Data Integrity Error Localization in Networked Systems with Missing Data

Yufeng Xin, Shih-Wen Fu, Anirban Mandal
RENCI, UNC - Chapel Hill
Chapel Hill, NC, USA

Ryan Tanaka, Mats Rynge, Karan Vahi, Ewa Deelman
ISI, USC
Marina Del Rey, CA, USA

***Abstract*—Most recent network failure diagnosis systems focused on data center networks where complex measurement systems can be deployed to derive routing information and ensure network coverage in order to achieve accurate and fast fault localization. In this paper, we target the wide-area networks to support the data-intensive distributed applications. We first present a new multi-output prediction model that directly maps the application level observations to localize the system component failures. In reality, this application-centric approach may face the missing data challenge as some input (feature) data to the inference models may be missing due to incomplete or lost measurements in the wide area networks. We show that the presented prediction model naturally allows the *multivariate* imputation to recover the missing data. We evaluate multiple imputation algorithms and show the prediction performance can be improved significantly in a large-scale network. As far as we know, this is the first study on the missing data issue and applying imputation techniques in the network failure localization.**

I. Introduction

Assurance of data integrity has been one of the most fundamental aspects of networked systems and Internet applications. Different mechanisms of error tolerance, detection, and mitigation have been widely implemented and deployed in different layers of the compute, storage, network, and applications systems. Unfortunately, these measures are not sufficient to cover the data corruptions in large-scale networks. For example, It is well known that the Ethernet CRC and TCP checksums are too small for modern data sizes [1]. Facebook recently reported a CPU bug that caused severe data corruptions in its hyper-scale data centers [2].

Data integrity error is a representative "gray" network failure [3], for which, network component faults are probabilistic and often evasive from the monitoring system. Gray failure diagnosis in large-scale networks is often a latent act after disastrous results from the applications and services. It remains a guessing art in most systems that requires intensive manual debugging and a daunting amount of communication between operators from different organizations that often takes days or weeks, primarily due to the infeasibility of accurate network models and incomplete coverage of system monitoring.

In recent years, machine learning (ML) techniques have found phenomenal success in solving the failure diagnosis challenges. The strength of ML models in learning the complex mapping between the failure root causes and the system level measurement observations effectively relieves the need for accurate domain models and full element level monitoring. However, the majority of these systems targeted the data center networks, where detailed network topology and traffic routing information can be determined and costly active and passive measurement systems can be deployed by the operators. The main technical contributions have been developing scalable inference models and measurement systems of complete coverage for fine classification or regression performance [4], [5], [6].

Accurate and fast fault localization requires complete network coverage from the measurement system within a limited time window. From ML perspective, training and test data sets with complete features are required for the model and inference. This is why substantial efforts were made in the existing work to develop complex measurement systems to ensure the network failure coverage.

In this study, we target a completely different network environment, Internet scale network, where multiple networks of different administrative domains are used to support distributed applications. Different from the data center networks, while the scalability might be smaller depending on topological and diagnostic granularity, the challenges on the available system information and measurement data are severed by the multi-domain nature of the Internet and applications. In this typical "opaque" network setting, deploying active probing to gather the network information and instrument the diagnosis in the production network is normally not feasible. It is also not realistic to deploy always-on passive monitoring system over the edges of the whole network.

On the other hand, modern Internet application software systems have built-in measurement and monitoring capabilities to ensure the application performance and mitigate the effects of failures in the network system layer [7], [8], [9]. From the gray failure diagnostic perspective, the fundamental ML based solution approach appears to be a very viable choice with its promise in learning a model to map the observations to internal faulty behaviors of the network. In our case, the design goal is application-centric: to infer the fault information at the component level from the application level measurement and monitoring information.

A big challenge in end host based application-centric solution approach is the data completeness being discussed above. First of all, the end hosts deployed on the network by an application may not need all the network components in forwarding the traffic. Even we limit our scope to exclude the part

of the network not being covered, in both training data set and testing data set, some features data may be missing in some data samples. In extreme case, some features may be missed totally for the entire data set. This data missingness challenge is exacerbated to deal with the data integrity errors for which the failure rate is normally very low and the application traffic may be very imbalanced among the source and destination hosts at the edge of the network. They may be caused by either incomplete coverage of the measurement, or the lost measurement records as the distributed measurement subsystem itself is not completely reliable in real-time, or there are just no measurements available for a part of application traffic during some time windows.

Missing data has been a prominent research topic in statistics and is garnering more active research interests in ML applications in the areas of medical science [10] and sensor applications [11]. The simple imputation techniques, like using the basic statistics (zero, mean, min, max, etc) of the existing feature data does not apply to our network failure diagnosis model because of the strong dependency between the application flows (features) and the system components (targets). The more suitable choices are the multivariate imputation algorithms that use the whole feature space to estimate (impute) the missing data in particular features.

In this paper, we first present a new multi-output ML prediction model that directly maps the application level observations to localize the system component failures. This model not only captures the fact that one faulty component would cause failures in multiple application flows, but also naturally allow the application of proven imputation methodologies to address the *missing data* challenge. Instead of pursuing more system information and complete measurement coverage, we focus on the multivariate imputation algorithms, parameter tuning, and quantifying their performance in improving the inference accuracy of failure localization. We also looked into the most recent algorithms based on the Generative Adversarial Nets (GAN) framework to generate the missing data [12], [13].

As far as we know, this is the first study on the missing data issue and applying imputation techniques in the area of network failure diagnosis. The evaluation results show satisfactory prediction accuracy. This model approach and missing data imputation results also present opportunities to the development and deployment of economical measurement capabilities in practical network settings.

II. A Network Failure Localization Model

In this section, we present the inference model to localize the failures based on the path level measurements. 1 shows a simple example network we target in this paper. It consists of several sites interconnected by a network cloud where only the network nodes and their interfaces are known (otherwise the failure localization is meaningless). A distributed application will incur traffic flows along an unknown path between pairs of end hosts. These flows are subject to the measurement of the applications to test if they are corrupted. Two such flows, $c3-d2$ and $c2-d4$ are shown in the figure. Intuitively, if

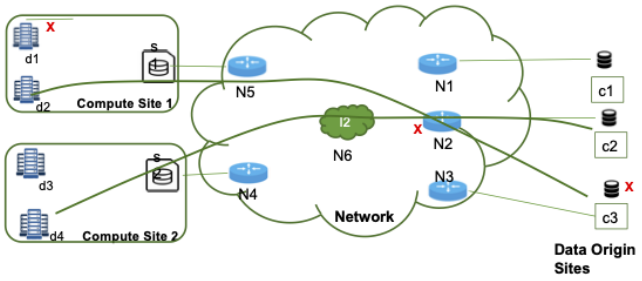


Fig. 1: An Example Network

both flows suffer from data corruption, the interface with the cross mark in Node $N2$ should be inferred to be the culprit.

In most existing work on in gray failure localization, the network is modeled as a simple graph $G(V, E)$ with a set of nodes V connected by a set of links E. The following bipartite mapping formula was used to capture the relationship between the link failures and the path level measurements [4], [5], [6]. The failure being considered is packet loss. The reasonings behind these models are similar: due to scalability or privacy constraints, monitoring every component of interest in a large-scale network is not feasible, while the path level measurement is more practical to deploy and instrument.

$$P(No\ Failure\ in\ Path\ i) = \prod_{j \in Path\ i} P(component\ j\ is\ normal) \tag{1}$$

This can be transformed to a familiar linear regression model for every identifiable path in the network after taking log on the equations.

$$p_i = \sum_{j \in Path\ i} c_j\ \forall i \in P \tag{2}$$

Here P represents the set of paths that are measured and a fundamental assumption underscoring these models is that routing of every path in P over link set E needs to be obtained. In addition, it was also assumed the measurement system have the access to all network nodes to instrument path measurements, *i.e.*, the source and destination of a path can be any nodes. In summary, to establish the regression model to obtain satisfactory inference performance, substantial efforts were made to (i) identify the routing of the paths, (ii) determine the path set for good coverage, and (iii) enable constant measurement of paths. Then the path measurements (p_i) will be obtained to estimate the link error probability (c_j) using this model.

In our wide-area multi-domain network setting, as we discussed in an earlier work [9], it is not practical to identify the routing of the paths over the network and even the network topology beforehand. This means that it is difficult to establish a model like Eq. (2). We also can not assume access to nodes in the network domains to instrument or measure paths.

We thereafter distinguish between application end hosts (that generate and receive data) and networking devices

(routers or switches), *i.e.*, V includes H end hosts and R routers. And we redefine E to be the set of network components where failures are supposed to be localized, specifically all the network interfaces on V and the end hosts H. We further constrain that only passive path measurements are available from certain applications on the end hosts, *i.e.*, P in our system only consists of paths originating from and ending in end hosts in H, which implies a much smaller identifiable path set. In the example network Fig. 1, none of the network nodes, $N1, \ldots, N6$, can be the source or destination of a path. And for a path between two end hosts, its routing is unknown. The only knowledge our model has is the bag of nodes and their interfaces for traffic forwarding.

We further observe that one component failure (e.g., x_j) could cause multiple paths erroneous while one erroneous path may be the result of a failure at different components. The standard model Eq. (2) ignores the correlation between multiple paths sharing a common component. We thereafter inverse the equations to represent the component failure probability as a function of the path failure probabilities as the following prediction model.

$$\begin{aligned} Y &= F(x_1, \cdots, x_p, \cdots, x_{|P|}) \\ &= \sum_{p \in P} w_p x_p + w_0 \end{aligned} \tag{3}$$

Specifically, Y represents a vector space $(y_1, \ldots, y_v, \ldots, y_{|V|})$ where y_v represents the failure probability of component $v \in V$. $X = (x_1, \ldots, x_p, \ldots, x_{|P|})$ forms the feature space that is defined by the combinations of the path failure probability. As shown in Eq. (3), we can further make it a linear regression model, which produces excellent performance as we will show in the evaluation section. We note, unlike in the existing work where failure localization is on network links, the network components in our model are the nodes and their interfaces because we assume the network topology is unknown.

Since any component failure only affects a small number of paths that go through it, plus multiple simultaneous failures are rare in reality, it is reasonable to expect both the feature matrix and the coefficient matrix is sparse, representing the samples collected during one inference window. This suggests using the regularization technique to make most of the estimated coefficient to be zero. The most efficient technique to achieve this intention is to add a L1-norm constraint is known as Lasso [5], where the regression optimization objective is defined as:

$$\hat{W} = \underset{W \in R^{|P|}}{\arg\min} ||\mathbf{Y} - \mathbf{X}W||_2^2 + \lambda ||W||_1 \tag{4}$$

Here $\mathbf{Y}$ and $\mathbf{X}$ are the sample matrix. This technique has proven extremely efficient in dealing with overfitting. The vector definition of $Y \in R^{|C|}$ means for each sample n, all entries but one in Y^n are zeros. Compared to the scalar variable of a specific component failure probability, this multi-output model captures the independence between all the failures and would help the training and prediction quality.

III. Missing Data and Imputation

As we discussed in Section I, missing data is pervasive in reality due to lost or unavailable measurement data. It means that some samples, in the training set or the test set, have missing features. Using the example network in Fig. 1 to illustrate, during a diagnosis time window, the application may not incur traffic between $c1$ and $d1$, or it never needs to transfer data between the origin sites, or the application measurement system may corrupt or lose some measurement data for some traffic flow. All these will lead to 'holes' in the feature columns in the data sets. In the first two cases, entire feature columns will be missing in our model 3.

Missing data can be categorized into three types: (i) the data is missing completely at random (MCAR) if the missingness does not depend on any of the observed and unobserved variables, (ii) the data is missing at random (MAR) if the missingness is dependent only on the observed variables, (iii) the data is missing not at random (MNAR) if the missingness is neither MCAR nor MAR, *i.e.*, the missingness depends on both observed variables and the unobserved variables. The majority of existing studies used the MCAR assumption [12].

There are many kinds of missing data recovery methods commonly used in the literature. These methods largely fall into three categories.

The *univariate* methods impute values in a feature dimension using only non-missing values in that feature dimension. It simply replaces the missed values with certain statistics of the non-missing values such as the zero, mean, median, mode, max, or min.

The *multivariate* imputation algorithms use the entire set of available feature dimensions to estimate the missing values based on the assumption of correlations between the feature dimensions. Each feature with missing values is modeled as a function of other features, and therefore the imputation itself is modeled as a regression problem that is trained and used to estimate the imputation. In order to achieve the best performance, especially to avoid the overfitting from certain features, it is conducted in a series of regression iterations: at each step, a feature is used as the output of other features and the resulted model is used to estimate the missing feature. After all features are processed or the designated max iteration is reached, the results of the final estimation are used to impute the missing data.

Our prediction model in Eq. (3) uses the path (flow) measurements as the input. It naturally fits the multivariate imputation approach because the path failures caused by a common component failure are correlated. In contrast, the existing models based on Eq. (2) use the component failure as the input variables that are independent of each other. The *multivariate* imputation does not seem to make sense. The *univariate* method is deemed not applicable due to the sparse nature of the feature matrix and lack of reasonable explanation.

Most recently, the Generative Adversarial Nets (*GAN)* framework has shown good performance to generate the missing data. In this model, the generator's goal is to accurately impute missing data, and the discriminator's goal is to distinguish between observed and imputed components. The discriminator is trained to minimize the classification loss (when classifying which components were observed and which have been imputed), and the generator is trained to maximize the discriminator's misclassification rate. Thus, these two networks are trained using an adversarial process [12], [13].

There are off-the-shelf libraries that support both univariate and multivariate imputations in popular software packages like R and Scikit-learn [14], [15]. The imputation can also be performed multiple times with different random number seeds to generate multiple imputations. This is important if the statistical analysis is needed, *e.g.*, in the medical domain.

Most of existing missing data studies focus on minimizing the imputation errors of the data in the feature space. However the ultimate goal is the performance of the prediction models after missing data is imputed.

Corresponding to our model in Eq. (3), missing data will cause values of some x_p to be null. The feature space is defined in a $|P|$-dimensional space $\mathbf{X} = \mathbf{X_1} \times \ldots \times \mathbf{X_{|P|}}$. Following the MCAR assumption on the missing data, we can define a mask vector $M = (M_1, \ldots, M_{|P|})$ taking random values in $(0,1)^{|P|}$. A sample vector $X = (X_1 \times \ldots \times X_{|P|})$ can be masked by M to generate a corresponding sample vector with missing data $\tilde{X} = \tilde{X_1} \times \ldots \times \tilde{X}_{|P|}$ as follow:

$$\tilde{X_p} = \begin{cases} X_p & \text{if } M_p = 1 \\ null & \text{otherwise} \end{cases}$$

From an arbitrary missing rate $r \in (0,1)$, a random mask vector M_r can be created to emulate missing data from a given feature matrix X. For a particular missing feature $X_r \in X$, the imputation essentially creates a regression model that makes X_r the output variable and all the other features the input variables.

$$X_r = F(x_1, \cdots, x_p, \cdots, x_{|P|}),\ p \neq r \tag{5}$$

At the end of the imputation, a recovered data set $\hat{X_r}$ is generated. The goal is to make these as close as possible.

Our main results are based on the MCAR missing data model, the *multivariate* imputation algorithms, and regularized regression model, which can be summarized in the following pipeline definition with Scikit_Learn.

```
estimator = make_pipeline(
    IterativeImputer(random_state=0,
     missing_values=np.nan,
     estimator=impute_estimator),
    PolynomialFeatures(poly),
    br_estimator
)
```

In the pipeline, the *impute_estimator* specifies the regressor for missing data imputation and the *br_estimator* specifies the regressor to infer the localized failure probability. We added a PolynomialFeatures element to evaluate if polynomials of higher degree perform better than the linear regressor.

This pipeline construct allows us to systematically evaluate the performance of multiple regressors in both *impute_estimator* and *br_estimator*, as well as tuning their hyperparameters. As we discussed earlier, in theory, Lasso should be a suitable regressor in both places. We also evaluated other popular regressors that include Ridge, BayesianRidge, ExtraTreesRegressor, and KNeighborsRegressor.

IV. Experiments and Evaluation

A. Emulation

In this section, we validate the performance of missing data imputation with the proposed failure localization prediction model in an emulated network as shown in Fig. 2. This topology mimics the Internet 2, the US Research and Education backbone network that has been used by many large-scale distributed middleware and applications. It is created in a high-fidelity emulation environment we built in the NSF ExoGENI cloud testbed [9] that can automatically create a virtual network system with virtual machines (VMs) running full software stack, bootstrap the network routing, initiate data transfers, and inject arbitrary integrity errors into the virtual router interfaces and end hosts to generate labeled training data and test data.

The network consists of 15 end hosts that can originate and receive data transfers and 87 network interfaces that a simulated distributed application may transfer data over. The application can check the data integrity of every data transfer at the receiving end hosts. During the emulation, errors with a given probability will be injected into the every end host and network interfaces in sequence, and one hundred data transfers are activated between all pairs of the end hosts in each round and the data transfer failure rates are computed. Therefore the training data set has 210 features, each of which represents a path between a pair of end hosts. We alter the error probability to generate multiple samples that are used to train the regression model.

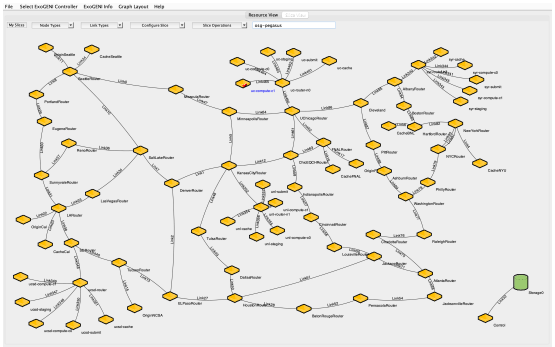
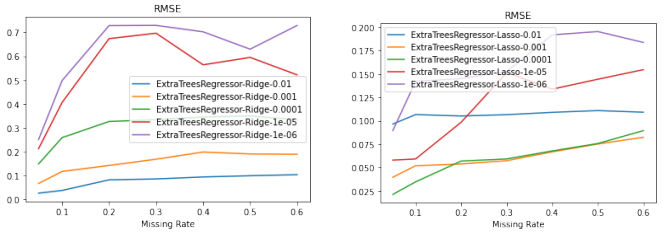


Fig. 2: Emulation Network Topology

To emulate the missing data, for a given missing rate, the mark vector M is generated randomly and applied to the data site to create the missing data in the input matrix. The imputation performance is measured by the root mean square error (RMSE) between the original training data and the data recovered by the imputation algorithm.

B. Missing data imputation performance

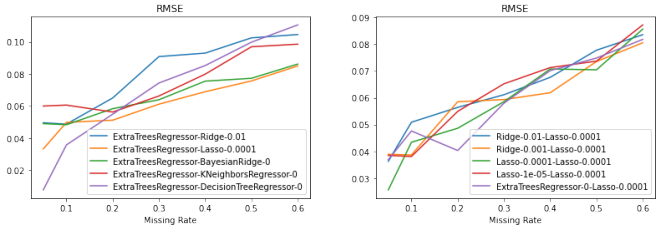
As we discussed in the last section, the regression models with a regularization term are normally suitable to sparse input matrices. We evaluated two representative L_1 regularization models, Ridge and Lasso with different penalty constant λ in Fig. 3. When varying the missing rate from 5% to 60%, $\lambda = 0.01$ for Ridge and $\lambda = 0.001$ and $\lambda = 0.0001$ for Lasso are the clear winners. Comparing the two figures, Lasso performs better than the Ridge counterpart.



(a) Imputation with Ridge **(b)** Imputation with Lasso

Fig. 3: Imputation with Regularized Regressors

We further evaluate three other regressors against the tuned Lasso and Ridge regressors in Fig. 4a. It confirms that the Lasso with a small penalty constant ($\lambda = 0.0001$) outperforms all other regressors except for the cases of very low missing rate. The BayesianRidge regressor performs closest to Lasso, the K-Neighbors regress the second, and the ExtraTrees regressor the next. When the missing rate is low ($< 20\%$), the ExtraTrees regressor performs much better than the other algorithms.



(a) Different Imputation Algorithms **(b)** Different Prediction Algorithms with Lasso Imputation

Fig. 4: Imputation Performance

Since we implement the imputation regressor in the pipeline consisting of the ultimate prediction model, we also tried different prediction algorithms with the best Lasso imputation regressor. From Fig. 4b, their impacts on the imputation performance seem to be small, though different with different missing rates.

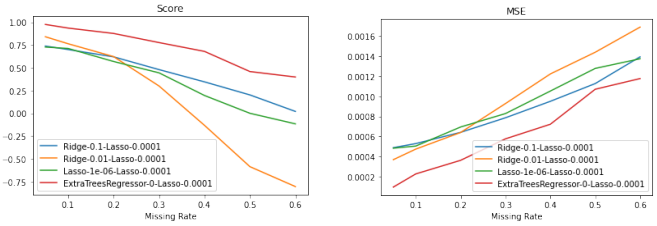
As we discussed in the last section, the latest GAN based imputation algorithms showed some promising results in the medical domain. We applied the Generative Adversarial Imputation Nets (GAIN) developed in [12] to our data set and compared its performance with Lasso in Table I. The results actually do not the GAIN model outperforms the Lasso algorithm. We'll leave the further validation and customization of GAN framework to our future work.

TABLE I: RMSE vs. Missing Rate)

	0.1	0.2	0.3	0.4	0.5	0.6
GAIN	0.1872	0.1794	0.214	0.2194	0.2666	0.2986
Lasso	0.05	0.0512	0.0612	0.0689	0.0756	0.085

C. Inference performance with imputed missing data

Many missing data imputation studies stoped after evaluating imputation performance as we did in subsection IV-B. However, our ultimate goal is to improve the inference accuracy of failure localization. In Fig. 5, we first evaluate the regression performance with different prediction algorithms on the best Lasso imputation regressor in terms of the coefficient of determination R^2 (Score) and the mean squared error (MSE) on the predicted output, the component failure probability vector Y in Eq. (3). We recall the higher score (best 1) and lower MSE mean better regression performance.



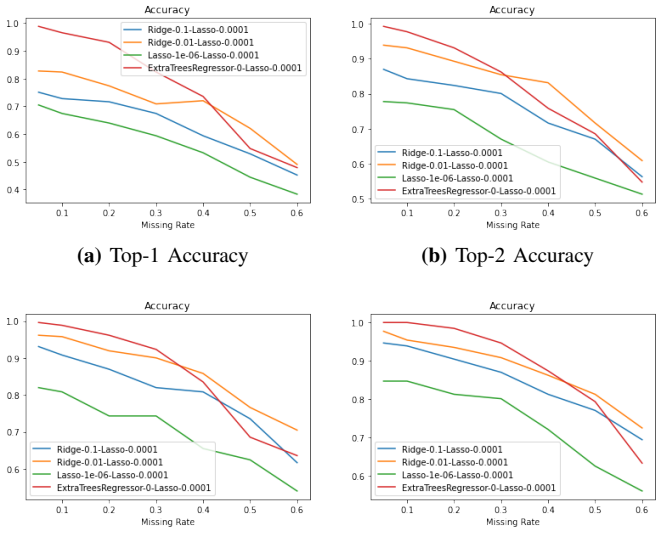
(a) Prediction Score **(b)** Prediction MSE

Fig. 5: Prediction Performance with Missing Data Imputation

The two metrics show the ExtraTrees regressor actually achieves the highest score and lowest MSE. The Ridge regressor with a small penalty constant (0.01) results in the worst performance. The Ridge with a bigger penalty constant (0.1) and the Lasso with a very small penalty constant (0.000001) result in a similar performance, not far from the ExtraTrees algorithm.

We proceed to attempt to determine the failure locations with the above combinations of prediction and imputation algorithms. For each test sample, after the predicted component failure probability vector $\hat{Y}$ is calculated, we sort the vector elements in descending order. In anticipating higher missing rate will likely significantly deteriorate the localization accuracy, we evaluate the $Top-K$ accuracy with small $k \leq 4$, which counts the correct label falls in the first k elements in the sorted Y vector. Fig. (6) shows the localization accuracy when $k = 1, 2, 3, 4$.

The results are promising. For the exact match ($Top-1$) in Fig. (6a), two best algorithms achieves over 50% accuracy at the missing rate 60%. When the missing rate is higher than 40%, the Ridge (0.1)+Lasso combination starts to outperform the ExtraTrees+Lasso algorithm, whose performance drops below the Ridge (0.01)+Lasso algorithm when $k > 1$. The results show some disparities from the regression performance results in Fig. 5.



(a) Top-1 Accuracy **(b)** Top-2 Accuracy

(c) Top-3 Accuracy **(d)** Top-4 Accuracy

Fig. 6: Top-k Localization Accuracy With Imputed Missing Data

V. Conclusions and Future Work

In this paper, we studied the diagnosis of data integrity error in a wide-area network that supports data-intensive distributed applications. Due to its multi-domain nature, information of the network topology and traffic routing, and network layer measurement system are either not feasible or too costly. Nevertheless, it is viable to construct Machine Learning models that rely on the application layer measurements to infer the failures inside the network. We first present a new multi-output ML prediction model that directly maps the application level measurements to the possible failure locations at the network components.

In reality, this application-centric approach may face the *missing data* challenge as some input (feature) data to the inference models may be missing due to incomplete or lost measurements in the wide-area networks. Missing data and the associated imputation techniques have been prominent research topics in statistics and are garnering more active research interests in ML applications as it is a pervasive problem in reality.

As our prediction model uses the path (flow) measurements as the input, it naturally allows the multivariate imputation because the path failures caused by a common component failure are correlated. We introduced several imputation algorithms under different missing data scenarios. Using a high-fidelity emulation environment we built in a Cloud testbed we evaluated the performance of the prediction model and the imputation techniques. The results showed fine-tuned regression model with regularization is very efficient in terms of missing data recovery performance and failure localization prediction accuracy.

For our future work, we plan to experiment with larger networks with different graph characteristics. We note full network coverage is not the focus of this study. As we explained, traffic in a particular application may not pass through all the network components. We will explore mechanisms to federate measurements from multiple applications to achieve higher network failure diagnosis coverage.

References Cited

[1] J. Stone and C. Partridge, "When the crc and tcp checksum disagree," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 309–319, Aug. 2000. [Online]. Available: https://doi.org/10.1145/347057.347561

[2] [Online]. Available: https://www.nextplatform.com/2021/03/01/facebook-architects-around-silent-data-corruption/

[3] P. Huang, C. Guo, L. Zhou, J. R. Lorch, Y. Dang, M. Chintalapati, and R. Yao, "Gray failure: The achilles' heel of cloud-scale systems," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*. New York, NY, USA: ACM, 2017.

[4] C. Tan, Z. Jin, C. Guo, T. Zhang, H. Wu, K. Deng, D. Bi, and D. Xiang, "Netbouncer: Active device and link failure localization in data center networks," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, Boston, MA, 2019.

[5] Q. Zhang, G. Yu, C. Guo, Y. Dang, N. Swanson, X. Yang, R. Yao, M. Chintalapati, A. Krishnamurthy, and T. Anderson, "Deepview: Virtual disk failure diagnosis and pattern detection for azure," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Renton, WA, Apr. 2018.

[6] B. Arzani, S. Ciraci, L. Chamon, Y. Zhu, H. Liu, J. Padhye, B. T. Loo, and G. Outhred, "007: Democratically finding the cause of packet drops," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, April 2018.

[7] E.-S. JUNG, R. KETTIMUTHU, and S. CHUNG, "High-performance end-to-end integrity verification on big data transfer," *IEICE TRANSACTIONS on Information and Systems*, vol. E102-D, no. 8, 2019.

[8] M. Rynge, K. Vahi, E. Deelman, A. Mandal, I. Baldin, O. Bhide, R. Heiland, V. Welch, R. Hill, W. L. Poehlman, and F. A. Feltus, "Integrity protection for scientific workflow data: Motivation and initial experiences," in *Practice and Experience in Advanced Research Computing on Rise of the Machines Learning (PEARC)*, New York, NY, 2019.

[9] Y. Xin, S. Fu, A. Mandal, I. Baldin, R. Tanaka, M. Rynge, K. Vahi, E. Deelman, I. Abhinit, and V. Welch, "Root cause analysis of data integrity errors in networked systems with incomplete information," in *12th International Conference on Information and Communication Technology Convergence*. Jeju Island, South Korea: IEEE, Oct. 2021.

[10] A. R. T. Donders, G. J. van der Heijden, T. Stijnen, and K. G. Moons, "Review: A gentle introduction to imputation of missing values," *Journal of Clinical Epidemiology*, vol. 59, no. 10, pp. 1087–1091, 2006.

[11] J. Du, M. Hu, and W. Zhang, "Missing data problem in the monitoring system: A review," *IEEE Sensors Journal*, vol. 20, no. 23, pp. 13 984–13 998, 2020.

[12] J. Yoon, J. Jordon, and M. van der Schaar, "Gain: Missing data imputation using generative adversarial nets," *ArXiv*, vol. abs/1806.02920, 2018.

[13] S. E. Awan, M. Bennamoun, F. Sohel, F. M. Sanfilippo, and G. Dwivedi, "Imputation of missing data with class imbalance using conditional generative adversarial networks," *Neurocomputing*, vol. 453, pp. 164–171, 2021.

[14] S. van Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of Statistical Software*, vol. 45, no. 3, pp. 1–67, 2011.

[15] M. Mera-Gaona, U. Neumann, R. Vargas-Canas, and D. M. López, "Evaluating the impact of multivariate imputation by mice in feature selection," *PLOS ONE*, vol. 16, no. 7, pp. 1–28, 07 2021.