Securing Edge-based IoT Networks with Semi-Supervised GANs

Santosh Kumar Nukavarapu
Virginia Commonwealth University
Richmond, USA
nukavarapuskk@mymail.vcu.edu

Tamer Nadeem
Virginia Commonwealth University
Richmond, USA
tnadeem@vcu.edu

***Abstract*—With the phenomenal growth of IoT devices and their exponentially increasing applications at the edge of the network, it has become imminent to provide secure, flexible, and programmable network services to support user privacy, security, and quality of service. However, such services can be only be enabled and effectively applied when the edge systems automatically identify these devices. Existing IoT classification systems are based on supervised training methods that require labeled data and manual feature extraction. Such approaches suffer from many challenges such as privacy, labeling efforts, and adaptability to new devices. This paper develops a multi-stage multi-class classifier based on semi-supervised generative adversarial networks that perform automatic feature extraction with minimal labeled data. The classifier can identify devices with 96% accuracy when only 3% of the training data is labeled. Moreover, the classifier can infer the device type (IoT, Non-IoT, and anomaly) of any new device correctly with 90% accuracy. We also show how our model can support novelty detection, such as zero-day malware attacks.**

***Index Terms*—Edge Computing, Generative Adversarial Networks, Internet Of Things, IoT Privacy, IoT Security, Semi Supervised GAN**

I. Introduction

The rapid growth of IoT devices, their use cases, and autonomous communication behavior have created new challenges for the edge infrastructure. A number of recent works have shown new network-based security and privacy attacks [5]. Therefore, the edge-based infrastructure should provide secure and reliable services for IoT devices. For example, a network-based security service should effectively identify and isolate malware traffic as soon it appears in the network. Given these requirements, the edge-based systems should efficiently and rapidly identify any new devices joining the network. Moreover, such device identification should support different scenarios, such as identifying the instances of known devices or malware, identifying the type of any new device (unknown), and anomalies. However, most of the current IoT classification approaches are based on supervised machine learning models [14], [15] and suffer from many challenges such as labeling efforts, privacy challenges with data collection, and difficulty identifying new devices that were not part of the training.

Generative adversarial networks (GANs) [9] have shown to model unknown complex distributions well and generate synthetic data close to the training data's true distribution. Given this capability, many recent extensions of GAN have been proposed for the Generative and classification Tasks. Very recently, semi-supervised GAN (SGAN) [17], [20], an extension to the GAN model, has shown excellent results in image classification, spam detection, and medical data where the availability of labeled data is very limited. The SGAN model consists of a multi-class classifier that can perform automatic feature extraction with very limited labeled data. Moreover, in the training of SGAN, the classifier is exposed to the additional synthetic data points with noise that give the classifier better generalization capability. Several features of the SGAN model, such as multi-class classification, automatic feature extraction, and the capture of the hidden distribution of different classes, are aligned with the requirements and objectives of building a real-world IoT classifier. Motivated by the above observations, in this paper, we develop IoT classifiers for real-world scenarios using SGAN. We summarize the contributions of this paper as follows:

- We design and develop *iKnight*, a Semi-supervised GAN-based multistage multi-class classifier for classifying IoT devices for different scenarios such as known device, unknown device, malware, and anomaly with very few labeled data.
- We evaluate the features of *iKnight* using real IoT devices and malware network traffic datasets.

II. Related Work & Background

A. Related Work

In recent years, there has been a significant interest in IoT device fingerprinting research [6], [8], [15] and activity inference analysis [3], [5]. While some works proposed supervised classifiers based on features such as device-dependent features for device identification [15], others used time series based features [3] and flow level metadata [5] to detect activities of IoT devices. Recently, authors in [14] used CNN and RNN based deep learning classifier for network classification, but this technique is also based on supervised learning that is highly dependent on labeled data availability. Recently, researchers have proposed a unsupervised clustering approach for IoT device classification [22]. Also, in [18], authors utilize IoT based raw network packets as their features for classifying IoT devices using autoencoders and bayesian modeling. We believe

This material is based upon work partially supported by the US National Science Foundation under Grant No. CNS-1764185.

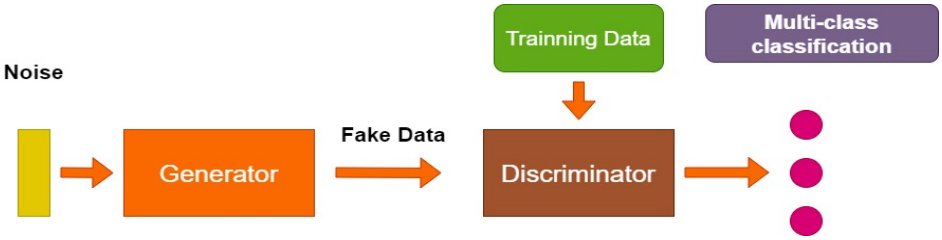

Fig. 1: Semi-supervised GAN (SGAN) Architecture

that multi-class classifiers are much more useful for real-world scenarios given their ease of deployment, maintenance, and scalability.

A very recent work has used a semi-supervised based approach towards classifying network applications belonging to QUIC, VPN, and Non-VPN datasets [11]. However, it is a well-known fact that IoT based network traffic is very different from traditional network traffic. Furthermore, IoT environments have a number of unique challenges, such as malware infections and anomaly detection, which are addressed in our work.

B. Semi-supervised GAN

The amount of labeled data has a significant impact on the accuracy of classifiers in machine learning models. However, many real-world problems have fewer labeled data and significant labeling efforts would be expected. Some researchers have recently proposed semi-supervised learning using GANs, generally known as semi-supervised GAN (SGAN) [17], [20]. The SGAN has shown promising results for image classification problems with sparsely labeled data. The SGAN model, as shown in Figure 1, consists of a generator, discriminator, and an additional classifier model whose weights are shared with the discriminator [20]. Like the vanilla GAN model, the Generator and Discriminator are trained in a min-max adversarial game where the Generator's objective is to generate fake data and the objective of the discriminator is to identify between fake and real data. However, in SGAN, in addition to this, the classifier model learns to classify the K classes of the training data.

In SGAN, the discriminator is trained on unlabeled data (from training) and the fake data from Generator, and the classifier is trained on a very few labeled data. The rationale behind this is that the feature representations learned from the discriminator, in order to identify the real data distribution, improve the classifier efficiency. Furthermore, the min-max game-based adversarial training will force the Generator to create synthetic samples that create additional data points to the classifier; thus, improving its efficiency [17]. In SGAN training, both the discriminator and the classifier are exposed to real labeled data, real unlabeled data, and fake data from the Generator. This approach generalizes better than other semi-supervised methods that do not have access to additional synthetic data points during training.

III. iKnight Framework

In this paper, we design and implement *iKnight*, an IoT network classifier for Edge-based systems. Figure 2 shows an overview of the *iKnight* framework that consists of an Encoding Engine and Device Discovery Engine. In *iKnight*, the encoding engine encodes the new device's flow and passes the encoded flow to the device discovery engine. The device discovery engine uses a multi-stage multi-class SGAN based classifier to identify a new device class (device name) or its type (IoT vs Non-IoT). We discuss below the implementations of these two components.

A. Encoding Engine

The *iKnight* encoding engine encodes the raw network flow packets of new devices using an encoding scheme. An encoding scheme is a process of mapping the raw network packets to a feature matrix that we define as a flow encoded matrix, which is sent to the device discovery engine for device type identification. The multi-stage multi-class SGAN model in the device discovery engine uses these raw encoded features to extract the hidden features and identify the device type automatically. An efficient encoding scheme is a significant factor for the efficiency of the device type identification. In *iKnight*, we use two encoding schemes; flow-only and flow-with-inter-arrival-time schemes in which we discuss their implementations below.

Flow Encoding Scheme: In the flow-only encoding scheme, we use the raw network byte-stream representation of the packets as features. We first convert each byte-stream representation of the packet of the flow into their equivalent hexadecimal integer value. This hexadecimal flow array is then encoded as a three-dimensional image matrix; a flow encoded NumPy matrix with height, width, and depth. Each row of the flow encoded matrix consists of a packet hex stream arranged as a 2-dimensional array with a size of 56*56 (accommodates the maximum 1500 bytes of a MAC packet). The packet rows are ordered in the flow matrix as per their arrival sequence. This encoding scheme is similar to pixel intensity arrangement in a non-gray image matrix where each row is a 2D matrix representing the different channels of the image. The flow encoded matrix's depth is the number of packets in the flow, a configurable parameter, which we find as the first 5 packets based on experiments. If the flow has less than 5 packets, then the rest of the packets rows of the flow encoded matrix are appended with zeros, an approach similar to [18].

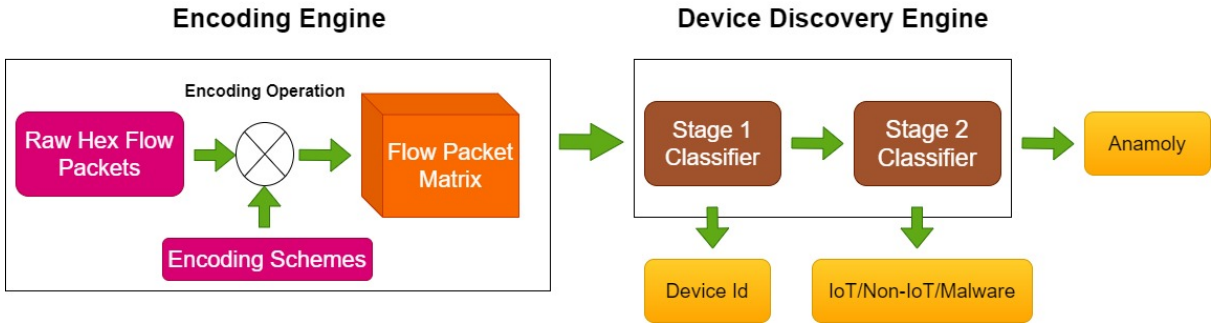

Fig. 2: *iKnight* Architecture

Flow and Inter-arrival Encoding Scheme: Similar to Flow encoding scheme, but using use of both the raw network byte stream and the packets' inter-arrival time for the flow-with-inter-arrival-time scheme. We arrange the 2D packets in the flow encoded matrix as rows similar to the flow-only scheme. However, we append an additional row at the end containing the inter-arrival times of the packets in their arrival sequence. Moreover, we convert the inter-arrival values to a hex decimal value similar to the packet data. We discuss the impact of these encoding schemes on device type inference in the evaluation section.

B. Device Discovery Engine

The *iKnight* Device Discovery Engine is a multi-stage classifier that can identify the device under different scenarios, such as identifying the device ID for a known device, and the device type for an unknown device. We define a device as known if it is available to the Classifier during training and unknown otherwise. In *iKnight*, we take a two-stage approach for device discovery. In the first stage, *iKnight* uniquely identifies all known devices by classifying them into the corresponding classes. However, for any unknown device, *iKnight* classifies it as unknown class, unlike other multi-class classifiers that will incorrectly identify an unknown class as one of the known classes. In the second stage, *iKnight* infers the device type of the unknown class as either an IoT, non-IoT, malware, or unknown. However, we refer to the unknown class in stage #2 as an anomaly, given it does not fall into any known network traffic classes. This type of inference at stage #2 will help the network administrators to identify the types of unknown devices and, consequently, to adapt network policies. Some of these policies can give higher network priority to specific device types (IoT vs. Non-IoT), enforce any known anti-malware patches for malware, and issue alerts in case of an anomaly for immediate actions. We use data sets from publicly available IoT and malware datasets to implement stage #1 and stage #2 classifiers. Then we perform data pre-processing and training on both classifiers as per their objectives using an SGAN CNN model. We discuss below the datasets, training procedure, and the model architecture for both stages in detail.

Publicly available IoT and Malware Datasets: In both stages, we utilized semi-supervised GAN in our classification models. In our experiments, we use the publicly available UNSW [21] and IoT-23 datasets [4]. We use the network traffic traces from different classes of IoT devices and non-IoT devices from the UNSW dataset as shown in Table I. The IoT-23 consists of labeled IoT malware network traffic flows from twenty different malware binary files (each different class) executed on a raspberry PI device in a controlled environment with flows labeled as either benign or malicious. We select 60,000 flows from the UNSW dataset and 10000 malicious flows from four different malware classes, Mirai, Trojan, Hakai, and Torii, for our experiments. For the stage #1 model, we train the model with twenty IoT devices, while for the stage #2 model, we use the combined network flows of twenty IoT devices, four Non-IoT devices, and four malware classes.

TABLE I: The list of IoT and non-IoT devices used in experiments

Device Name	Device Type
Smart Things	Hub
Amazon Echo	Speaker
iHome	Speaker
Triby Speaker	Speaker
Netatmo Welcome	Camera
TP-Link Day Night Cloud Camera	Camera
Samsung SmartCam	Camera
Dropcam	Camera
Insteon Camera	Camera
Withings Smart Baby Monitor	Camera
Nest Dropcam	Camera
Belkin Wemo Switch	Acutator
TP-Link Smart Plug	Acutator
Light Bulbs LiFX Smart Bulb	Acutator
NEST Protect Smoke Alarm	Sensor
Netatmo Weather Station	Sensor
Withings Smart Scale	Sensor
Blipcare Blood Pressure Meter	Sensor
Withings Aura smart Sleep	Sensor
PIX-Star Photo-frame	Digital Frame
Laptop	Non-IoT
Android Phone	Non-IoT
iPhone	Non-IoT
Samsumg Galaxy Tab	Non-IoT

Data Pre-Prepossessing: Figure 3 shows the *iKnight* stage #1 and stage #2 model training pipeline. We first extract flows from the raw pcap files using the pkt2flow tool [1], which splits the pcap files into individual flows. We then convert all the packets of a flow into raw byte stream using *scapy* tool [2]. For both models, we use raw TCP and UDP flows

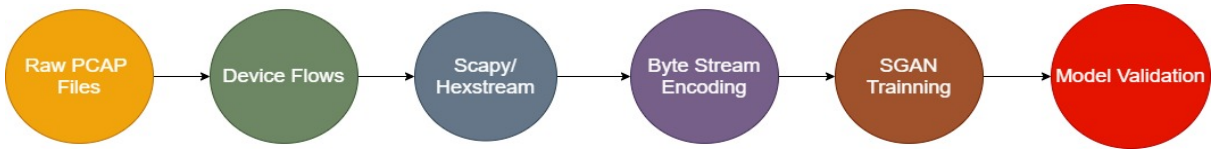

Fig. 3: *iKnight* stage #1 and stage #2 model training pipeline

for our training data. We remove the MAC layer header and device-dependent fields (source address, a destination address, destination port, source port, and checksum) from IP and transport layer headers to make our training data network independent. We then train the model at each stage with two types of training data; network-independent-headers-payload and only payload. We discuss the effect of the selection of each training data in the evaluation section. The training data at each stage consists of randomly sampled flows for all classes with balanced representation. We use the Encoding Engine discussed earlier to create the flow encoded matrix for each flow as the input to the SGAN training.

Training: In stage #1 SGAN model training, we split the dataset with 80% for training and 20% for testing. As discussed earlier in Section II-B, SGAN training consists of training both the Discriminator and Classifier with mini-batches of data in each epoch. Therefore, we select labeled flow samples for training the Classifier and unlabeled flows for training the Discriminator. Then, we train the SGAN model for multiple epochs, where each epoch consists of multiple batches of training data. We then validate the stage #1 SGAN model with the testing data. The SGAN training process for stage #2 is similar to stage #1, except that we filter the flows of a specific device and use the filtered flows to infer the device type class during validation. For example, we filter out all the flows of Dropcam during training and then evaluate the trained model with these filtered flows to validate whether the trained model is capable of inferring the correct type of device as IoT. We repeat this process for all the device classes (IoT, Non-IoT, Malware) and report the average accuracy. This process effectively evaluates the ability of the Classifier to infer unknown devices, which is discussed in the evaluation.

Model architecture: Our model architecture consists of a Generator, Discriminator, and Classifier. The Classifier shares the weights with the Discriminator [20] and has a Softmax layer [16] used to classify the flows' device or device type classes label. For our Generator and Discriminator models, we adopt an architecture very similar to that of DCGAN [19]. Note that the approach of using DCGAN for SGAN is also very recently used by [11]. However, we design our model with layers and hyper-parameters that is optimized specifically for byte stream based flow encoding scheme training data which consists of inputs of a 56*56*n Numpy array, where n is the number of packets the flow. For both the Discriminator and GAN model, we use Adam optimizer [12].

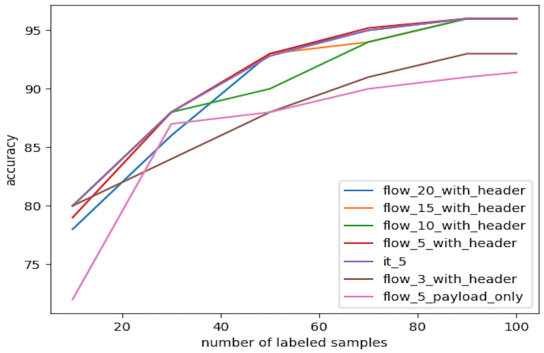

Fig. 4: The accuracy of stage #1 classifier in *iKnight*'s Device Discovery Engine

IV. Evaluation

A. Experiment setup

We implement *iKnight* Device Discovery Engine using Keras [7] and python, and train it on a GPU enabled machine with installed and 32 GB of memory. In the below sections, we evaluate *iKnight* for the different features of a real-world classifier discussed earlier in section V.

B. Impact of training data and encoding schemes

Figure 4 shows how the accuracy of the stage #1 classifier improves with the number of labeled samples. Therefore, the device type classification depends on the number of labeled samples, as discussed earlier. In our experiments, for both stage #1 and stage #2 classifiers, we achieve the highest accuracy with only 3% labeled flows of each device. Therefore, the best configuration of *iKnight* can achieve high accuracy with very few labeled data. We show the confusion matrix for stage 1 classifier using packets-with-payload-only encoding scheme in Figure 5.

In addition, Figure 4 also shows the increase in the accuracy of stage #1 classifier from 92% to 96% when the training data selection is changed from packets-with-payload-only to packets-with-network-independent-header. Therefore, the network-independent-header fields for IoT devices significantly improve the IoT device classification due to its features that help the classifier to uniquely identify each of the device classes more than packets-with-payload-only. Similarly, Figure 6 shows that the stage #2 classifier's accuracy significantly improves by 9% with the change of the encoding scheme from packets-with-payload-only to packets-with-network-independent-header while identifying the device type of an unknown device (i.e., Dropcam). Therefore, the network-independent-header improves the unknown device type classification.

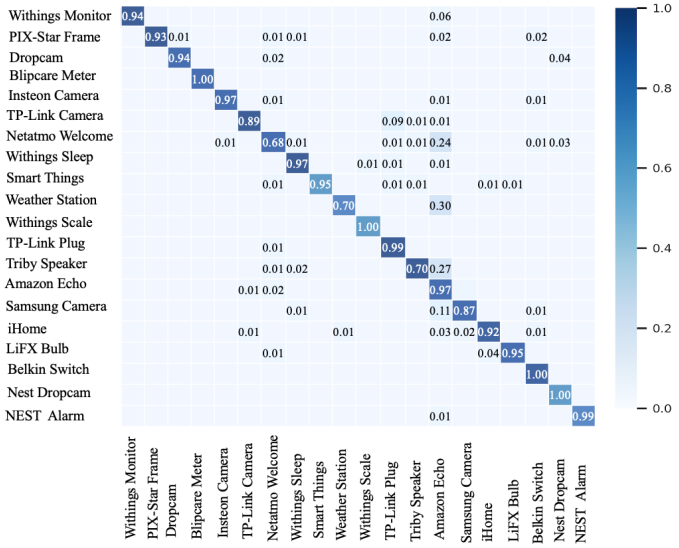

Fig. 5: The confusion matrix for *iKnight* stage #1 classifier with packets-with-payload-only

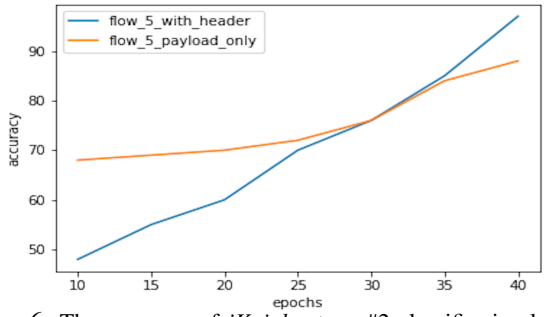

Fig. 6: The accuracy of *iKnight* stage #2 classifier in classifying an unknown device (i.e., Dropcam) as IoT device

Moreover, we observe from Figure 4 that the system's accuracy increases by 3% with an increase in the number of flow packets from three to five. However, a further increase in the number of packets does not affect the classifier's accuracy. Additionally, the encoding scheme flow-with-inter-arrival-time with five packets also does not increase the accuracy. The rationale behind this could be that the weights of the SGAN classifier have learned the best possible parameters from the raw encoded packet data. The additional meta-data information (inter-arrival time) does not improve their ability to increase the classifier's overall feature space representation. However, we note that a better choice of encoding design, such as the location and the representation of inter-arrival times inside the flow encoding matrix, can improve the system's accuracy. However, finding the optimal system encoding scheme for *iKnight* is outside the scope of this paper. Therefore, the best configuration of *iKnight* is the first five packets of the flow with network-independent-header and the encoding scheme flow-only.

C. Performance comparison to other classifiers

One of the advantages of SGAN is its ability to achieve high accuracy compared to supervised machine learning algorithms with less labeled data. Therefore, we compare *iKnight* with the most popular classification algorithms used for IoT network classification tasks. We compare the SGAN stage #1 classifier using packets-with-payload-only encoding scheme with both supervised and unsupervised classifiers as shown in Table II. Each of the supervised classifiers was trained with 3% labeled data, and each of the unsupervised classifiers was trained with all the unlabeled data. Finally, the SGAN classifier was trained with both the 3% labeled and the unlabeled data. We use the same packets-with-payload-only encoding scheme for all the classifiers. Table II shows that SGAN, which was trained with the same number of labeled flows, outperforms all other supervised classification algorithms. Moreover, SGAN also exceeds the unsupervised K means algorithm, where both SGAN and K means were trained using the same number of unlabelled training samples. While SGAN extensively performs well over all other algorithms, we have around 2% increase in accuracy over CNN. We believe this can be further enhanced by a better choice of encoding schemes, which can help SGAN extract the training data's distribution very well. For example, a combination of flow-level metadata and flow-byte stream encoding can help GANs model with a lot of information during training and, consequently, build a better latent space representing the training data.

TABLE II: The accuracy of *iKnight* stage #1 classifier using packets-with-payload-only encoding scheme across twenty IoT devices shown in Table I

Classification Method	**Type**	**Accuracy**
Kmeans	unsupervised (all unlabelled)	0.01
Kmeans + PCA	unsupervised (all unlabelled)	0.03
KNN	supervised (3% labeled)	0.85
Decision Tree	supervised (3% labeled)	0.86
Random Forest	supervised (3% labeled)	0.88
CNN	supervised (3% labeled)	0.90
SGAN	semi supervised (3% label + all unlabelled)	0.92

D. Unknown device type and anomaly detection performance

In this section, we evaluate *iKnight* for the different unknown device type and anomaly detection scenarios.

For evaluating *iKnight* stage #2, unknown device type identification scenario, we randomly choose five IoT devices, three Non-IoT devices, and all the four Malware classes. For each of the unknown devices, we first filter out the corresponding flows from the training data and validate the trained model with the filtered out flows of the unknown device, as discussed in section IV. We repeat this process for all the selected devices. Figure 7 shows the stage #2 device type classification accuracy using packets-with-network-independent-header encoding scheme for different classes of unknown devices. The accuracy of the unknown IoT and malware device types is higher than that of unknown Non-IoT device types. We believe this could be due to the few number of Non-IoT devices in the training data in which a better representation of each device type class in the training data can further improve the classification accuracy of the stage #2 classifier. The average accuracy for

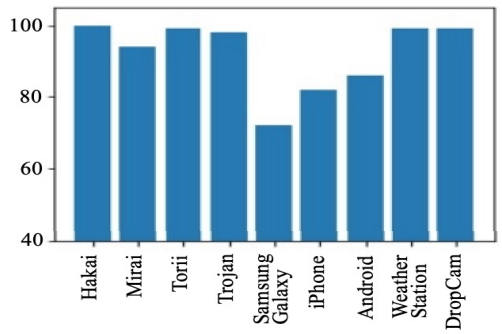

Fig. 7: The accuracy of *iKnight* stage #2 classifier of unknown devices using packets-with-network-independent-header for different type classes (i.e., IoT, Non-IoT, Malware)

each of the device type classes for the unknown scenario is IoT device type with 94%, Non-IoT with 80%, and the Malware with 97%. The overall unknown accuracy for different classes of device types is 90%.

Figure 6 shows the Stage #2 classifier's ability in identifying the type of an unknown device (i.e., Dropcam) as an IoT device with 97% accuracy. The classifier's accuracy is higher when the model is trained with packets-with-network-independent-header than when trained with packets-with-payload-only. This shows that *iKnight* can effectively capture the hidden feature space distribution for each of the IoT, Non-IoT, and Malware known device types. Moreover, this enables *iKnight* to have a better-generalized feature space representation of each device type class, which improves its ability to identify the type of unknown devices.Therefore, *iKnight* is capable of identifying the class types of unknown devices and known devices.

Semi-supervised GANs have recently shown good performance in identifying unknown classes as unknown (anomaly) [13], also known as novelty detection. This SGAN capability helps the Stage #1 classifier to identify any device that is not part of the training data as unknown, and also benefits the Stage #2 classifier to identify the type of unknown device as IoT, Non-IoT, Malware, or any anomaly (i.e., novelty detection). As future work, we are planning to run more experiments to evaluate SGAN capability in anomaly detection.

V. Discussion

In this paper, we evaluated and implemented the features of SGAN based IoT classification system and highlighted that SGAN based approach has several features that benefit the design of an IoT classifier. However, in IoT based scenarios, the user can add new devices to the network that were not part of the initial training set. Moreover, the device can have updated packet signatures due to a firmware change. In our future work, we plan to extend *iKnight* with continuously learning to identify new devices or devices with updated firmware by utilizing the Generator of the SGAN. The Generator can produce synthetic training data of old classes that can be used with new classes (ground truth) to incrementally train the classifier, an approach similar to [23] recently proposed for continuous image generation. Additionally, we would also transform *iKnight* into a very lightweight model using Knowledge distillation [10] for efficient deployment on edge devices.

References

[1] pkt2flow. https://github.com/caesar0301/pkt2flow. Accessed: 2020-25-06.

[2] scapy. https://scapy.net/. Accessed: 2020-25-06.

[3] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac. Peek-a-boo: I see your smart home activities, even encrypted! In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 207–218, 2020.

[4] M. J. E. Agustin Parmisano, Sebastian Garcia. Stratosphere laboratory. a labeled dataset with malicious and benign iot network traffic. https://www.stratosphereips.org/datasets-iot23, January 22th ,2019.

[5] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster. Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. *arXiv preprint arXiv:1708.05044*, 2017.

[6] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray. Iotsense: Behavioral fingerprinting of iot devices. *arXiv preprint arXiv:1804.03852*, 2018.

[7] F. Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[8] S. Dong, Z. Li, D. Tang, J. Chen, M. Sun, and K. Zhang. Your smart home can't keep a secret: Towards automated fingerprinting of iot traffic. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 47–59, 2020.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[10] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[11] A. S. Iliyasu and H. Deng. Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks. *IEEE Access*, 8:118–126, 2019.

[12] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[13] M. Kliger and S. Fleishman. Novelty detection with gan. *arXiv preprint arXiv:1802.10560*, 2018.

[14] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access*, 5:18042–18050, 2017.

[15] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma. Iot sentinel: Automated device-type identification for security enforcement in iot. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017.

[16] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[17] A. Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[18] J. Ortiz, C. Crawford, and F. Le. Devicemien: network device behavior modeling for identifying unknown iot devices. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, pages 106–117, 2019.

[19] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[20] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[21] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2018.

[22] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman. Inferring iot device types from network behavior using unsupervised clustering. In *2019 IEEE 44th Conference on Local Computer Networks (LCN)*. IEEE, 2019.

[23] C. Wu, L. Herranz, X. Liu, J. van de Weijer, B. Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. In *Advances in Neural Information Processing Systems*, 2018.