



Enhancing Cybersecurity Through Comparative Analysis of Deep Learning Models for Anomaly Detection

Kateřina Mackov¹^a, Dominik Benk¹ and Martin řrot²^b

¹Novicom, s.r.o., Na schodech 65/1, 140 00, Prague, Czech Republic

²Faculty of Transportation Sciences, Czech Technical University in Prague, Konviktsk 20, 110 00, Prague, Czech Republic

Keywords: Cybersecurity, Deep Learning, System Log, Dataflow.

Abstract: With the increasing complexity of cyber attacks, traditional methods for anomaly detection in cybersecurity are insufficient, leading to the necessity of integrating deep learning and neural network approaches. This paper presents a comparative analysis of the most powerful deep learning methods for such anomaly detection. We analysed existing datasets for syslog and dataflow, compared several preprocessing methods and identified their strengths and weaknesses. Additionally, we trained and evaluated several deep learning models to provide a comprehensive overview of the current state-of-the-art in cybersecurity. The CNN model achieves excellent results, with 0.999 supervised and 0.938 semi-supervised F1-score in syslog anomaly detection on the BGL dataset and 0.985 F1-score in dataflow anomaly detection on the NIDS dataset. This research contributes to the field of cybersecurity by aiding researchers and practitioners in selecting effective deep-learning models for robust real-life anomaly detection systems. Our findings highlight the reusability of these models in real-life systems.

1 INTRODUCTION


Cybersecurity, a subset of computer science focused on defending computer systems and networks, faces a rising tide of sophisticated attacks. Traditional methods using keywords and pattern detection fall short, leading to the adoption of deep learning (DL) and neural network techniques for improved anomaly detection.


In this paper¹, we present a summary of existing datasets for syslog and dataflow and compare different preprocessing methods with emphasis on their benefits and disadvantages. We compare different approaches of AI models based on the most common deep learning methods to provide a general summary of the current state-of-the-art in the area of cybersecurity. Moreover, we emphasized the reusability of these models in real-life systems and we evaluate them accordingly. For the syslog, we compared several models and the syslog preprocessing approaches

on several datasets from the LogHub collection. We reached the best results on the BGL dataset with the CNN model with 0.999 supervised and 0.938 semi-supervised F1 score. We compared NIDS and CTU-13 datasets and different approaches to dataflow preprocessing. We reached the best results with the CNN model on the NIDS dataset with a 0.985 F1 score highlighting the reusability of CNN models in real-life systems.

2 MOTIVATION

Syslog and dataflow, easily accessible from net flow, provide crucial insights into network activities and cybersecurity threats. While other network data is challenging to preprocess, syslog and dataflow offer comprehensive coverage for anomaly and attack detection. In our paper, we chose these sources due to their accessibility and comprehensive representation of network aspects. Utilizing deep learning technologies, we can create robust systems for monitoring network security and detecting attacks. Deep neural networks outperform traditional methods and manual rule creation (Chen et al., 2021), (Tang et al., 2016), thanks to their multiple layers that efficiently handle

^a <https://orcid.org/0000-0001-9815-2763>

^b <https://orcid.org/0000-0003-0049-4381>

¹This research is a part of Project TM03000055, cofinanced from the state budget by the Technology Agency of the Czech Republic under the DELTA2 Programme for applied research, experimental development and innovation.

complex features and computational tasks.

Existing papers analyse either anomaly detection in syslog or anomaly detection in dataflow. They present different approaches and algorithms on several datasets. Therefore, it is impossible to compare their results and the methods directly. We present a transparent analysis and comparison of existing approaches on the most common datasets for syslog Loghub (He et al., 2020) and the most common datasets for dataflow NIDS (Sarhan et al., 2021) and CTU-13 (Garcia et al., 2014). This paper is organized into two main sections. In the first part, we present current methods and datasets followed by experiments and the discussion of the results for syslog and in the second part we do the same for dataflow. In the end, we sum up both of them in a common conclusion.

3 SYSLOG

A device, system or application performs actions with certain parameters during its operation. The information about these actions is stored in the system log (syslog). Syslog is an unstructured text message written by the programmer during the development of a given application, system or software. It consists of characteristics of the syslog source (IP, MAC, DHCP, system information, type, name) and operational information that describes what happened on that source. However, as there is no general structure, the contents of these messages can vary a lot and the collection of such data and their parsing for further anomaly and attack detection is complicated. It is necessary to develop general systems able to process all these data and extract important information from them to reveal possible attacks.

3.1 Literature Review

Most of the current experiments on anomaly detection in syslog were performed on the largest collection of labelled and unlabeled datasets for syslog anomaly detection LogHub (He et al., 2020) dataset which is the only dataset used in all the related work. Unfortunately, no other labelled large enough datasets were found.

The best results are obtained using deep learning-based models which are significantly surpassing classical ML methods (Chen et al., 2021) and (He et al., 2016). They have strong generalization capabilities that can help with handling unknown logs. Several DL approaches and processing methods are compared in related work but a comprehensive overview of all

Table 1: Sizes and proportions of train and test data of individual syslog datasets.

Dataset	HDFS.1	BGL	Hadoop	Open Stack	Thunderbird(5%)
Labelled	✓	✓	✓	✓	✓
Time span	38.7h	214.7d	-	-	244d
#Messages	11,175,629	4,747,963	394,308	207.82	10,560,610
(% anomalies)	(2.58%)	(7.34%)	(93.47%)	(8.87%)	(3.77%)
Data size	1.47GB	708.76MB	48.61MB	60.01MB	1.48GB
#Templates	30	619	298	51	4040
#Train sessions	460048	662	782	1656	449
(% anomalies)	(2.92%)	(50.3%)	(82.23%)	(9.48%)	(44.77%)
#Test sessions	115013	166	196	414	113
(% anomalies)	(2.94%)	(45.18%)	(85.71%)	(10.14%)	(42.48%)

combinations of the best deep learning-based methods with the best log preprocessing methods is not available in any.

3.1.1 Datasets

Loghub (He et al., 2020) is a collection of 17 system log datasets with a total size of 77GB created by a group of scientists at the Chinese University of Hong Kong (CUHK). Logs come from real traffic in a laboratory environment from 6 different systems: distributed supercomputer systems, operating systems, mobile systems, server applications and standalone software. It contains a wide variety of different data natures and formats, of which 5 datasets are labelled by human annotators and can be used for training deep learning models.

In this paper, we used only 5 labelled datasets from the whole Loghub collection: BGL, Hadoop, HDFS_1, OpenStack, and Thunderbird. HDFS_1, OpenStack and Hadoop were collected from distributed file system logs, and BGL and Thunderbird were generated from Blue Gene/L supercomputer log. We selected these datasets as they are labelled and both supervised and semi-supervised methods can be easily evaluated on them. They differ in the source of logs, the size of the datasets, the time span during which the data were collected and the number of sessions and percentage of anomalies, see Table 1. Note that, from Thunderbird, we used only 5% of its original size for train and test session creation because of the limited capacity during the computation.

3.1.2 Models

This section describes the deep learning models for syslog anomaly detection evaluated on BGL and HDFS datasets.

DeepLog. Deeplog (Du et al., 2017) is a method based on recurrent neural networks (RNN) which allows the output from certain nodes to affect subsequent input to the same nodes and is widely used in natural language processing tasks. They preserve dependencies in time and remember patterns to predict

the next likely scenario based on previously seen data. DeepLog parses syslog and converts them to vectors called embeddings using word2vec. Word2vec uses a neural network model to learn word associations from a large unlabelled corpus of text and creates embeddings keeping similar words close and distant words far from each other.

LogAnomaly. LogAnomaly (Meng et al., 2019) is another RNN-based model, which is similar to DeepLog but instead of word2vec uses a template2vec approach to create syslog embeddings. Template2vec uses a special language model that was retrained to take into account synonyms, antonyms and similar word relationships in the context of logs and not from general texts as in the case of word2vec.

Logsy. Logsy (Nedelkoski et al., 2020) is a deep learning technique based on Transformer language models which are deep learning architecture that relies on the parallel multi-head attention mechanism that allows words from a given text to focus only on the important words from the text requiring less training time than RNN. Transformer learns to distinguish between normal data from systems and anomalies and also it can predict the next log in the sequence based on language semantics and syntax learned from large amounts of unlabeled data.

Autoencoder. Autoencoder (Zhang et al., 2019) is a type of neural network mostly used in unsupervised setups which learns to efficiently encode unlabeled data to a representation with a reduction of the data dimensionality. It consists of two neural networks: the encoder maps the message to a code, and a decoder reconstructs the message from the code optimally with zero difference. During the detection of anomalies in syslog, only the encoder is important to extract and encode the most valuable information.

Convolutional Neural Network. Convolutional neural networks (CNN) (Lu et al., 2018) is a type of neural network that learns features via filter (or kernel) optimization using a special function called convolution and it also can detect anomalies in logs under a supervised setup.

3.2 Datasets Preprocessing

3.2.1 Sessions and Windows

According to the original paper (He et al., 2020), the data were grouped into sessions which contain a set of related logs and the results are evaluated on those. Sessions are useful for preserving relations among logs and therefore, we kept the sessions in this paper too. In HDFS_1 (and OpenStack), the sessions are marked by identifier block_ID (and instance_id).

Hadoop has no identifier but grouping is easily possible by featured application_id and container_id. BGL (and Thunderbird) could only be grouped into sessions by time, the most appropriate length was set to be 6 hours (and 1 hour) (He et al., 2016), see Table 1

During the evaluation, some of the sessions were too long so we also grouped all of the datasets by fixed-length windows of 10 logs to speed up the detection process. However, during the experiments, we also tested different lengths. During training, we also used sessions as mentioned above. The whole session was considered anomalous if at least one of the logs was labelled as an anomaly. After the creation of sessions and windows, we performed data preprocessing: the log is parsed into templates and the templates are converted into numerical vectors. we compared several techniques to find the most suitable for each dataset to deduce general rules.

3.2.2 Template Parsing

We parsed the original log message into the template using three methods: 1) Sophisticated template parsing, 2) Regular expression to remove variables from the full log message, and 3) Regular expression. Each log is parsed into a template by extracting variables and parameters such as numbers, time stamps, IP addresses or dates to keep only the textual part. Sophisticated template parsing uses special parsing algorithms for this: Spell (Du and Li, 2016), Drain (He et al., 2017) or IPLoM (Makanju et al., 2009). Firstly, we used Spell on all datasets but the number of parsed templates was often far from reality so we selected the best-performing algorithm for each dataset based on experiments performed by authors of (Zhu et al., 2019): Spell for the HDFS_1, IPLoM for the OpenStack, Drain for for BGL, Hadoop, and Thunderbird.

Although sophisticated methods work with regular expressions in the initial step to enhance parsing accuracy, we exclusively extracted main benchmark parameters (Zhu et al., 2019). Evaluation involved comparing the number of extracted templates in the training dataset, testing dataset, and out-of-vocabulary templates in the test dataset for all three methods (see Table 2).

3.2.3 Template Embedding

We converted created templates into numerical vectors called embeddings by two approaches: 1) Random embeddings and 2) Log2Vec semantic embeddings (Liu et al., 2019). Random embeddings (often referred to as sequential) extract unique templates and randomly number them and no semantic meaning of the logs is included. Log2Vec is based on

Table 2: Number of templates from train and test logs in all syslog datasets.

Dataset	Data type	# Train templates	# Test templates	#Test OOV templates
BGL	Drain templates	1645	614	218
	Regex full	3381	1548	369
	Regex content	1461	666	153
HDFS.1	Spell templates	35	36	2
	Regex full	71	63	1
Hadoop	Regex content	71	63	1
	Drain templates	657	405	51
	Regex full	770	623	23
OpenStack	Regex content	722	593	21
	IPLoM templates	1705	445	413
	Regex full	5233	1503	1017
Thunderbird (5%)	Regex content	2090	539	470
	Drain templates	7443	1091	165
	Regex full	3060	1903	69
	Regex content	1786	1128	39

Table 3: Numbers of triplets, synonyms and antonyms.

Dataset	Triplets	Synonyms	Antonyms
BGL	423	1070	54
HDFS.1	29	95	1
Hadoop	305	933	59
OpenStack	883	229	185
Thunderbird (5%)	630	3021	173

the idea of word2vec but in the context of logs involving the semantics of words in the embedding creation. The emphasis is put on antonyms, synonyms and meaning triples. Especially logs with opposite words should be far from each other as they have a significant impact on anomaly detection². We trained the Log2Vec model for every dataset separately as the creation of one Log2vec model for all gave very poor results. Moreover, a special model called MIM-ICK³ was trained to predict embeddings for out-of-vocabulary words and to generate embeddings for unseen logs. We computed the numbers of synonyms, antonyms and triplets found in each dataset, see Table 3.

3.3 Experiments

We tested two approaches: 1) Anomaly prediction and 2) Next-log prediction. Anomaly prediction is a supervised approach to predicting an anomaly or normal behaviour in the sequence of logs. Next-log prediction is a semi-supervised approach to predicting the probability distribution of all logs that can follow

²For example embeddings for *TURN ON* and *TURN OFF* would be close in word2vec, which is not desirable in case of logs.

³Model mimicking the prediction of Log2Vec but on a character-level.

Table 4: F1 scores of all syslog experiments after selecting the best model for each supervised setup.

Dataset	Data type	Anomaly predictions					
		Label type		Session			
		Eval type	Without	With	Without	With	
BGL	semantics	RF ^a	RC ^b	TEM ^c	RF ^a	RC ^b	TEM ^c
	sequentials	0.999	0.999	0.993	0.992	0.991	0.996
HDFS.1	semantics	0.999	0.999	0.993	0.981	0.982	0.995
	sequentials	0.979	0.973	0.959	0.635	0.641	0.637
Hadoop	semantics	0.967	0.975	0.98	0.64	0.637	0.641
	sequentials	0.899	0.899	0.923	0.992	0.992	0.994
OpenStack	semantics	0.899	0.899	0.923	0.992	0.992	0.994
	sequentials	0.158	0.156	0.191	0.154	0.154	0.181
Thunderbird	semantics	0.214	0.175	0.188	0.15	0.15	0.172
	sequentials	0.999	0.999	0.99	0.993	0.993	0.993
		0.999	0.999	0.99	0.994	0.994	0.994

the given log sequence. If the observed log is different from the top k^4 predictions, the anomaly is raised.

Additionally, we compared several deep-learning techniques to find the best model for anomaly detection. We involved LSTM, CNN and Transformers for both supervised and semi-supervised approaches. These models were selected based on the experiments in (Chen et al., 2021) and the extensive overview of methods provided in (Li and Jung, 2022).

During the training process, we considered several preprocessing methods of template creation as described in Section 3.2.2 which we combined with both techniques for embedding creation described in Section 3.2.3. We tested other hyperparameters including the percentage of training data, window size, window stride size, whether adding an attention layer helps and the optimal number of predictions in a semi-supervised setup. We selected the best models based on the F1 score.

3.4 Results

We compared the results of the best-performing models from different combinations of setups described in the previous section, see Table 4 and 5. In this set of experiments, we fixed the following parameters for all models: $K = 10$, $strides = 1$, $window_size = 10$, and no attention.

Datasets. The results vary a lot among the datasets. Real-life use cases would require to selection of the dataset closest to the real data. BGL and Thunderbird behave similarly, they have a high difference in supervised and semi-supervised performance. HDFS.1 has very accurate predictions in both supervised and semi-supervised approaches but has a very low number of templates indicating its unsuitability in reality. The results

⁴Hyperparameter optimized for each dataset.

Table 5: F1 scores of all syslog experiments after selecting the best model for each semi-supervised setup.

Dataset	Label type		Next log predictions				
	Data type	Session			Without		
		Eval type	RF ^a	RC ^b	TEM ^c	RF ^a	RC ^b
BGL	semantics	0.678	0.709	0.688	0.54	0.668	0.87
	sequentials	0.678	0.712	0.701	0.512	0.664	0.885
HDFS_1	semantics	0.956	0.964	0.963	0.387	0.369	0.388
	sequentials	0.969	0.958	0.954	0.393	0.369	0.414
Hadoop	semantics	0.901	0.903	0.923	0.97	0.976	0.714
	sequentials	0.903	0.903	0.923	0.962	0.962	0.558
OpenStack	semantics	0.168	0.196	0.184	0.154	0.145	0.17
	sequentials	0.165	0.188	0.184	0.146	0.146	0.168
Thunderbird	semantics	0.604	0.604	0.604	0.658	0.626	0.62
	sequentials	0.604	0.604	0.604	0.826	0.658	0.636

^a Regex full, ^b Regex content, ^c Sophisticated template parsing.

Cell colours: CNN, LSTM, Transformer.

from other datasets are ambiguous: Hadoop has very low variability in F1 score and OpenStack is very difficult to train in general⁵. This might also be the reason that the literature usually employs only BGL and HDFS_1.

Sessions and Windows. Sessions have significantly higher F1 scores and they are commonly used for evaluation in literature. However, the sessions are often not marked in real data and the evaluation of several hours long time sessions can cause delay for early warning or adequate reaction in practice. Therefore, we considered a window evaluation more suitable for the reality.

Supervised and Unsupervised. A supervised approach generally yields better results, but labelled data is usually not available in reality. Therefore, a semi-supervised approach should provide more valuable insights about real performance. Even though some of the semi-supervised models appear to be underperforming compared to (Chen et al., 2021), we believe this can be fixed by fine-tuning, especially the K parameter.

Template Creation. Sophisticated parsing and regex full and regex content methods reach similar results. Sophisticated parsing usually is slightly better, but requires knowledge about the dataset and slows down the whole preprocessing. An interesting observation was that none of the sophisticated parsing algorithms was universal for all datasets. Therefore, the regular expression parsing methods in reality are more suitable.

Embeddings Creation. The results when comparing Log2Vec and embeddings without semantics are also surprising. According to (Liu et al.,

⁵Probably caused by difficulties during template parsing as suggested by experiments (Zhu et al., 2019).

2019), Log2Vec should significantly improve the results but as to our analysis, the results were nearly the same and Log2Vec only slowed the process. Nevertheless, we still believe that Log2Vec has a lot of exploring potential if the model is further tuned and optimized, however, its inclusion in reality is not suitable.

Model Architectures. When comparing different model architectures, all of the models performed reasonably well. In general, LSTM and CNN have better results than Transformers. LSTM appeared to be more suitable for the semi-supervised approach, but it was significantly slower than CNN. Therefore, CNN is expected to be the best choice for the real-life use case as proved in (Chen et al., 2021).

Train Data Size. We experimented with different train set sizes and found out that in the case of datasets with very few templates, only 1% percentage of data was enough to achieve reasonable performance. This indicates that these datasets are far from reality, where thousands of different log templates appear.

Additionally, we ran another experiment with the usage of a semi-supervised CNN trained on Regex content with semantic embeddings and only 50% of data to save time. The results are the following.

Number of Predictions (K). The number of predicted most probable logs had a significant impact and its increasing size helped to achieve much better results in BGL and Thunderbird. For example, if we change the parameter K from 10 to 200 in BGL, we get an astounding improvement in F1-score from 71.2% to 93.8% which is much closer to the findings from literature (Chen et al., 2021).

Window Size. Increasing window size helps to improve results during window evaluation, but longer windows are significantly more demanding on memory. This is not surprising as each window can consider more information and make more accurate predictions.

Other Hyperparameters. According to the size of strides, there is no general rule to improve the results. We believe increasing the size might only be beneficial during training. The final observation was that the attention layer had a minimal impact on the results.

4 DATAFLOW

Dataflow includes vital information detailing network data flow characteristics between two devices: IP and

MAC addresses of source and target devices, communication ports and protocols, packet and byte sizes, and communication start time with duration. Due to computational demands, specific exchanged data is excluded. Typically formatted in standardized IPFIX or Netflow formats, parsing and preprocessing complexity is significantly reduced.

4.1 Literature Review

Dataflow is usually collected in a standardized format so its processing is not as demanding as in the case of syslogs. It includes a selection of the most important features either based on general knowledge or using tools such as PCA or correlation matrix to reduce the dimensionality of data with minimal loss of information. Except for IP addresses, all features can be preprocessed by categorisation and normalisation. IP addresses can be preprocessed using categorisation or IP2Vec (Ring et al., 2017). In comparison to syslog, the field of dataflow research exhibits weaker performance, potentially attributed to the lower complexity of data.

4.1.1 Datasets

Most of the current experiments on anomaly detection in dataflow were performed on NIDS (Sarhan et al., 2021), CTU-13 (Garcia et al., 2014) or KDD Cup 1999 (Hettich, 1999) datasets, which are the most common datasets for dataflow anomaly detection. As KDD Cup 1999 is very small compared to the others, we omit it and perform the experiments only on NIDS and CTU-13.

NIDS (Sarhan et al., 2021) dataset is a collection of 5 datasets of dataflow collected from several sources, which reflect normal behaviour combined with several types of attacks. We used only *NF-UQ-NIDS*, which is a combination of all other 4 datasets *NF-UNSW-NB15*, *NF-BoT-IoT*, *NF-ToN-IoT*, and *NFCSE-CIC-IDS2018*. In total, NIDS contains 11,994,893 flows from which 2,786,845 (23.23%) are anomalies.

CTU-13 (Garcia et al., 2014) is a collection of 13 scenarios of botnet attacks mixed with normal flow. The idea is to capture real botnet traffic mixed with normal and background traffic in several scenarios. A specific malware that uses particular protocols and performs different anomalies is simulated in each scenario. For our purposes, we combined all of these scenarios which resulted in a total of 19,474,237 flows from which 331,852 are anomalous (1.70%).

4.1.2 Models

Deep learning-based models reach the best results for detecting anomalies in dataflow and they significantly outperform classical ML methods (Tang et al., 2016). Similarly, as in the case of syslog, the most promising existing experiments include RNN, CNN and Autoencoders. They were trained along with several types of classical neural networks and Generative Adversarial Networks on a preprocessed and cleaned version of the original KDD dataset called KDD Cup 99 in (Podder et al., 2021) and the best results were obtained with RNN. The convenience of RNN was also confirmed by other experiments in (Kim et al., 2016). However, CNN achieved comparable results (Kim and Cho, 2018) so both of these approaches were combined into a C-LSTM model. It consists of several convolutional and recurrent layers in a linear structure, where each layer extracts different features. Models were trained on the Webscope S5 dataset, which contains 367 time series and each has a length of 1500 dataflows. According to the results, traditional RNN and CNN approaches were significantly outperformed by C-LSTM. We decided to employ larger data and verify these results along with different IP preprocessing approaches.

4.2 Datasets Preprocessing

All the characteristic features in dataflow are numerical or categorical except for IP. We preprocessed features either by normalization⁶ or categorization⁷. We employed several techniques to preprocess IP addresses as they have a more complicated structure.

Normalization. During normalization, we performed a standard min-max normalization. We normalized the features sum of TCP flags and flow duration columns independently. In the case of packets and bytes features, we normalised both input and output columns together using common min and max values to preserve the relationship between those values.

Categorization. We limited the number of categories in some features to improve the generalization of the model. In feature protocol number (0-255), all protocols up to 145 were kept as unique, and protocols higher than this were merged to a single category⁸. Similarly, port numbers (0-65536) higher than 49151 were merged to one category⁹ as all those numbers

⁶packets and bytes, sum of TCP flags, flow duration

⁷ports, protocols, flow direction

⁸<https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>

⁹<https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>

Table 6: Number of rows, anomalies and unique values in dataflow datasets.

		NIDS		CTU-13	
		Train	Test	Train	Test
Number of rows		8996169	2998724	15579389	3894848
Number of anomalies		2091125	695720	265649	66203
Unique values	source_ip	65613	38015	1787875	708206
	target_ip	25698	18122	469689	206522
	source_port	4064	3969	4165	3969
	target_port	4082	3944	4010	3889
	protocol	147	147	7	6
	17_protocol	127	122	-	-
	input_bytes	37635	24388	82339	38882
	output_bytes	64700	35061	-	-
	total_bytes	-	-	220784	104478
	input_packets	4310	2086	-	-
	output_packets	2830	1667	-	-
	total_packets	-	-	6149	12668
	sum_tcp_flags	48	49	-	-
	flow_duration	108417	72597	4298221	1320011
	direction	21	10	6	6
label	2	2	2	2	
attacks	21	10	12	12	

are unassigned. The other ports were grouped based on the first word of the name⁹.

IP Addresses. During the IP preprocessing, we experimented with several approaches: 1) convert each unique IP to a new category, 2) grouping all IPs into 6 categories (host, private network, subnet, documentation, internet, software)¹⁰ and 3) IP2Vec. IP2Vec (Ring et al., 2017) is a new method for IP preprocessing based on the same principle as word2vec. The context of IPs is learnt from other features and then a vector embedding is calculated, grouping IPs with similar behaviour close to each other, while keeping IPs with distinct behaviour far away.

We created 12 numerical features for NIDS and 9 for CTU-13. Along with the binary label of whether the flow is anomalous, we trained several AI models for anomaly prediction. We computed the descriptive statistics of all features from both datasets split into train and test, see Table 6. Note that NIDS are approximately half the size of CTU-13 and that anomalies are significantly less common than normal flows. We solved the insufficient amount of anomalies by appropriately upsampling.

4.3 Experiments

We employed several DL techniques to find the best model for anomaly prediction evaluated on CTU-13 and NIDS datasets. We used CNN, and RNN, and we

¹⁰Inspired by (Cotton et al., 2013)

Table 7: Results of all dataflow experiments on NIDS dataset.

		Raw IP		Cat IP		IP2Vec	
Model	Upsample	No	Yes	No	Yes	No	Yes
NN	Accuracy	0.980	0.983	0.870	0.508	0.873	0.528
	Precision	0.965	0.952	0.985	0.319	0.990	0.330
	Recall	0.946	0.976	0.448	0.991	0.456	0.999
NN_HC	F1	0.956	0.964	0.616	0.483	0.625	0.496
	Accuracy	0.768	0.234	0.864	0.869	0.867	0.386
	Precision	0.000	0.233	0.987	0.966	0.992	0.274
RNN	Recall	0.000	0.999	0.418	0.450	0.430	0.999
	F1	0.000	0.377	0.587	0.614	0.600	0.430
	Accuracy	0.966	0.947	0.987	0.984	0.993	0.989
CNN	Precision	0.933	0.835	0.989	0.959	0.986	0.954
	Recall	0.920	0.962	0.954	0.971	0.983	0.999
	F1	0.927	0.894	0.971	0.965	0.984	0.976
RCNN	Accuracy	0.986	0.985	0.986	0.983	0.993	0.989
	Precision	0.970	0.958	0.989	0.959	0.994	0.955
	Recall	0.970	0.979	0.952	0.970	0.976	0.999
RCNN	F1	0.970	0.968	0.970	0.965	0.985	0.976
	Accuracy	0.986	0.983	0.986	0.983	0.993	0.988
	Precision	0.966	0.950	0.989	0.958	0.990	0.953
RCNN	Recall	0.972	0.979	0.953	0.969	0.979	0.999
	F1	0.969	0.964	0.970	0.964	0.985	0.976

also experimented with RCNN (Kim and Cho, 2018), which combines the best from CNN and RNN to achieve higher performance. Additionally, we compared those techniques with simple neural networks (NN) and simple neural networks with high capacity (NN_HC) to verify the necessity of advanced deep learning methods.

Also, as we already noted, the ratio of anomalies in both training datasets is very low. Therefore, we included upsampling and evaluated the models with and without it. In the case of NIDS, anomalies were upsampled 4 times, while in CTU-13 we upsampled all scenarios individually to balance the total number of anomalies.

4.4 Results

We compared the results of all neural network models on both NIDS and CTU-13 datasets, see Tables 7, 8. The interpretation is split into the following parts.

Datasets. The NIDS dataset has significantly better results than CTU, as confirmed in other papers (Maimó et al., 2018). We reached similar results with NIDS as in (Sarhan et al., 2021) and (Kim and Cho, 2018). With CTU-13, the results were much worse except for models with IP2Vec, which improves the results in deep-learning models up to an F1 score and accuracy of 0.99. This demonstrates the strong dependency on IP addresses in the CTU-13 dataset, which is the reason why addresses are completely excluded during training in other litera-

Table 8: Results of all dataflow experiments on CTU-13 dataset.

Model	Upsample	Raw IP		Cat IP		IP2Vec	
		No	Yes	No	Yes	No	Yes
NN	Accuracy	0.983	0.983	0.993	0.945	0.999	0.999
	Precision	0.000	0.000	0.869	0.230	0.988	0.974
	Recall	0.000	0.000	0.678	0.955	0.946	0.947
	F1	0.000	0.000	0.762	0.371	0.967	0.960
NN_HC	Accuracy	0.983	0.017	0.993	0.907	0.999	0.999
	Precision	0.000	0.017	0.851	0.149	0.993	0.958
	Recall	0.000	0.999	0.693	0.954	0.945	0.979
	F1	0.000	0.033	0.764	0.258	0.968	0.968
RNN	Accuracy	0.987	0.968	0.994	0.985	0.999	0.999
	Precision	0.877	0.296	0.800	0.525	0.999	0.999
	Recall	0.288	0.652	0.826	0.929	0.999	0.999
	F1	0.434	0.407	0.813	0.671	0.999	0.999
CNN	Accuracy	0.983	0.921	0.993	0.980	0.999	0.999
	Precision	0.496	0.168	0.811	0.460	0.999	0.999
	Recall	0.024	0.923	0.764	0.930	0.999	0.999
	F1	0.045	0.284	0.787	0.616	0.999	0.999
RCNN	Accuracy	0.983	0.912	0.988	0.980	0.999	0.999
	Precision	0.528	0.154	0.869	0.453	0.999	0.999
	Recall	0.049	0.927	0.365	0.932	0.999	0.999
	F1	0.089	0.264	0.514	0.610	0.999	0.999

ture (Nguyen et al., 2022). Surprisingly, NIDS does not have this problem and it might be caused by more features in the source dataset.

Upsampling. We reached better results without upsampling the anomalies in training data in almost every type of model. This was not expected as the model's learning is usually improved by balancing the data. We are not sure about the origin of this and it might be interesting to perform more experiments to verify this problem.

IP Preprocessing. In all models except for simple NN, IP categorization into 6 categories helps to achieve better results compared to keeping original IPs as single categories. This indicates that simple NN is strongly dependent on particular IP addresses. IP addresses are not stable and every time one device logs into the network, it can have different IPs. Therefore, ungrouped IP addresses should be definitely omitted. IP2Vec can slightly improve the results but it involves a high risk of dependency on IPs. The IP2Vec creation significantly slows the processing and therefore, we do not see much benefit in real-life models in spite as claimed in (Ring et al., 2017).

Model Selection. Deep-learning-based methods (CNN, RNN, RCNN) have significantly better results in general confirming the complexity of data and the necessity for advanced predictive models. The best results seem to be obtained by RNN. However, the recurrence in these is not within the following flows, but instead, it is involved in the features in one flow caused by the missing time information in the NIDS,

making it impossible to order the flows and evaluate the dependencies among them. Therefore, we selected CNN as the best model, it slightly overperforms RCNN and we do not confirm the benefits claimed in (Kim and Cho, 2018). As to the simple NN and simple NN_HC, our hypothesis with an insufficient capacity of simple NN was not confirmed as simple NN_HC had significantly worse results and the necessity of advanced learning methods was confirmed.

5 CONCLUSION

In conclusion, our study underscores the vital role of deep learning models in cybersecurity anomaly detection, surpassing traditional methods that prove insufficient in the face of increasing cyber threats. The adaptability of deep learning to evolving attacks is crucial, enabling the processing of unstructured data, uncovering dependencies, and enhancing attack detection.

For syslog anomaly detection, key findings include segmenting log sequences into windows for insightful evaluation. Preference leans towards supervised over semi-supervised approaches, with LSTM models excelling in the latter. While Log2Vec embeddings and sophisticated parsing techniques show uncertain benefits, CNN models prove superior in supervised scenarios, demonstrating suitability for real-world applications. Optimizing hyperparameter values is crucial for performance.

In net flow anomaly detection, dataset selection is pivotal, favoring NIDS over CTU-13 for its richer features and diverse data sources. Upsampling anomalies provides no benefit, and dataflow preprocessing should focus on categorization and normalization, excluding IP2Vec. Deep learning methods, particularly CNN, enhance prediction results effectively.

Our general analysis identifies CNN models as highly effective for both syslog and dataflow datasets, demonstrating versatility in network data anomaly detection. The reusability of these models in real-life systems bodes well for practical implementation, though further research with realistic datasets is essential to enhance accuracy and efficiency. Leveraging deep learning strengthens cybersecurity models, safeguarding sensitive information and business processes against evolving threats.

REFERENCES

- Chen, Z., Liu, J., Gu, W., Su, Y., and Lyu, M. R. (2021). Experience report: Deep learning-based system log analysis for anomaly detection. *arXiv preprint arXiv:2107.05908*.
- Cotton, M., Vegoda, L., Bonica, R., and Haberman, B. (2013). Special-purpose ip address registries. Technical report, IETF.
- Du, M. and Li, F. (2016). Spell: Streaming parsing of system event logs. In *2016 IEEE 16th International Conference on Data Mining*, pages 859–864. IEEE.
- Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298.
- Garcia, S., Grill, M., Stiborek, J., and Zunino, A. (2014). An empirical comparison of botnet detection methods. *computers & security*, 45:100–123.
- He, P., Zhu, J., Zheng, Z., and Lyu, M. R. (2017). Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE international conference on web services (ICWS)*, pages 33–40. IEEE.
- He, S., Zhu, J., He, P., and Lyu, M. R. (2016). Experience report: System log analysis for anomaly detection. In *2016 IEEE 27th international symposium on software reliability engineering (ISSRE)*, pages 207–218. IEEE.
- He, S., Zhu, J., He, P., and Lyu, M. R. (2020). Loghub: a large collection of system log datasets towards automated log analytics. *arXiv preprint arXiv:2008.06448*.
- Hettich, S. (1999). Kdd cup 1999 data. *The UCI KDD Archive*.
- Kim, J., Kim, J., Thu, H. L. T., and Kim, H. (2016). Long short term memory recurrent neural network classifier for intrusion detection. In *2016 international conference on platform technology and service (PlatCon)*, pages 1–5. IEEE.
- Kim, T.-Y. and Cho, S.-B. (2018). Web traffic anomaly detection using c-lstm neural networks. *Expert Systems with Applications*, 106:66–76.
- Li, G. and Jung, J. J. (2022). Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges. *Information Fusion*.
- Liu, F., Wen, Y., Zhang, D., Jiang, X., Xing, X., and Meng, D. (2019). Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1777–1794.
- Lu, S., Wei, X., Li, Y., and Wang, L. (2018). Detecting anomaly in big data system logs using convolutional neural network. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 151–158. IEEE.
- Maimó, L. F., Gómez, Á. L. P., Clemente, F. J. G., Pérez, M. G., and Pérez, G. M. (2018). A self-adaptive deep learning-based system for anomaly detection in 5g networks. *Ieee Access*, 6:7700–7712.
- Makanju, A. A., Zincir-Heywood, A. N., and Milios, E. E. (2009). Clustering event logs using iterative partitioning. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1255–1264.
- Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., Chen, Y., Zhang, R., Tao, S., Sun, P., et al. (2019). Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI*, volume 19, pages 4739–4745.
- Nedelkoski, S., Bogatinovski, J., Acker, A., Cardoso, J., and Kao, O. (2020). Self-attentive classification-based anomaly detection in unstructured logs. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1196–1201. IEEE.
- Nguyen, V. Q., Nguyen, V. H., Hoang, T. H., and Shone, N. (2022). A novel deep clustering variational auto-encoder for anomaly-based network intrusion detection. In *2022 14th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–7. IEEE.
- Podder, P., Bharati, S., Mondal, M., Paul, P. K., and Kose, U. (2021). Artificial neural network for cybersecurity: A comprehensive review. *arXiv preprint arXiv:2107.01185*.
- Ring, M., Dallmann, A., Landes, D., and Hotho, A. (2017). Ip2vec: Learning similarities between ip addresses. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 657–666.
- Sarhan, M., Layeghy, S., Moustafa, N., and Portmann, M. (2021). Netflow datasets for machine learning-based network intrusion detection systems. In *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings 10*, pages 117–135. Springer.
- Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., and Ghogho, M. (2016). Deep learning approach for network intrusion detection in software defined networking. In *2016 international conference on wireless networks and mobile communications (WINCOM)*, pages 258–263. IEEE.
- Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., et al. (2019). Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 807–817.
- Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., and Lyu, M. R. (2019). Tools and benchmarks for automated log parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice*, pages 121–130. IEEE.