Behavioural Classification of Network Devices using Graph Structure of Private Networks

Vojtěch Outrata[1,2], Jaroslav Hlaváč[1,3] and Martin Kopp[1]

[1]TD&R Data Science, Cisco Systems, Karlovo nám. 10, 120 00 Praha 2, Czech Republic
[2]Czech Technical University, Faculty of Electrical Engineering, Karlovo nám. 13, 120 00 Praha 2, Czech Republic
[3]Charles University, Faculty of Mathematics and Physics, Ke Karlovu 3, 121 16 Praha 2, Czech Republic

Abstract

Classifying computer network devices by their behaviour is crucial for keeping track of today's dynamically changing network environments. In the field, it is mainly done either manually or using heuristics. The commonly used solutions completely overlook the computer network's inherent graph structure. In this paper, we show that leveraging the graph structure using graph convolutional neural networks is worth the added computational burden. Our evaluation on real-world networks with user-defined classes includes three baseline models and a graph neural network. The experimental results highlight the models' proficiency in learning diverse device classes, with the graph-based models exhibiting superior performance. We also show that the graph-based models struggle to adapt to the ever-changing structure of the network and measure the cost of their retraining.

Keywords

behavioural classification, device classification, graph neural networks, positional features

1. Introduction

Properly managing and monitoring large private networks is necessary for companies to avoid network failures that could cause business disruptions. An essential part of this process is knowing the role and importance of each device in the network. Devices crucial for business operations must be differentiated from the less important ones. The impact of losing connection to a smart light bulb differs vastly from losing a production server. Nowadays, manual device classification is still common. But it is time consuming and requires extensive domain and local network environment knowledge. This makes it impractical for large-scale networks. Automated or semi-automated heuristic solutions exist, but the dynamic nature of networks, where devices are constantly added and removed, further complicates the labelling process. As a result, many devices in networks remain unlabelled or mislabelled.

Our approach to the device identification problem is as a semi-supervised classification. We assume that by leveraging the graph structure of the computer network, we can improve the classification efficacy. We test our assumption by comparing methods that classify a device using only information about the device itself with graph neural networks leveraging the natural graph structure of the computer networks.

For this purpose, we designed features that capture important device information without building a graph and features computed from the graph structure. We train and test various models on real-world, large-scale networks spanning multiple industries and numerous distinct device types. Consequently, the model evaluation provides valid information on the usability of the models in the real world.

2. Prior art

Behavioural device classification based on network traffic can be based solely on static information about the device. This group of methods is in this paper represented by the three standard ML models: Support Vector Machine (SVM) [1], Random Forest (RF) [2], and AdaBoost [3].

The other way is to leverage the graph structure of computer networks. There are multiple approaches to creating node embeddings for classification. The Matrix-factorisation leverages the eigendecomposition of a proximity matrix as the embedding for each node [4]. While straightforward, this approach is not further used as factorisation is computationally expensive.

The second approach, inspired by the word2vec algorithm [5], obtains the proximity of nodes by random walks over the graph. This approach is used by, e.g., DeepWalk [6], LINE [7], and node2vec [8]. The node2vec is the most recent and most general out of the listed ones, and it has been previously used to model computer networks [9].

The most recent approaches are based on Graph Convolutional Networks (GCN) [10]. Localised convolution,

ITAT 23: Information Technologies – Applications and Theory, September 22–26, 2023, Tatranské Matliare, Slovakia
voutrata@cisco.com (V. Outrata); jhlavac@cisco.com (J. Hlaváč); markopp@cisco.com (M. Kopp)

Id	source	destination	proto	hash	ts
a	***ip***:192.168.8.18 ***port***: 59496	***ip***:192.168.0.19 ***port***: 80	TCP	fff	16
b	***ip***:10.4.22.12 ***port***: 64270	***ip***:1.2.91.29 ***port***: 80	TCP	123	18

Table 1
Connection log examples showing when a device communicated, where it communicated, using what protocol and the sha256 hash of the binary that created the connection.

introduced in GraphSage [11], is an innovative way to incorporate both node features and the graph structure. The embedding of each node contains aggregated information from its K-hop neighbourhood, and due to its locality, it scales much better than the previous approaches. The GraphSage has been surpassed on common benchmarks by other architectures, such as GAT [12], GATv2 [13], P-GNN [14], each defining their own aggregating scheme. In this paper, we employ one of the most recent architectures, the GATv2, further referred to as the GNN.

3. Data description and feature engineering

In this paper, we work with the network communication logs collected by a lightweight application (collector) installed on the endpoint device. The data format is similar to the NetFlow [15] but enriched by additional information about the device. In the field, several implementations of such collectors exist, each collecting different features. To make our work generally applicable, we selected the information present in endpoint connection logs from any collector. Table 1 shows an example of the communication logs with the selected features.

For feature selection and engineering, we used connection logs from 8 real-world networks of various sizes (from thousands to hundreds of thousands of devices), different industries (logistics, healthcare, finance, …) and labelled by different logic (by location, function, or combination).

3.1. Device features

Each device in the network can be described by a set of features engineered from a device's communication logs without building the local network's graph. In total, we use 122 features, each computed from the device communication spanning a single day. The comprehensive list of features can be found in Appendix A. We divide these features into four categories - port, time, networking and hash. The port number can indicate what services are running on the device and what services it is accessing over the network. Time features describe the activity of the device during the day. Networking features focus on the ratios of inbound and outbound connections over transport layer protocols and the ratio of internal/external network communication. Hash features are computed from the prevalence of the binary that initiated the connection.

The baseline classification algorithms use only the above described features. In GNN, they are used as part of node description in a graph.

3.2. Graph-based features

To leverage the structure of the computer network, we consider devices to be nodes, and communication between two devices indicates an edge. To prevent the graph from expanding to unreasonable sizes by representing the public internet, the graphs are constructed only from communications within the internal network.

Correctly classifying devices of the same function (e.g., domain controllers) in different locations to geographically defined classes is not possible using only features described in 3.1. Therefore, we enrich the node features with proposed features that reflect the global structure of the graph.

The novel positional features are constructed as follows: We choose c nodes with the highest PageRank centrality [16] as central nodes for the network. These central nodes then serve as anchors in the graph. Each node's global position in the graph is then encoded into c distances to these central nodes. Knowing the global position of the nodes greatly improves the classification results for networks that are labelled by location. Figure 1 shows results on Company A, which has a geographically segmented private network. To ensure statistical relevance, ten runs were performed for each number of nodes, and 95% confidence intervals based on the Student's t-distribution are displayed. The steep improvement in the results occurs at 20 central nodes, corresponding to the number of important nodes in the respective graph with respect to the PageRank centrality (according to a deeper graph analysis). Since other networks may have a larger graph with more central nodes, e.g. Company C has 30-40 central nodes, we have decided to set the number of considered central nodes to 50 throughout the experiments. Further increasing the number of central nodes did not have any impact on the classification accuracy.

3.3. Device labels

There is no universal template for categorising devices within private networks. The type of devices in networks varies greatly alongside the industry the company is operating in. Consequently, the companies usually design

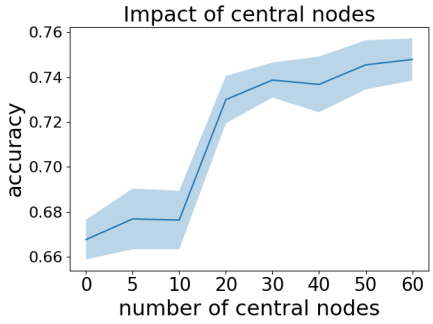

Figure 1: The effects of the number of central nodes on classification accuracy for Company A. The 95% confidence intervals are based on the Student's t-distribution.

the device classes themselves, resulting in different types of grouping. One company may want to group the devices by their geographic location, while another one by the function of the device in the network. From the eight networks used for feature selection, we chose the three most diverse to use in the final experiments. The labels for our experiments were provided by the respective companies themselves. Table 2 shows the device counts and labels for three networks.

The ratios of labelled devices within the networks further support the semi-supervised learning scenario. The models can not only extrapolate learned knowledge to unlabelled devices, but the GNN model leverages the information from unlabelled devices in its aggregation scheme while classifying the labelled devices. The number of devices within each class indicates severe class imbalance for Company B and Company C. During the evaluation, special countermeasures had to be taken for both companies.

	label	device count	labelled devices
Company A	City1	57	0.27
	City2	136	
	City3	136	
	City4	102	
	City5	246	
	City6	60	
	City7	51	
	City8	36	
Company B	Domain Controller	11	0.52
	Protect	298	
	Protect - IT	20	
	Server	69	
Company C	Loc. A - servers	514	0.67
	Loc. A - workers	1 022	
	Loc. A Building Services	10	
	Loc. A IS	24	
	Loc. A Lab	34	
	Loc. B - servers	26	
	Loc. B - workers	22 236	
	Loc. B App Packaging	15	
	Loc. B Cardiology EEG	20	
	Loc. B Cardiology PACS	176	
	Loc. B Medical Device	27	
	Loc. B Philips Software	49	
	Loc. B Radiology	84	
	Loc. C General	92	
	Loc. C General Srvs	27	

Table 2
Device labels for the three selected companies used in the final experiments. Company A is grouped by geographic location, Company B by device function, and Company C has a combination of both.

4. Experimental evaluation

We designed two experiments to evaluate the usability of proposed solutions in the real world. The first experiment confirms whether models can learn the diverse behaviour of network devices when training on each day separately. The second experiment tries to find out how long models maintain good classification results before they need to be retrained.

4.1. Experimental setup

In the experiments, we worked with a dataset collected over 14 days on three private networks of various sizes and structures, see Table 2. Both device and graph features were calculated for each day.

One of the substantial challenges in the training process was the class imbalance. In Company B, class weights [17] were applied for training SVM, Random Forest, and GNN. AdaBoost performed well without class weighting. For Company C, the majority class covers $\sim 90\%$ devices. Therefore, it was subsampled before the class weighting.

The models were set up by hyperparameter grid search with 4-fold cross-validation, based on the study [18]. Regarding GNN architecture, initial experiments revealed that one hidden layer aggregating a 1-hop neighbourhood is sufficient. Deeper architectures resulted in worse performance. Additional GNNs parameters, such as the number of heads in the attention mechanism or dimension of the final embedding, were optimised using the Tune framework [19], utilising the Adam [20] optimiser.

4.2. Training stability

The first experiment was designed to evaluate models' performance over a working week of telemetry from two perspectives. Firstly, how well can the models learn the diverse classes. Secondly, to assess whether the mod-

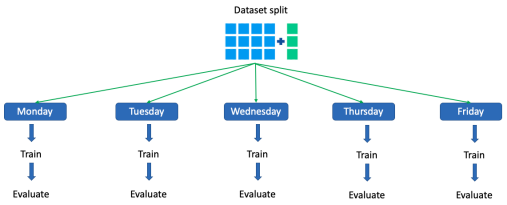

Figure 2: Setup of the first experiment. Models are trained on five working days, taking data from 20% of devices each day for testing purposes to evaluate the stability of the training process throughout the week.

els' training is stable throughout the week. Stability is an essential metric for potential real-world use. To validate training stability, models were trained and evaluated on ten train/test splits each day, as shown in Figure 2. Specifically, for each train/test split and day, each model was trained and evaluated on data from that particular day. We provide experimental analysis for each company separately to discuss the caveats of different networks.

4.2.1. Company A

Figure 3 contains classification accuracy of each method with the 95% confidence intervals inferred based on the Student's t-distribution. The graph-based approach consistently achieves better classification results than the baseline models, which indicates that utilising graph structure is beneficial for classifying the location-based device classes.

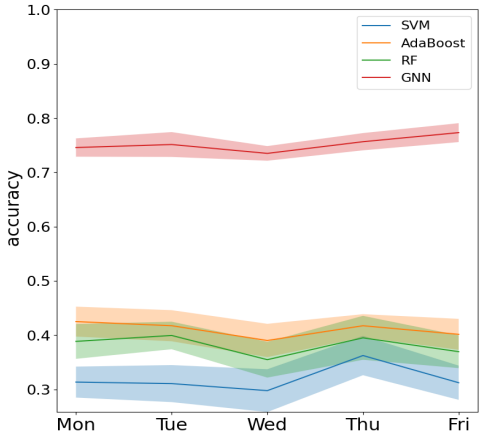

Figure 3: Classification accuracy for each model with the 95% confidence interval for Company A.

4.2.2. Company B

Due to severe class imbalance in Company B, we present the macro average recall and precision, rather than accuracy in Figure 4.

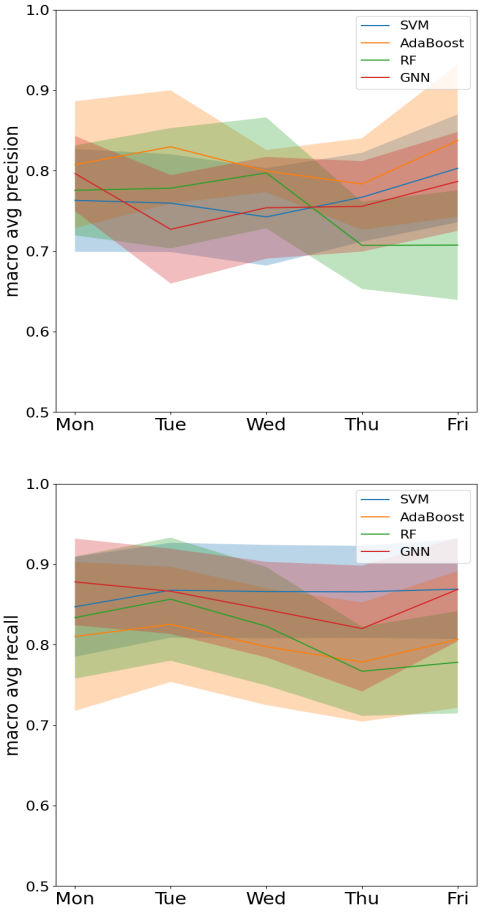

Figure 4: Classification precision and recall with 95% confidence intervals for Company B. The wide confidence intervals are caused by low prevalence classes.

The class imbalance, specifically the low number of devices in certain classes of the testing dataset, causes substantial fluctuations in macro-averaged metrics, resulting in unstable reported scores and wide confidence intervals. To illustrate the problem, an example of classification results of the GNN model is shown in Table 3. In this example, the misclassification of one domain controller and one device from *Protect-IT* class would cause the reported macro average for recall to fall by more than 10%.

	precision	recall	support
Domain Controller	0.83	1.00	5
Protect	1.00	0.90	69
Protect-IT	0.50	1.00	5
Server	0.86	0.92	13
macro avg	0.80	0.96	
accuracy	0.91		

Table 3
Example of classification results of one run of the GNN model on Company B.

4.2.3. Company C

The classification results for Company C are also affected by class imbalance. The classification performance depicted in Figure 5 shows that the GNN model with AdaBoost is consistently more precise than other baseline models.

Further study of the confusion matrix of the GNN model, in Figure 6, shows two extremes, either the model can distinguish the class well or almost not at all. The model usually classified all devices from classes *Loc. A Lab*, *Loc. A IS*, *and Loc. A Building Services* entirely into the *Loc. A—workers* class. This is not surprising as all three classes fall within the *Loc. A—workers* definition (workstations, not servers, in the same location). Similar results from other models indicate that these three classes cannot be easily separated by our features, so the models assign them to the most general class.

A similar explanation also applies to other low prevalence classes that fall into the broader category of *Loc. B—workers*. On the other hand, classes representing servers were well classified.

4.2.4. Summary

The presented experimental analysis confirms that diverse device types can be classified using the representation introduced in Section 3. Furthermore, the results demonstrated that all models' training procedures were stable throughout the week. The results also indicate that by leveraging the positional features, the GNN was able to learn higher prevalence classes and makes reasonable predictions for the classes with only a few devices.

4.3. Prediction stability

The second experiment tests how long the models can maintain their predictive performance. All models are trained on a single day (Wednesday) and evaluated on the next six days (Thursday - Tuesday), as illustrated in Figure 7. Ten-fold cross-validation was used on the Wednesday data for training the models. Then they were evaluated on the test data from Wednesday and the whole dataset from each of the following six days.

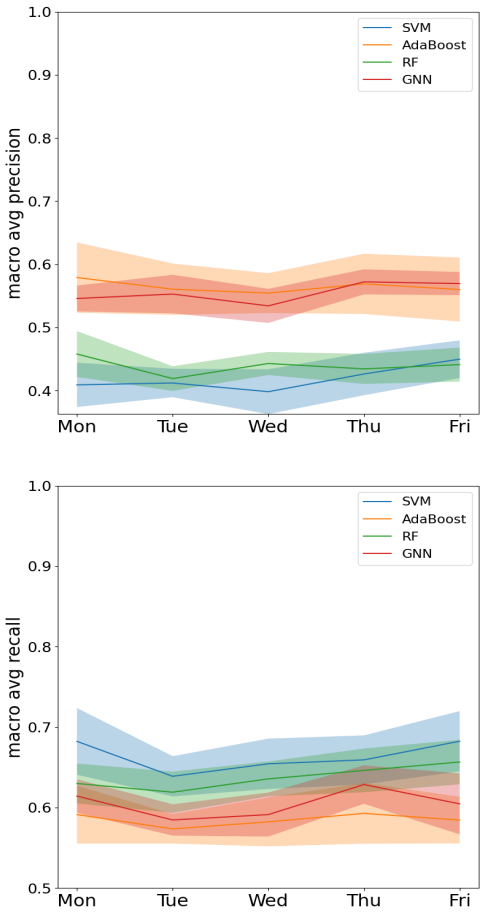

Figure 5: Classification precision and recall with 95% confidence intervals for Company C. GNN and AdaBoost has consistently higher precision with recall similar to other models.

4.3.1. Class distribution

First, we examine the occurrences of devices from each class over the week. Figure 8 represents Company B and exhibits an expected pattern; the general class *Protect* shrinks the most during the weekend as it contains workstations of regular employees. The other classes, representing servers and IT workstations, remain stable even throughout the weekend. This pattern holds for the other two companies as well.

While expected, this observation still complicates the selection of central nodes for features representing the global structure of the graph. As explained in Section 3.2, the positional features represent the distance to a set of 50 central nodes present in the graph. The central nodes set from the Wednesday data are used as anchor nodes

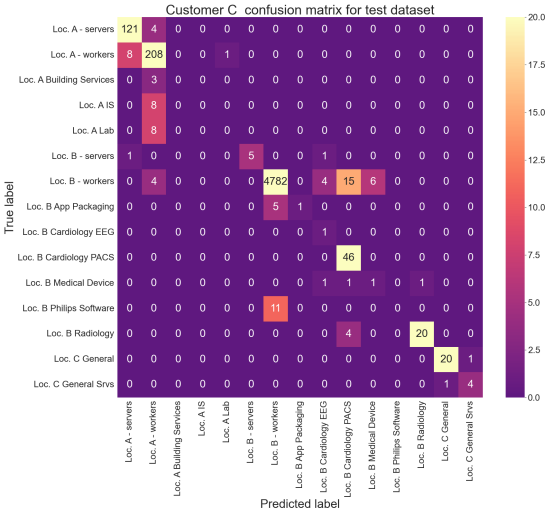

Figure 6: Confusion matrix of the test dataset for Company C. Most of the misclassified devices were assigned to logically similar but more prevalent classes.

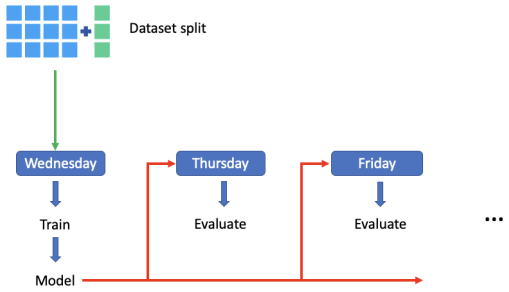

Figure 7: For the prediction stability experiment models were trained on data from 80% of devices from Wednesday and evaluated on the following 6 days to see how long the model can hold the prediction performance.

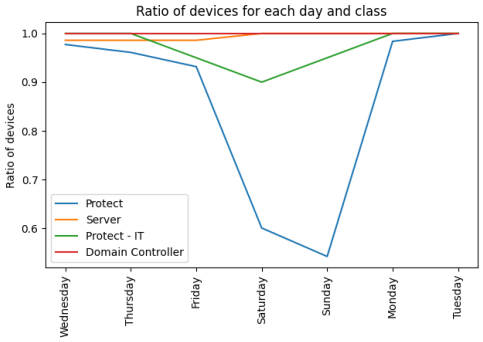

Figure 8: Ratio of active devices throughout the week for Company B.

for evaluation on the remaining six days even though the network graph structure changes.

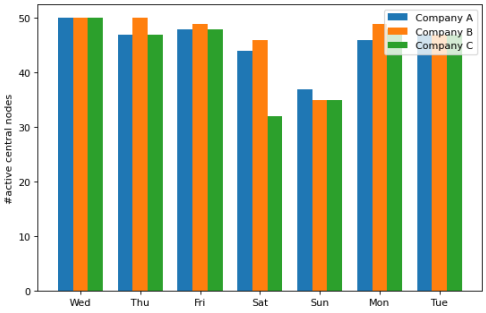

Figure 9: The number of active central nodes in data during the week.

Figure 9 shows that for each company, the number of active central nodes substantially drops during the weekend. Since they are not present in the data, the distance to them cannot be computed and is set to -1 (unreachable). We assume that it is one of the reasons for the poor performance of the graph-based models on weekend data.

4.3.2. Company A

Figure 10 shows the accuracies for all models in Company A. The performance of the GNN model, which was by a large margin best-performing model for this company in the previous experiment, deteriorates rapidly and over the weekend falls below the performance of the baseline models. The network graph structure changes particularly during the weekend as many devices (including central nodes are inactive). Therefore, the GNN model's performance falls very fast and only surpasses the other models on Tuesday, when most devices are active again. The fact that all timestamps of communication logs are stored in UTC, and the private network of Company A lies in multiple different timezones explains why Monday still partially exhibits weekend behaviour.

4.3.3. Company B and Company C

Again, due to the severe class imbalance, we report the macro averages of precision and recall, rather than accuracy, for Companies B and C in Figure 11 and Figure 12, respectively.

Since there is no train/test split for datasets from Thursday to Tuesday and the whole day's worth of data from all active devices used instead, the low prevalence classes have more devices, and the issue with unstable confidence intervals from the previous experiment does not occur. For Company B, the models generally have a sim-

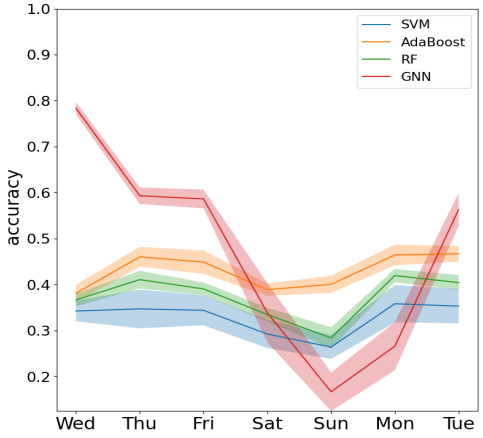

Figure 10: Models' accuracy with 95% confidence intervals for Company A. GNN model's performance deteriorates during the weekend as the network structure changes and several central nodes are inactive.

ilar drop in performance over the weekend but retain their performance on other days.

For Company C, the performance of the GNN degrades notably faster than for the baseline models that do not utilise the graph structure because Company C has devices grouped partially by location, similar to Company A. This claim is further supported by the graph-based models not having this issue for Company B, with devices grouped by function and not location.

4.3.4. Summary

The results indicate that the period for which the models retain their performance depends on a particular network, the nature of labels, and the model itself. For Company A, the by-far best GNN model leverages the graph structure for predictions, but it can't handle the change of the graph structure during the weekend. For Company B, the models' performance drops during the weekend, but the models retain their performance the rest of the days. For Company C, the GNN model degrades, while the other models retain their performance for a longer period. From the previous experiment's perspective, daily retraining of the graph models would prevent performance loss. Appendix B provides a study of the training times of each method to support the viability of daily retaining. Improving graph-based models' time stability will be part of our following research.

5. Conclusion

We have presented a semi-supervised approach to classifying network devices using the inherent graph structure

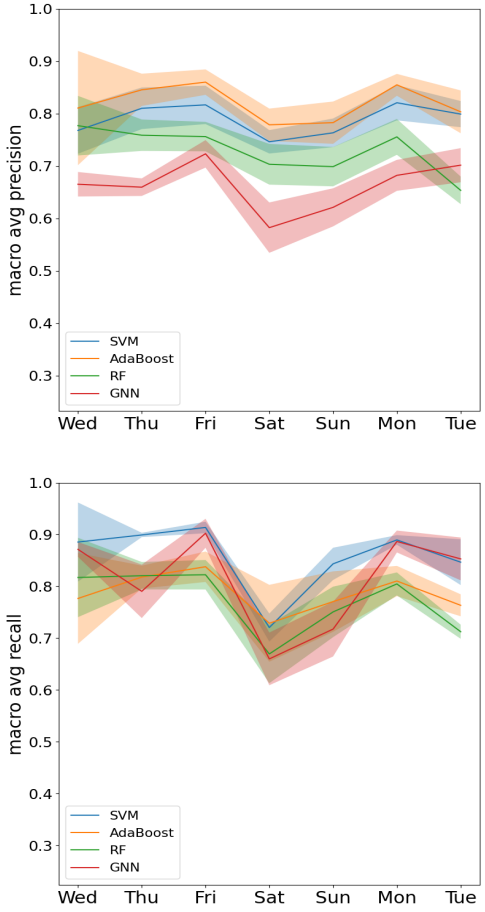

Figure 11: Classification precision and recall with 95% confidence intervals over the week for Company B.

of computer networks. We encoded the global position of the devices in the network by calculating distances to a set of central nodes selected by their PageRank centrality and used graph convolutional neural networks, specifically GATv2 architecture. Compared with the three baseline models, the GNN performed better on geographically structured private networks but struggled with maintaining its performance over the weekend. To overcome this problem, we plan to experiment with methods for classification on dynamic graphs in the future.

References

[1] C.-C. Chang, C.-J. Lin, Libsvm: a library for support vector machines, ACM transactions on intelligent systems and technology (TIST) 2 (2011) 1–27.

[2] A. Criminisi, J. Shotton, E. Konukoglu, et al., Deci-

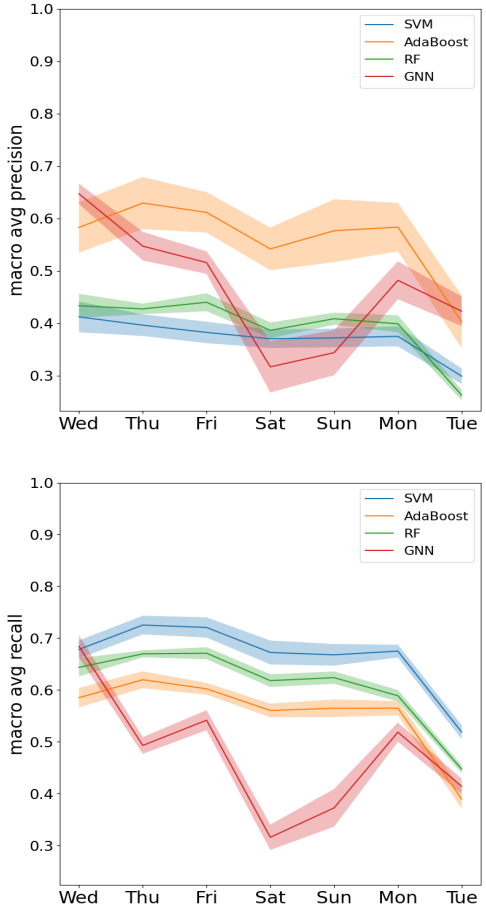

Figure 12: Classification precision and recall with 95% confidence intervals over the week for Company C.

sion forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning, Foundations and Trends® in Computer Graphics and Vision 7 (2012) 81–227.

[3] Y. Freund, R. E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: Computational Learning Theory: Second European Conference, EuroCOLT'95 Barcelona, Spain, March 13–15, 1995 Proceedings 2, Springer, 1995, pp. 23–37.

[4] L. Tang, H. Liu, Leveraging social media networks for classification, Data Mining and Knowledge Discovery 23 (2011) 447–478.

[5] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).

[6] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[7] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th international conference on world wide web, 2015, pp. 1067–1077.

[8] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[9] T. Anglade, C. Denis, T. Berthier, A novel embedding-based framework improving the User and Entity Behav- ior Analysis, 2019. URL: https://hal.sorbonne-universite.fr/hal-02316303, working paper or preprint.

[10] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[11] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems 30 (2017).

[12] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).

[13] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks?, arXiv preprint arXiv:2105.14491 (2021).

[14] H. Cui, Z. Lu, P. Li, C. Yang, On positional and structural node features for graph neural networks on non-attributed graphs, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 3898–3902.

[15] B. Claise, Cisco Systems NetFlow Services Export Version 9, RFC 3954, 2004. URL: https://www.rfc-editor.org/info/rfc3954. doi:10.17487/RFC3954.

[16] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: Bringing order to the web., Technical Report, Stanford InfoLab, 1999.

[17] G. King, L. Zeng, Logistic regression in rare events data, Political analysis 9 (2001) 137–163.

[18] J. N. Van Rijn, F. Hutter, Hyperparameter importance across datasets, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2367–2376.

[19] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, I. Stoica, Tune: A research platform for distributed model selection and training, arXiv preprint arXiv:1807.05118 (2018).

[20] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[21] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, V. Sivaraman, Classifying iot devices in smart environments using network traffic characteristics, IEEE Transactions on Mobile Computing 18 (2019) 1745–1759. doi:10.1109/TMC.2018.2866249.

[22] E. F. de S. Soares, C. A. V. Campos, S. C. de Lucena, Online travel mode detection method using automated machine learning and feature engineering, Future Generation Computer Systems 101 (2019) 1201–1212. URL: https://www.sciencedirect.com/science/article/pii/S0167739X19305874. doi:https://doi.org/10.1016/j.future.2019.07.056.

[23] H. Zhang, Z. Yu, G. Dai, G. Huang, Y. Ding, Y. Xie, Y. Wang, Understanding gnn computational graph: A coordinated computation, io, and memory perspective, Proceedings of Machine Learning and Systems 4 (2022) 467–484.

A. Comprehensive list of features

This appendix covers all the device features used in our research. We inspected the traffic on 14 days of 8 private networks of various sizes to design the following features that can be divided into four categories:

1. Port features
2. Time features
3. Networking features
4. Hash features

The utilized device features (except the graph positional features) are standard features for capturing information in similarly structured data. Before training and evaluating the models, min-max scaling was applied to the described features.

A.1. Port features

Source and destination ports of network communication are well-known sources of behavioural information about the device [21]. They contain information about what services are most likely running on the device and what services it is communicating to.

Therefore, the incoming versus outgoing telemetry ratio to each service listed in Table 4 is calculated and used as features, together with the total number of unique source and destination ports observed.

service	ports
FTP	20,21
SSH	22
Telnet	23, 992
SMTP	25
DNS	53
DHCP	67,68
HTTP	80, 8080, 8008, 8081
NetBios	135-140
BGP	179
LDAP	389
HTTPS	443
LDAP secure	636
FTP secure	989,990
SMB	445
Kerberos	88
SNMP	161
NTP	123
IPP	631
Certificate Management Protocol	829
ISAKMP	500
Sun RPC	111
RLZ DBase	635
webservice unassigned	81
SNPP	444
Multicast DNS	5353
SSDP	1900
Remaining well-known	0-1023
Remaining registered	1024-10 000
Remaining private/ephemeral	>10 000

Table 4
Prevalent services observed in the communication logs with their port numbers.

A.2. Time features

The device's activity is described by the times it communicated or was silent. Firstly, we added 24 features, representing 24 hours, and calculated percentages of the total communication for each respective hour. Then, the amount of daily active hours to see how much it communicates daily. Lastly, we compute skewness and kurtosis of time differences between each communication to represent whether the device is active periodically or in bursts. We chose skewness and kurtosis based on their success on the time series presented in [22].

A.3. Networking features

The networking features cover communication statistics in different directions and over different transport protocols. The following list shows the features we use:

- number of all logs
- number of inbound logs
- number of outbound logs
- percentage of outbound logs
- percentage of TCP logs
- percentage of UDP logs
- percentage of private logs

- number of IP addresses in all logs
- number of IP addresses in incoming logs
- number of IP addresses in outgoing logs

A.4. Hash features

The hash features are only available by collecting connection logs directly on the endpoint. The binaries that initiated the connections are grouped according to their prevalence to three intervals: the top 10%, in the range 10% - 50%, and the rest. Then, the ratios of binaries the device uses over the sum of all logs in a given day are calculated. This help to identify device specific vs company-specific applications.

B. Training time comparison

This section analyses the time needed for training each model. Because GNN requires additional graph-based features, we included preprocessing times in the evaluation.

Figure 13, and Figure 14 display the standalone training times for all models on Company A and C. The measured times for Company B are almost identical to Company A. The times for GNN are reported for training on the CPU and GPU for AI inference, the NVIDIA T4 Tensor Core GPU. The GPU training is around five times faster compared to the training on a CPU.

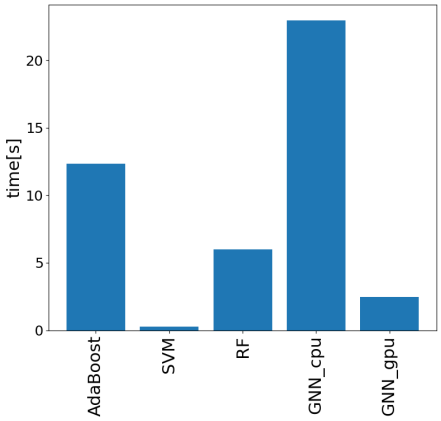

Figure 13: Training times of each model for Company A. The training times were averaged over ten runs.

There is a considerable difference in resource consumption among the models, especially on the largest network of Company C. For better overall assessment, all data processing must be considered. Therefore, Figures 15, and Figure 16, respectively, show the time taken to process raw logs, create the respective dataset, and train each

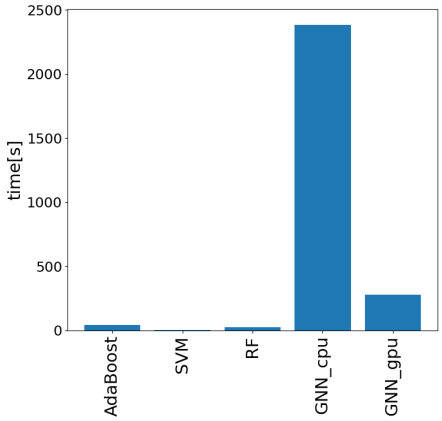

Figure 14: Training times of each model for Company C. The training times were averaged over ten runs.

model. The device-matching algorithm is a constant specific to the endpoint data. Overall, device matching and computation of device features consume the majority of the time.

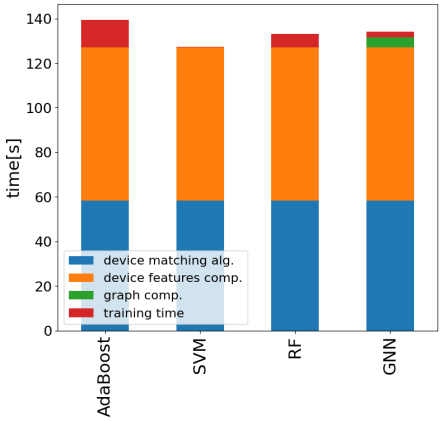

Figure 15: End-to-end processing times for each of the models on Company A.

For the largest network, Company C, the GNN takes longer due to issues discussed, for example, in [23]. The training times could be reduced by, e.g., a neighbourhood sampling mechanism or a simpler neighbourhood aggregating scheme, resulting in smaller computational graphs and faster forward and backward passes during training.

One of the key takeaways from the presented comparisons is that most of the time is consumed by preprocessing raw logs rather than by model training. Furthermore, the models do not have to be retrained from scratch but only fine-tuned for the current data, further reducing the required training time.

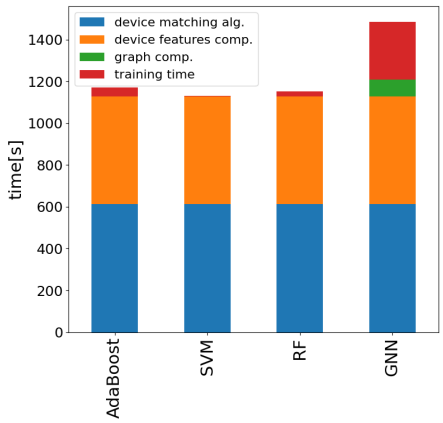

Figure 16: End-to-end processing times for each of the models on Company C.

C. GNN hyperparameters

Table 5 presents the resulting optimal hyperparameters of the model and training procedure for each company found through a hyperparameter grid search.

parameter	Cust. A	Cust. B	Cust. C
learning rate	0.003	0.01	0.004
num_heads	8	4	10
embedding_dim	4	4	40
l2_reg	0.004	0.04	0.001
dropout	0.4	0.2	0.1
n_epochs	400	200	250

Table 5
The obtained set o optimal hyperparameters for the GNN model for each company. These hyperparameters were then used for each model training in the described experiments.

Adam optimizer was used for training the neural networks with the respective learning rate and hyperparameters.