A Multi-Class Intrusion Detection System Based on Continual Learning

Chrysoula Oikonomou
Information Technologies Institute
Thessaloniki, Greece
chrisaoikon@iti.gr

Ilias Iliopoulos
Information Technologies Institute
Thessaloniki, Greece
ilias_nbp@iti.gr

Dimosthenis Ioannidis
Information Technologies Institute
Thessaloniki, Greece
djoannid@iti.gr

Dimitrios Tzovaras
Information Technologies Institute
Thessaloniki, Greece
Dimitrios.Tzovaras@iti.gr

2023 IEEE International Conference on Cyber Security and Resilience (CSR) | 979-8-3503-1170-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CSR57506.2023.10224974

***Abstract*—With the proliferation of smart devices, network security has become crucial to protect systems and data. In order to identify and categorise different network threats, this study introduces a flow-based Network Intrusion Detection System (NIDS) based on continual learning with a CNN backbone. Using the LYCOS-IDS2017 dataset, the study explores several continuous learning techniques for identifying threats including denial-of-service and SQL injection. Unlike previous approaches, this work treats intrusion detection as a multi-class classification problem, rather than anomaly detection. The findings show how continuously learning models may identify network intrusions with high recall rates and accuracy while generating few false alarms. This study contributes to the development of an adaptive NIDS that can handle attack classification simultaneously with detection, and that can be trained online without periodic offline training. Additionally, utilising the improved version of the dataset adds value to the research on LYCOS-IDS2017 by presenting results for untested models.**

***Index Terms*—Continual Learning, Intrusion Detection, Multi-class Classification, Network Security, Supervised Learning, Online learning**

I. Introduction

During the last few years there is a rapid increase in the number of smart and embedded devices, and this growth is projected to continue due to the ongoing development of the Internet of Things (IoT) ecosystem [1]. Such devices are a basic component in multiple environments, such as smart homes, industries, and even e-Health systems. The information exchanged is often private and its breach may result in violation of individual rights, as well as in increased economic risks in industrial environments. The breach of such information becomes a great risk for organisations, as malicious users develop sophisticated cyber attacks to take advantage of possible network security vulnerabilities. In order to ensure the CIA triad (Confidentiality, Integrity, and Availability) of the data, security policies and mechanisms should be deployed. NIDSs are regarded as one of the most crucial elements of a company's network security, since they act as the first line of defence against cyber threats and are in charge of discovering potential network intrusions. They also have an important role in industrial environments, as they prevent the potential consequences of a security threat that have an impact on both economic and legal level.

NIDSs based on continual learning algorithms receive a lot of research attention, as they give the opportunity to be kept updated with the continuously evolving cyber attacks. Continual Learning (CL) provides the ability to learn, maintain, and fine-tune progressively obtained information. Furthermore, CL based NIDSs significantly reduce the costs of required periodic retraining as they can recognise new patterns given only a small batch of network data. During the retraining process, is important that the data serving as an input are different from the previous inputs, so that there is no chance of overfitting. A CL based NIDS also offers the ability to adapt to new patterns of benign traffic. As stated in the Longitudinal Study presented in [2], apart from the continuously evolving attack vectors, normal traffic patterns also present multiple variations through time. Hence, a CL based NIDS can adapt to these changes as well, and can reduce false alarms significantly, progressively. In this paper a NIDS trained with CL techniques is presented. The authors tested twelve different CL algorithms to compare their performance, in terms of their efficiency in correctly detecting and classifying malicious traffic. The algorithms are trained and tested with the use of the LYCOS-IDS2017 dataset[3]. The contribution of this paper is two-fold:

1) The authors fairly compare state-of-the-art methods in CL and determine which works best at a specific memory and architecture setting.
2) It is argued that Multi-Class NIDS on the LYCOS-IDS2017 dataset has promising results and is worth of further experimenting.

The rest of the paper is organised as followed. Section II includes a brief analysis of the Continual Learning concept and methods. Section III presents the related work on network-based IDSs based on CL techniques. In Section IV, there is a description of the dataset used, as well as all the prepossessing applied on it. Section VI includes a detailed analysis of the methodology followed for the training and testing procedures, and discusses the results of the implementation, in terms of

efficiency. Finally, the conclusions of this work as well as discussion on future work are presented in Section VII.

II. Continual Learning

Since the nature of normal traffic in an underlying network change over time, the need for a continually evolving NIDS becomes apparent. Continual Learning (CL), also known as Incremental Learning or Life-long Learning, describes the ability of a model to learn continually from a stream of data without forgetting previously obtained knowledge. The learning process of a CL based framework is the following : the model is trained incrementally on task (x_{T_i}, y_{T_i}) where x is the feature vector and y the target vector of task T_i,for $i = 1, 2, ..N$ and N the number of tasks in the scenario. The grand goal of CL is to strike a balance between learning stability and plasticity. Some of the most important challenges involved in CL are:

- **Catastrophic Forgetting (CF):** NNs have the tendency to drastically forget previously acquired knowledge upon learning new information.
- **Class Imbalance (CI) :** Real life data, especially network data sets, have a lot of under-represented classes which makes unbiased training a challenge. For example, the amount of normal traffic that passes through a network is usually much bigger than the malicious attacks.

CL methods have typically been taxonomized into three major categories based on the techniques they use to tackle the aforementioned problems: regularisation-based, memory-based and parameter-isolation based.[4] **Regularisation based** methods impose constraints on the parameters of the model to avoid a considerable shift in their values during the novel input. **Memory-based** techniques store a subset of samples from previous tasks for either replay while training on a new task or for regularisation purposes. When storing samples is not possible due to privacy or storage concerns, an alternative is Generative Replay which trains a deep generative model such as GAN to generate pseudo-data that mimic past data for replay. The main disadvantages here are that it takes a long time to train generative models. Knowledge Distillation methods which present an effective way for knowledge transfer between networks. For the online incremental learning setting, most existing methods rely on the rehearsal strategy and focus on better utilisation of the memory. In addition to the aforementioned CL challenges, the existing literature has focused on anomaly detection, treating the problem of intrusion detection as binary, and not as a multi-class classification of attacks. It's important to note here that, to the best of the authors' knowledge, this is the first attempt to create an online CL framework capable of performing multi-class network intrusion detection.

III. Related Work

While the use of Machine Learning techniques is very common in the development of NIDSs, the majority of the works published focus on offline models. This section includes a literature review of the papers that introduced NIDSs using CL algorithms. All of the papers examined concern anomaly-based detection. Intrusion detection using anomaly-based NIDSs does not require signatures, namely pre-defined patterns to detect the attacks. The theory behind anomaly-based detection is to discover malicious instances by modelling normal traffic. Hence, any divergence from this pattern is seen as suspicious[5].

In[6], the authors investigated the possibility for continuous learning algorithms to maintain prior knowledge while gradually learning new data patterns. Using the NSL-KDD and the CICIDS 2017 datasets, they conduct an experimental and analytical evaluation of advanced intrusion detection systems employing three important continual learning approaches: learning without forgetting, experience replay, and dark experience replay. Their models outperformed state-of-the-art works in terms of accuracy and FPR while they were able to progressively learn newer data patterns. They concluded by highlighting the shortcomings of conventional statistical and machine learning methods that do not use gradients to solve the covariate shift problem.

Authors of [7] explored the application of generative replay (GR), initially introduced in [8], for continuous learning to the issue of variational autoencoder (VAE) based anomaly detection. They suggested a straightforward yet effective technique to reduce catastrophic forgetting. Their method takes advantage of the VAE's generating capabilities, which is not often employed for anomaly detection.

Class Imbalance (CI) is a challenging issue in CL. In order to reduce CI in a Class Incremental Setting, the authors of [9] investigated algorithmic level solutions to the CI issue that arises when using network data and the CL framework, and proposed a CL-based A-NIDS (CIS). Their approach provided positive outcomes, and they also investigated how the buffer size affected the performance indicators, which calls for careful consideration. The same research team introduced also a CL-based anomaly-based NIDS (A-NIDS) framework in [10]to recognise malicious traffic with new attack patterns. The suggested architecture is among the most scalable options since it never uses the outdated training data whenever a new threat emerges. They discovered that Domain Incremental Learning (DIL) configuration is more similar to actual traffic patterns than Class Incremental Learning (CIL), which is more vulnerable to Task Execution Order Sensitivity (TEOS). Hence, DIL paired with sophisticated memory population techniques is a suitable option for creating practical CL-based A-NIDS.

IV. LYCOS-IDS2017 Dataset and Data Preprocessing

A. LYCOS-IDS2017 Description

A lot of researchers have focused on designing IDSs for network attacks based on machine learning techniques. One of the main challenges faced when developing such systems is the use of a suitable dataset. A dataset should meet a number of criteria to contribute to the effectiveness of the IDS. One of the most important characteristics to be considered is the

size of the dataset. There should be enough data in order to efficiently form patterns and hence increase the classification performance. In addition, the quality of the data should also be assessed. In other words, the feature describing each entry of the dataset should be adequate to successfully characterise the entry. This is achieved both by having a sufficient amount of features and at the same time by minimizing the noise on it like missing values. Last but not least, a dataset should be also evaluated, in terms of up-to-dateness. Taking into consideration that cyber attacks are continuously adapting according to the technological advancements of systems and networks, as well as new, more sophisticated attacks are introduced, it is crucial that the dataset that an IDS is trained on, is capable of describing as much of the possible current threats as possible.

The authors of [3] were based on CIC-IDS2017, a dataset previously introduced by [11]. The initial dataset, which was captured on a real network, comes with 50 GB of raw data in PCAP format and 84 flow-based characteristics stored in CSV files. The program that divides packets into flows and retrieves their properties is called CICFlowMeter [12]. According to [13] a network flow is defined as *A network flow is defined as an unidirectional sequence of packets between given source and destination endpoints. Flow records include details such as IP ad- dresses, packet and byte counts, timestamps, Type of Service (ToS), application ports, input and output interfaces, etc.*. They proposed an updated version of that dataset, by applying several improvements. In detail, their contribution was divided into four sections. First, they highlighted certain fundamental issues with the intial CSV files, and -as there was no labeling tool available for this dataset- they suggested a flow extractor called LycoST, to resolve the issues uncovered. Their tools were used to create a new dataset named LYCOS-IDS2017. Both the tools and the dataset they produced are openly accessible.

LYCOS-IDS2017 includes network traffic that is normal, meaning it may not cause damage to the network and/or the system, while also it contains 14 different types of known network attack types. In total 1837498 entries are included and every entry in the dataset is described by 83 features (including the label). The labels were created by a labelling process included in the LycoSTand tool. In Table I there is a class distribution of the dataset.

TABLE I
LYCOS-IDS2017 CLASS DISTRIBUTION

Label	**Number of Instances**	
	Initial	After Remediation
Benign	1395675	1140804
Bot	735	735
DDoS	95683	95180
DoS GoldenEye	6765	6765
Dos Hulk	158988	157996
DoS Slowhttptest	4866	4674
Dos Slowloris	5674	4725
FTP-Patator	4003	3998
Heartbleed	11	11
PortScan	160106	67217
SSH-Patator	2959	2957
Web Attacks-Brute Force	1360	*(merged class)*
Web Attacks-SQL Injection	12	*(merged class)*
Web Attacks-Sql Injection	661	2031
Total	1837498	1487093

B. Data preprocessing

Data preprocessing is a crucial step when designing an IDS based on machine learning algorithms. The four main refinements performed in the dataset are listed below.

1) **Handle missing values:** Eliminated rows including missing values, as they were coming from classes represented by numerous of instances, so the deletion did not cause any problem to the training process.
2) **Handle features with irrational values:** There were three features, that described packet length and time duration, occasionally represented by negative values. Those indexes were eliminated from the dataset.
3) **Handle features with zero variance:** Features that were represented by the same value across all samples of the dataset were completely removed, as they could not offer any statistical information, in the model training phase.

After eliminating the rows and columns mentioned in the previous steps, the process continued by identifying features which present high correlation. High correlation features are more linearly dependant and therefore nearly equally affect the dependent variable. As a result, when two features have a significant correlation, one of the is excluded. To detect such features Pearson Correlation Coefficient was calculated [14]. The correlation coefficient's value ranges from +1 to -1 depending on the strength of the association. The degree of absolute correlation between the two variables is indicated by a value of 1. As the correlation coefficient value gets closer to zero, the relationship between the two variables will get weaker. For the purposes of this implementation, features that presented a coefficient value than $|0.97|$ were removed. Following the methodology suggested by [3], the input set was sized down to 10 features, using recursive feature elimination (RFE). The final step for the data remediation was to check for duplicate indexes and remove them too. After completing all data remediation procedures, the final dataset contained 1633101 entries, described by 10 features each (including the label).

Another challenging characteristic of the dataset, as it can also be observed in I, is the large unevenness in the distribution of instances between the classes. To address this, two more modifications were implemented. At first,three different web-attack classes were merged into one as they have a high value of similarity concerning, their behaviour network-wise [15] and finally up-sampling and down-sampling was performed.

V. COMPARED METHODS

A. Regularization-based methods

Elastic Weight Consolidation (EWC): imposes a quadratic penalty to regularise the update of model parameters that were important to previous tasks. The importance of parameters

TABLE II
THE PERFORMANCE RESULTS OF THE DIFFERENT AGENTS USED FOR THE NIDS

Model	Accuracy	Recall Rate	False Alarms	Precision	F1-Score
GDUMB	97,11%	93,32%	8.3%	93,77%	91,29%
ICARL	90,21%	80,10%	24.72%	83,40%	73,11%
GSS	87,12%	89,79%	9.34%	91,05%	87,27%
ER	84,64%	95,57%	5.12%	95,52%	94,48%
ASER	84,18%	59,39%	51.79%	74,40%	50,11%
MIR	82,85%	94,45%	6.4%	94,66%	92,88%
AGEM	56,69%	93,35%	6.4%	95,04%	91,94%
SSR	51,16%	37,82%	65,74%	69,25%	22,47%
NCM	49,09%	66,36%	28.01%	67,52%	60,43%
STREAMINGLDA	44,37%	62,04%	30.79%	61,07%	54,86%
EWC	49,07%	59,21%	23.08%	81,61%	60,19%
LWF	8,32%	76,68%	0.05%	99,91%	86,76%
OFFLINE	99,23%	96,01%	5.11%	96,20%	94,66%

is approximated by the diagonal of the Fisher Information Matrix.[16]

B. Knowledge Distillation methods

Learn without Forgetting (LwF: . incorporates a teacher model which is the model after learning the last task, and a student model which is the model trained with the current task. It's worth noting that LwF is heavily reliant on the relatedness between the new and old tasks. Thus, it may not perform well on tasks with different distributions.[17]

C. Memory-based methods

The basic concept behind memory-based methods is the following: For every incoming batch of samples, the method retrieves another batch from a memory buffer, updates the model using both the incoming and memory batches and then updates the buffer with the incoming batch. What differentiate various memory-based methods are the memory retrieval strategy, model update and memory update strategy.[4]

Averaged GEM (AGEM): prevents CF by penalising the parameter update with the samples on the memory buffer. At every training step, GEM ensures that the loss of the memory samples for each individual preceding task does not increase, while A-GEM ensures that the average loss for all past tasks does not increase.[18][4]

Incremental Classifier and Representation Learning (iCaRL): incorporates a loss function which includes a classification loss to encourage the model to predict the correct labels for new classes and a Knowledge Distillation loss to prompt the model to reproduce the outputs from the previous model for old classes.[19][4]

Experience Replay (ER): is a very effective replay-based method which applies reservoir sampling for memory update and random sampling for memory retrieval. Reservoir sampling ensures that all data points have the same probability to be selected for storage in the memory buffer. ER trains the model with the incoming and memory batches together using the cross-entropy loss. Despite its simplicity, recent research has shown that ER outperforms many specifically designed CL approaches with and without a memory buffer[20][4].

Adversarial Shapley Value Experience Replay (ASER): is a very interesting variation of ER which uses the efficient KNN Shapley Value (KNN-SV) computation for it's memory retrieval and update stratiegies. ASER scores memory data samples according to their ability to preserve latent decision boundaries for previously observed classes while interfering with latent decision boundaries of current classes being learned. [21][4].

Maximally Interfered Retrieval (MIR): focuses on improving the aforementioned ER by applying a different memory retrieval strategy. It selects memory samples that are maximally interfered (the largest loss increases) by the parameter update with the incoming batch and applies reservoir sampling for memory update. [22][4]

Gradient based Sample Selection (GSS): focuses on a new memory update strategy. Specifically, it tries to diversify the gradient directions of the samples in the memory buffer. To this end, GSS maintains a score for each sample in the buffer, and the score is calculated by the maximal cosine similarity in the gradient space between the sample and a random subset from the buffer. When a new sample arrives and the memory buffer is full, a randomly selected subset is used as the candidate set for replacement. The score of a sample in the candidate set is compared to the score of the new sample, and the sample with a lower score is more likely to be stored in the memory buffer.[23][4]

Greedy Sampler and Dumb Learner (GDUMB): greedily updates the memory buffer from the data stream with the constraint to keep a balanced class distribution. At inference, it trains a model from scratch using the balanced memory buffer only.[24][4]

D. Classifier Replacements

The authors also compared four online continual learning methods for updating the classifier F using pre-trained universal image features from backbone B. These methods were chosen due to their ability to learn one sample at a time in a single pass over a dataset without task labels (i.e., online continual learning) with low memory and compute requirements.

TABLE III
DETAILS OF THE CNN ARHITECTURE.

Layer Type	Output Shape	Param
Convd1 - 1	[BS, 64, 10]	256
MaxPool1 - 2	[BS, 64, 5]	-
ReLu - 3	[BS 64, 5]	-
Flatten - 4	[BS, 320]	-
Linear1 - 5	[BS, 128]	41088
Dropout1 - 6	[BS, 128]	-
Linear2 - 7	[BS, 12]	1548

Nearest Class Mean (NCM): maintains one running mean vector per class each with an associated counter denoting the number of samples represented in each mean. During inference, it assigns the label of the nearest class mean to a new example (Euclidean distance is used).[25]

Streaming Linear Discriminant Analysis (SLDA): maintains one running mean vector per class with an associated counter, which are updated in the same way as NCM, and one shared running covariance matrix among classes. Intuitively, SLDA makes predictions by assigning a new example the label of the closest Gaussian in feature space defined using the running class means and shared covariance matrix. NCM is a special case of SLDA where the covariance matrix is equal to the identity matrix.[25][26]

Streaming Softmax Replay: maintains a memory buffer which equally distributes examples among classes seen so far. When the buffer is full, it randomly replaces an example from the most represented class with a new example. During training, it randomly defines a number of examples from the replay buffer, combines them with the new example, and makes a single update using stochastic gradient descent with a cross-entropy loss. While effective, streaming softmax replay can be memory intensive due to the storage of its memory buffer. [25]

VI. System Architecture and Experimental Results

A. The proposed NIDS Architecture

The same backbone CNN architecture was used for the comparison of the different online CL methods. Details on the layers and number of parameters used can be found at III. For the sake of fair comparison across agents, the same basic training settings we re used: learning rate $lr = 10e-5$, memory size $M = 10000$ samples and optimiser *Adam*. The online incremental training scenario has twelve tasks $T_i, i = 1, ...12$ and one class per task T_i.

B. Evaluation metrics

In order to compare the performance of the models tested, the evaluation of their performance was based on the following metrics:

- **Accuracy:** Accuracy is defined as the proportion of correct predictions produced by an algorithm out of all predictions made by the algorithm.
- **Recall Rate:** The percentage of successfully predicted outcomes to all forecasts is known as recall. It is sometimes referred to as sensitivity or specificity.
- **Precision:** The percentage of positive identifications that was actually correct.
- **F1-Score:** A metric that combines recall and accuracy. It is referred to as the harmonic mean of the two.
- **False Alarms:** Number of alerts that falsely indicates that malicious activity is detected.

Table II includes the evaluation metrics for all models tested. Besides the aforementioned evaluation metrics, an offline training scenario, with a higher learning rate $lr = 10e-3$, was added for the evaluation of the online CL methods' performance, which is included in TableII.

C. Performance Results

Upon observing the performance results of the different agents used for comparison shown at Table II, the authors extract the following conclusions:

- *GDUMB has the best multi-classification performance* (which is excellent compared to the offline training scenario) but it lags behind in regard to false alarms which is a very important metric for a NIDS.
- *LwF has the lowest false alarm rate* but the worst classification performance. The main reason behind this is the aforementioned CI issue between normal traffic and attacks. Upon training on task 9 which contains the vastly over-represented benign class, the model fails to preserve knowledge from previous classes leading to extremely low false alarms and classification performance.
- *ER seems to be the best performing method (generally)* since it has the finest trade-off between multi class accuracy and false alarm predictions. It also has the best Precision, Recall and F1-Score than all other methods.
- All *classifier replacements had a worse performance than memory-based methods*, which can be attributed to the fact that the CNN architecture that was used, although not deep, it was not pre-trained and weight initialisation plays a very important role to the feature extraction. It is important to note that the classifier replacements could have a much better performance with a different backbone.

VII. Conclusions and future Work

In this paper, the authors tested twelve different CL based algorithms to develop a network IDS. This is the first published work that deals with intrusion detection as a multi-class classification problem using CL. In detail, the authors built a multi-class classifier that is able to classify a network flow in one of 12 different categories, one representing normal traffic and 11 representing different network attack types. These algorithms were trained and tested using the LYCOS-IDS2017 dataset. All algorithms were compared using the following metrics; accuracy, recall, precision, f1-score and false alarm rates. The results induced show that memory based methods (GDUMB,ER) perform better in terms of accuracy and false

alarm predictions and that multi - class prediction of network attacks in an online incremental learning scenario is possible and future scientific work should be pointed towards that direction.

As a next step the authors plan to test different backbone models, to investigate how they affect the performance of the models. In addition, it is intended to further explore continual learning algorithms, to implement a NIDS that handles the classification as an unsupervised problem.

ACKNOWLEDGMENT

This work is co-funded by the European Union (EU) within the RECLAIM project under grant agreement number 869884. The RECLAIM project is part of the EU Framework Programme for Research and Innovation Horizon 2020.

REFERENCES

[1] E. Perspectives and C. Report, "Cisco annual internet report - cisco annual internet report (2018–2023) white paper," 2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity," ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1723–1732. [Online]. Available: https://doi.org/10.1145/3097983.3098163

[3] A. ROSAY, F. CARLIER, E. CHEVAL, and P. LEROUX, "From cic-ids2017 to lycos-ids2017: A corrected dataset for better performance," in *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, ser. WI-IAT '21. New York, NY, USA: Association for Computing Machinery, 2022, p. 570–575. [Online]. Available: https://doi.org/10.1145/3486622.3493973

[4] "Online continual learning in image classification: An empirical survey," *Neurocomputing*, vol. 469, pp. 28–51, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221014995

[5] T. Hamed, R. Dara, and S. C. Kremer, "Chapter 6 - intrusion detection in contemporary environments," in *Computer and Information Security Handbook (Third Edition)*, third edition ed., J. R. Vacca, Ed. Boston: Morgan Kaufmann, 2017, pp. 109–130. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128038437000065

[6] S. Prasath, K. Sethi, D. Mohanty, P. Bera, and S. R. Samantaray, "Analysis of continual learning models for intrusion detection system," *IEEE Access*, vol. 10, pp. 121 444–121 464, 2022.

[7] F. Wiewel and B. Yang, "Continual learning for anomaly detection with variational autoencoder," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3837–3841.

[8] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *Advances in neural information processing systems*, vol. 30, 2017.

[9] S. K. Amalapuram, T. T. Reddy, S. S. Channappayya, and B. R. Tamma, "On handling class imbalance in continual learning based network intrusion detection systems," in *The First International Conference on AI-ML-Systems*, 2021, pp. 1–7.

[10] S. K. Amalapuram, A. Tadwai, R. Vinta, S. S. Channappayya, and B. R. Tamma, "Continual learning for anomaly based network intrusion detection," in *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 2022, pp. 497–505.

[11] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018.

[12] "Cicflowmeter, a network traffic biflow generator and analyzer (formerly iscxflowmeter)." [Online]. Available: https://www.unb.ca/cic/research/applications.html

[13] V. Marinov and J. Schönwälder, "Design of an ip flow record query language," in *Resilient Networks and Services: Second International Conference on Autonomous Infrastructure, Management and Security, AIMS 2008 Bremen, Germany, July 1-3, 2008 Proceedings 2*. Springer, 2008, pp. 205–210.

[14] G. M. Davis, *Pearson Correlation Coefficient*. CRC Press, 2018, p. 1–4.

[15] P. Toupas, D. Chamou, K. M. Giannoutakis, A. Drosou, and D. Tzovaras, "An intrusion detection system for multi-class classification based on deep neural networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 1253–1258.

[16] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 532–547.

[17] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[18] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," 12 2018.

[19] S. Rebuffi, A. Kolesnikov, and C. H. Lampert, "icarl: Incremental classifier and representation learning," *CoRR*, vol. abs/1611.07725, 2016. [Online]. Available: http://arxiv.org/abs/1611.07725

[20] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. S. Torr, and M. Ranzato, "Continual learning with tiny episodic memories," *CoRR*, vol. abs/1902.10486, 2019. [Online]. Available: http://arxiv.org/abs/1902.10486

[21] Z. Mai, D. Shim, J. Jeong, S. Sanner, H. Kim, and J. Jang, "Adversarial shapley value experience replay for task-free continual learning," *CoRR*, vol. abs/2009.00093, 2020. [Online]. Available: https://arxiv.org/abs/2009.00093

[22] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars, *Online Continual Learning with Maximally Interfered Retrieval*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[23] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," *Advances in neural information processing systems*, vol. 32, 2019.

[24] A. Prabhu, P. H. Torr, and P. K. Dokania, "Gdumb: A simple approach that questions our progress in continual learning," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 524–540.

[25] T. L. Hayes and C. Kanan, "Online continual learning for embedded devices," 2022. [Online]. Available: https://arxiv.org/abs/2203.10681

[26] ——, "Lifelong machine learning with deep streaming linear discriminant analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.