MINI PROJECT

Botnet Attacks Detection in IoT Devices Using Autoencoders

Submitted By:
Saumya Jain IEC2020088
Madhav Badewad IEC2020030

Contents

Introduction

With the increasing demand and growth in the Internet of Things (IoT) automated network system, the IoT models are getting complicated day by day. IoT devices use a wireless medium to broadcast data which makes them an easier target for an attack. Normal communication attack in the local network is limited to local nodes or small local domains, but attack in IoT systems expands over a larger area and has devastating effects on IoT sites. Therefore, a secured IoT infrastructure is necessary for the protection from cybercrimes. The security measures that have been used become vulnerable with the vulnerability of IoT devices. For some stakeholders and entrepreneurs, data is the money for their business. For the government and some private agencies, some data are classified and confidential. Vulnerability in IoT nodes makes a backdoor for an attacker to gather confidential data from any important organization.

A **botnet** is a group of Internet-connected devices, each of which runs one or more bots. Botnets can be used to perform Distributed Denial-of-Service (DDoS) attacks, steal data, send spam, and allow the attacker to access the device and its connection. The owner can control the botnet using command and control (C&C) software. The word "botnet" is a portmanteau of the words "robot" and "network".

A botnet comprises a large number of malware-infected client computers that are controlled by a remote server to perform malicious acts. A remote command and control server can control botnet computers to perform these types of attacks:

- Denial-of-service attacks
- Sending spam and viruses
- Stealing private data from clients

For detecting attacks launched from IoT bots we have used N-BaloT, a network-based approach for the IoT that uses deep learning techniques to perform anomaly detection. Specifically, we extract some features that capture behavioral snapshots of benign IoT traffic, and train a deep autoencoder (one for each device) to learn the IoT device's normal behaviors. The autoencoders attempt to compress snapshots. When an autoencoder fails to reconstruct a snapshot, it is a strong indication that the observed behavior is anomalous (the IoT device has been compromised and is exhibiting an unknown behavior). An advantage of using deep autoencoders is their ability to learn complex patterns—for example, of various device functionalities. This results in an anomaly detector with hardly any false alarms.

Literature Review

Sr. No	Author Name	Paper Title	Conference/ Journal Paper	Methods/ Approaches Used	Dataset used	Advantages / Result	Disadvantages / Future Work
1.	Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, Yuval Elovici	N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders	IEEE Computer Society	Autoencoders	N-BaIoT Dataset to Detect IoT Botnet Attacks	Autoencoders can be performed in a semi online manner (train on a batch of observations and then discard). The training is therefore practical, and there is no storage concern. Additionally, our method is network-based so it does not consume any computation, memory, or energy resources from the (typically constrained) IoT devices.	Different autoencoders can be used and transfer learning could be adopted. Evaluate transfer learning techniques by assessing the accuracy of models trained on specific devices when they are applied to identical devices, possibly when connected to other organizational networks.
2.	Xiaoran Yang, Zhen Guo, Zetian Mai	Botnet Detection based on Machine learning	International Conference on Blockchain Technology and Information	CTU13 and ISOT-2010, is used to obtain grayscale images that can be used for image	The malicious traffic comes from the CTU-13 data set. Normal traffic	This approach fills the gap in the data set of botnet detection in related research directions. In addition, the research has obtained a	The limitations of deep learning itself, as well as the complexity of actual problems, there are still

			Security (2022)	recognition.	comes from the ISOT-2010 data set	model that can be used for botnet detection and classification. And the accuracy rate of multi-class classification is as high as 98.7% or more. The accuracy of binary classification is 99.7%, and it has the ability to identify unknown types of traffic. This reduces the false alarm rate of detection and has certain practical significance.	some problems in the practical application of deep learning in the field of botnet detection.
3.	Orestis Kompougias , Dimitris Papadopoulos , Evangelos Mantas , Antonis Litke , Nikolaos Papadakis , Dimitris Paraschos , Akis Kourtis , George Xylouris	IoT Botnet Detection on Flow Data using Autoencoders	1 IEEE International Mediterranean Conference on Communications and Networking 2021	DNS blacklisting, sliding window of some size	CICIDS2017 This dataset contains real-world data with a simulated behaviour of 25 users within the network, capturing packets of the most	Successful in establishing a baseline of normal network behaviour, outputting a higher reconstruction error when fed with botnet flows which was used as an indicator of malicious activity given a predefined decision boundary.	Outlier detection in network traffic by using heuristic-based approaches such as neural networks is an inherently complex problem since the activities of each user in a network can vary dramatically making the underlying

					common HTTP, HTTPS, FTP, SSH, and email protocols.		distribution hard to model
4.	Mohsin Munir, Muhammad Ali Chattha, Andreas Dengel, Sheraz Ahmed	A Comparative Analysis of Traditional and Deep Learning-based Anomaly Detection Methods for Streaming Data	IEEE International Conference on Machine Learning and Applications	KNN, Local Outlier Factor (LOF), Connectivity-based Outlier Factor (COF), Local Correlation Integral (LOCI), Autoencoder, Isolation Forest (iForest), One-class SVM	Yahoo Webscope, Numenta Anomaly Benchmark	Deep learning-based anomaly detection methods are superior to other methods in most of the evaluation metrics for the Yahoo Webscope data set.	Aim to extend the comparison to other anomaly detection methods on streaming data sets, especially the methods which are performing good in the image modality
5.	Hichem Sedjelmaci, Sidi Mohammed Senouci, Mohamad Al-Bahri	A Lightweight Anomaly Detection Technique for Low-Resource IoT Devices: A Game-Theoretic Methodology	IEEE ICC 2016 - Mobile and Wireless Networking Symposium	Game - theoretic methodology	-	Lightweight anomaly detection approach requires a low energy consumption to achieve a high security level, i.e. high detection and low false positive rates	Embed our lightweight anomaly detection technique in largescale IoT devices and will deal the accuracy detection, energy consumption and network delay

6.	Ibrahim Alrashdi, Ali Alqazzaz, Esam Aloufi, Raed Alharthi, Mohamed Zohdy, Hua Ming	AD-IoT: Anomaly Detection of IoT Cyberattacks in Smart City Using Machine Leaming	IEEE	Machine Learning Algorithms	UNSW-NB15, KDD99, NLS-KDD	AD-IoT can significantly detect malicious behavior using anomalies based on machine learning through the evaluation of the UNSW-NB 15 dataset to detect the binary labeled classification before distributing on fog nodes.	Model is not finalized yet and needs more development
7.	Mahmudul Hasan, Md. Milon Islam, Md Ishrak Islam Zarif, M.M.A. Hashem	Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches	Internet of Things 7 (2019)	SVM, ANN, LR, DT, RF	Distributed Smart Space Orchestration System (DS2OS)	Rather than depending upon a single model they used different algorithms for comparison and found that DT,RF and ANN performed better.Trained over large data ,ensures more accuracy during testing.	This paper is confined to static data or preprocessed data. However, we are going to overcome this using data streaming in this project .

Supervised vs Unsupervised Anomaly Detection

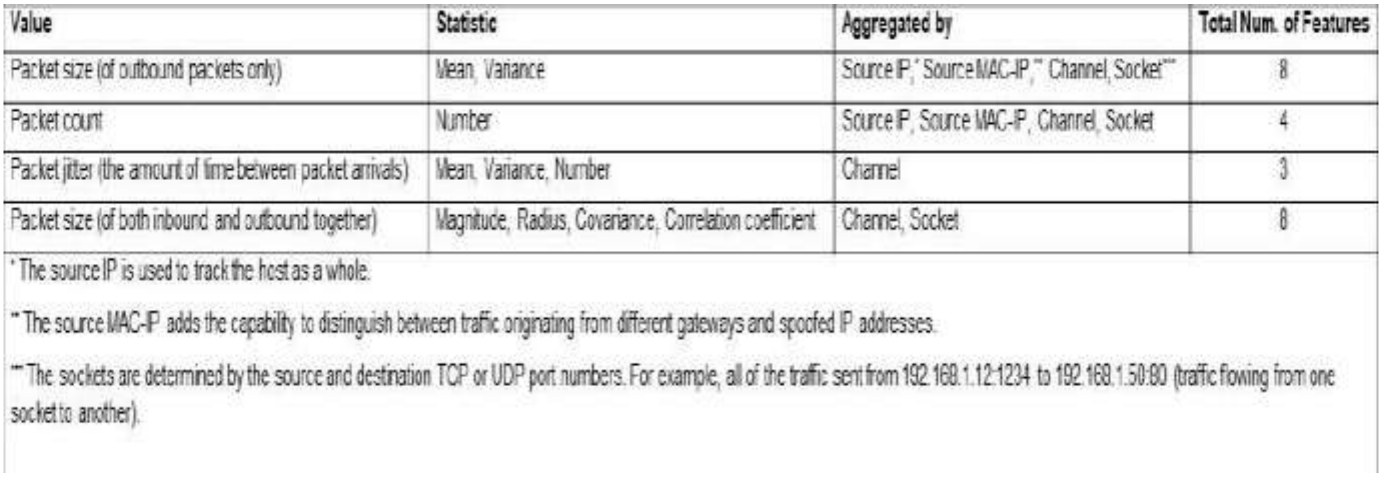
The most common version of anomaly detection is using the unsupervised approach. In there, we train a machine-learning model to fit to the normal behavior using an unlabeled dataset. In that process, we make an important assumption that the majority of the data in the training set are normal examples. However, there can be some anomalous data points among them (a small proportion). Then any data point which differs significantly from the normal behavior will be flagged as an anomaly. In supervised anomaly detection, a classifier will be trained using a dataset that has been labeled as 'normal' and 'abnormal'. When a new data point comes, it will be a typical classification application. There are pros and cons in both of these methods. The supervised anomaly detection process requires a large number of positive and negative examples. Obtaining such a dataset will be very difficult since anomalous examples are rare. Even though you obtain such a dataset, you would only be able to model the abnormal patterns in the gathered dataset. However, there are many different types of anomalies in any domain and also future anomalies may look nothing like the examples seen so far. It will be very hard for any algorithm to learn from anomalous examples; what the anomalies look like. That is why the unsupervised approach is popular. Capturing the normal behavior is much easier than capturing the many different types of abnormalities.

Dataset

1. N-BaIoT Dataset to Detect IoT Botnet Attacks:

This dataset contains real traffic data of 9 commercial IOT devices which were infected by Mirai and BASHILTE botnet attacks. It contains a total of 7062606 samples and 115 features. There are 11 instances of data collected for each device, 10 of which are data collected when the device is infected by 10 types of attacks carried out by 2 botnets, plus one instance of uninfected data.

The data is collected by analyzing data packets. A set of 23 features are extracted from analyzing the packet at 5 different time intervals: 100ms, 500ms, 1.5sec, 10 sec and 1 min totalling to 115 features. The features are given below:



Value	Statistic	Aggregated by	Total Num. of Features
Packet size (of outbound packets only)	Mean, Variance	Source IP,* Source MAC-IP,** Channel, Socket***	8
Packet count	Number	Source IP, Source MAC-IP, Channel, Socket	4
Packet jitter (the amount of time between packet arrivals)	Mean, Variance, Number	Channel	3
Packet size (of both inbound and outbound together)	Magnitude, Radius, Covariance, Correlation coefficient	Channel, Socket	8

* The source IP is used to track the host as a whole.

** The source MAC-IP adds the capability to distinguish between traffic originating from different gateways and spoofed IP addresses.

*** The sockets are determined by the source and destination TCP or UDP port numbers. For example, all of the traffic sent from 192.168.1.12:1234 to 192.168.1.50:80 (traffic flowing from one socket to another).

Source: [1]

2. IOT device identification:

This dataset was taken from chapter 5 of Machine learning cookbook for cyber security.

This dataset contains traffic data of about 10 devices: Baby monitor, lights, monitor sensor, security camera, smoke detector, socket, thermostat, TV, watch and water sensor. The data set contains 296 traffic attributes: ip port, packets, ack, bytes, reset, daystime and more. Each device has about 200 samples.

Source: [link]

3.CTU-13 DATASET

The botnet traffic dataset known as CTU-13 was recorded in 2011 at the CTU University in the Czech Republic. The dataset's aim was to have a sizable capture of actual botnet traffic mixed in with other types of traffic and background activity. The CTU-13 dataset consists of thirteen captures of various botnet samples, sometimes known as scenarios. On each scenario, they ran a particular piece of malware that utilised a number of protocols and carried out various tasks. However, because autoencoders only train on normal data and test on aberrant data, this dataset is divided into normal and abnormal traffic.

Methodology

- Our proposed method for detecting IoT botnet attacks relies on deep autoencoders for each device.
- In this project we worked on total 3 datasets, 3 different autoencoders models and transfer techniques
- We have trained our model on statistical features extracted from benign traffic data.
- When applied to new (possibly infected) data of an IoT device, detected anomalies may indicate that the device is compromised.

- This method consists of three main stages: data collection, training an anomaly detector, and testing the model.
- We have also used Transfer Learning technique, sparse autoencoders, Autoencoders Ensemble.

List of ML algorithms to be used

1. Autoencoders

Autoencoders are a specific type of feedforward neural networks where the input is the same as the output. They compress the input into a lower-dimensional code and then reconstruct the output from this representation. The code is a compact “summary” or “compression” of the input, also called the latent-space representation.

An autoencoder consists of 3 components: encoder, code and decoder. The encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code.

If an autoencoder is trained on benign instances only, it will succeed at reconstructing normal observations but fail at reconstructing abnormal observations (unknown concepts). When a significant reconstruction error is detected, we classify the given observations as anomalous.

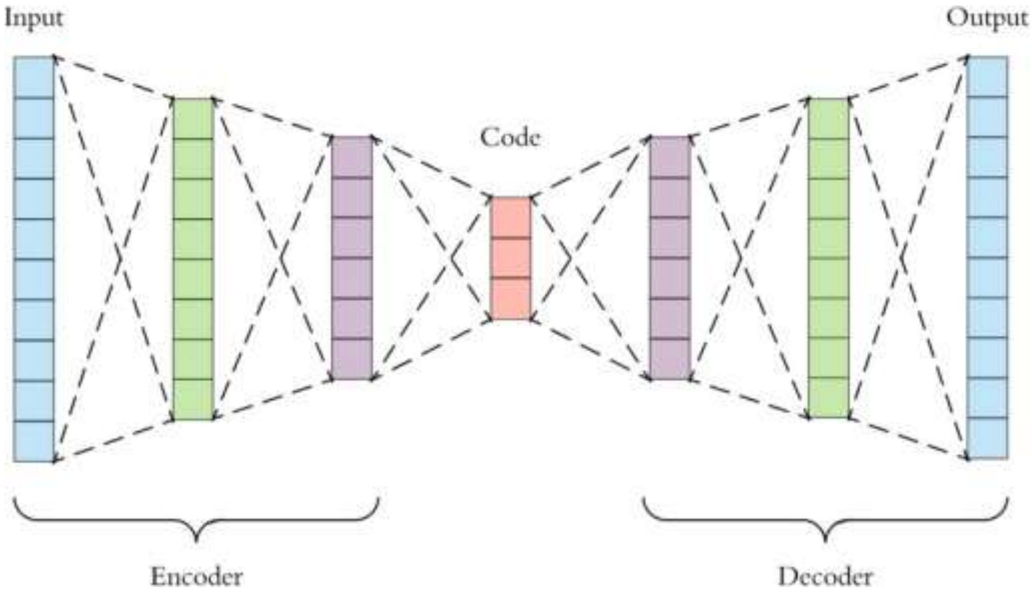


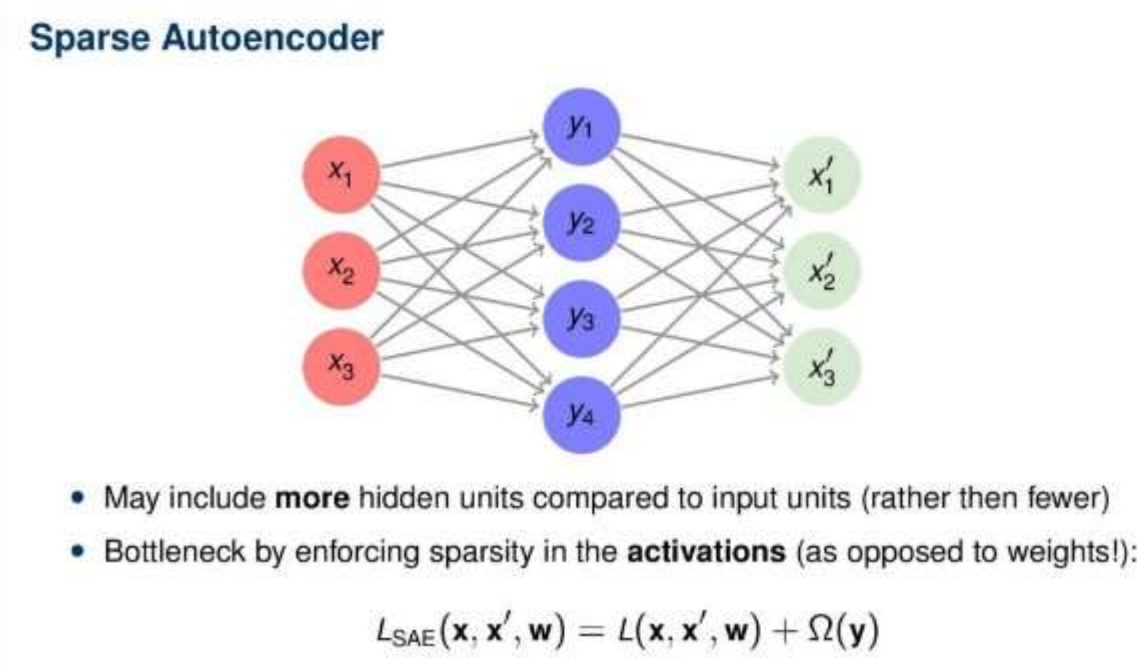
Image Source: [Link]

2. Sparse Autoencoders

An autoencoder that uses a sparsity penalty as part of its training criteria is known as a sparse autoencoder. Typically, we would build our loss function by penalising hidden layer activations, which would encourage just a small number of nodes to activate when a single sample is input into the network.

The autoencoder can be made regular by applying a sparsity constraint.

Only fraction nodes are permitted to do forward and backward propagation during regularisation. These nodes are known as active nodes since they have non-zero values.



Sparse Autoencoder

- May include **more** hidden units compared to input units (rather then fewer)
- Bottleneck by enforcing sparsity in the **activations** (as opposed to weights!):

$$L_{\text{SAE}}(\mathbf{x}, \mathbf{x}', \mathbf{w}) = L(\mathbf{x}, \mathbf{x}', \mathbf{w}) + \Omega(\mathbf{y})$$

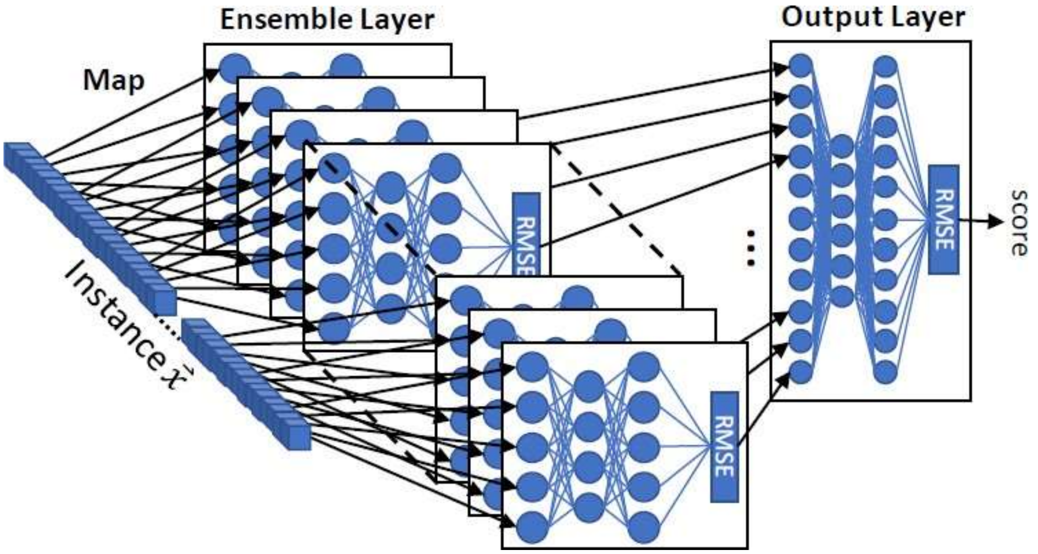
Source :link

3. Autoencoder Ensemble

Ensemble learning methods are algorithms that combine the predictions from different base detectors in order to create more robust results. The basic idea is that predictive algorithms often have a natural variance in the scores, depending on the choice of the data or the design of the base model. Therefore, by using multiple executions of the model and taking a central estimator of the scores (such as the mean or median), the variance is reduced.

A number of samples are analyzed to group the features into disjoint subsets. This is done by calculating the pairwise correlation distances between all features of these samples. The subsets are selected with agglomerative hierarchical clustering. So each subset is a group of features that are highly intercorrelated. The maximum size of a subset is a parameter of the algorithm.

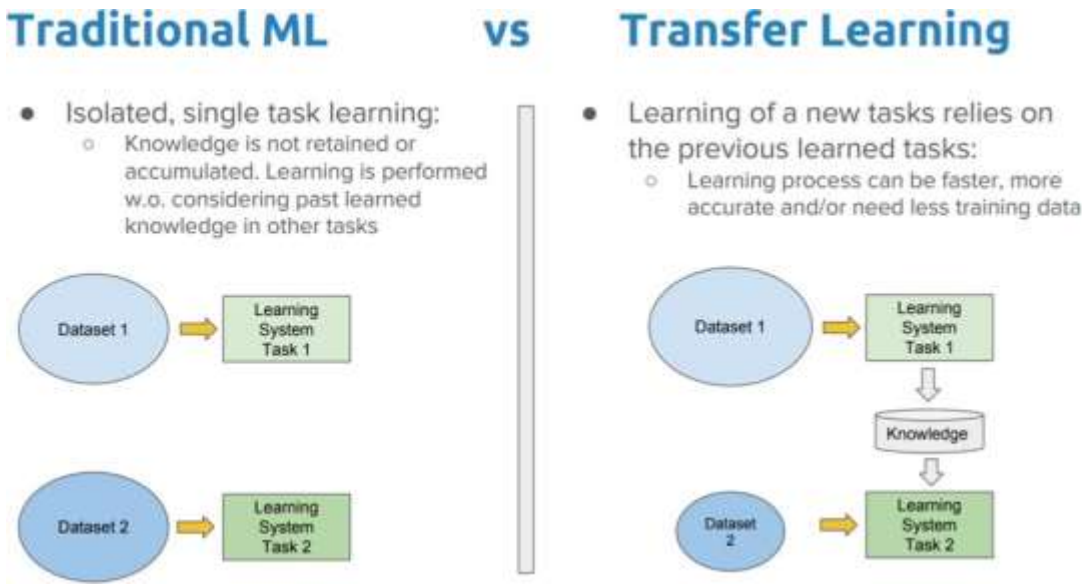
After that, an autoencoder is created for each subset of features. These autoencoders form the ensemble layer. Each autoencoder sees only its subset of features of each sample. The reconstruction errors of all ensemble members form the input for the final autoencoder. That is, it learns how well the ensemble can reconstruct the training data. The final autoencoder's own reconstruction error is used for the output, i.e., surpassing a given threshold is reported as an anomaly.



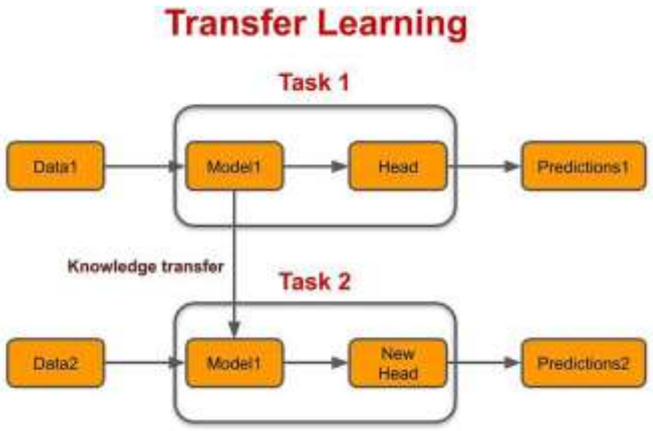
Source: [link]

4. Transfer Learning

Conventional machine learning and deep learning algorithms, so far, have been traditionally designed to work in isolation. These algorithms are trained to solve specific tasks. The models have to be rebuilt from scratch once the feature-space distribution changes. Transfer learning is the idea of overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones.


Traditional ML
vs
Transfer Learning
Isolated, single task learning:
Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
Learning of a new tasks relies on the previous learned tasks:
Learning process can be faster, more accurate and/or need less training data
Dataset 1
Learning System Task 1
Dataset 2
Learning System Task 2
Dataset 1
Learning System Task 1
Knowledge
Dataset 2
Learning System Task 2

Source: [Link]



Source: link

Performance Analysis

Confusion matrix

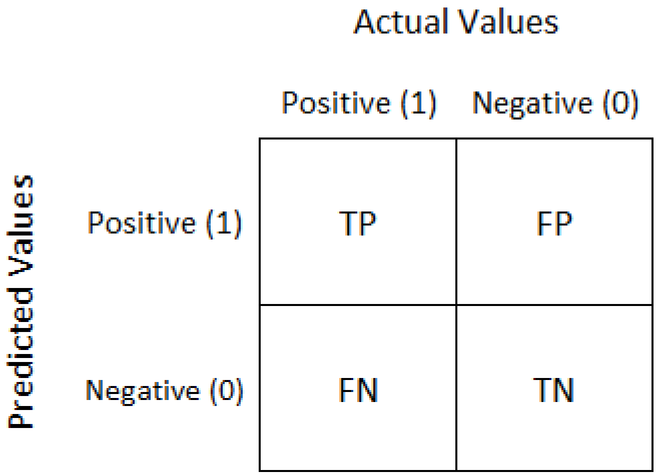
In Machine Learning, a confusion matrix, also known as error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa. It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions. It has 4 grids (as shown in figure below) indicating 4 values:

True positive (TP): A test result that correctly indicates the presence of a condition or characteristic.

True Negative (TN): A test result that correctly indicates the absence of a condition or characteristic.

False Positive (FP): A test result which wrongly indicates that a particular condition or attribute is present.

False Negative (FN): A test result which wrongly indicates that a particular condition or attribute is absent.



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Source: link

Results

True Positive: Autoencoder predicted as abnormal and it's true.

True Negative: Autoencoder predicted as normal and it’s true.

False Positive: Autoencoder predicted as abnormal and it’s false.

False Negative: Autoencoder predicted as normal and it’s false.

S.No	Device	Method	TP	FP	TN	FN
1.	Danmini Doorbell	Normal	100%	0.28%	99.72%	0%
		Transfer learning without training	100%	100%	0%	0%
		Transfer learning with training	100%	0.53%	99.47%	0%
2.	Ecobee Thermostat	Normal	100%	0%	100%	0%
		Transfer learning without training	100%	1.91%	98.09%	0%
		Transfer learning with training	100%	0%	100%	0%
3.	Ennio Doorbell	Normal	100%	4.47%	95.53%	0%
		Transfer learning without training	100%	2.26%	97.74%	0%
		Transfer learning with training	100%	4.67%	95.33%	0%
4.	Philips B120N10 Baby Monitor	Normal	100%	19.07%	80.93%	0%
		Transfer learning without training	100%	10.57%	89.43%	0%
		Transfer learning with training	100%	16.74%	83.26%	0%
5.	Provision PT 737E Security Camera	Normal	100%	11.57%	88.43%	0%
		Transfer learning without training	100%	100%	0%	0%
		Transfer learning with training	100%	5.69%	94.31%	0%
6.	Provision PT 838 Security Camera	Normal	100%	19.79%	80.21%	0%
		Transfer learning without training	100%	59.31%	40.69%	0%
		Transfer learning with training	100%	19.60%	80.40%	0%
7.	Samsung SNH 1011 N Webcam	Normal	100%	52.77%	47.23%	0%
		Transfer learning without training	100%	100%	0%	0%
		Transfer learning with training	100%	56.74%	43.26%	0%
8.	SimpleHome XCS7 1002 WHT Security Camera	Normal	80%	5.50%	94.50%	20%
		Transfer learning without training	100%	100%	0%	0%
		Transfer learning with training	100%	10%	90%	0%

9.	SimpleHome XCS7 1003 WHT Security Camera	Normal	80%	3.08%	96.92%	20%
		Transfer learning without training	80%	0%	100%	20%
		Transfer learning with training	80%	3.14%	96.86%	20%

Transfer Learning on Same Device

S.no	Dataset properties			Optimized hyperparameters of autoencoders				Botnet infections	
	Device	No of benign instances	Training time (seconds)	Learning rate	No of epochs	Anomaly Threshold	Window Size	Mirai	Bashlite
1.	Danmini Doorbell	49,548	3516	0.012	800	0.042	82	✔	✔
2.	Ecobee Thermostat	13,113	322	0.028	250	0.011	20	✔	✔
3.	Ennio Doorbell	39,100	1197	0.003	350	0.011	22	-	✔
4.	Philips B120N10 Baby Monitor	175,240	1704	0.016	100	0.030	65	✔	✔
5.	Provision PT 737E Security Camera	62,154	1826	0.026	300	0.035	32	✔	✔
6.	Provision PT 838 Security Camera	98,514	2606	0.008	450	0.038	43	✔	✔
7.	Samsung SNH 1011 N Webcam	52,150	502	0.013	150	0.074	32	-	✔
8.	SimpleHome XCS7	46,585	622	0.017	230	0.056	23	✔	✔

	1002 WHT Security Camera								
9.	SimpleHome XCS7 1003 WHT Security Camera	19,528	502	0.006	500	0.004	25	✔	✔

Reconstruction of training state of base paper

S.no	Device	TP	FP	TN	FN
1.	Danmini Doorbell	80%	0%	100%	20%
2.	Ecobee Thermostat	100%	1.00%	99.00%	0%
3.	Ennio Doorbell	60%	0%	100%	40%
4.	Philips B120N10 Baby Monitor	100%	3.86%	96.14%	0%
5.	Provision PT 737E Security Camera	100%	21.05%	78.95%	0%
6.	Provision PT 838 Security Camera	100%	0%	100%	0%
7.	Samsung SNH 1011 N Webcam	60%	1.55%	98.45%	40%
8.	SimpleHome XCS7 1002 WHT Security Camera	100%	13.94%	86.06%	0%
9.	SimpleHome XCS7 1003 WHT Security Camera	100%	4.17%	95.83%	0%

Base Paper reconstruction results

S.No	Device	TF	FP	TN	FN
1	Danmini Doorbell	100%	14.9%	85.1%	0%
2	Ecobee Thermostat	100%	21.6%	78.4%	0%
3	Ennio Doorbell	100%	26.5%	73.5%	0%
4	Philips B120N10 Baby monitor	100%	12%	88%	0%
5	Provision PT 737E Security Camera	100%	19.5%	80.5%	0%
6	Provision PT 838 Security Camera	100%	28.9%	71.1%	0%
7	Samsung SNH 1011 N Webcam	100%	35.2%	64.8%	0%
8	SimpleHome XCS7 1002 WHT Security Camera	100%	30.9%	69.1%	0%
9	SimpleHome XCS7 1003 WHT Security Camera	99.9%	22.2%	78.8%	0.1%

Autoencoder Ensemble on N-BaloT Dataset

S.no	Device	Learning rate	No of Epochs	TP	FP	TN	FN
1.	Baby monitor	0.0005	56	100%	0%	0%	0%
2.	Lights	0.0005	25	100%	0%	0%	0%
3.	Monitor sensor	0.0005	12	100%	0%	0%	0%
4.	Security Camera	0.0005	141	100%	0%	0%	0%
5.	Smoke Detector	0.0005	22	100%	0%	0%	0%
6.	Socket	0.0005	17	100%	0%	0%	0%
7.	Thermostat	0.00001	800	100%	0%	0%	0%
8.	TV	0.01	8	100%	0%	0%	0%
9.	Watch	0.000001	800	100%	0%	0%	0%
10.	Water Sensor	0.01	6	100%	0%	0%	0%

Autoencoders on IOT Device Identification Dataset

- Accuracy when we trained our model on one device and tested it for another similar device.

S.No.	Device	Device Name	TP	FP	TN	FN
1.	Doorbell	**Trained On:** Danmini Doorbell				
		Tested On: Ennio Doorbell	100.0%	1.3%	98.7%	0%
2.	Security Camera	**Trained On:** Provision PT 737E Security Camera				
		Tested on: Provision PT 838 Security Camera	100%	48.33%	51.61%	0%
		SimpleHome XCS7 1002 WHT Security Camera	100%	100%	0%	0%
		SimpleHome XCS7 1003 WHT Security Camera	79.16%	4.56%	95.44%	20.84%

Conclusion

Medan and others [1] presented use of autoencoders as an effective means for detecting DDoS traffic generation from Bashlite and Mirai infected devices. We were able to replicate their results and advanced it by evaluating performance of the trained models across different devices.After working on different autoencoders we come to conclusion that deep autoencoder performed better for all datasets N-BaIoT, IOT device identification and CTU -13.

We further implemented transfer learning by saving the autoencoder trained on one device and using it to predict for other similar devices. Transfer-learning is often considered when only limited data is available for a similar problem.

References

1. Y. Meidan et al., "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," in IEEE Pervasive Computing, vol. 17, no. 3, pp. 12-22, Jul.-Sep. 2018, doi: 10.1109/MPRV.2018.03367731.
2. X. Yang, Z. Guo and Z. Mai, "Botnet Detection Based on Machine Learning," 2022 International Conference on Blockchain Technology and Information Security (ICBCTIS), 2022, pp. 213-217, doi: 10.1109/ICBCTIS55569.2022.00056.
3. O. Kompougias et al., "IoT Botnet Detection on Flow Data using Autoencoders," 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), 2021, pp. 506-511, doi: 10.1109/MeditCom49071.2021.9647639.
4. M. Munir, M. A. Chattha, A. Dengel and S. Ahmed, "A Comparative Analysis of Traditional and Deep Learning-Based Anomaly Detection Methods for Streaming Data," 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019, pp. 561-566, doi: 10.1109/ICMLA.2019.00105.
5. H. Sedjelmaci, S. M. Senouci and M. Al-Bahri, "A lightweight anomaly detection technique for low-resource IoT devices: A game-theoretic methodology," 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1-6, doi: 10.1109/ICC.2016.7510811.
6. I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy and H. Ming, "AD-IoT: Anomaly Detection of IoT Cyberattacks in Smart City Using Machine Learning," 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), 2019, pp. 0305-0310, doi: 10.1109/CCWC.2019.8666450.
7. Hasan, Mahmudul, et al. "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches." Internet of Things 7 (2019): 100059.
8. P. P. Kundu, T. Truong-Huu, L. Chen, L. Zhou and S. G. Teo, "Detection and Classification of Botnet Traffic using Deep Learning with Model Explanation," in IEEE Transactions on Dependable and Secure Computing, 2022, doi: 10.1109/TDSC.2022.3183361.
9. B. Sharma, L. Sharma and C. Lal, "Anomaly Detection Techniques using Deep Learning in IoT: A Survey," 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), 2019, pp. 146-149, doi: 10.1109/ICCIKE47802.2019.9004362.
10. J. B. Borges, J. P. S. Medeiros, L. P. A. Barbosa, H. S. Ramos and A. A. Loureiro, "IoT Botnet Detection based on Anomalies of Multiscale Time Series Dynamics," in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2022.3157636.