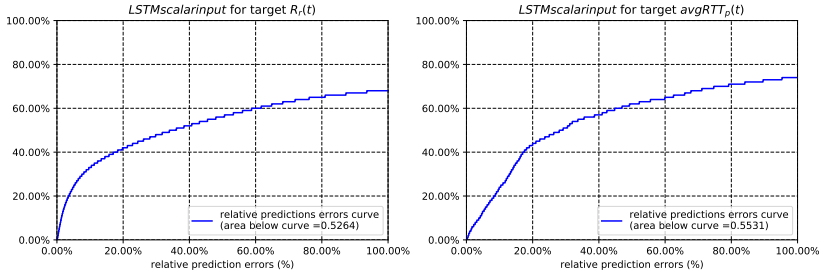


Figure 7: Relative prediction errors for $R_r(t)$ and $avgRTT_P(t)$ of LSTM models with scalar input.

Target	$R_r(t)$	$avgRTT_P(t)$
LSTM-scalar-input	0.5264	0.5530

Table 2: Experiment results(*ABC*) of LSTM models with scalar input.

feed LSTM with the previous observed scalar without any statistics of it. **Fig.7** and **Tab.2** show the results.

As we can see, for target $R_r(t)$, the *ABC* of LSTM is larger than random forest with *data-1* and *data-2*, but is much smaller than random forest with *data-3-ws10*. This shows empirical evidence that there is temporal locality with $R_r(t)$ of a path. We can infer $R_r(t)$ of a path from its recent `traceroute` measurements better than using current features only.

By contrast, for target $avgRTT_P(t)$, the *ABC* of LSTM is smaller than random forest in all cases, which may indicate that $avgRTT_P(t)$ is less likely to have the temporal locality.

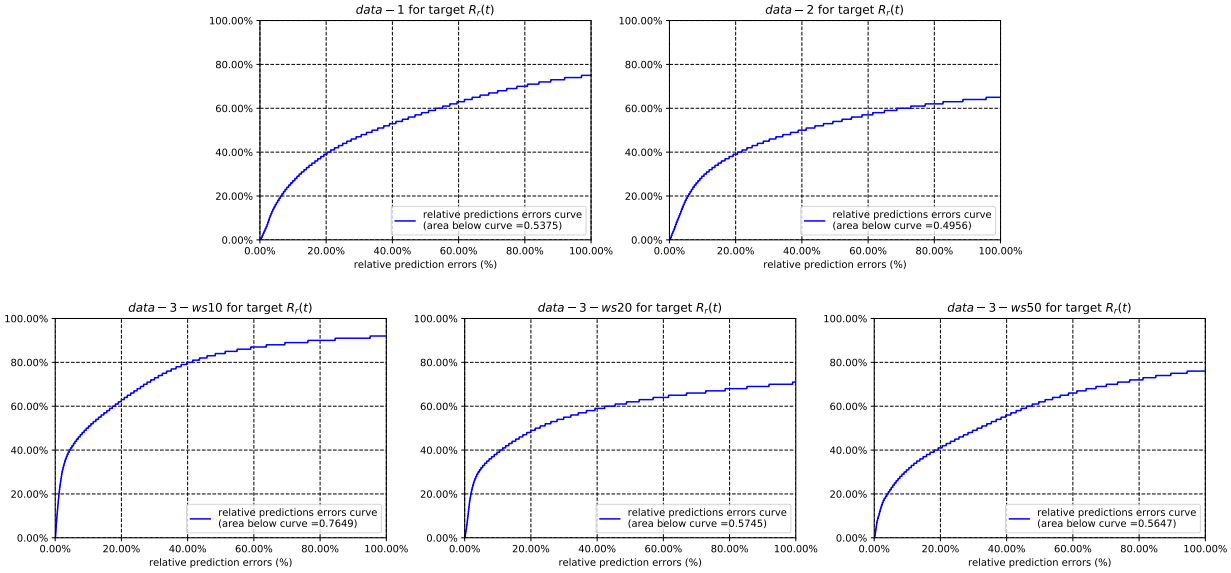


Figure 8: Relative prediction errors for $R_r(t)$ of LSTM models with vector input.

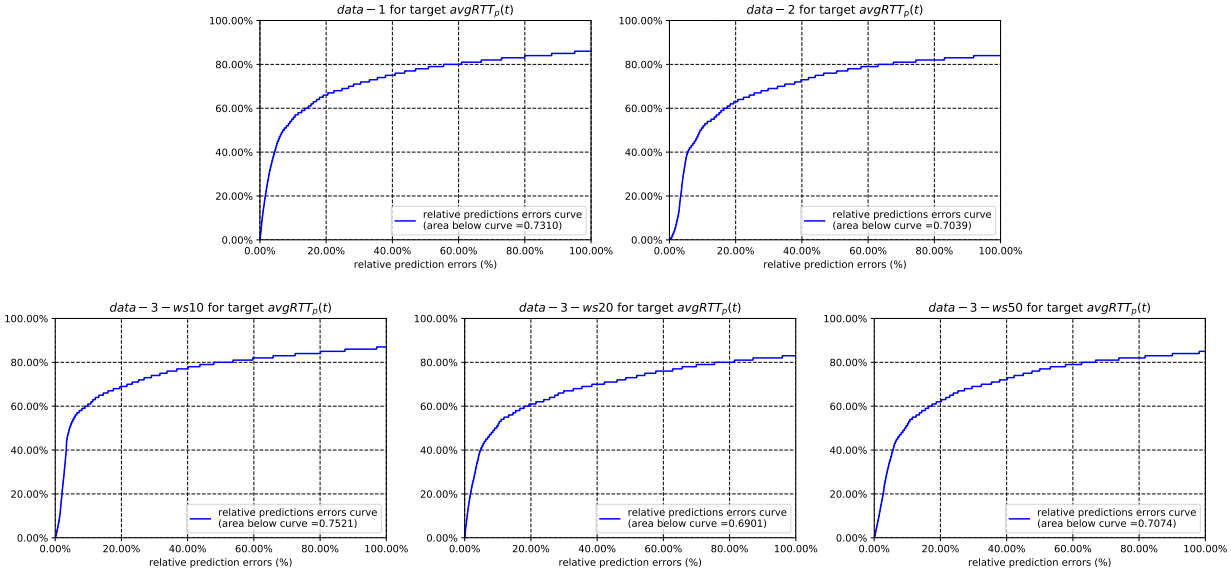


Figure 9: Relative prediction errors for $avgRTT_P(t)$ of LSTM models with vector input.

For both tasks, using scalar input for LSTM cannot outperform random forest steadily.

5.2.2 Vector Input. In order to better leverage the data to get better performance with our LSTM model, we take the same featues for LSTM model as those for random forest model. More specifically, Features for target $R_r(t)$ and features for target $avgRTT_P(t)$. The results are shown in **Fig.8**, **Fig.9** and **Tab.3**.

In general, the results consist with our previous observation that $R_r(t)$ has more temporal locality while $avgRTT_p(t)$ seems to have less or no temporal locality.

More specifically, for target $R_r(t)$, LSTM model outperforms random forest model in all five data processing manners with a maximum $ABC = 0.7649$. And when fed with vector input, the LSTM has a significant lift in performance than LSTM with scalar input, which shows that high quality features can be beneficial to not only classic shadow models like random forest, but also *deep* models like LSTM model.

However, when it comes to target $avgRTT_p(t)$, although LSTM models do have better performance when fed with vector input rather than scalar input, their *ABC*s are still smaller than those of random forest. We believe it is because there is less temporal

Target	$R_r(t)$	$avgRTT_p(t)$
data-1	0.5374	0.7310
data-2	0.4956	0.7039
data-3-ws10	0.7649	0.7521
data-3-ws20	0.6513	0.6901
data-3-ws50	0.5647	0.7074

Table 3: Experiment results (*ABC*) of LSTM models with vector input.

locality with reference to $avgRTT_p(t)$. The more data we observed before, the better we can predict $avgRTT_p(t)$ in the future. Thus LSTM model cannot maximize its ability in predicting temporal patterns.

6 CONCLUSIONS

In this paper, we study the problem of predicting Internet path dynamics and performance with `traceroute` measurement and machine learning models. We reprocess one dataset in several more rational ways and conduct extensive experiments on the reprocessed dataset with random forest and LSTM model. It is found that some features, e.g., $R_r(t)$, are more likely to have temporal locality than other features like $avgRTT_p(t)$, and this insight can guide us to select the most suitable model facing such tasks in computer network scenarios.

ACKNOWLEDGMENTS

This work is successfully produced thanks to the kind guidance from and discussion with Prof. Linghe Kong. His revealed us common shortcomings of ML-CN works and inspired us to build models that suits *network* better.

REFERENCES

[1] Md Zahangir Alom, VenkataRamesh Bontupalli, and Tarek M Taha. 2015. Intrusion detection using deep belief networks. In *Aerospace and Electronics Conference (NAECON), 2015 National*. IEEE, 339–344.

[2] Elaheh Biglar Beigi, Hossein Hadian Jazi, Natalia Stakhanova, and Ali A Ghorbani. 2014. Towards effective feature selection in machine learning-based botnet detection approaches. In *Communications and Network Security (CNS), 2014 IEEE Conference on*. IEEE, 247–255.

[3] Yung-Tsung Hou, Yimeng Chang, Tsuhan Chen, Chi-Sung Laih, and Chia-Mei Chen. 2010. Malicious web content detection by machine learning. *Expert Systems with Applications* 37, 1 (2010), 55–60.

[4] Umar Javed, Italo Cunha, David Choffnes, Ethan Katz-Bassett, Thomas Anderson, and Arvind Krishnamurthy. 2013. Poiroot: Investigating the root cause of interdomain path changes. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 183–194.

[5] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C Suh, Ikkyun Kim, and Kuinam J Kim. 2017. A survey of deep learning-based network anomaly detection. *Cluster Computing* (2017), 1–13.

[6] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2015. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2015), 865–873.

[7] Vern Paxson. 1996. End-to-end routing behavior in the Internet. *ACM SIGCOMM Computer Communication Review* 26, 4 (1996), 25–38.

[8] Supranamaya Ranjan. 2014. Machine learning based botnet detection using real-time extracted traffic features. (March 25 2014). US Patent 8,682,812.

[9] Kamaldeep Singh, Sharath Chandra Guntuku, Abhishek Thakur, and Chittaranjan Hota. 2014. Big data analytics framework for peer-to-peer botnet detection using random forests. *Information Sciences* 278 (2014), 488–497.

[10] Xiaoling Tao, Deyan Kong, Yi Wei, and Yong Wang. 2016. A Big Network Traffic Data Fusion Approach Based on Fisher and Deep Auto-Encoder. *Information* 7, 2 (2016), 20.

[11] Marina Thottan and Chuanyi Ji. 2003. Anomaly detection in IP networks. *IEEE Transactions on signal processing* 51, 8 (2003), 2191–2204.

[12] Sarah Wassermann, Pedro Casas, Thibaut Cuvelier, and Benoit Donnet. 2017. *Predicting Internet Path Dynamics and Performance with Machine Learning*. Technical Report. AIT Austria.

[13] David Zhao, Issa Traore, Bassam Sayed, Wei Lu, Sherif Saad, Ali Ghorbani, and Dan Garant. 2013. Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security* 39 (2013), 2–16.