Quantify Co-Residency Risks in the Cloud Through Deep Learning

Jin Han, Wanyu Zang, Meng Yu, *Member, IEEE*, and Ravi Sandhu, *Fellow, IEEE*

Abstract—Cloud computing, while becoming more and more popular as a dominant computing platform, introduces new security challenges. When virtual machines are deployed in a cloud environment, virtual machine placement strategies can significantly affect the overall security risks of the entire cloud. In recent years, the attacks are specifically designed to co-locate with target virtual machines in the cloud. The virtual machine placement without considering the security risks may put the users, or even the entire cloud, in danger. In this article, we present a fine-grained model to quantify the risk level caused by co-residency. Using a large scale dataset collected from Microsoft Azure Platform, we profile the behavior patterns of normal service subscribers (tenants) using our proposed feature metrics. Tenants are clustered into multiple categories. After the baseline is established based on the normal behavior pattern, the derivation can be evaluated for each category and the high-risk group can be labeled accordingly. With the labeled datasets, a classification component and a quantification component are constructed to dynamically quantify the co-residency risks for a specific virtual machine. Our experimental results demonstrate the robustness of our model to the new data and the accuracy is verified by examination of F-score Matrix.

Index Terms—Cloud security, deep learning, co-resident attack

1 Introduction

For a cloud computing system, one of the most critical and fundamental components is the Virtual Machine (VM). Multi-tenant deployment of VMs significantly improves cloud resources utilization, which would benefit the cloud service provider. On the other hand, sharing resource among clients makes the cost of service more affordable and minimizes the maintenance overhead of the computing resources. Along with these advantages, cloud computing acts as more and more important roles in our daily life, and also attracts more attention from attackers. Since VMs could be exposed to various types of security threats, lots of research efforts [1], [2], [3], [4], [5], [6], [7] have been made into this particular field.

The logical isolation between VMs deployed on the same server is provided by virtualization techniques [8]. Under the control of a hypervisor, one VM can only access its own assigned partition of hardware resources, such as memory space or storage blocks. In the ideal situation, one VM should never be compromised by another VM on the same physical computer. However, in the real world, many stealthy attacks can happen. For example, cache utilization can determine the execution time of cache read operations [9]. As the result, various types of side channels between attacker's VMs and victim's VMs could be built and sensitive information could be extracted [1], [9], [10], [11], which are called co-resident attacks. One of the most influential co-resident attacks is to extract private keys [2] through a side-channel. Construct a useful side-channel needs to overcome challenges to handle processor core migration, numerous sources of channel noise, and the preemption of the victim with sufficient frequency to extract fine-grained information from it. The authors [2] demonstrated the attack in a lab setting by extracting an ElGamal decryption key from a victim using the most recent version of the libgcrypt (v1.5.0) cryptographic library [12].

To address the problem, a lot of research have been done to defend against co-residency attack. Most of them [13], [14], [15], [16], [17], [18] focused on preventing the construction of side channels. One major requirement of those methods is that substantial changes to the existing system are needed, which can be very expensive to a cloud provider. Other researchers investigated the problem from different perspectives. Sundarewaran, *et al.* [19] proposed a defense mechanism to identify anomalies in CPU and RAM utilization, or cache miss rate. To improve the overall security level of the cloud system, we proposed a four-dimensional security evaluation model to cover possible attack paths in a cloud [20]. In this work, the co-resident risk is taken into consideration and a coarse-grained method was proposed to quantify the co-residency risk through individual VMs, although the effectiveness of our model was not verified via real world case. Another major issue is we believe the co-residency risk factor should be determined by the tenants, instead of individual VM. If a tenant is identified as an attacker, all of his owned VMs should be considered with high risk level. To better address this issue, Yi Han *et al.* [21] proposed a mechanism to identify the co-resident attacker's

- *Jin Han is with Samsung Research, Austin, TX 78746 USA. E-mail: jin.han@utsa.edu.*
- *Wanyu Zang and Meng Yu are with the Department of Computer Science, Roosevelt University2460, Chicago, IL 60605 USA. E-mail: {wzang, myu04}@roosevelt.edu.*
- *Ravi Sandhu is with the Department of Computer Science and the Institute for Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249 USA. E-mail: ravi.sandhu@utsa.edu.*

Manuscript received 15 Jan. 2020; revised 22 Sept. 2020; accepted 4 Oct. 2020. Date of publication 21 Oct. 2020; date of current version 9 July 2021. (Corresponding author: Jin Han.)
Digital Object Identifier no. 10.1109/TDSC.2020.3032073

behaviors in the absence of any defense, and proposed the semi-supervised learning algorithm, Deterministic Annealing Semisupervised SVM (DAS3VM) [6], to classify normal users and high risk users in the cloud [22]. Followed with his lead, multiple works were done [7], [23], [24] based on Game Theory Model. However, the effectiveness of game was determined by the accuracy of the classification of attackers. In their experiment, the clustering was either done with a simulation or small size dataset, which had a very limited chance to contain real attackers.

Compared with previous work [6], [21], [22], we are taking challenge of dealing with real-world datasets of clouds in this paper. To better understand and predict workloads for improved resource management of large cloud platforms, Microsoft construct an open VM trace dataset source, which contained over 25,000 service subscriber accounts with 4.6 million VMs and 3.1 billion utilization reading records [25]. A. George *et al.* [26] provided another four new traces from two private and two high-performance computing clusters to conquer the over-fit issue with original dataset when evaluating the generality of new research. The large-scale dataset is widely used in the workload related research field [27], [28] and we believe that it also provides a great opportunity to profile normal service subscriber's behaviors, which could separate attacker out. One major improvement in our current work (classification module) is that our work is based on above large-scale real-world datasets. In our previous works [20], [29], we proposed a set of coarse-grained metrics to quantify the risk level of the entire Cloud. Co-residence risk acts as the most important portion of risk estimate in our proposed metrics. Due to the limitation of simulated data used, we calculate the co-resident risk level based on the vulnerabilities of each VM, which is insufficient for more complicate situations. The introducing of Azure dataset leads us to a better mechanism (deep learning method) to evaluate the co-residency risk on tenant level, instead of individual VM. Based on this new mechanism, we evaluated these new virtual machine datasets to quantify the security risks of the entire cloud system. The unique contributions of our work are as follows.

- Based on a large scale real-world dataset, we profile the behavior patterns of normal service subscribers (tenants).
- We propose an effective feature metrics abstracted from real-world dataset, which are used to profile the behavior of tenants more accurately.
- We propose an innovative framework to dynamically quantify the co-residency risk level for a specific tenant and his VMs.
- Based on our experimental results using F-score Matrix [30], our model demonstrates the robustness to new seen datasets.

One major challenge we conquered is how to handle the extremely imbalanced large-scale dataset, which contained huge amount of traces from normal tenants, but very tiny amount of traces from attacks.

The rest of our paper is organized as follows. In Section 2, we discuss the related work. Section 3 provides an overview of our proposed approach. Section 4 describes the design details. Evaluation results are discussed in Section 5, and Section 6 summarizes our work.

2 Related Work

Logical isolation between VMs deployed into the same server was provided by Virtualization techniques [8]. Under the control of hypervisor, one VM could only access to its own assigned partition of hardware resource, such as computing power or data storage. One VM should never be affected or compromised by another VM's behavior. However, in real world, many stealthy attack could happen. For example, cache utilization rate would determine the execution time of cache read operations [9]. As a result, various types of side channels between attacker's VMs and victim's VMs could be built and sensitive information could be extracted [1], [9], [10], [11], which also was called as co-resident attack [30]. There is another threat generated by this risk type is VM image manipulation [31], where the victim's VM image could be used by attack to create some new VMs for attacking.

Resource sharing in cloud computing raises a threat of Cache-Based Side Channel Attack (CSCA). It is proposed to detect and prevent guest virtual machines from CSCA. Cache miss patterns were analyzed in this solution to detect side channel attack. CSCA is divided into two types and those are time-driven cache attacks and trace-driven cache attacks. It is based on a cloud setting with two VMs installed on the same physical machine using a bare-metal hypervisor sharing a highest-level cache. One of the most influential one is to extract private keys [2]. Author successfully constructed such a side-channel requires overcoming challenges including core migration, numerous sources of channel noise, and the difficulty of preempting the victim with sufficient frequency to extract fine-grained information from it, and demonstrates the attack in a lab setting by extracting an ElGamal decryption key from a victim using the most recent version of the libgcrypt cryptographic library. Since the Last Lever Cache was used as the side channel for attack, Yinqian Zhang *et al.* [23] describer a Home Alone system that makes tenant verification of exclusive utilization of VMs over physical machine. The primary concept behind Home Alone is to reverse the essential application of side channels. Indeed of making the best advantage of side channel to be vector of any attack, Home Alone utilizes a side-channel (in Last Level Cache) as a new protective detection tool. By examining the utilization of cache while the period in which friendly VMs collaborate to eliminate segments of cache, the tenant with home Alone can identify co-resident activity of foe VM.

To defend this particular type attack, lots of other research effort was devoted into different directions. One of them is to enforce the isolation of virtual resources. Khalid Bijon *et al.* [32] proposed a formal Trusted Virtual Datacenter (TVD) management model to manage strong isolation among virtual resources and also develop an authorization model for the cloud administrative-user privilege management in the system. In their following works [33], [34], they presented attribute-based constraints specification and enforcement as a mechanism to mitigate such co-resident risks. Conflicting attribute values are specified by the tenant

or by the cloud IaaS system as appropriate. Based on conflict specifications, author developed a conflict-free virtual machine scheduling framework. Our proposed framework is also inspired by their work and focused on the scheduling strategy of deployment of Virtual Machines.

Another approach is to prevent the construction of a specific side channel. Aviram *et al.* [13], C. Vattikonda *et al.* [15] and J. Wu [16] focus on eliminating the timing channels. Cache-based channel is another major defense target. J. Shi [4], T. Kim [17] and Y. Zhang [18] worked on mitigate cache-based side channel attack to better protect cloud system. As we discussed, these defense mechanisms are limited by their specific target and are not universal solution. Another major common requirement of these methods is that substantial changes to the existing system are needed, which can be very expensive to the cloud provider or very difficult to utilize these defenses.

Some of existing works focused on the analysis of the potential data traffic information. Adam Bates *et al.* [35] depicts about Co-resident watermarking, traffic analysis attack, which facilitates dangerous co-resident VM to introduce watermark signature over the internet flow of target instance. The final examination illustrates that there is a careful hardware design to be utilized in the cloud environment. Similarly, there has been many techniques developed for the security of cloud computing. Several security metrics were proposed to better protect the cloud. Taous Madi *et al.* [36] propose a quantitative model and a set of multi-level distance metrics for multi-tenancy in the cloud at different layers. Followed by their lead, N. Alhebaishi *et al.* [37] model both cross-layer and co-residency attacks on the NFV stack and mitigate such threats through optimizing the virtual machine (VM) placement with respect to given constraints.

Another interesting direction is based on game theory. Yi Han *et al.* [21] proposed a mechanism to identify the co-resident attacker's behaviors in the absence of any defense. Overall speaking, the attacker has to start a much larger number of VMs than the normal user to achieve the co-residency with their target. Based on the assumption, they used the semi-supervised learning algorithm, Deterministic Annealing Semisupervised SVM (DAS3VM) [6], to classify normal users and high risk users in the cloud. In their subsequent work [22], they proposed a semi-supervised learning based defense strategy to increase the attack cost. Followed with his lead, multiple works were done [7], [23], [24] based on Game Theory Model. C. zhang *et al.* [38] proposed an information Model to increase user's value and improve user experience by reducing job failures. In those works, the two-player security game will be utilized. However, the effectiveness of the game was determined by the accuracy of the classification of attackers. In their experiment, the clustering work either done with simulation or a small size dataset, which had a very limited chance to contain real attackers. To conquer this weakness, our proposed framework is based on real-world large scale dataset, and it also could dynamically adapt to keep-evolving environment.

Our previous work [39] proposed a VM placement scheme based on security risk of each VM, and Yuchi and Shettey [40] extended it to the VM placement initialization. Both of them mainly focused on the dependency relations. Yuchi and Shettey's method also oversimplified the problem and did not reflect the potential risk caused by co-resident VMs, whose importance was discussed in [41], [42]. In [42], the author studied the characteristics of different PaaS cloud and the co-resident threat in placement policies. They implemented a memory-bus based covert-channel detection for co-residence and presented an efficient launch strategy. Their experiment concluded the risk caused by co-residency was real in popular PaaS clouds. Previously, we have investigated to periodically migrate VMs based on the game theory, making it much harder for the adversaries to locate the target VMs in terms of survival capability measurement [43]. But we did not consider the risk caused by the co-resident VMs in the same physical machine.

3 Overview of Our Approach

In our previous work, we proposed a set of coarse-grained metrics to quantify the risk level of the entire Cloud. The co-resident risk level is based on the vulnerabilities of each VM, which is insufficient for more complex situations. To solve the problem, we develop a fine-grained model to better quantify the co-resident risk based on service subscribers(tenants). In our new design, we consider large scale dataset for real world cloud risk estimate. Furthermore, we design our framework in the way adapting to the dynamic changing environment.

3.1 Threat Model and Security Assumptions

To accomplish the co-resident attack, attackers have to make effort to deploy their own VMs on the same physical server with the victim's VMs. In other words, one major portion of the attacker's cost would be determined by the procedure of co-locating with the target. Service providers should have the capability to collect necessary information they need to profile a service subscriber. For stealth purposes, the attacker wants to reduce the possibility of being detected as much as possible.

We have the following assumptions specifically for profiling and classification of service subscribers: 1. Service provider has no knowledge about the attacker's presence; 2. Attacker's behavior pattern could be evolved and different in every time period; 3. Service provider can verify a limited amount of normal service subscriber (mark as legal); 4. Attacker has no way to compromise the data collected by the service provider for profiling purpose; and 5. Attacker cannot compromise the data processing components described in the paper.

The above assumptions ensure that our proposed defense mechanism works properly. Also, the attacks should not be common or standard events in the cloud. We assume that the major portion of data collected by service providers should belong to normal activities instead of attacks.

3.2 Patterns of The Co-Resident Attacks

To successfully accomplish co-resident attack, an attacker needs to achieve co-residency with the targets. Some previous work [21] focused the problem. A general procedure can be summarized as follows to launch co-resident attacks.

1) A large number of virtual machines are launched in the cloud.

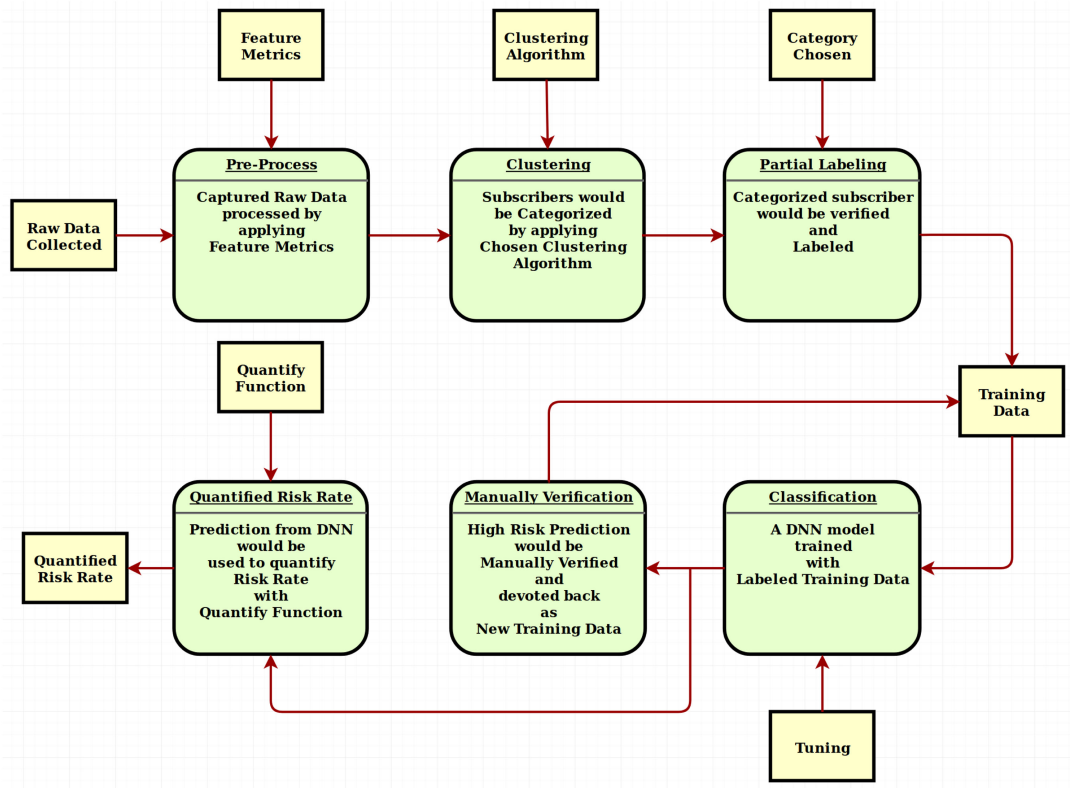

Fig. 1. Overview of framework.

2) Check if any of attacker's virtual machines is deployed on the same physical machine with the target.
3) In order to save the cost, turn off virtual machines which failed to co-locate with the target. This step is optional.

The above steps may be repeated several times by the attacker until co-residency is achieved. In contrast to the normal service subscriber, a large number of virtual machines have to be started by the attacker. To reduce the cost of attacks, virtual machines which failed to achieve co-residency will be short-lived. Otherwise, a high CPU utilization would be observed due to active long-lived virtual machines. In reality, the attacker might use a single subscriber account or multiple subscriber accounts to accomplish the attack. Nowadays, the Cloud service provider allows a service subscriber to start multiple VMs simultaneously in a single account. Thus, we cannot simply use the number of VMs owned by a subscriber to tell an attacker from legitimate service subscribers.

Note that the presence of attack can be rare in the real world, which means the major portion, if not all, of a dataset from a cloud, is from normal service subscribers. We believe that the behavior pattern of normal service subscribers should demonstrate some derivation from the above pattern of attacks. In other words, if service providers can successfully profile normal service subscribers, the attacker has to adjust their behavior pattern to match the normal pattern to reduce the chance being detected, which will increase the attacker's cost accordingly.

3.3 Proposed Processing Framework

Fig. 1 demonstrates the diagram of our proposed framework to classify service subscribers and better quantify the Co-Resident Risk Rate. It contains five necessary components to generate a quantified co-resident Risk Rate and one optional component to make our framework more adaptive to the practical environment. In the figure, the *Pre-Process* component organizes raw data using the feature metrics determined by the service provider. More detailed explanation of feature metrics are in Section 4. A service provider can update the feature metrics based on their unique requirements. The *Clustering* component classifies service subscribers into different categories. As the outputs, candidate categories for subscribers are sent to *Partial Labeling* Component. The *Partial Labeling* component examines the candidate categories of subscribers and determines the labeling of each category. Some categories of subscribers may be dropped in this component they are not relevant to quantifying the risks. In the *Classification* component, a Deep Neural Network is trained to perform the classification task for incoming subscriber. Hyper-parameter tuning is performed to increase training efficiency. In the *Quantified Risk Rate* component, the risk level of incoming subscriber is evaluated based on the prediction of classification component and pre-defined quantify function. The *Manual Verification* is an optional component to make our processing structure more adaptive to the real world. The prediction of new subscriber data can be verified manually by the service provider and send it back to training data to make our proposed DNN model adapt to the latest environment.

4 Design

In this section, we provide more detailed discussions and explanations of our design.

4.1 Feature Metrics to Profile Normal Service Subscriber(Tenant)

In our proposed framework, the feature metrics are used in the Pre-Process component to process collected raw data. As we mention earlier, service provider can define their own feature metrics according to their unique business requirement. In our experiments, we propose a six dimension feature metrics to profile service subscriber:

1) N - the total amount of VMs created and deployed by a specific service subscriber.
2) T - the average interval time between starting two VMs. Note that it is the time between starting the ith and the (i+1)th VMs, rather than difference between the stopping ith and the (i+1)th VMs.
3) M - the median memory size among VMs for a specific service subscriber.
4) A - the overall active rate for a specific service subscriber. Detail explanation would be explained by following section.
5) W - the average amount of active VM at each time stamp for a specific service subscriber.
6) I - the median of average CPU utilization rate among all VMs at each time stamp for a specific service subscriber.

Among these features, feature 1) to 4) are the overall analysis factors for each service subscriber. Feature 5) and 6) are detail features for a specific service subscriber. The reason we have two types features, because not only we want to know the whole picture of subscriber's behavior, we also want to more accurately profile each subscriber's behavior pattern and build the characteristic image with detail inside information.

M - Median Memory Size. For each service subscriber, the memory requested by the VM_i is recorded as M_i. The feature M is calculated as:

$$M = \mathrm{median}(\mathrm{M_i}). \tag{1}$$

We believe that the requested memory size could represent the tenant's expectation of potential workload for the VM.

A - Overall Active Rate. In our experiment, for a subscriber, its VM_i at time stamp t_j to be considered as active if its CPU utilization is over 15. Assume at the time stamp i, one subscriber has total NT_i alive VMs, where NA_i VMs are active. Then, the Overall Active Rate for the subscriber would be:

$$A = \left(\sum_{i=1}^{n}(NA_i/NT_i)\right)/n. \tag{2}$$

W - Average Active VM Amount. Feature A presents the overall active rate among all timestamps. We need a feature to demonstrate the overall workload requested by the subscriber. Assume at the time stamp i, one subscriber has a total NT_i VMs alive, where NA_i VMs are active. Then, the average active VM amount for the subscriber would be:

$$W = \left(\sum_{i=1}^{n} NA_i\right)/n. \tag{3}$$

I - Median of Average CPU Utilization. Features A and W present the activeness level for a subscriber, but they do not cover any difference of each VM even with the same activeness rate. We introduce feature I : median of Average CPU Utilization Rate in overall period per subscribers.

Assume at the time stamp i, one subscriber has total NT_j VMs alive. For each VM, its CPU utilization is CPU_j. Then, the median of average CPU Utilization Rate in all time period for the subscriber would be:

$$C = \mathrm{median}\left(\left(\sum_{\mathrm{j}=1}^{\mathrm{NT_j}}\mathrm{CPU_j}\right)/\mathrm{NT_j}\right). \tag{4}$$

We use the median value rather than the mean value is because the median value is more robust with outliers.

Since we are taking a data-driven approach in our classification module, the effectiveness of features acts as the most dominant factor. Also, our proposed framework is general enough and can be used in different cloud service platforms. The service provider can easily adjust above feature metrics to meet their own business need.

4.2 Choice of Clustering Algorithm

In the clustering component, the service provider can choose a specific clustering algorithm per their own need. In our work, we test several clustering algorithms and chose DBSCAN in the end due to it's robustness towards the outlier detection. There are two parameters in the DBSCAN algorithm, ε and MinPts, where ε is the maximum distance between two neighbors, and MinPts is the minimum number of points in a cluster. Once MinPts is set, ε can be determined by drawing a k-distance graph (k=MinPts). In other words, ε can be considered as a function of MinPts. MinPts should not be too small, otherwise, noise in the data will result in spurious clusters.

In our experiments, we use DBSCAN (ε=35, Minpts=10) to initially cluster subscribers into total 6 categories. Then we manually verify every category with the activeness level and partially label them into three major types: Inactive (Normal), Periodically Active, and Extremely Active. Generally speaking, most of original category 1 and 2 was put into Inactive type. Extremely Active type are most from original category 0 (outliers) and category 5. The labeled data will be used as the training data of the classification component.

4.3 Dynamically Quantify Co-Residency Risks

To better quantify the co-resident risk level, we combine classification and quantification in our proposed framework. Though the cloud service provider could collect any information they want and use them to detect malicious activities, the presence of malicious activity is a very rare case and majority of the collected data belongs to normal service subscriber, which makes the classification task very difficult, if not impossible. However, the rich information of the normal behavior enables the service subscriber to successfully profile normal tenant's behavior patterns. The service provider is be able to determine a risk rate by measuring the derivation rate from the normal pattern. In order to do so, the service provider needs to build a non-

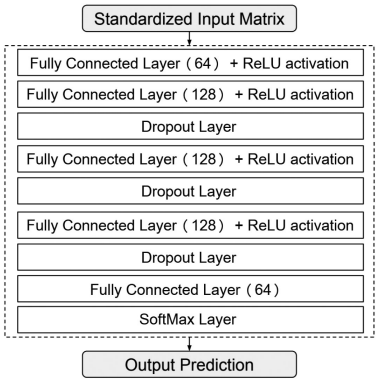

Fig. 2. Architecture of proposed deep neural network.

rule based general system which can dynamically adapt to the environment changes.

4.3.1 Deep Neural Network Architecture

Fig. 2 shows the Deep Neural Network model used in our experiments, which was implemented in Tensorflow Framework. It consists of 9 layers. First, the fully connected layer has 64 nodes (or neurons). Afterward, three fully connected layers with 128 neurons and Relu activation, followed with the dropout layer. The last layer is a 3-node softmax layer that returns an array of 3 probability scores that sum to 1. In our current experiment, during the feature abstraction procedure, detail active rate curve for each tenant was converted into a vector. Fully connected(FC) layer is designed to handle vector info efficiently. At the same time, even transfer training for new seen data can be finished offline, it might be important for some provider to enable online transfer training. The depth of model should be carefully determined. We applied five layers of FC layers here, but it can be adjusted by service provider to achieve the balance between the complexity of target model(Better detection accuracy) and acceptable online transfer training time for new seen data.

Dropout is a regularization technique that turns neurons on/off in each layer to force them to go through a different path. This operation improves the generalization of the network and prevents over-fitting. To reduce overfitting, we use a dropout [44] layer after the second and third fully connected layers, since its effectiveness is proved in [45] and the dropout regularization works really well with the fully connected layers. Rectified linear unit (ReLU) is a type of activation function. It is simply defined as $f(x) = \max(0, x)$. ReLU usually works better in practice than other activation functions, such as sigmod function. It is one of the most commonly used activation functions in the neural network.

The model is trained using back-propagation for Adam Optimizer [46], a stochastic gradient descent that automatically adapts the learning rate. The optimizer works on minimizing the loss function. We use the mean cross-entropy as a loss function in our experiment. The model is also trained with a batch setting, which is not reflected in the layers described above.

4.3.2 Hyper-Parameters Tuning

Hyper-Parameters tuning is always a very challenging problem in a deep learning task. Normally, the accuracy rate of classification tasks could be dramatically affected by the setting of hyper-parameters. In order to achieve the best classification accuracy, multiple methods were tested, such as random search [47], and grid search tested afterward. After carefully studies and experiments, we set up the hyper-parameters for training as follows. We set the Dropout rate to 0.5 in each dropout layer. Learning rate, another important parameter, determines how fast we move toward the optimal weights in our network. If it is too large, it will skip optimal values. On the other hand, if it is too small, it will take too much time to converge to the optimal values, and it may get stuck in local minima. Fortunately, Adam Optimizer provides a very detailed and flexible API to set up the initial rate and decay rate. We finish our training by setting the initial learning rate started as 1e - 3 (0.001), beta1 as 0.9, beta2 as 0.999 and decay rate as 0.1.

4.3.3 Handling Imbalanced Data

The imbalance is common in the real world. Most classification data sets do not have an exactly equal number of instances in each class, while a small difference often does not matter. But when there is a modest class imbalance like 4:1 in the dataset or above, it can cause problems [48]. Sometimes imbalance is not just common, it is expected. For example, in datasets like those that characterize fraudulent transactions are imbalanced. The vast majority of the transactions will be in the Not-Fraud class and a very small minority will be in the Fraud class. We had the same situation in our own dataset, most of the subscribers should be legal and only very rare events could be caused by the malicious attacker.

There are several methods available for dealing with the imbalance situation: *Change Performance Metrics* Since we are dealing with an extremely imbalanced dataset, the accuracy rate is not enough to accurately evaluate the performance model. F-score Matrix [29] was applied in our experiment. It contains three major metrics as below [48]:

- Precision: the number of true positives divided by all positive predictions. Precision is also called Positive Predictive Value. It is a measure of a classifier's exactness. Low precision indicates a high number of false positives.
- Recall: the number of true positives divided by the number of positive values in the test data. Recall is also called Sensitivity or the True Positive Rate. It is a measurement of classifier's completeness. Low recall indicates a high number of false negatives.
- F-score: the weighted average of precision and recall

Oversample Minority Class. Oversampling can be defined as adding more copies of the minority class. In our experiment, we implement oversampling by constructing the batch with a solid number of periodically and extremely active types, which was randomly chosen each time.

4.3.4 Quantify the Co-Residency Risk

In our framework design, the quantification component takes the output from the classification component and applies it into a pre-defined quantify function. In our experiment, for simplicity, we use a softmax activation function in

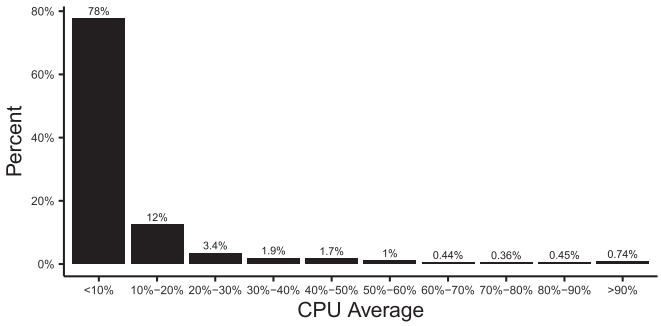

Fig. 3. Average CPU utilization distribution among VMs.

the classification component and directly output the prediction of the probability of each category. The probability rate of the normal category will be used to calculate the co-resident risk. In other words, we believe it represents the derivation rate from normal behavior patterns.

5 Experimental Evaluation

We conduct our experiments on a Dell Precision Tower T5810 Workstation. The Dell Workstation has an Intel Xeon E5-1620, 32G RAM, and Nvidia Quadro P5000 Graphic Card for GPU acceleration. GPU acceleration is applied in our Model Training procedure and it shortens the training time dramatically.

5.1 Insight of Azure Dataset

We use real-world large-scale dataset, Azure public Dataset, in our experiments. Azure Public Dataset [25] provides two portion of the Virtual Machine (VM) workloads of Microsoft Azure collected in 2019 and 2017. This dataset contains over 12,000 service subscribers with their over 5 million VMs and the corresponding 3.1 billion CPU utilization record sampled every five minutes over one month. The total size of the dataset is over 500GBs. To finish the statistical task for the proposed feature metrics, we divide the task into multiple portions of data in each temporary step. We summarize the observations as follows:

A - Overall Active Rate. Fig. 3, shows the average CPU utilization distribution among VMs. The CPU utilization indicates

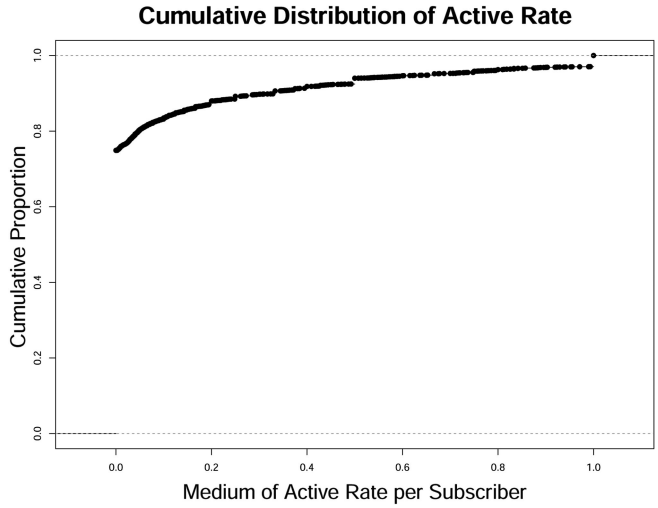

Fig. 4. Cumulative distribution of active rate per tenant.

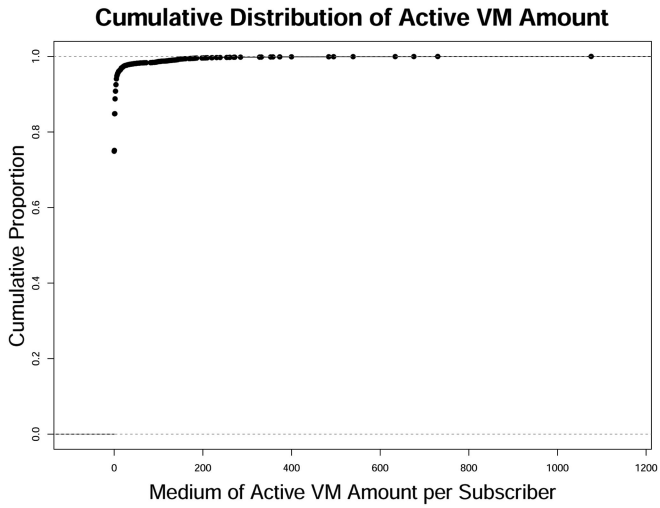

Fig. 5. Cumulative distribution of average active VM amount per tenant.

the overall active rate for tall subscribers. We noticed that more than 90 percent VMs' CPU utilization rate is under 15 percent, which means most of VMs stay under very low workload status.

The cumulative distribution of active rate per subscriber is shown in Fig. 4. Generally speaking, over 80 percent subscribers have less than 10 percent active rate.

W - Average Active VM Amount. The cumulative distribution of average active VM amount per subscriber is shown in Fig. 5. Combine it with feature A, we could filter out the type of extremely active subscriber.

I - Median of Average CPU Utilization Rate. Fig. 6 shows the cumulative distribution of CPU utilization per subscriber. Over 90 percent subscribers have been identified as inactive type in Fig. 8

Based on above feature metrics, we obtained a draft profile of the service subscribers. In the next section, we will continue to use our proposed feature metrics to cluster the subscribers.

5.2 Clustering of Subscribers

Azure Dataset contains over 12,000 service subscribers, therefore, we clustered them into several candidate groups for the future study.

5.2.1 Aspect From Activeness Rate

Based on observation of Fig. 7, around 65 percent subscribers create one VM, and 25 percent subscribers created less than five VMs in their lifetime.

Combined with the cumulative distribution of feature I, we could conclude that most service subscribers should only maintain a small amount VMs at inactive level.

To verify our conclusion, we did the active rate curve in all timestamps for all subscribers. According to our observations on all curve diagram, we define the subscribers as following categories based on the activeness rate:

- Inactive Subscriber: In Fig. 8, the very top section is a typical curve for the inactive type subscriber we defined, and over 80 percent of subscribers fall into this category. In this category, a very small amount

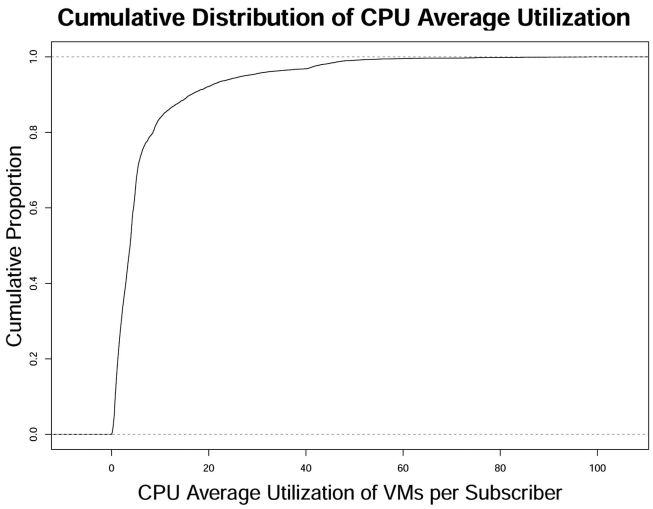

Fig. 6. Cumulative distribution of CPU average utilization per tenant.

of VMs was created by the subscriber, and most of the time the VM stay inactive.

- Periodically active subscriber: The middle portion of Fig. 8 is a typical curve for the Periodically active subscriber we defined. Multiple VMs are created by the subscriber and the activeness level is shifted dramatically over the subscriber's whole lifetime.
- Extremely active subscriber: The typical curve for this type of subscriber is located at the bottom of Fig. 8. This type of subscriber maintains an extremely high activeness rate over their whole lifetime.

We performed multi-step testing in our experiment. Initially, we use DBSCAN to cluster subscribers. And then, we tested the MinPts ranging from 5 to 50 since we believe most data should comes from the normal service subscriber. We expect that there would have at least one clustering output contains a major portion of the subscribers. After careful testings and studies, we set the MinPts to 10 and ε to 35 in our experiment. In the further step, we manually verify all user's activeness level curves among the whole lifetime. Three major categories were separated, shown in Fig. 8. Thus, if the attacker tries to pretend a normal user, the cost

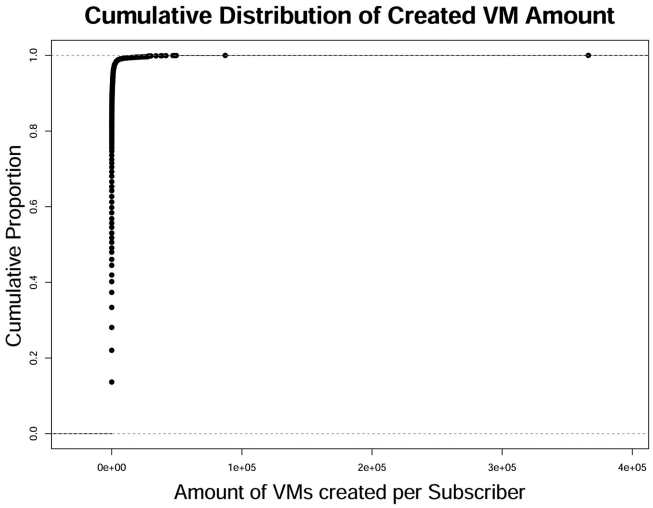

Fig. 7. Cumulative distribution of created VMs amount per tenant.

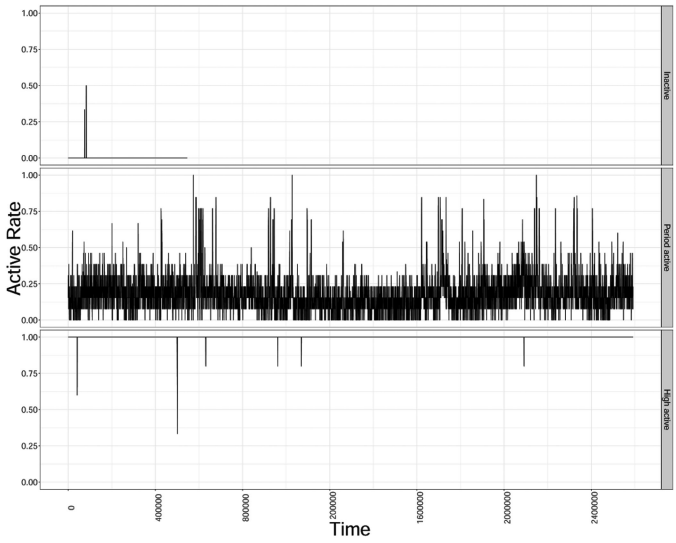

Fig. 8. Subscriber categorized with activeness level.

will be high. We demonstrated the coefficient relationship between extremely active subscribers and others in Fig. 9.

We demonstrated our clustering results in Figs. 10 and 11. Most of the service subscribers were clustered into one category, which fits to our expectations. More important, 78 subscribers were identified as the outlier. By checking with their detail active rate curve, we believe their behavior pattern would be the closest to potential high-risk ones.

5.3 Training and Evaluating Deep Neural Network

After we initially clustered service subscribers, we manually labeled them into three major categories: normal subscriber, periodically active subscriber, and extremely active subscriber, shown in Table 1:

Oversampling Minority Class. 78 subscribers are located into extremely active category, and only 379 subscribers are located into periodically active category. Comparing with a total of 11573 normal subscriber, we are facing an extremely imbalanced dataset. We divided the whole dataset into two parts: Training set (90 percent) and Test set (10 percent). For each training session of our model, we did 100 batches in each epoch and we finished a total of 100 epochs. To handle this imbalance, we have to oversample these two minor classes. We kept choosing 300 unrepeated from the normal type, 100 from periodically active type, and 30 from extremely active type, shuffling them to form a batch of the training set. The training set will be used in one step. Since all training tasks were finished off-line, the inference procedure for new subscriber data only needs few seconds to generate the final prediction.

We reserved 10 percent of total service subscribers' data for test purposes. The data is new and it has never been used to optimize our model in the training procedure. We only recorded the checkpoint when the accuracy rate of the test dataset has been improved. We noticed the loss value was shifted several times in our experiment in Fig. 12. Fig. 13 display the accuracy rate of the test dataset.

Even we tried to use dropout to avoid model overfitting issue, we still could observe that when training accuracy keeps rising from 94 to 96 percent, the accuracy rate for test set actually declined from 97.9 to 95 percent. Considering

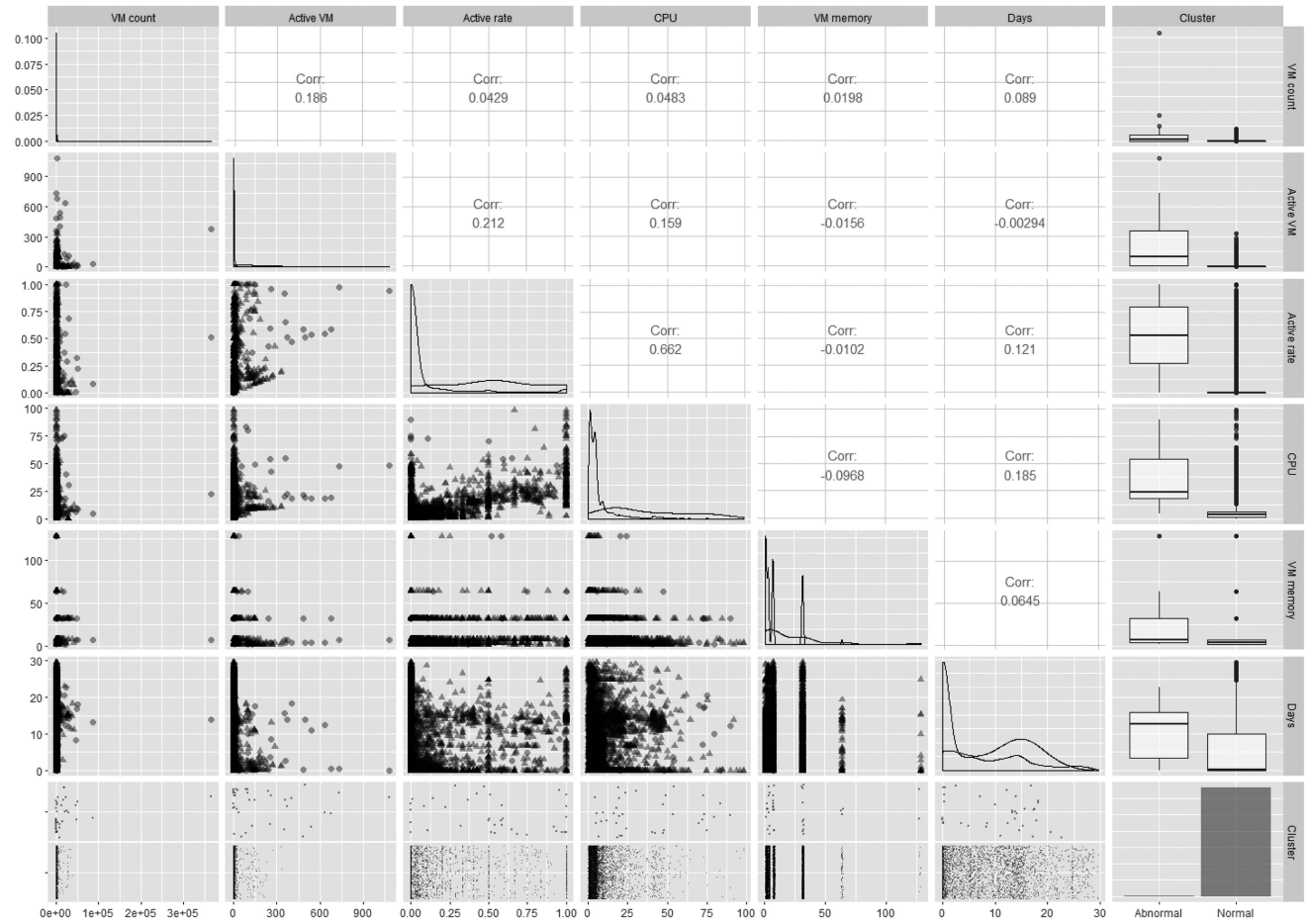

Fig. 9. Cluster pairing.

an extreme imbalanced dataset was processed here, we need to use several other performance metrics to further evaluate our model.

When the accuracy rate is above 98 percent, we calculated out all the recall, precision and F-score for all three categories as in Table 2:

The test dataset only contains eight extremely active subscribers, which we believe should be able to simulate the practical environment since in the real world, the appearance of an extremely active pattern would be a very rare case. This extreme imbalance also existed in the training dataset. Table 2 demonstrates the best F-score result for test dataset with total 97.8 percent accuracy rate for test.

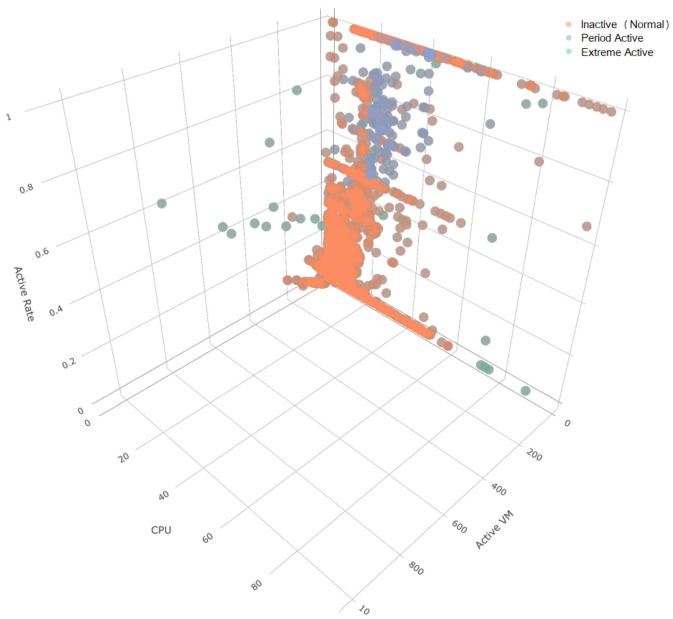

Fig. 10. Cluster result.

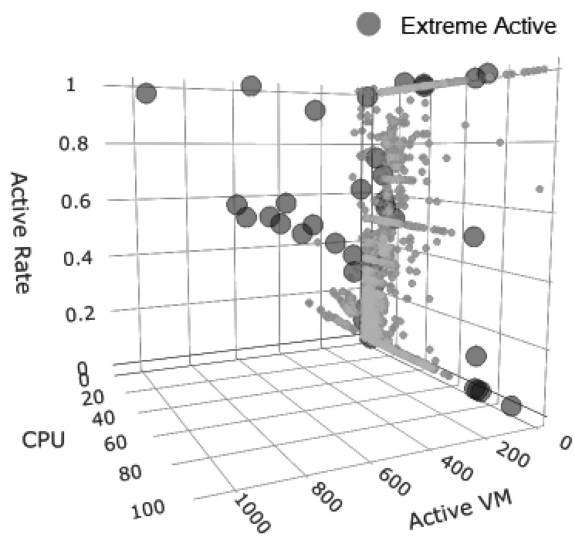

Fig. 11. Cluster result (extremely active).

TABLE 1
Subscribers in Each Category

Type	Amount
Normal	11573
Periodically Active	379
Extremely Active	78

TABLE 2
Best F-Score Result

Category	Recall	Precision	F-Score
Normal	0.987	0.994	0.99
Periodically Active	0.85	0.708	0.773
Extremely Active	0.636	0.583	0.609

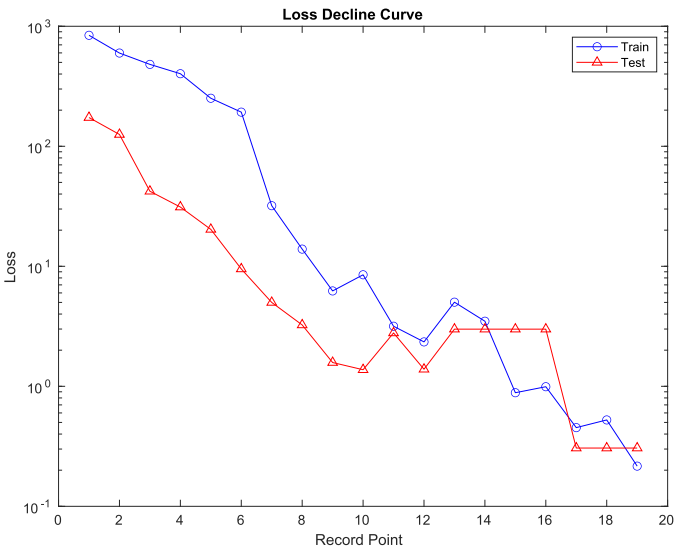

Fig. 12. Cross-entropy loss curve.

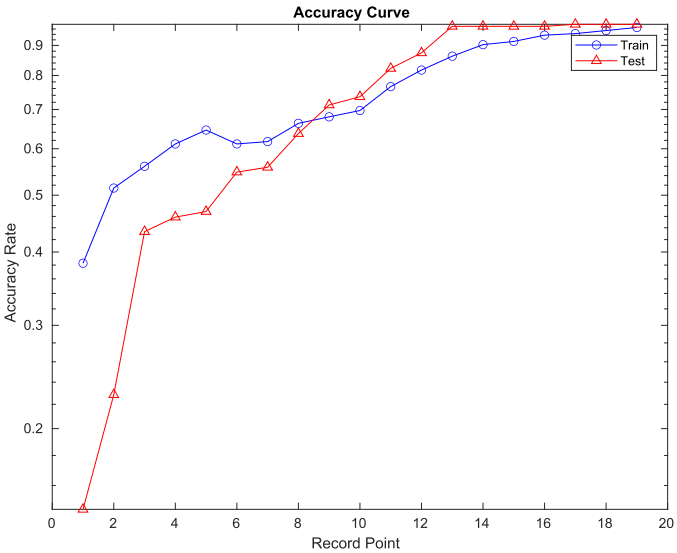

Fig. 13. Training accuracy curve.

According to the F-score matrix from Fig. 14, the performance of classification component on extremely active pattern was affected by the extreme data imbalance in our experiment. There are two possible explanations for this issue. First, even we applied multiple methods to handle the imbalanced training dataset, such as oversampling the minority category, the training of our model was still under noticeable affection of data imbalance. In the meanwhile, this is the major defect of the neural network, it has its limitation to handle unseen data. Another possible reason is that the initial clustering was not accurate enough. Since

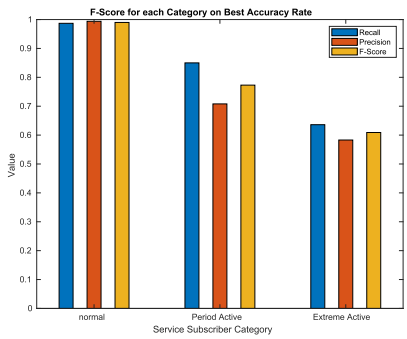

Fig. 14. F-score matrix.

there are only 78 service subscribers were labeled into the extremely active category, it is possible that too much detailed feature information was lost during the abstraction procedure of calculating feature metrics. To fix this issue, we proposed to construct a Convolutional Neural Network submodule for handling the detail active rate curve per each subscriber. In that case, this manual label procedure would be replaced by an automatic component, and a lot of improvement could be achieved in our future work. Also on the other hand, as we discussed above, this issue is expected. Since we only used the probability of inactive type in the reference procedure to calculate the risk level, which will not be affected by this issue. At the same time, our feature metrics can be used as good baseline to monitor tenant's online behavior. Although the sample time window is five minutes in Azure dataset, it is too long to the target of monitoring individual VM. Considering the trade of between expense added and more detailed sample info, this is will be tough choice to service provider. Thus we plan to extend our work to monitor tenant's behavior, which we believe it is feasible and much more reliable with current dataset. In our future work, we will construct one sub-module to analyze the real-time activeness level for individual tenant.

6 Conclusion and Future Work

In this paper, we proposed a comprehensive framework, using deep learning, to quantify the co-residency risks in a cloud. We mainly focused on the Infrastructure as a Service (IaaS) type of cloud services. Our work is based on Microsoft Azure virtual machine tracing dataset that is a large scale and real-world dataset. In this paper, we proposed a dynamic adaptive framework to better quantify the co-residency risks. A set of feature metrics was proposed to accurately profile the behavior patterns of normal service subscribers in the cloud. Based on the analysis of Azure dataset, we used DBSCAN algorithm to cluster into several groups of service subscribers and manually labeled them into three major categories: Inactive, Periodically Active, and Extremely Active.

With the labeled datasets, a classification component was trained to identify new datasets and the output was used in the quantification component to quantify the co-residency risk rate. Based on our experimental results, our classification component demonstrated robustness to new data. It also achieved an accuracy of 98 percent for test dataset and its performance was verified by examination the F-score Matrix.

In our future work, we will extend our approach in two directions. First of all, as we mentioned above, detailed information about the service subscriber were lost during the data abstract procedure. Also, we observed that extremely active pattern is a very rare event in all datasets, and this extreme imbalance highly affects the classification task. By analyzing the F-score matrix, there is still a lot of space for improvements to more accurately identify those rare events. To conquer the challenge, we propose to build another sub-module in the classification component to handle the detail feature diagram directly. In this way, we can more accurately profile normal subscriber's behavior patterns. Benefited from the improvement, we will be able to achieve a better classification at the same time. Second, our framework was trained and tested in one particular dataset. The general compatibility of our proposed framework may be questioned by a new dataset. With the newly collected dataset, a well-trained classification component should be able to adapt to a new working environment. With appropriate transfer training, We could extend applying our framework to test the general adaption capability. There is also another possible research direction, if we can successfully profile normal subscriber pattern, it can be used to predict specific subscriber's future workload request or detection of abnormal appearance.

Acknowledgments

This project was supported in part by ARO Grant W911NF-15-1-026 and NSF Grants under CNS-163441 CNS-1524462, CNS-1422355 and CREST Grant HRD-1736209. The authors highly appreciate the support provided by Samsung SARC Research Center at Austin. They would like to thank anonymous reviewers for their insightful comments.

References

[1] H. Hlavacs, T. Treutner, J. Gelas, L. Lefevre, and A. Orgerie, "Energy consumption side-channel attack at virtual machines in a cloud," in *Proc. IEEE 9th Int. Conf. Dependable Autonomic Secure Comput.*, 2011, pp. 605–612.

[2] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-VM side channels and their use to extract private keys," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 305–316. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382230

[3] S. Jin, J. Ahn, S. Cha, and J. Huh, "Architectural support for secure virtualization under a vulnerable hypervisor," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2011, pp. 272–283.

[4] J. Shi, X. Song, H. Chen, and B. Zang, "Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring," in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw. Workshops*, 2011, pp. 194–199.

[5] V. Varadarajan, T. Ristenpart, and M. Swift, "Scheduler-based defenses against cross-VM side-channels," in *Proc. 23rd USENIX Secur. Symp.*, 2014, pp. 687–702. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/varadarajan

[6] V. Sindhwani and S. S. Keerthi, "Large scale semi-supervised linear SVMs," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2006, pp. 477–484. [Online]. Available: http://doi.acm.org/10.1145/1148170.1148253

[7] S. Rethishkumar and R. Vijayakumar, "Defender vs attacker security game model for an optimal solution to co-resident dos attack in cloud," in *Intelligent Communication Technologies and Virtual Mobile Networks*, S. Balaji, Á. Rocha, and Y.-N. Chung, Eds. Cham, Switzerland: Springer, 2020, pp. 527–537.

[8] P. Barham *et al.*, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, Oct. 2003. [Online]. Available: http://doi.acm.org/10.1145/1165389.945462

[9] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 199–212. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653687

[10] Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter, "Homealone: Co-residency detection in the cloud via side-channel analysis," in *Proc. IEEE Symp. Secur. Privacy*, 2011, pp. 313–328.

[11] S. Yu, X. Gui, and J. Lin, "An approach with two-stage mode to detect cache-based side channel attacks," in *Proc. Int. Conf. Inf. Netw.*, 2013, pp. 186–191.

[12] W. Koch, "Libgcrypt," [Online]. Available: https://gnupg.org/software/libgcrypt/index.html

[13] A. Aviram, S. Hu, B. Ford, and R. Gummadi, "Determinating timing channels in compute clouds," in *Proc. ACM Workshop Cloud Comput. Secur. Workshop*, 2010, pp. 103–108. [Online]. Available: http://doi.acm.org/10.1145/1866835.1866854

[14] J. Szefer, E. Keller, R. B. Lee, and J. Rexford, "Eliminating the hypervisor attack surface for a more secure cloud," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, 2011, pp. 401–412. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046754

[15] B. C. Vattikonda, S. Das, and H. Shacham, "Eliminating fine grained timers in Xen," in *Proc. 3rd ACM Workshop Cloud Comput. Secur. Workshop*, 2011, pp. 41–46. [Online]. Available: http://doi.acm.org/10.1145/2046660.2046671

[16] J. Wu, L. Ding, Y. Lin, N. Min-Allah, and Y. Wang, "XenPump: A new method to mitigate timing channel in cloud computing," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, 2012, pp. 678–685.

[17] T. Kim, M. Peinado, and G. Mainar-Ruiz, "STEALTHMEM: System-level protection against cache-based side channel attacks in the cloud," in *Proc. 21st USENIX Secur. Symp.*, 2012, pp. 189–204. [Online]. Available: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/kim

[18] Y. Zhang and M. K. Reiter, "Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 827–838. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516741

[19] S. Sundareswaran and A. C. Squcciarini, "Detecting malicious co-resident virtual machines indulging in load-based attacks," in *Information and Communications Security*, S. Qing, J. Zhou, and D. Liu, Eds. Cham, Switzerland: Springer, 2013, pp. 113–124.

[20] J. Han, W. Zang, L. Liu, S. Chen, and M. Yu, "Risk-aware multi-objective optimized virtual machine placement in the cloud," *J. Comput. Secur.*, vol. 26, no. 5, pp. 707–730, Aug. 2018.

[21] Y. Han, J. Chan, T. Alpcan, and C. Leckie, "Using virtual machine allocation policies to defend against co-resident attacks in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 1, pp. 95–108, Jan./Feb. 2017.

[22] Y. Han, T. Alpcan, J. Chan, C. Leckie, and B. I. P. Rubinstein, "A game theoretical approach to defend against co-resident attacks in cloud computing: Preventing co-residence using semi-supervised learning," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 3, pp. 556–570, Mar. 2016.

[23] M. M. Hasan and M. A. Rahman, "Protection by detection: A signaling game approach to mitigate co-resident attacks in cloud," in *Proc. IEEE 10th Int. Conf. Cloud Comput.*, 2017, pp. 552–559.

[24] H. Wu and W. Wang, "A game theory based collaborative security detection method for Internet of Things systems," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 6, pp. 1432–1445, Jun. 2018.

[25] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proc. 26th Symp. Operating Syst. Princ.*, 2017, pp. 153–167. [Online]. Available: http://doi.acm.org/10.1145/3132747.3132772

[26] G. Amvrosiadis, J. W. Park, G. R. Ganger, G. A. Gibson, E. Baseman, and N. DeBardeleben, "On the diversity of cluster workloads and its impact on research results," in *Proc. USENIX Conf. Usenix Annu. Tech. Conf.*, 2018, pp. 533–546. [Online]. Available: http://dl.acm.org/citation.cfm?id=3277355.3277407
[27] R. Shi, Y. Gan, and Y. Wang, "Evaluating scalability bottlenecks by workload extrapolation," in *Proc. IEEE 26th Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, 2018, pp. 333–347.
[28] J. Han, W. Zang, S. Chen, and M. Yu, "Reducing security risks of clouds through virtual machine placement," in *Data and Applications Security and Privacy XXXI*, G. Livraga and S. Zhu, Eds. Cham, Switzerland: Springer, 2017, pp. 275–292.
[29] Y. Sasaki *et al.*, "The truth of the F-measure," 2007.
[30] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 199–212. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653687
[31] D. Gonzales, J. M. Kaplan, E. Saltzman, Z. Winkelman, and D. Woods, "Cloud-trust-a security assessment model for infrastructure as a service (IaaS) clouds," *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 523–536, Jul.–Sep. 2017.
[32] K. Z. Bijon, R. Krishnan, and R. Sandhu, "A formal model for isolation management in cloud infrastructure-as-a-service," in *Network and System Security*, M. H. Au, B. Carminati, and C.-C. J. Kuo, Eds. Cham, Switzerland: Springer, 2014, pp. 41–53.
[33] K. Bijon, R. Krishnan, and R. Sandhu, "Mitigating multi-tenancy risks in IaaS cloud through constraints-driven virtual resource scheduling," in *Proc. 20th ACM Symp. Access Control Models Technol.*, 2015, pp. 63–74. [Online]. Available: http://doi.acm.org/10.1145/2752952.2752964
[34] K. Bijon, R. Krishnan, and R. Sandhu, "Virtual resource orchestration constraints in cloud infrastructure as a service," in *Proc. 5th ACM Conf. Data Appl. Secur. Privacy*, 2015, pp. 183–194. [Online]. Available: http://doi.acm.org/10.1145/2699026.2699112
[35] S. Rethishkumar and R. Vijayakumar, "Two-player security game approach based co-resident dos attack defence mechanism for cloud computing," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 2, no. 4, pp. 552–559, Jun. 2017.
[36] T. Madi *et al.*, "Quantic: Distance metrics for evaluating multi-tenancy threats in public cloud," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci.*, 2018, pp. 163–170.
[37] N. Alhebaishi, L. Wang, and S. Jajodia, "Modeling and mitigating security threats in network functions virtualization (NFV)," in *Data and Applications Security and Privacy XXXIV*, A. Singhal and J. Vaidya, Eds. Cham, Switzerland: Springer, 2020, pp. 3–23.
[38] C. Zhang, V. Gupta, and A. A. Chien, "Information models: Creating and preserving value in volatile cloud resources," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2019, pp. 45–55.
[39] M. Li, Y. Zhang, K. Bai, W. Zang, M. Yu, and X. He, "Improving cloud survivability through dependency based virtual machine placement," pp. 321–326, 2012.
[40] X. Yuchi and S. Shetty, "Enabling security-aware virtual machine placement in IaaS clouds," in *Proc. IEEE Military Commun. Conf.*, 2015, pp. 1554–1559.
[41] V. Varadarajan, Y. Zhang, T. Ristenpart, and M. M. Swift, "A placement vulnerability study in multi-tenant public clouds," in *Proc. USENIX Conf. Secur. Symp.*, 2015, pp. 913–928.
[42] W. Zhang *et al.*, "A comprehensive study of co-residence threat in multi-tenant public PaaS clouds," in *Information and Communications Security*. Berlin, Germany: Springer, 2016, pp. 361–375.
[43] Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang, *Incentive Compatible Moving Target Defense against VM-Colocation Attacks in Clouds*. Berlin, Germany: Springer, 2012, pp. 388–399.
[44] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580
[45] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 1058–1066. [Online]. Available: http://proceedings.mlr.press/v28/wan13.html
[46] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2014.
[47] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 281–305, Feb. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2503308.2188395
[48] T. Boyle, "Dealing with imbalanced data," [Online]. Available: https://https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18

Jin Han received the PhD degree in computer science from the University of Texas at San Antonio. He is working for Samsung SARC research center at Austin, TX. His research has focused on compiler optimization for GPU on mobile device and efficiency improvement of parallel computing. His research interests include deep learning, computer vision in mobile device, and image quality metrics.

Wanyu Zang received the PhD degree in computer science from Nanjing University, China. She is an assistant professor of Roosevelt University. Her research interests include computer networks and network security, cloud computing and security. Her research has been funded by ARO and NSF.

Meng Yu (Member, IEEE) received the PhD degree in computer science from Nanjing University, China. He is a Robert Miner Endowed chair professor of Roosevelt University. His research interests include systems and network security, cloud computing, virtualization and security. His research has been funded by NSF. He is a member of ACM.

Ravi Sandhu (Fellow, IEEE) is the founding executive director with the Institute for Cyber Security, University of Texas at San Antonio, and holds an Endowed chair. He is an inventor on 29 patents. He is a past editor-in-chief of the *IEEE Transactions on Dependable and Secure Computing*, a past founding editor-in-chief of the *ACM Transactions on Information and System Security*, and a past chair of ACM SIGSAC. He founded ACM CCS, SACMAT, and CODASPY, and has been a leader in numerous other security conferences. His research has focused on security models and architectures, including the seminal role-based access control model. His papers have accumulated more than 26, 000 Google Scholar citations, including more than 6, 400 citations for the RBAC96 paper. He is a fellow of ACM and AAAS.