ISSN 2303-4521
Periodicals of Engineering and Natural Sciences *Original Research*
Vol. 12, No. 1, February 2024, pp.75-100

A survey about deep learning and federated Learning in cyberse-curity

Imad Tareq[1], Bassant M. Elbagoury[1,2], Salsabil El-Regaily[1], El-Sayed M. El-Horbaty[1]
[1]Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt
[2]Faculty of Computer Science and Computer Engineering King Salman International University, EI-Tor, Egypt

ABSTRACT

Advances in Artificial Intelligence (AI) technology have led to the strengthening of traditional systems' cybersecurity capabilities in a variety of applications. However, these embedded machine learning models have exposed these systems to a new set of vulnerabilities known as AI assaults. These systems are now attractive targets for cyberattacks, jeopardizing the security and safety of bigger systems that include them. As a result, DL approaches are critical to transitioning network and system protection from providing safe communication between systems to intelligence systems in security. Federated learning (FL) is a new kind of AI based on heterogeneous datasets and decentralized training. FL is a unique research topic that is currently in its early phases. It has not yet gained wide acceptance in the community, owing mostly to privacy and security considerations. In this research, we first shed light on its privacy and security risks that must be discovered, analyzed, and recorded. FL is favored in scenarios where privacy and security are paramount issues. An extensive understanding of risk factors allows an FL adopter and implementer to construct a safe environment successfully while giving researchers a clear perspective of possible study domains. The survey in this paper intends to include an analysis of cybersecurity and DL approaches and modern advances to improve enhanced protection methods. It proposes a complete examination of FL's security and privacy issues to assist in bridging the gap between the current level of federated AI and a future in which broad adoption is achievable. We also propose a range of cybersecurity datasets and the most recently used rating standards.

Keywords: AI, Cybersecurity, Deep Learning, Federated Learning

Corresponding Author:

Imad Tareq

Faculty of Computer and Information Sciences

Ain Shams University

Cairo, Egypt.

E-mail: emadtariq1982@gmail.com

1. Introduction

Since the introduction of artificial intelligence (AI) in 1956, AI technology has increasingly influenced human existence. As AI technology has advanced recently, different application industries have entered intelligence. With this advance, the frequency and range of cyberattacks have recently increased, quickly rendering conventional security techniques obsolete. As a result, the most cited dimension in the literature is controlling the security of networks and systems to identify cyberattacks after selecting and monitoring behavior sources. Artificial Intelligence (AI) techniques, notably Deep Learning (DL), have grown in popularity in the past few years [1]. Most malware detection systems that use AI rely on a central organization that collects data from many devices and trains global models. Nonetheless, this method is inappropriate for circumstances where device actions contain confidential or sensitive data that, if compromised, would dramatically impact security and privacy where integrity is critical [2]. As a result, there is an urgent need for a practical and effective

approach to alleviate the issues above and re-store the AI's life. The notion of "federated learning (FL)" arose against this backdrop. Google originally introduced the concept of FL in 2016, primarily to allow Android mobile phone users to upgrade models locally without disclosing sensitive personal data. Google then created an application-oriented FL system. FL is one of the most closely scrutinized technologies in privacy computing. FL has been the standard solution and product choice in many privacy computing applications due to its lightweight technological pathways and deployment strategy benefits. Moreover, as FL applications have evolved and im-proved, many research successes in the FL sector have developed [3]. Today's artificial intelligence is becoming more decentralized. New AI models are being trained collaboratively. Federated learning has evolved as a distributed confidentiality meth-od in recent years. As illustrated in Figure 1, In FL, algorithms are trained across servers or different devices utilizing decentralized data samples without sending actual data. This concept differs significantly from previously known strategies, like keeping data in a distributed architecture or uploading data to servers.

On the contrary, FL creates more secure models without sharing data, which leads to privacy protection while increasing security privileges and data access, which is the desired outcome [4]. Federated learning has been applied in multiple applications, like IoT, transportation, medical, healthcare, defense, and mobile apps. Despite FL's tremendous technique, certain technical components, including platforms, software, hardware, and others concerning data privacy and access, are still poorly understood [5]. This paper will discuss the benefits of the FL technique and explore many of the most significant cybersecurity applications these features enable. We then examine some ongoing obstacles in systems, networking, cybersecurity, practical concerns in real-world implementations, tool development that function as significant obstacles to federated learning, and the potential for overcoming these obstacles.

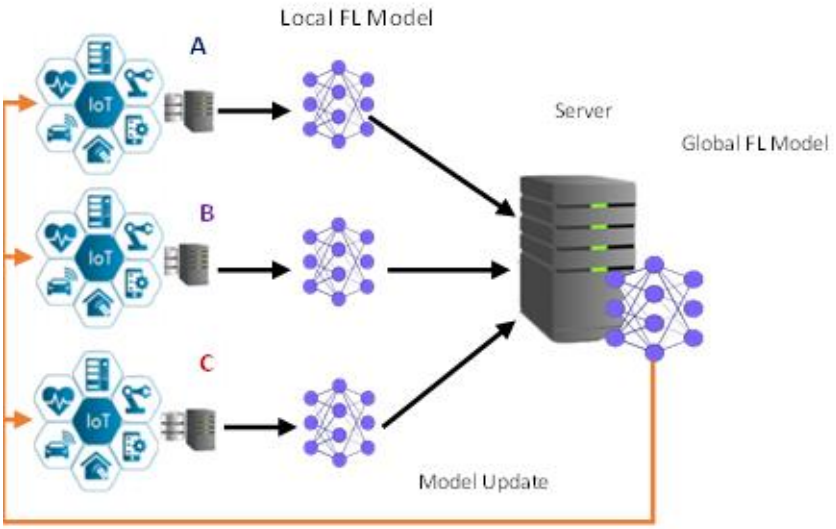

Figure 1. The federated learning

The study offers an overview of cybersecurity and Federated Learning, applications, FL platforms, challenges, algorithms, a wide range of datasets, and the most recently used evaluation criteria. Compared to previous survey papers, this work aims to present a thorough understanding of the most significant deep learning (DL) applications in cybersecurity and real-world FL use cases to assist data scientists in designing better privacy-preserving strategies for data that rely on FL. In addition, we present a summary of major challenges mentioned in recent literature. While considerable studies have been conducted on this subject, there has not been enough progress regarding knowledge of federated learning on a deeper level. Federated learning is new and poorly understood, with little application in most industries. As a result, we lack a comprehensive understanding of federated learning and cannot see the big picture of FL. We believe the proposed survey will provide a comprehensive overview of the issues and contributions raised. Compared to existing surveys, the following are the important contributions of this work.

1- We explain DL algorithms and review cybersecurity. In addition, it highlights DL use cases and cybersecurity attack detection using FL.

2- Compared to other survey studies, this report describes the more relevant FL platforms, software, and hardware, allowing researchers to learn about FL techniques.

3- We review to identify and analyze the critical contributions that address security and privacy issues. These are important considerations in FL because malevolent parties can still leak and taint data.

4- We thoroughly analyze the most used dataset in cybersecurity applied and evaluation criteria. The rest of the survey is organized as follows. The basic knowledge of cybersecurity is introduced, and datasets most used in Sect. 2. Sect. 3 introduces AI systems support to Cybersecurity and Applications for DL in Cybersecurity. Sect. 4 Overview of Federated Learning and FL's Motivations for Cybersecurity, Types, and FL Applications. Sect. 5 discusses Commonly used evaluation criteria. And conclusion in Sect. 6.

2. Cybersecurity

Cybersecurity is a skill set, procedures, and strategies to protect computers, data, and networks from malicious malware, disruption, and other dangers. Our lives have be-come more luxurious because of AI's fast growth, yet numerous hazards exist. AI attacks, Malware, IoT attacks, third-party attacks, MITM attacks, phishing, ransomware, denial-of-service attacks, and supply chain attacks are all examples of cyberat-tacks. Developers may find it challenging to incorporate essential cyber security in IoT networks with numerous levels and topologies [6]. Intelligent devices and connected technology are at the heart of the bulk of IoT applications and systems. Cyber-security is critical in AI applications because it ensures efficient management of interactions with people and goals. In cyberspace, intrusion detection systems are com-monly used to identify and manage hazards. It allows programmers to ensure safe functioning and the capacity to address any online threats that may jeopardize the system [7], as shown in Figure 2.

Figure 2. A form of cyberattack

2.1. Cybersecurity security dataset

Conducting relevant security research necessitates the proper selection and application of data. The size of the data set also influences DL model training. In most cases, there are two ways to collect security information: directly, which is extremely specific and ideal for gathering short-term or tiny quantities of data, and leveraging a current public dataset, which enables quick access to the various datasets required for research, this can minimize the time spent on data gathering and enhance the effectiveness of research.

2.2. Most used datasets

The process of protecting networks and connected Internet devices from cyber-attacks by identifying and monitoring risks and assisting in patching security flaws is known as security. Because of the dramatic increase in cybercrime, DL techniques are now being used to provide early detection solutions for cyber risks and to prevent them. DL algorithms perform best when trained on great, diverse data sets. The more commonly used data sets in security applications are briefly overviewed in this section and represented in Table 1.

Table 1. The most often used datasets in cybersecurity applications.

Dataset	year	Attack	Feature	Training set	Based dataset
NSL-KDD	2009	4	43	Feature	Network traffic-based
Botnet-IoT	2014	5	115	Feature	Internet-connected device
UNSW-NB15	2015	9	49	Feature	Network traffic-based
CICIDS-2017	2017	15	83	Anomaly	Network traffic-based
BOT-IOT	2018	5	43	Feature	IoT traffic-based dataset
N-BaIoT	2018	2	115	Feature	IoT Network traffic
CSE-CIC-IDS2018	2018	7	80	scenarios	Network traffic
IoTID20	2020	5	83	Feature	IoT network communication
IoT-23	2020	12	21	scenarios	Network traffic-based
MedBIoT	2020	3	-	Classifier	IoT Network traffic
MQTT-IOTIDS2020	2020	5	-	sensors	MQTT-based network
DS20S	2020	8	13	Scenarios	IoT-based services
TON-IoT	2021	10	-	Feature	Heterogeneous dataset
WUSTL-IIoT	2021	5	41	attributes	IIoT architected environment
Edge-IIoT	2022	5	61	Scenarios	IoT, IIoT protocol

- NSL-KDD dataset: This dataset addresses some of the flaws in the KDD Cup 99 dataset. The dataset contains network data attributes for each instance. It has 22 attack kinds di-vided into four basic attack groups. There are 127973 records in the training and 22544 records in the testing. Each traffic record has six symbolic features and 35 continuous features. These are the fundamental content and traffic characteristics [8].
- Botnet dataset: Diverse data is needed to imitate real-world traffic correctly for botnet detection systems. This dataset was suggested by Beigi et al [9] and is separated into training dataset that consists of seven distinct types of botnets (Neris, Virut, Rbot, Zeus, SMTP Spam, NSIS, and Zeus control) and sixteen test datasets (Menti, Neris, Sogou, Rbot, ...etc). Botnet topologies might be centralized, distributed, or random.
- The UNSW-NB15: The dataset was generated using the IXIA Perfect Storm tool, including attack traffic and normal user network traffic. Nine attack scenarios were deployed: DoS, analysis, fuzzes, backdoors, generic, exploits, shellcode, reconnaissance, and worms. The dataset has 2,540,044 streams, including 321,283 harmful and 2,218,761 benign, and it was used to extract 49 network traffic features [10].
- CICDS2017: contains network traffic samples gathered for the intrusion detection job. The dataset, which comprises almost 1.5 million PCAPs data simulating real-world traffic data transfers, extracts 83 network flow features with 15 class labels (14 attack+1 normal labels) and 3119345 instances after analyzing 25 user behaviors spanning a di-verse of network protocols, including HTTP and SSH protocols [11].
- Bot-IoT: The UNSW Canberra Cyber Center collected the Bot-IoT dataset using smart home equipment. The network environment contained both botnet traffic and normal. Among the over 72.00.000 records in this collection are DDoS and DoS, Service Scan, OS, Data Exfiltration attacks, and Keylogging. The authors used an MQTT protocol-based dataset to simulate the model behavior of IoT devices. Smart fridges, weather stations, motion-activated lights, smart thermostats, and remotely operated garage doors are also included in the dataset [12].
- N-BaIoT 2018: The dataset includes unusual network traffic collected from nine hacked Internet of Things devices (i.e., thermostats, doorbells, baby monitors, security, and web cameras). IoT devices

were linked with several access points, switches, routers, and servers to simulate a typical network data flow. BASHLITE and Mirai, two IoT-based botnets, were used to launch attacks on IoT devices. A separate CSV file containing the 23 features and every network traffic feature is provided for each attack [13].

- CSE-CIC-IDS2018: consists of seven attack scenarios, such as Heartbleed, DDoS, Web at-tacks, brute-force attacks, DoS, botnet attacks, and penetration. The dataset comprises 16,000,000 instances collected over ten days, 80 characteristics extracted from captured traffic using CICFlowMeter-V3, and each computer's network traffic and system logs [14].
- IoTID20: The dataset includes IoT hardware and network infrastructure built with the smart home devices SKT NGU. A smart home Wi-Fi router and EZVIZ Wi-Fi camera link these two IoT devices. Smartphones, tablets, and laptops are among the additional devices linked to the smart home router, which uses the CIC flowmeter program to create a CSV dataset format by extracting features from Pcap files. They comprise 80 net-work attributes and three label features containing various IoT attack types and families (Normal, DoS, Mirai, MITM, Scan) [15].
- IoT-23: This IoT device network traffic collection includes 23 different samples of IoT network traffic. Three network traffic samples from actual IoT devices and twenty net-work grabs from malicious IoT devices are employed in these scenarios. Three IoT de-vices— a smart door lock, an Amazon Echo Home intelligent personal assistant, and a Philips HUE smart LED bulb—were employed to gather network traffic in benign situations. It is critical to note that these three IoT devices are real, not simulated hardware. There are a total of 30,858,735 benign flows. Nonetheless, the dataset contains twenty-one feature attributes. The types of qualities vary, with some having time-stamp values and others being nominal or numerical [16].
- MedBIoT Dataset: A medium-sized dataset extracted from an IoT botnet; Tallinn University of Technology in Estonia will release this dataset in 2020. The Mirai, BashLite, and Torii botnets make up the entire dataset. This collection includes 83 genuine and simulated IoT devices. Furthermore, the spread of the botnet and C&C communication are the main topics of this dataset. The emulated devices (TP-Link smart switch, intelligent Tasmota Sonoff switch, and TP-Link light bulb) are used in addition to the actual devices (Lock, Switch, Fan, and Light). Each data source, botnet phase, and device type have its own set of pcap files. Every pcap file contains both legitimate and malicious traffic. During malware deployment, all network traffic was collected [17].
- MQTT-IOT-IDS2020: A simulated MQTT network topology is used to generate the dataset. The network comprises a fictitious camera, a broker, 12 sensors, and an attacker. Aggressive scan, regular operation, UDP scan, Sparta SSH brute-force, and MQTT brute-force assault are the five situations that have been observed. The features are extracted following the saving of the raw pcap data. Unidirectional flow features, bidirectional flow features, and packet features are all features that raw pcap files extract [18].
- DS2OS dataset: The DS2OS dataset contains various IoT-based services, such as temperature, window, and light controllers. The user's interactions with the services are recorded and saved in CSV file format. The dataset includes 357952 samples and 13 characteristics. The dataset contains eight classes, 10017 data anomalies, and 347935 normal data. The eight types of assaults are Data Type Probing, DoS, Malicious Control, Spying, Wrong Setup, Scan, Malicious Operation, and Normal [19].
- TON-IoT: The TON-IoT data collection includes heterogeneous data gathered from various sources and is intended to gather and evaluate different data sources from the IIoT and IOT. System network traffic, Linux system logs, Windows, and telemetry data from connected devices TON-IoT is a CSV dataset with a categorized column indicating the normal or attack behavior and the kind of attack subclass, consisting of nine various kinds of attacks, including DOS, XSS, Password Cracking Attack, DDOS, Reconnaissance, Backdoor, Injection Attack, MITM, and Ransomware [20,21].
- WUSTL-IIOT: Washington University in St. Louis developed a cybersecurity-focused network-driven dataset of IoT applications. Through the simulation and modeling of re-al-world industrial

systems, the architecture used to simulate actual industrial applications includes a variety of IoT sensors and actuators, a logger, an HMI, a PLC, and an alarming device. The data set contains 1,194,464 observations, with 87,016 for malicious and 1,107,448 for benign samples. It is made up of 2.7 GB of data that was collected over 53 hours. The dataset contains forty-one attributes chosen based on how their values changed throughout the attack phases. Attacks such as command injection, denial-of-service, reconnaissance, and backdoors are used in the testbed [22].

- Edge-IIoTset: This database examines and categorizes fourteen distinct kinds of attacks on IIoT and IoT protocols into five categories, like DoS, information gathering, MITM at-tacks, DDoS attacks, Injection attacks, and Malware attacks, which are used in intrusion detection systems. IoT data was collected from over ten devices (Humidity sensors and Low-cost digital temperature, Heart Rate sensors, PH Sensor Meters, Ultra-sonic sensors, Soil Moisture Sensor, Water level detection sensors, Flame sensors et). There is a strong correlation between 61 of the 1176 characteristics. Edge-typical IIoT set's attack statistics are 20952648, 11223940 normal, and 9728708 attacks [23].

3. AI support to cybersecurity

In this environment, organizations have begun utilizing AI to help them deal with an expanding range of cybersecurity threats, technological difficulties, and resource restrictions by boosting their systems' durability, resilience, and response. AI systems collaborate with security analysts to alter the rate at which tasks are done. In this sense, the interaction between security operators and AI systems should be viewed as a synergistic integration in which the specific added value of both AI systems and humans is kept and developed instead of being a rivalry between the two [24]. In cybersecurity, there are three primary types of AI applications. The following are the primary reasons for the increased use of AI in cybersecurity.

- Cybersecurity skill shortages continue to be a challenge: There is a global scarcity of cybersecurity expertise. This shortage forces the sector to automate procedures more quickly.
- Impact speed: In certain large assaults, the average effect time on organizations is four minutes. In addition, today's attacks are not confined to ransomware or focusing on vulnerabilities or certain systems; they may change and move in reaction to what the targets are doing. These attacks have a very quick impact, and few human contacts may occur in the interval.
- Operational complexity: Given the growth of cloud computing systems and the reality that these systems can be operationalized and provide services in milliseconds, there can only be a few humans in that loop, and you need to explore a more analytics-driven capability.

AI might assist security teams in the following manner: It can improve systems' robust-ness, resilience, and reactivity. First, AI may boost system resilience or its ability to pre-serve its original believed stable configuration even when it processes incorrect inputs due to self-assess and self-treatment software. This suggests that by outsourcing verification and validation to machines, AI systems might be used to increase robustness testing. Second, by increasing threat and anomaly identification, AI can increase system resilience, or a system's ability to withstand and sustain an attack. Third, AI can improve the ability of a system to respond independently to attacks or system reactions, detect vulnerabilities in other machines, and act strategi-cally by selecting what weaknesses to exploit and execute more aggressive retaliation. An organization's need to conduct a risk-impact assessment is linked to whether to transfer making choices and reactive actions to artificial intelligence [25].

3.1. Applications for artificial intelligence in cyber security

Artificial intelligence is crucial for identifying and preventing cyberattacks. AI may be applied in various cyber security solutions, including spam filtering software, fraud detection software, bot detection software, secure user authentication software, intrusion detection, and advanced malware detection [26]. Figure 3 depicts a few DL methods.

- Spam filter: Gmail employs artificial intelligence to prevent and detect spam and fake emails. Gmail's AI was trained by billions of actual Gmail consumers - every time you select "Spam" or "Not Spam" on an email, you're actually assisting the AI in identifying spam in the future. As a result, artificial intelligence has progressed to the point where it can recognize most minor spam emails disguising as "regular" emails.
- Improving Network Security: Network security prevents unauthorized access to data and files in the system and their malicious exploitation. It also safeguards the secrecy of an organization's network. Artificial intelligence may automatically ana-lyse network traffic for possible violations or

unauthorized access. In network security, your network architecture and security policies are crucial. The network architecture controls how a computer connects to the internet. A network policy is frequently used to formalize the ideas and practices necessary to maintain network security while managing network security. AI can apply security restrictions, and network traffic patterns can be mapped to them.

Figure 3. Some AI applications in cybersecurity

- Fraud Detection: Online financial transactions are growing more popular and rapid-ly expanding. And so is deception. Detecting the fraud after the incident has occurred is pointless. The capacity to detect suspicious conduct and prevent it from occurring is a blessing made available by AI. Fraudsters employ real-time technology such as machine learning and big data analytics. Fraudsters target the weakest link in the chain. Machine learning makes fraud detection and prevention much easier. With various machine learning approaches, big data analysis is feasible. With the analysis, suspicious behavior may be brought to the notice of authorities and remedied. Machines identify and prevent threats rather than depending solely on people. Streaming data may be analyzed in real-time, and fake signal patterns can be identified. A vast amount of data must be supplied to the machines to achieve high accuracy. Accuracy improves when the system self-learns and discovers defects on its own, solving them.
- Detecting Malware: Malware is a severe danger to organizational security that is rapidly spreading. One of the more notable successes of artificial intelligence in Cybersecurity is the exact identification of malware, which is made possible by the availability of massive quantities of data for training deep learning models. Artificial intelligence cybersecurity danger detection systems are particularly good at detecting malware pro-grams that may alter frequently to avoid detection (e.g., meta-morphic malware and polymorphic). Metamorphic malware is a harmful program that changes to evade detection. However, it is much more difficult to detect than polymorphic malware. The primary distinction between polymorphic and metamorphic malware is that the first scenario completely changes its source code. In contrast, the latter retains certain sections while simply altering others. The rationale for rewriting the complete source code is to elude anti-malware technologies more successfully. Traditional cybersecurity techniques make it incredibly tough to identify this malware. As a result, AI cybersecurity's adaptive and learning capabilities are required to detect and respond to these constantly developing threats.
- Botnet Detection: An infected computer network is nothing more than a botnet. It transmits infection via DDoS attacks and spamming tactics such as overflowing mailboxes or distributing infections. A botnet is a network of computers (bots) infected with the same software and controlled by hackers. Botnet detection is often based on network request patterns and time. A master script of orders frequently administers the botnets. A large-scale botnet assault will generally involve numerous "users" executing the same or similar website requests. This might involve failed login attempts (a botnet brute-force attack), network vulnerability assessments, and other vulnerabilities. As a result, botnet identification and removal is a critical duty in the cyber security arena. The deep learning algorithm determines the accuracy of botnet identification and removal utilized.
- Network Intrusion Detection: Intrusion detection identifies actions that seek to compromise a resource's confidentiality; the purpose is to detect malicious activities. The intrusion detection system (IDS) is the most significant component that may be utilized to identify cyber-attacks or malicious activity. AI is crucial in this case for identifying intrusions and customizing IDS.
- Secure User Authentication: AI and its subsets, such as ML and DL, provide accurate identification processing, verification, and authentication. Machine learning is particularly beneficial in determining

whether a consumer is genuine. At the time of the transaction, AI software examines the person's regular behavior, how they conduct their transaction, the devices they utilize, and how they move the mouse or tap the screen. The program does the checks to ensure that the user or per-son is an authorized user of the accounts.

- Hacking Incident Forecasting: may anticipate a hacking event before it occurs. In the re-al world, such foresight may save a lot of money. To accomplish this, we need a comprehensive dataset that includes the most recent occurrences, reports, and other attributes that can be seen outside. Passively collected data is used to establish a forecasting system for cybersecurity infrastructure. As the major goal of the rating system is to evaluate cyber security infrastructure using some metrics based on data obtained passively from the internet, it may be regarded as one of the processes in predicting.

3.2. Deep learning in cyber security

What is more intriguing about DL in cybersecurity is its capacity to detect and prevent as-saults before they occur. Most cyber technologies are reactive, relying on symptoms of a breach to detect a danger. They normally only recognize known threats but are ineffective versus zero-day or unknown attacks. Deep learning methods employ deep neural networks to "think" like a human mind and may adapt to the data qualities on which they are trained. This makes it simpler to adapt to the large number of dangers automatically. While machine learning needs more human interaction to respond quickly enough, DL evolves and learns over time to recognize hazards it has not encountered be-fore and prevent them from taking effect. DL can be successful for intrusion detection and prevention, as it identifies harmful network activity and stops bad actors from entering a network. Previously, machine learning was utilized for these defenses. However, ML methods produced too many false positives, making it more difficult for security personnel to isolate the true issues. Deep neural networks can make IP/ID systems intelligent by analyzing traffic more precisely and distinguishing between good and harmful behavior. Early cyber-attack detection is critical in the event of an attack to limit damage to both individuals and businesses [27].

3.3. Deep learning algorithms

DL techniques need vast processing power and data to solve complex problems. However, they can work with almost any data type. The usual unsupervised, supervised, and hybrid techniques are all used to secure networks and systems, with the following deep learning algorithms being the most popular.

3.3.1 Convolutional neural networks (CNN)

CNN is one of the most significant deep learning models. It refers to a neural network model that operates on two-dimensional data. It can, however, work with both one-dimensional and three-dimensional data. A CNN's pre-processing requirements are far lower than those of other classification algorithms. Even though the filters are hand-engineered using simple approaches, CNN can learn these filters/characteristics with the right training. CNN and FNN are comparable in the following ways: each neuron includes inputs, an activation function, which is utilized in many products, neurons where weights and bias must be learned, and a loss function in the final layer (completely connected) that detects and evaluates the difference between anticipated and predicted value. CNNs are multi-layer neural networks; the first layer is a convolutional layer that is in charge of extracting features; if we have a CNN input shape i × i × d, where i is the input size, and d is the dimension of the convolution layer, it operates as follows:

$$i_{out}\,\frac{i-f+2p}{s}+1 \qquad (1)$$

Where f is the kernel size, p is padding, s is stride, i is the input, d is the dimension, and i_out It is the output of the convolution layer.

The second layer is a pooling layer that reduces the dimensionality of the down-sampled features while maintaining the most significant information. If we have an $i \times i \times d$ feature map, stride s, and a kernel size of f, the output of max pooling can be:

$$i_{out}\,\frac{i-f}{s}+1 \qquad (2)$$

Where i is the input, s is stride, f is the kernel size.

Finally, the third layer, the 3-layer, is a fully linked categorization layer that provides the most crucial information. Deep convolution, pooling, and classification layers have enabled the development of novel CNN applications. This type of network has been used in gaming, video recognition, and image processing. CNN's main benefit in pattern recognition tasks such as object detection and picture recognition are its accuracy; CNN is superior to FNN since it requires fewer parameters than FNNs. Otherwise, they have downsides, such as high processing costs, massive quantities of training data, and the effort required to set up the network effectively for the situation at hand [28].

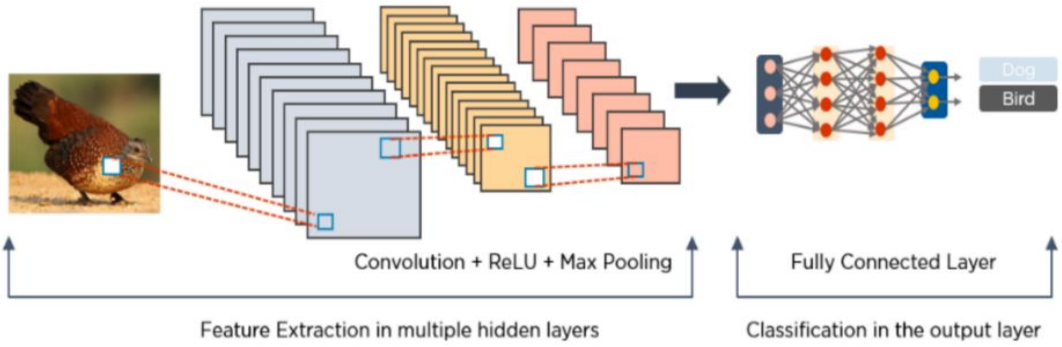

Figure 4. An example of CNN architecture

CNNs are built on three functional concepts: (a) linked weights, (b) spatial sub-sampling, and (c) local receptive fields. Every unit in the convolutional layer gets input from a collection of neighboring units in the previous layer via local receptive fields. This helps cells remove fundamental visual features like borders and corners. Following convolutional layers incorporate these qualities to discover higher-level features; moreover, the theory of bound weights adds the assumption that basic feature detectors that work well on a portion of a picture are probably beneficial throughout the full image. The definition of connected weights requires a collection of units to weigh the same. A convolutional layer's units are arranged in levels. All levels of units are the same weight. As a result, each level is responsible for building a unique feature. Feature maps are the outputs of levels. A convolutional layer can be composed of many levels, allowing the development of numerous feature maps in each place. The vast number of parameters that need to be learned, which might lead to over-fitting difficulties, is one of the obstacles that may arise with CNN training. Strategies such as data augmentation, stochastic pooling, and dropout have been developed for this purpose. Furthermore, CNNs are typically subjected to pre-training, which starts. Pre-training will accelerate the learning process and increase the network's generalization capacity. The most prevalent CNN designs are GoogLeNet (inception), ZFNet, VGGNet, ResNet, and AlexNet [29].

3.3.2 Recurrent neural networks (RNN)

RNN is a form of neural network that can successfully handle sequential input by employing recurrent connections to acquire and use information from previous time steps. RNNs may successfully handle sequences of any length by unfolding the recurrent connections over time. This is accomplished by applying the same weights to all time steps, allowing the network to learn and capture long-term relationships in the data. The inclusion of recurrent connections allows the network to absorb and use information from earlier time steps while processing the present input. This enables RNNs to model the temporal dependencies present in sequential data [30,31]. Figure 5 shows the basic architecture of RNN.

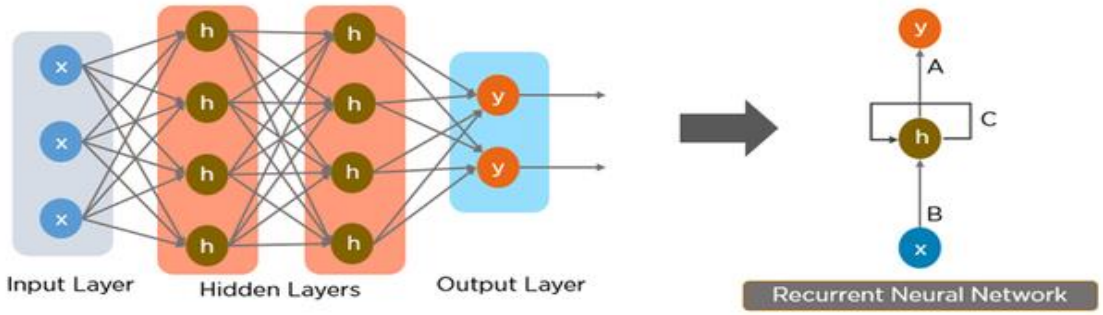

Figure 5. RNN architecture layer

The layer structure of the network is replicated during the period. A is the concealed state at time-step t, regarded as the network's memory. X_t is the vector of size N at time-stamp t in this case. The memory state is calculated using the previous concealed state and the current time-step input. Eq3. shows the computation.

$$A_t = f(WX_t + UA_{t-1}) \quad (3)$$

Where W are the input weights, and U are the weights of the previous state. f is a nonlinearity that is employed to produce the ultimate cell condition. RNN can handle long-term dependencies and is beneficial for predicting time-series. It accepts input of any length, and the model size remains constant as the input size expands. The weights remain static over time and provide prior information weight. RNN operations are slower, and training may be difficult. It has explosion and gradient fading issues.

3.3.3. Long short-term memory networks (LSTM)

Schmid Huber and Hochreiter presented LSTM as an upgraded variation of RNN in 1997 that tackled the problem of bursting and vanishing gradients. LSTM aims to prevent long-term dependency difficulties so that long-term dependencies can be learned and remembered. This is because patterns can be remembered specifically and for an extended period. They are also useful for time series prediction due to their ability to retain past inputs. The three layers (Input Gate, Forget Gate, and Output Gate) of an LSTM interact uniquely compared to other structures. Standard applications for LSTMs include the detection of irregularities in network traffic or standards (sneak detection systems) [30, 32]. Figure 6 displays the three cells, which are also known as gates, of the LSTM:

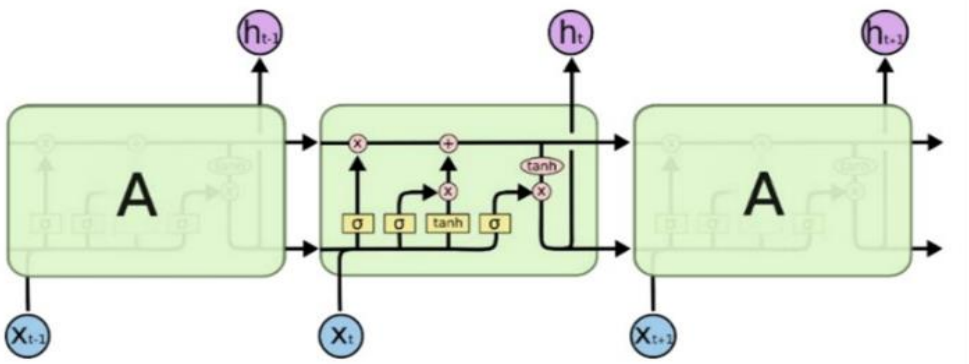

Figure 6. The gates in long, short term memory

- Forget Gate: The cell determines whether the previous cell's information should be retained or forgotten. A sigmoid function is used, which examines the concealed state of the preceding time-stamp $ht-1$ and current time-stamp and returns a value between 0 and 1 as output, with one representing storing the state and 0 representing erasing the state. The forget gate is calculated using Eq. 4.

$$f_t = \partial(W_f.[h_{t-1}, x_t] + b_f) \quad (4)$$

- Input Gate: the input gate controls the storage and quantification of new data. The initial layer uses a sigmoid function to identify which cell values must be changed. In contrast, the subsequent layer takes the new data. It employs the tanh activation function, which converts the new data between -1 and 1. The two sections are joined, and the state is updated.

$$i_t = \partial(W_t.[h_{t-1}, x_t] + b_i), \hat{\mathrm{C}}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \quad (5)$$

The formulation input gate is presented in Eq5.

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \quad (6)$$

- Output Gate: The output layer uses a sigmoid function to select which features of the cell condition will be output. The cell condition is transmitted via the tanh activation function, which returns values ranging from $-1\ to\ 1$.

$$o_t = \partial(W_0.[h_{t-1}, x_t] + b_o) \quad , \quad h_t = o_t * tanh\,(C_t) \qquad (7)$$

3.3.4 Deep belief networks and deep Boltzmann machines

Deep Belief Networks (DBN) and Deep Boltzmann Machines (DBM) are deep learning architectures from the "Boltzmann family," with the Restricted Boltzmann Machine (RBM) being used in the learning module. The RBM is a type of randomized neural network. Direct connections to the lowest levels and undirected connections to the upper two layers construct an RBM. The DBM's tiers all feature undirected connections. Figure 7 depicts a graphical depiction of DBNs and DBMs. After introducing the RBM to their fundamental model, we will define the DBNs and DBM in the following parts.

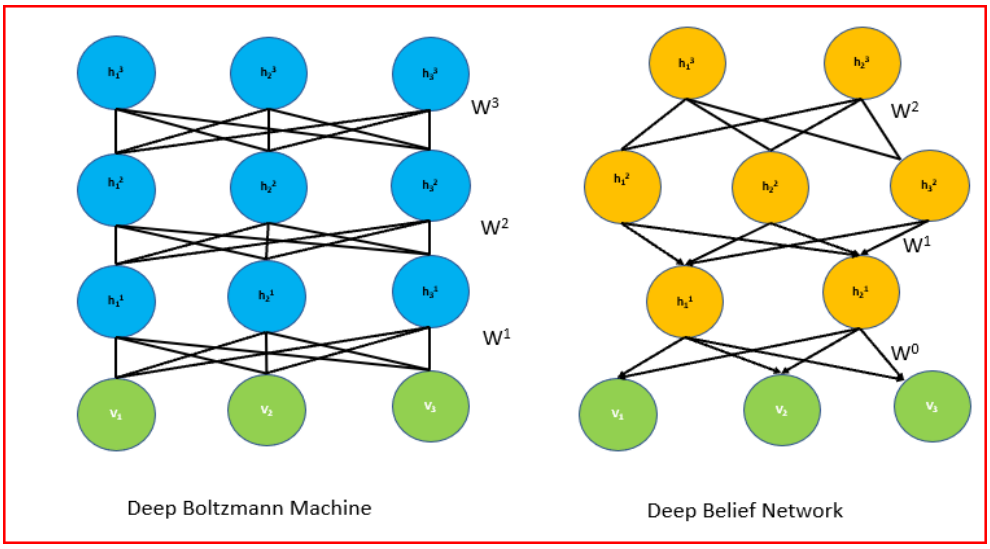

Figure 7. Architecture for deep belief network and deep Boltzmann machine

RBM can also be referred to as a stochastic neural network; this is a popular DL framework due to its ability to learn about the distribution of supervised and unsupervised input probabilities. RBM differs from Boltzmann machines in that the restricted connection exists inside the layer between the modules. RBM is an undirected two-layer visual model with visible and concealed variables. An RBM is distinct from a Boltzmann machine because it requires the hidden and visible units to create a bipartite graph [33]. This constraint improves the effectiveness of training methods where Eq8 defines the function $E\ (v, h)$.

$$\mathrm{E\ (v,\ h;\ \theta)} = -\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij}\, v_i\, h_j \ - \sum_{i=1}^{D} b_i v_i - \sum_{j=1}^{F} \alpha_j\, h_i \qquad (8)$$

Where F and D are the numbers of concealed and visible units, where $\theta = \{a, b, W\}$ are the model parameters; that is, Wij are real-valued weights that indicate the relationship between hidden unit j and visible unit i, and bi, aj are real-valued prejudices. The following equations can be used to compute the joint distribution through hidden and visible units: Eq9.

$$\mathrm{P\ (h,\ v;\theta)} = \frac{1}{\mathrm{Z(\theta)}} \exp\ (\mathrm{-E\ (h,\ v;\ \theta)}), \quad \mathrm{Z(\theta)} = \sum_{\mathrm{v}}\sum_{\mathrm{h}} \exp(-\mathrm{E(v, h;\ \theta)}) \qquad (9)$$

Where $Z(\theta)$ is the normalizing constant, Eq9 and Eq10 may be used to calculate conditional distributions over visible v and hidden h vectors.

$$\mathrm{P(h|v;\theta)} = \prod_{\mathrm{j=1}}^{\mathrm{F}} \mathrm{p(h_j \mid v)},\ \mathrm{P\ (v \mid h;\ \theta)} = \prod_{\mathrm{i=1}}^{\mathrm{D}} \mathrm{p(v_i \mid h)} \qquad (10)$$

DBMs are deep models that use RBM as their building component. DBM is similar to RBM, except DBM includes more hidden layers and variables. The DBN design differs because the lowest levels create a driven generative model. At the same time, the two upper layers form an undirected graphic model. In contrast, all connections in the DBM are undirected. Undirected connections are employed throughout all tiers. A DBM trains layers of a shared unsupervised model concurrently during network training, and the DBM uses a randomized maximum probability approach to maximize the lower limit on probability. Because of the

connections between the concealed neurons, estimating the distribution across the posterior hidden neurons from the visible neurons is sometimes impossible. DBM can discover more complicated internal representations, which are a promising solution to handle recognition difficulties. Furthermore, in semi-supervised learning circumstances, high-level representations are frequently created from relatively little labeled data, and a substantially large amount of unlabeled inputs may subsequently be utilised to modify the model for particular tasks. Furthermore, it may include top-down feedback and an initial bottom-up pass, allowing DBM to more robustly disperse, distribute, and cope with ambiguous inputs [34]. DBNs are generative models that offer a probability distribution on labels and data, and they are another sort of RBM. They are built by stacking and greedily training RBM, with the numerous hidden layers learning by utilizing the hidden output of one RBM as input data for training the following layer of RBM. A DBN utilises an effective layer-by-layer greedy learning technique to initialise the deep network and complete all weights and outputs in the sequel [35]. A DBN with l hidden layers reflects the sum of the hidden layer. h^k, and the visible layer x, distributions. where $k = 1, 2, \ldots, l$, as follows:

$$P\,(x, h^1, \ldots, h^l) = \left(\prod_{k=0}^{l-2} p\left(h^k \mid h^{k+1}\right)\right) P\,(h^{l-1}, h^l) \qquad (11)$$

where $x = h^o, p(h^k R / h^{(k+1)})\ is$ a conditional distribution for level k of visible units dependent on hidden RBM units at level $k + 1$, and $p(h^{(1-1)} / h1)$ is a combined distribution between visible and hidden layers at the top-level. DBNs with RBMs may be implemented as the constructing blocks for every layer using greedy, layer-wise, unsupervised training ideas.

3.3.5 Generative adversarial networks (GANs)

GAN is a framework for machine learning in which two networks of neurons compete to enhance their prediction accuracy utilizing deep learning techniques. GANs are frequently unsupervised and learn in a cooperative zero-sum game setting in which one individual's gains equal another individual's losses. A GAN comprises two neural networks: the discriminator and the generator. The discriminator is a deconvolutional neural network, while a convolutional neural network is the generator. The generator's objective is to produce outputs that may be mistaken for genuine data. Generative models, in essence, create their training data. The discriminator's objective is to identify if the outputs it receives were created on purpose. The discriminator network is taught to distinguish between the generated data and genuine samples. At the same time, the generator is trained to generate phony data. The generator is penalized if the discriminator rapidly recognizes the generator's deceptive data, such as a picture that is not a human face. When the adversarial networks continue their feedback loop, the discriminator becomes increasingly skilled at identifying wrongly created data. At the same time, the generator produces output that is more reliable and of greater quality [36,37].

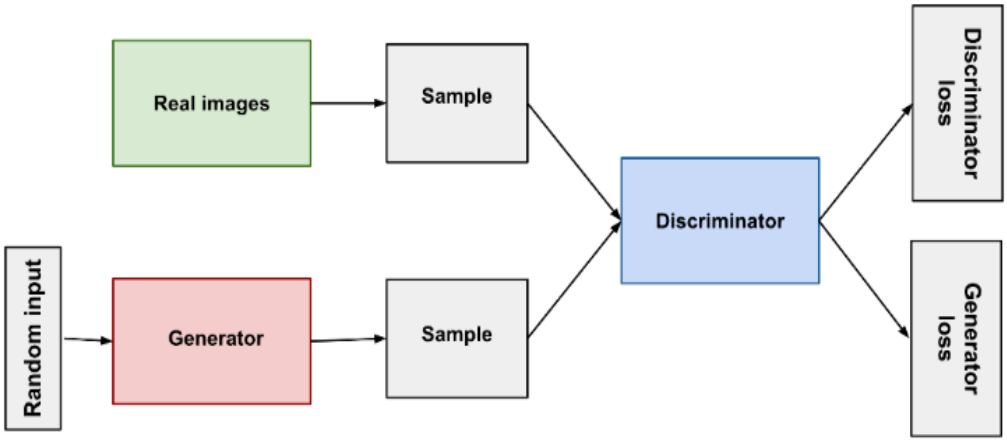

Figure 9. GAN architecture

GANs are often classified into three categories:

- Generative. This discusses data using a probabilistic model.
- Adversary. In an adversarial scenario, a model is trained.
- Networking. Deep neural networks can train artificial intelligence (AI) algorithms.

Establishing the intended result and compiling an initial training data set based on those parameters constitute the first step in creating a GAN. After that, the generator is fed this data randomly until the desired level of basic output accuracy is reached. The discriminator is then fed the generated samples or visuals with actual data points from the original notion. Following the data digestion process of the generator and discriminator models, backpropagation optimisation takes place. As it goes through the data, the discriminator gives each picture a probability between 0 and 1, indicating its validity: 1 for real photographs and 0 for fakes. Once the intended result is achieved, these values are manually verified, and the process is repeated.

3.3.6 Autoencoders (ACOD)

Autoencoder is a neural network that learns how to understand the dimensionality reduction in the input dataset and rebuild the original dataset using an unsupervised technique. The learning algorithm is based on using backpropagation. It is a generative model using a non-linear feature extraction method, Autoencoder, which uses data-driven learning to extract features. It is unsupervised since it is trained to replicate the input vector instead of applying class labels. The number of neurons in the input layer is the same as in the output layer, as illustrated in Figure 10, with complete neuronal connections from one layer to the next. Autoencoder contains three layers: the input layer, the hidden layer, and the output layer. That tries to recreate its output layer input. As a result, the output layer has the same number of units as the input layer. Typically, the visible layer contains more neurons than the hidden layer. Try encoding or expressing the input more compactly. It has certain principles with RBM; instead of stochastic units with a specified distribution, it generally utilises a deterministic distribution, like in RBM. This network uses nonlinear mapping to approximate an ideal function to minimize input and output errors. As a result, it is clear that the abstract representation at the hidden layer contains critical data about the original input and may be regarded as a high-level function [38].

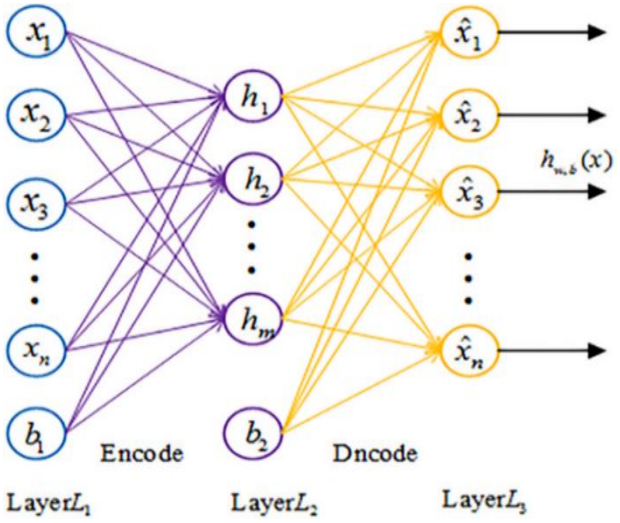

Fig 10. Autoencoder structure

In general, the functioning of an autoencoder, as illustrated in Figure 10, may be separated into two stages: encoding and decoding. Both encoding and decoding are forward propagation techniques that execute non-linear transformation using an activation function. Encoding converts the original input into an abstract representation, and decoding returns the representation while attempting to minimise reconstruction error. The following equations describe an encoding and decoding operation during the encoding process.

$$y' = f(w\,x + b) \tag{12}$$

Where x is the input vector, w is the weight matrix, f is a nonlinear activation function, b is the bias vector is the parameters to be modified, and y is the hidden representation in decoding stages.

$$x' = f(w'\,y' + c) \tag{13}$$

Where x' Is the reconstructed input at the output layer, c is the bias to the output layer, and w' Is the transposition of w; using the following equations the autoencoder parameters can be updated:

$$w\_new = W - \eta\partial E/\partial W, \quad b\_new = b - \eta\partial E/\partial b \tag{14}$$

Where E is the reconstruction error of the input at the output layer, $w\_new\ and\ b\_new$ At the end of the current iteration are the modified parameters for w and b.

4. Overview of federated learning

FL is a decentralized ML technique that permits several devices or entities to train a shared model while maintaining their data locally collaboratively. It addresses privacy and data ownership issues by minimizing the need to share raw data. Deep learning algorithms can be applied within the context of FL to train complex models. This allows enterprises to create a common global model without storing training data in a centralized location. FL enables multiple players to collaborate on constructing a single, robust system without sharing data, privacy, access to heterogeneous data, and addressing critical challenges to data access rights and security are just a few examples.

A centralized server or coordinator in FL initializes the model architecture and distributes it to the entities or participating devices. This model serves as a foundation for training. Each training cycle involves the participation of a subset of devices. This choice process can be based on various criteria, such as device capabilities, availability, or data diversity. Each selected device downloads the current model and performs training on its local data. DL algorithms can be used for this training. The devices use their local data to update the model parameters iteratively, typically through techniques like stochastic gradient descent (SGD) or its variants [38]. After local training, the devices return their updated model parameters to the centralized server. The server aggregates these model updates using techniques like Federated Averaging. This involves averaging the model parameters from cooperating devices to make a globally updated model. The centrally aggregated model is then distributed back to the participating devices, replacing their previous models. The devices repeat the local training process with the updated model in the next training round; this work is repeated for multiple rounds to refine the shared model further. The number of rounds can be pre-determined or based on convergence criteria. Federated Learning protects privacy by storing the training data on local devices. The central server only periodically receives model updates, typically encrypted to protect sensitive information, to the model evaluation; a separate validation dataset or a subset of devices may be employed to evaluate the efficacy of the shared model. This evaluation helps monitor progress and determine convergence. Federated Learning with deep learning algorithms offers several advantages, including data privacy, reduced communication overhead, and leveraging distributed computing resources. It enables the training of complex deep learning models while respecting privacy constraints, making it suitable for scenarios where data cannot be directly shared or centralized [4,39]. Federated learning helps AI systems learn from a wide variety of data from many locations. Google already uses FL, enabling incredible predictive input features for the Android keyboard (Gboard), on-device search for phones, and other applications. Recent advancements have concentrated on eliminating statistical barriers and increasing FL security. To enable sensitive-privacy systems where training data is decentralized at the edge and where the costs and dangers involved with sensitive data management are high, leading service providers have employed FL methods. Additionally, it allows for the development of intelligent apps while giving consumers control over their data [40].

4.1 FL's motivations for cybersecurity

The traditional approach to cybersecurity makes it more difficult to acquire and share data in a privacy-invading manner. Similarly, data aggregation from several data providers is a difficult undertaking. FL might be used to mitigate cyber-attacks while simultaneously protecting data privacy and security. Various variables impact the usage of FL for Cybersecurity and the methodologies employed. The following are the reasons for adopting FL for Cybersecurity[40].

- Data Privacy: User information is distributed among several entities in the feature space, with each entity monitoring a unique data characteristic relevant to all users. Instead of transferring raw data to the server, every entity in the fundamental network communicates the parameters of the local model learned using locally gathered data characteristics. This helps to safeguard privacy.
- Confidentiality: Any unauthorised access to data creates a data breach and a cyber threat. Only authorised individuals have access to privileged and sensitive information. While FL is used, local training of edge device models ensures authorized access.
- Data Security: Using FL to secure information from different assaults is possible since raw information and data are not exchanged via the network; only updates are delivered to the server.

- Availability: Access to user information must be available when needed. Availability is tied to system uptime and dependability, both harmed by malicious threats such as cyber-attacks and unauthorized access. The local model is available on the edge device when FL is employed. In contrast, the global model is accessible to users via the cloud.
- Integrity: Keeping data consistent, correct, and complete is crucial in cybersecurity. A hacker may modify the data sent by the sender before it reaches the receiver. Because the FL approach safeguards privacy, sensitive data is not transmitted outside the local context.

4.2 Centralized vs decentralized vs federated approach

To comprehend federated learning, we will contrast it with more traditional centralized and decentralized approaches. as shown in Figure 11.

- Centralized learning: This method gathers data for learning models from many sources and connects it to a cloud server to produce a common model that may be applied to various devices. The key feature of centralised learning is that the model may employ generalisation information from a group of devices and immediately work with more pertinent ones. On the other hand, traditional centralized learning has certain drawbacks, such as bandwidth, privacy, latency, and connectivity [22].
- Distributed Learning: Distributed models are trained the same way as centralised models, only that they are trained on numerous participants individually [41]. Participants train their models separately and communicate weight changes to the central server during the training stage of a distributed algorithm. At the same time, the central server gets updates from participants and averages output. Following certain communication cycles, the central cloud server does convergence testing.
- Federated learning (FL): is essentially an ML method for training algorithms via decentralized edge devices while retaining data samples locally. It applies model training to data spread over millions of devices. Simultaneously, it allows you to enhance outcomes achieved at the periphery and in the center—a step-by-step federated machine learning process. Selecting a model that has already been trained on the central server or has not been trained at all, the initial model would then be distributed to customers as the next phase (devices or local servers). Each customer continues to train it on-site with its local data. The crucial thing to remember is that this training data can be kept private. When locally trained models are relayed to the central server over secured communication channels, the server receives no real data but trained model parameters. All client updates are averaged and pooled into a common model, boosting accuracy. This model is then returned to all devices and servers [22,42].

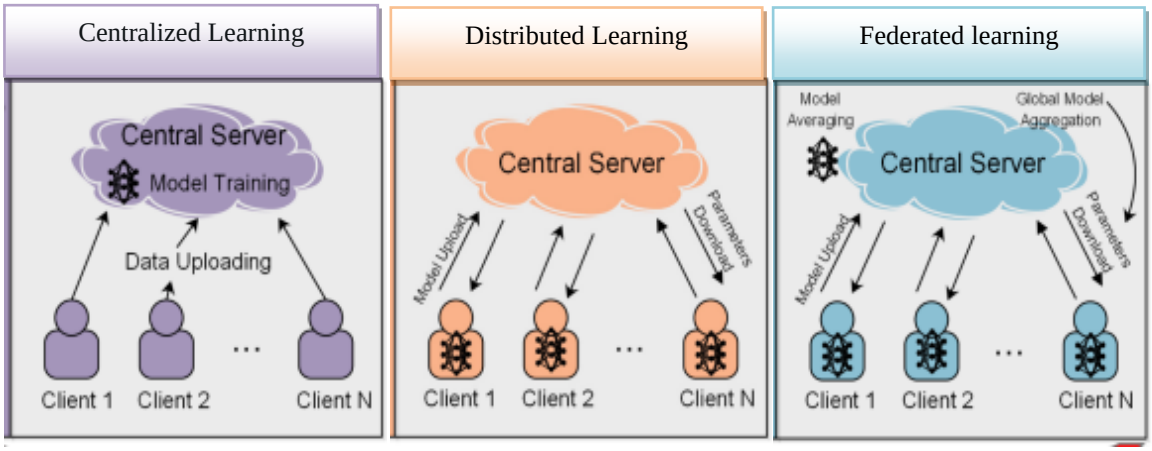

Figure 11. The centralized, distributed, and federated learning model classifier [43]

4.3 Types of federated learning

FL has taken on different shapes and forms over time in the computing realm. The variations depend on Schemes and Data Partitions.

4.3.1. Types based on the schemes

Cross-Silo Federated Learning Model: The cross-silo FL architecture is made up of numerous silos connected to a central server. Many businesses, for example, can interact through a single network while keeping their raw data separate in silos. It allows organizations to process massive amounts of data while protecting privacy [42]. The architecture consists of end users from several enterprises, a silo from each enterprise, and a central server. As seen in Figure 12.

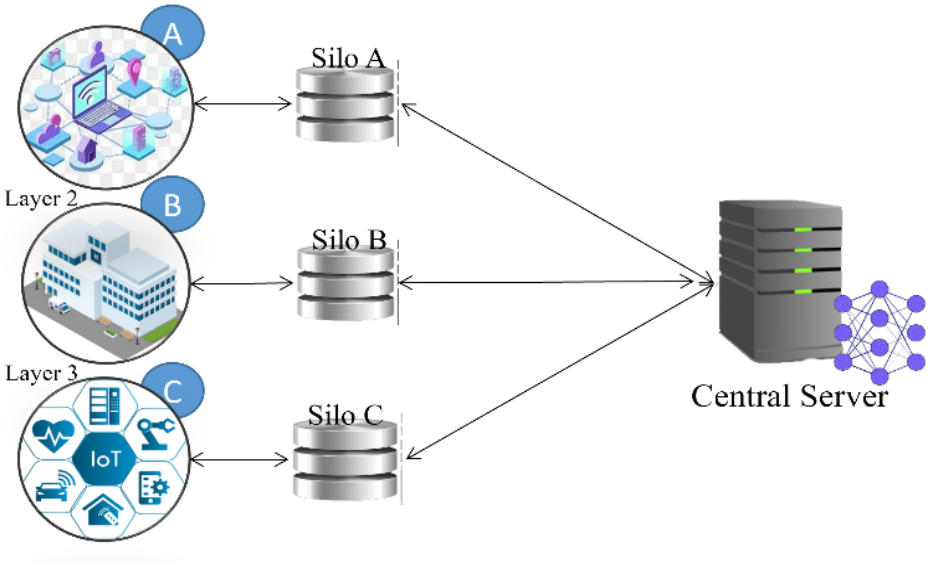

Figure 12. Cross-Silo FL model

Cross-Device Federated Learning Model: This federated learning approach uses data from individual network-connected devices. Several end-user devices, including laptops and smartphones, serve as data sources for locally training the model. The central server then brings these together to construct a global model. Based on their data, each user device builds a locally trained model. Because there are so many end-user devices, training the model with many data points increases its forecast accuracy using one or more steps. Each device then computes a stochastic gradient of the sequential model. Finally, a global model aggregates each device's determined gradients (parameters) with the central server [42].

4.3.2 Types based on the data partitions

Based on these two architectural standards, data is designed into three structural shapes (Horizontal, Federated Transfer, and Vertical) Learning models.

Horizontal Federated Learning Model: This structural type makes advantage of a shared feature space shared by numerous network clients, yet each sample stays unique, as illustrated in Fig 13(a). It is called sample-based FL; it typically works with clients with comparable datasets. The horizontal federated learning technique permits the development of a multi-task federated learning system [43].

Vertical Federated Learning Model: As shown in Fig 13(b), a vertical network deals with many features using the same sample area. Another name for it is feature-based FL; this concept is most typically used in business-to-business contact and data sharing when many organizations working with the same clients shares a common network—the popular vertical federated learning system PyVertical [44].

Federated Transfer Learning Model: This federated learning paradigm includes horizontal and vertical federated learning systems. It can work with datasets with diverse feature spaces and sample spaces, as seen in Fig 13(c). In addition, it enables several entities to use a global model without sharing a common feature space, ensuring the privacy of their data. The idea is to train a model for a specific problem on a large dataset and then apply it to another problem in a related area [45].

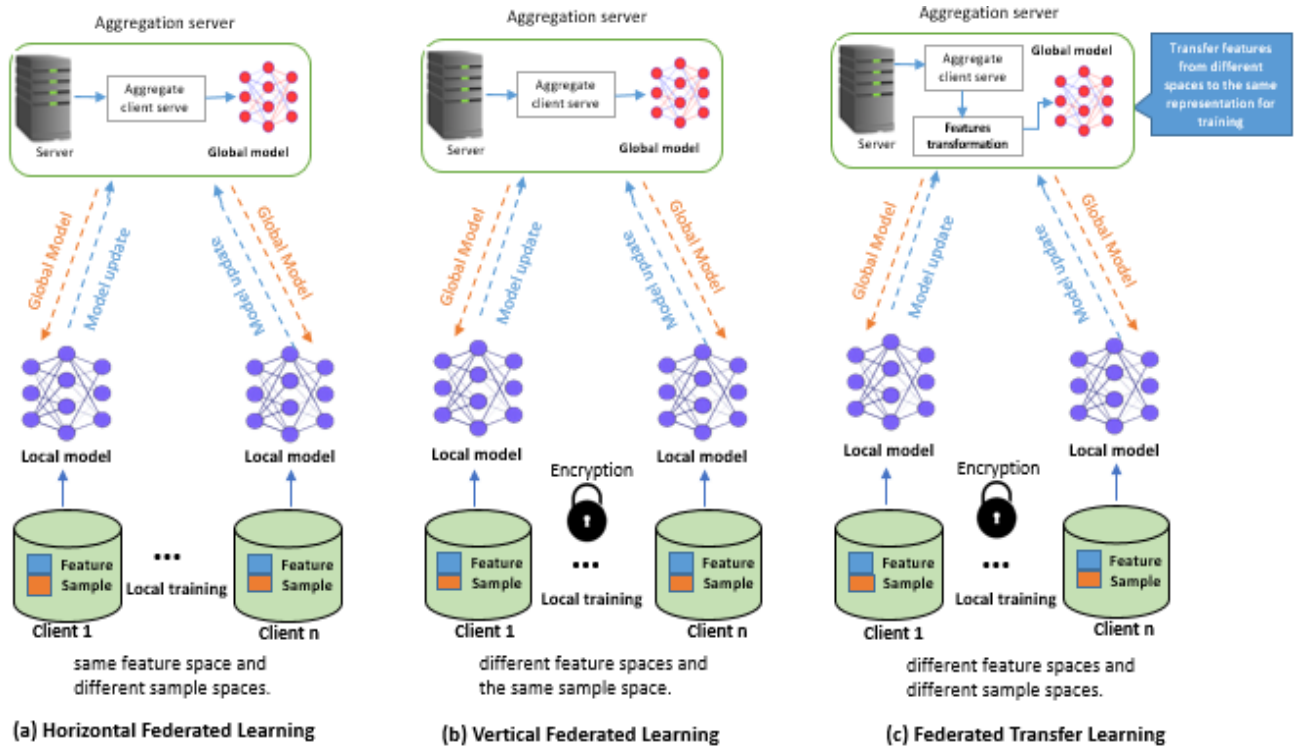

Figure 13. The FL classifications. (a) A database that shares the same feature space but has various sample spaces. (b) A database with the same sample space but the various feature spaces. (c) Database with various feature spaces and sample spaces.

4.4. Federated learning applications

Federated Learning has enabled numerous significant applications. We will go through some of the more important ones in this section. Statistical models are used in smartphones to power apps like face recognition, voice recognition, and next-word prediction by learning user behavior over a vast pool of phones. To preserve their privacy, users can choose not to disclose their data. FL can produce precise smartphone forecasts without disclosing private information or affecting the user interface. Entire institutions or organizations may be called "devices" in federated learning. For example, hospitals retain massive volumes of patient data that predictive healthcare programs may access. Hospitals, however, follow rigorous privacy standards that require data to be kept locally. FL is a fantastic alternative for these applications since it reduces network overhead and facilitates private learning across numerous devices/organizations [4].

Sensors are used in modern IoT networks, such as wearable technology and smart homes, to gather and interpret data in real-time. An autonomous vehicle fleet, for example, may require a simulation of pedestrian, construction, or traffic behavior to work successfully. However, constructing aggregate models in this case behavior may be difficult because of privacy concerns and each device's limited connection. Federated learning approaches enable the development of models that swiftly respond to alters in these systems while safeguarding user privacy. Integrating financial, medical, and other data from many sources is necessary when creating a data service platform for the insurance sector. An insurance company must consider multi-party data to improve its risk management capabilities and commercial expansion. In the insurance industry, effective data utilization without violating individual privacy is a major challenge [46,47].

4.5. FL implementations

This section provides a brief overview of popular FL implementations, which are mentioned in Table 2 according to their focus and supporting software. The interested reader could also refer to more extensive explanations and comparisons of the various implementations found in [48,49]. Regarding framework selection, it gives a uniform criterion for assessing the most popular FL frameworks regarding capability, usability, and performance.

Table 2. Summarises FL platforms and the software that supports them.

platform	focus	Supporting software
PySyft	secure and private	Python
TFF	Training decentralized data	Python, Tensor Flow
FATE	security and privacy	all FATE projects
Open FL	sensitive information	Haxe,JavaScript, TypeScript
IBM	learning topologies, security	Python
Tensor/IO	Mobile device	Tensor Flow
FFL-ERL	A real-time system, Parallel computing	Erlan
CrypTen	privacy-preserving	PyTorch

PySyft is an open-source Python library for confidentiality and security. PySyft separates private data from model training using FL, SMPC, and DP. The Open Mined community designed it, and it mostly works with deep learning technologies such as TensorFlow and PyTorch. Both dynamic computations over hidden data and static computations—graphs of computations that may be carried out later in a different setting—are supported by PySyft. It largely functions with PyTorch and TensorFlow, two deep learning frameworks. PyGrid2 facilitates FL on the web, mobile, and other devices because PySyft does not support network communication. PySyft is not yet ready for production since it is still in beta release [50].

Tensor Flow Federated (TFF) is a Python-based, open-source framework created by Google for training machine learning models using decentralized data. TFF operates at two key application programming interface (API) layers: The FL application programming interfaces provide high-level APIs that allow developers to import existing machine learning models into TFF without fully grasping how federated learning works. Federated Core API (FC) provides low-level APIs for creating unique federated algorithms [51].

Federated AI Technology Enabler (FATE) is an open-source platform built on HE and SMPC. It includes ML methods such as tree-based algorithms, logistic regression, and other DL and TL approaches. FATE supports standalone and cluster installations and may be deployed on Linux or Mac systems. It creates safe computation protocols by combining multi-party computing and homomorphic encrypting [52].

Open Federated Learning (Open FL) is a Python 3 library for FL framework that enables organizations to train a model cooperatively without disclosing sensitive information. Statistical models can be trained using any deep learning framework, such as Tensor Flow or PyTorch, via a plugin method [53].

IBM Federated learning (FL) is an open-source Python library designed to facilitate the easy implementation of FL in productive environments. IBMFL is an enterprise-level solution that provides a basic FL layer over which more advanced features can be added. It incorporates both unsupervised and supervised learning methods and reinforcement learning, as well as DNNs, while facilitating the easy implementation of new FL algorithms [54].

Tensor/IO is a cross-platform lightweight on-device ML toolkit that adds Tensor Flow and Tensor Flow Lite support to iOS, React Native, and Android apps. Tensor/IO does not execute ML directly but collaborates with an underlying library, such as Tensor Flow, to facilitate the installation and use of smartphone models [50].

Functional Federated Learning in Erlang (FFL-ERL) is a framework for FL written in Erlang, a dynamically typed, structured programming language with parallel processing capabilities that may be used to develop real-time systems [55].

CrypTen is a framework based on PyTorch that makes it simple to research safe and privacy-preserving ML. CrypTen allows machine learning researchers, not cryptography specialists, to experiment with ML models using safe computing approaches by integrating with the widely used PyTorch API [50].

4.6. Aggregation algorithm

The aggregation algorithm is critical in any federated learning and topology style setting. The logic aggregates the local model updates from all customers who participated in the training cycle. Many suggested techniques aim to improve the privacy of local model changes, which can be turned into a global model that the entire system may utilize, conserve communication bandwidth, or permit concurrent client updates. Federated averaging differs depending on the pre-configuration settings of each FL implementation. The current algorithms are discussed below:

- SGD - Stochastic Gradient Descent: This algorithm iteratively goes down a function's gradient. The basic goal is to reduce the gradient to the smallest value. When executing SGD, the client's database is used to construct a single stochastic gradient for the particular loss function. The central server receives all of these gradients from numerous clients and averages them. It generates a synchronous model by bringing all data to a single gradient point for each client before averaging it. One downside of SGD is the sluggish processing of data [56].
- Federated Averaging (FedAvg) is a common algorithm used in FL to aggregate model updates from several devices or clients. Google researchers introduced it in their paper [39]. The goal of FedAvg is to leverage the local model updates from associated devices to create a global model that represents the knowledge learned from the distributed data without directly accessing the raw data. FedAvg achieves this by averaging the model updates received from each device, thereby effectively aggregating the knowledge from multiple devices [57].
- Stochastic Controlled Averaging for Federated Learning (SCAFFOLD): This method's calculations are more controlled, as the name implies. It addresses the FedAvg convergence problem for heterogeneous data by introducing a correction value at each gradient calculation in the iterations performed locally. Similar to the previous approaches, the outcomes of these local iterations are subsequently relayed to the centralized server for averaging. While the process is sped up due to local efficiencies in gradient calculations, the correction term ensures that the data is centralized and easily converges [57].
- FSVRG: The algorithm aims to perform a complete computational operation on each client, followed by multiple updates. Iterating over random data permutations and performing a single update are used to perform the changes. The FSVRG method is primarily concerned with sparse data. Some traits appear only occasionally in the data set [57].
- FedProx: FedAvg and FedProx are comparable in that each iteration necessitates the selection of device groups. Local updates are performed and then aggregated to provide a global update. FedProx is intended to be an improvement on the FedAvg algorithm. Where small changes are made to boost performance and diversity, because various FL devices have different restrictions, it would be unrealistic to expect them to do the same amount of work. The algorithm, in particular, accepts partial work rather than uniform labor. Tolerating partial work allows system heterogeneity and improves stability over the FedAvg technique by default [58].
- Federated Matched Averaging (FedMA): FedMA was created to aid in the federated learning of modern neural network designs. To begin, the data center collects the first layer weights from clients and utilizes one-layer matching to generate the federated model's first layer weights. These weights are then sent to clients, who use them to train every one of the layers on their datasets. This process is continued until the last layer is reached; at this point, a weighted average is computed based on the fraction of data points per client. In FedMA, communication is also included. In order to make their local models the same size as the original models, clients first obtain the global model at the beginning of a new cycle. Sizes might be decreased, making them simpler to manage [59].
- Brain Torrent: In a medical environment, the algorithm was used. Brain Torrent runs in a peer-to-peer environment; the purpose of Brain Torrent is for all centers to communicate with one another rather than rely on the primary server as in standard FL. Brain Torrent was created to assist mobile device users [60].

4.7. Challenges in federated learning

FL is a new sort of AI that is developed for model training in a distributed and heterogeneous. Various challenges act as fundamental hurdles to enabling FL on the possibility of millions of devices achieving the full potential of FL in applications.

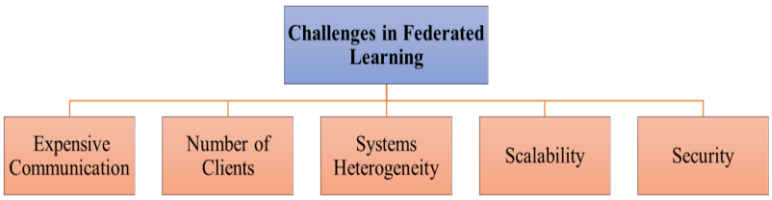

Figure 14. FL challenges

4.7.1. Expensive communication

Federated learning is creating a one-of-a-kind, worldwide statistical model using data saved on possibly millions of remote devices linked to the network via wireless and wired connections. The communication network will be worse than local computing because of restricted resources and the rising number of user devices and equipment. Communication in such networks may be far more costly than in standard data center configurations. Implementing federated training methodologies on standard mobile networks results in costly communication. An intelligent communication protocol that can efficiently send short training messages rather than the complete dataset via the network is required to deal with expensive communication. Edge computing allows the FL to overcome communication issues [60,61].

4.7.2. Number of clients

In federated, the number of participants (clients) is essential for storing and analysing collaborative learning models. These clients typically refuse to participate in the training intentionally or accidentally. Intentionally signals that the client is not passionate about federated learning: Methods, Future Directions training, and Challenges, but mistakenly is due to a terrible network problem, a lack of resources, a low battery backup, and so on. Managing such many clients is tough and a substantial barrier [62].

4.7.3. Systems heterogeneity

A current network is considered to have various heterogeneous levels regarding network, hardware, application, data storage, devices, and battery levels. The existence of many types of networks, such as LAN, WAN, MAN, and PAN, is referred to as network heterogeneity. Mobile phones, tablets, laptops, smartphones, and other portable devices that link with other devices are examples of device heterogeneity [63]. This wide range of options is a challenge in federated. Furthermore, any system may have a unique data storage structure, and the independently and symmetrically distributed criteria may be violated, complicating model analysis. Furthermore, since each participant device gathers data based on its unique usage pattern and local environment, which may differ from other participants, different data distributions among all participants are possible because data on contributor devices is collected by the devices themselves [64].

4.7.4. Scalability

Scalability is another prevalent problem in federated since many customers might approach a specified limit. In contrast, communication via the parameter server can be simplified to just one round for both players and the server [65]. It additionally reduces communication costs for each client. On the other hand, communication via parameter server continues to be a challenge for connection-efficient distributed training since download and upload to/from the server requires effective compression to lower transmission costs, time, and energy [66].

4.7.5. Security

Another major concern in federated is security. Federated learning protection is tied to the participant (client) and communication network, which may violate FL key security standards such as authentication, confidentiality, and integrity. Cyberattacks, for example, pose significant network security risks to FL. Clients

can provide sensitive information to an invader, a third party, or a fictional central server. FL should take the lead in data preservation by releasing model updates rather than raw data [67,68].

5. Commonly used evaluation criteria

Various indicators and measurements for each mission can be used to evaluate any learning model. A confusion matrix is a formal method for outlining the specifics of the learning model. A confusion matrix is a table that sums up a prediction's classification model or performance. A confusion matrix divides the outcomes of binary or multiple categorizations into four groups. It returns the classification outcomes in the form of True Negative (TN), False Negative (FN), True Positive (TP), and False Positive (FP) values, which are then used to construct additional measures. In addition to the mistake rate, other criteria should be prioritised, such as space difficulty, time difficulty, and the flexibility of learning algorithms. Moreover, the significance of the metric varies depending on the application. Assume that it is necessary to consider FN when determining whether a financial transaction is legitimate or fraudulent. A financial transaction with a single value of FN could result in a massive financial loss. The above terms also calculate the confusion matrix's metrics [69,70,71].

- Accuracy: the percentage of samples and applications correctly classified in a dataset. The higher accuracy value indicates that the classifier is accurate.

$$Accuracy = (TN + TP)/(TP + FN + FP + TN) \quad (15)$$

- Precision: measures how many benign, positive samples and applications were correctly identified in the dataset. When the precision value of a classifier is higher, it performs better and is more desirable.

$$Precision = TP/(FP + TP) \quad (16)$$

- F1-Score: the F1 score represents the balance of a classifier's precision and recall in a single metric by taking the harmonic mean of these two values.

$$F1 - score = 2.(Precision * recall) / (precision + recall) \quad (17)$$

- True Negative Rate (TNR): the percentage of accurately categorized attacks, malicious, negative samples, and applications in the collection.

$$True\ Negative\ Rate = TN/(FP + TN\ ) \quad (18)$$

- Recall: This measure computes the fraction of true positive predictions among all potential positive predictions.

$$Recall = TP/TP + FN \quad (19)$$

- False Negative Rate (FNR) calculates the fraction of benign, positive samples and applications wrongly categorized with the total number of applications in the dataset.

$$False\ Negative\ Rate = FN/\ (FN + TP) \quad (20)$$

- Error Rate: This metric calculates the proportion of samples and applications in the dataset that were incorrectly classified.

$$Error\ Rate = (FN + FP)/(TN + FN + FP + TP) \quad (21)$$

- False Discovery Rate (FDR): It quantifies the proportion of apps comprised of dangerous, harmful samples incorrectly classified as all attacks—and malicious, negative models and applications correctly classified in the dataset.

$$False\ Discovery\ Rate = FP/(FP + TP) \quad (22)$$

6. Conclusion

As technology advances, the quantity and sophistication of cyberattacks increase. Traditional cybersecurity solutions struggle with many challenges and security issues to identify unknown threats like new malware variants and zero-day attacks in such a complex technical environment. Cybersecurity systems have used ML approaches to solve these difficulties, albeit with little effectiveness against unplanned or unpredictable threats. Meanwhile, deep learning approaches improve learning procedures and yield promising outcomes in various applications, including cybersecurity. To a large extent, the success of DL is dependent on new advances in software engineering and the vast supply of training data. This overview study examines deep learning algorithms for detecting and classifying all sorts of cyberattacks. To that aim, the notion of cybersecurity is explained, and a thorough investigation of DL techniques is performed, encompassing all elements of cybersecurity, intrusion detection, privacy protection, numerous security concerns, and software attack detection. We analyze the architecture of all the works examined, emphasizing the DL approach based on FL employed, its implementation, and the data sets utilized for testing. We compared the performance of the various surveys wherever feasible. This research is anticipated to assist academics in the field of cybersecurity based on FL in understanding the development and present research status of FL, as well as give strong support for FL's future growth. Future FL research will continue to concentrate on privacy and security protection mechanisms, client cooperative training mode and fairness, robustness, personalized federated learning mechanism, and so on, to simplify the deployment and implementation of FL technology for in-depth investigation.

Conflict of Interest

The authors declare that they have no conflict of interest, and all of the authors agree to publish this paper under academic ethics.

Author Contributions

All the authors contributed equally to the manuscript.

Funding

The work was not supported by any official Institute or company, it was completed by depended only on our efforts.

References

[1] S. A. Jebur, K. A. Hussein, H. K. Hoomod, and L. Alzubaidi, “Novel deep feature fusion framework for multi-scenario violence detection,” *Computers*, vol. 12, no. 9, 2023.

[2] V. Rey, P. M. Sánchez, A. H. Celdrán, and G. Bovet, “Federated learning for malware detec-tion in iot devices,” *Computer Networks*, vol. 204, 2022.

[3] L. Lavaur, M.-O. Pahl, Y. Busnel, and F. Autrel, “The evolution of federated learning-based intrusion detection and mitigation: A survey,” *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 3, pp. 2309–2332, 2022.

[4] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, “Federated learning for the internet of things: Applications, challenges, and opportunities,” *IEEE Internet Things M.*, vol. 5, no. 1, pp. 24–29, 2022.

[5] R. Shao, H. He, H. Liu, and D. Liu, *Stochastic channel-based federated learning for medi-cal data privacy preserving*. Arxiv, 2019.

[6] R. Kozik, M. Choraś, M. Ficco, and F. Palmieri, “A scalable distributed machine learning approach for attack detection in edge computing environments,” *J. Parallel Distrib. Comput.*, vol. 119, pp. 18–26, 2018.

[7] P. Zhang, M. Zhou, and G. Fortino, “Security and trust issues in Fog computing: A survey,” *Future Gener. Comput. Syst.*, vol. 88, pp. 16–27, 2018.

[8] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *International journal of advanced research in computer and communication engineering*, vol. 4, pp. 446–452, 2015.

[9] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, *Towards effective feature selec-tion in machine learning-based botnet detection approaches*. IEEE, 2014.

[10] N. Moustafa, J. Slay, and G. Creech, "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks," *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 481–494, 2019.

[11] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems," *International Journal of Engineering & Technology*, vol. 7, pp. 479–482, 2018.

[12] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, 2019.

[13] Y. Meidan *et al.*, "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, 2018.

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

[15] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous ac-tivity detection in iot networks," in *Canadian Conference on Artificial Intelligence*, Cham: Springer, 2020, pp. 508–520.

[16] V. Dutta, M. Choraś, M. Pawlicki, and R. Kozik, "A deep learning ensemble for network anomaly and cyber-attack detection," *Sensors (Basel)*, vol. 20, no. 16, p. 4583, 2020.

[17] A. Guerra Manzanares, J. Medina-Galindo, H. Bahsi, and N. S. Medbiot, "Generation of an IoT botnet dataset in a medium-sized IoT network," *InICISSP*, pp. 207–218, 2020.

[18] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset)," in *Selected Papers from the 12th International Networking Conference*, Cham: Springer International Publishing, 2021, pp. 73–84.

[19] L. Chen, Y. Li, X. Deng, Z. Liu, M. Lv, and H. Zhang, "Dual auto-encoder GAN-based anomaly detection for industrial control system," *Appl. Sci. (Basel)*, vol. 12, no. 10, p. 4986, 2022.

[20] N. Moustafa, "A new distributed architecture for evaluating AI-based security sys-tems at the edge: Network TON_IoT datasets," *Sustainable Cities and Society*, vol. 72, 2021.

[21] I. Tareq, B. M. Elbagoury, S. El-Regaily, and E.-S. M. El-Horbaty, "Analysis of ToN-IoT, UNW-NB15, and Edge-IIoT datasets using DL in cybersecurity for IoT," *Appl. Sci. (Basel)*, vol. 12, no. 19, p. 9572, 2022.

[22] "WUSTL-IIOT-2021 Dataset for IIoT Cybersecurity Research," *Wustl.edu*. [Online]. Available: http://www.cse.wustl.edu/~jain/iiot2/index.html. [Accessed: 20-Mar-2024].

[23] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138509–138542, 2021.

[24] S. M. Hassan and J. Wasim, "Study Of Artificial Intelligence In Cyber Security And The Emerging Threat Of Ai-Driven Cyber Attacks And Challenges," *Journal of Aeronau-tical Materials*, vol. 43, pp. 1557–1570, 2023.

[25] M. Taddeo, T. McCutcheon, and L. Floridi, "Trusting artificial intelligence in cybersecurity is a double-edged sword," in *Philosophical Studies Series*, Cham: Springer International Publishing, 2021, pp. 289–

297.

[26] R. Prasad and V. Rohokale, *Cyber security: the lifeline of information and communica-tion technology*. Cham, Switzerland: Springer International Publishing, 2020.

[27] T. R. Reshmi, "Information security breaches due to ransomware attacks-a systematic literature review," *International Journal of Information Management Data Insights*, vol. 1, no. 2, 2021.

[28] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, p. 53, 2021.

[29] Z. Diame, M. ElBery, M. Salem, and M. Roushdy, "Experimental comparative study on autoencoder performance for aided melanoma skin disease recognition," *Int. J. Intell. Comput. Inf. Sci.*, vol. 22, no. 1, pp. 88–97, 2022.

[30] S. M. Elgayar, S. Hamad, and E. S. El-Horbaty, "Revolutionizing Medical Imaging through Deep Learning Techniques: An Overview," *International Journal of Intelligent Com-puting and Information Sciences*, vol. 23, no. 3, pp. 59–72, 2023.

[31] M. Abdelazim, W. Hussein, and N. Badr, "Automatic Dialect identification of Spoken Ara-bic Speech using Deep Neural Networks," *International Journal of Intelligent Compu-ting and Information Sciences*, vol. 22, no. 4, pp. 25–34, 2009.

[32] M. A. Mead, "HCLM) for Short-Term Traffic Volume Prediction," *International Journal of Intelligent Computing and Information Sciences*, vol. 22, no. 4, pp. 51–61, 2022.

[33] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Restricted Boltzmann Machine and Deep Belief Network: Tutorial and survey," *arXiv [cs.LG]*, 2021.

[34] S. Cheon, J. Kim, and J. Lim, "The use of deep learning to predict stroke patient mortality," *Int. J. Environ. Res. Public Health*, vol. 16, no. 11, p. 1876, 2019.

[35] S. Dupond, "A thorough review on the current advance of neural network structures," *Annual Reviews in Control*, vol. 14, no. 14, pp. 200–230, 2019.

[36] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," *arXiv [cs.LG]*, 2016.

[37] S. A. Jebur, K. A. Hussein, H. K. Hoomod, L. Alzubaidi, and J. Santamaría, "Review on deep learning approaches for anomaly event detection in video surveillance," *Electronics*, vol. 12, 2022.

[38] D. Pratella, S. Ait-El-Mkadem Saadi, S. Bannwarth, V. Paquis-Fluckinger, and S. Bottini, "A survey of autoencoder algorithms to pave the diagnosis of rare diseases," *Int. J. Mol. Sci.*, vol. 22, no. 19, p. 10891, 2021.

[39] B. Mcmahan, M. E. Ramage, D. Hampson, and S. Arcas, *Communication-efficient learning of deep networks from decentralized data. InArtificial intelligence and statistics*. PMLR, 2017.

[40] M. Ammad-Ud-Din *et al., Federated collaborative filtering for privacy-preserving personalized recommen-dation system*. 2019.

[41] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "MLbase: A Distributed Machine-learning System," *InCidr*, vol. 1, pp. 2–3, 2013.

[42] C. Huang, J. Huang, and X. Liu, "Cross-silo federated learning: Challenges and opportunities," *arXiv [cs.LG]*, 2022.

[43] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," *arXiv [cs.AI]*, 2019.

[44] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv [cs.LG]*, 2017.

[45] S. Saha and T. Ahmad, "Federated transfer learning: Concept and applications," *Intell. Artif.*, vol. 15,

no. 1, pp. 35–44, 2021.

[46] P. Singh, M. K. Singh, R. Singh, and N. Singh, "Federated learning: Challenges, methods, and future directions," in *Federated Learning for IoT Applications*, Cham: Springer International Publishing, 2022, pp. 199–214.

[47] I. A. Aljazaery, S. Alrikabi, and H. T. Alaidi, "Encryption of Color Image Based on DNA Strand and Exponential Factor," *International Journal of Online & Biomedical Engineering*, vol. 18, no. 3, 2022.

[48] I. Kholod *et al.*, "Open-source federated learning frameworks for IoT: A comparative review and analysis," *Sensors (Basel)*, vol. 21, no. 1, p. 167, 2020.

[49] S. A. Jebur, A. K. Nawar, L. E. Kadhim, and M. M. Jahefer, "Hiding Information in Digital Im-ages Using LSB Steganography Technique," *International Journal of Interactive Mo-bile Technologies*, vol. 17, 2023.

[50] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on ena-bling technologies, protocols, and applications," *IEEE Access*, vol. 2020, pp. 140699–140725.

[51] "TensorFlow federated," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/federated. [Accessed: 20-Mar-2024].

[52] *FATE: An Industrial Grade Federated Learning Framework*. .

[53] "Welcome to the open federated learning (OpenFL) documentation! — OpenFL 2024.2 documentation," *Readthedocs.io*. [Online]. Available: https://openfl.readthedocs.io/en/latest/index.html. [Accessed: 20-Mar-2024].

[54] *Mybluemix.net*. [Online]. Available: https://ibmfl.mybluemix.net. [Accessed: 20-Mar-2024].

[55] G. Ulm, E. Gustavsson, and M. Jirstrand, *Functional federated learning in erlang (ffl-erl). InFunctional and Constraint Logic Programming: 26th International Workshop*. Frankfurt/Main, Germany: Springer International Publishing, 2018.

[56] *Nimbleedge.ai*. [Online]. Available: https://blog.nimbleedge.ai/types-of-federated-learning/. [Accessed: 20-Mar-2024].

[57] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms. InProceedings of the second workshop on distribut-ed infrastructures for deep learning," pp. 1–8, 2018.

[58] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimiza-tion in heterogeneous networks. Proceedings of Machine learning and systems," vol. 2, pp. 429–450, 2020.

[59] Z. Wang, W. Zhang, X. Wu, and X. Wang, "Matched averaging federated learning gesture recognition with WiFi signals," in *2021 7th International Conference on Big Data Computing and Communications (BigCom)*, 2021.

[60] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "BrainTorrent: A peer-to-peer environment for decentralized federated learning," *arXiv [cs.LG]*, 2019.

[61] L. R. Ali, S. A. Jebur, M. M. Jahefer, and B. N. Shaker, "Employing Transfer Learning for Diag-nosing COVID-19 Disease," *International Journal of Online & Biomedical Engineer-ing*, vol. 18, no. 15, 2022.

[62] R. Singh, N. Singh, and A. G. Dinker, "Performance analysis of TCP variants using AODV and DSDV routing protocols in MANETs," *Recent Advances in Computer Science and Communications*, vol. 14, no. 2, pp. 448–455, 2021.

[63] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019.

[64] P. Singh and R. Agrawal, "A game-theoretic approach to maximise payoff and customer retention for differentiated services in a heterogeneous network environment," *Int. J. Wirel. Mob. Comput.*, vol. 16,

no. 2, p. 146, 2019.

[65] S. Liu, G. Yu, X. Chen, and M. Bennis, "Joint user association and resource allocation for wireless hierarchical federated learning with IID and non-IID data," *IEEE Trans. Wirel. Commun.*, vol. 21, no. 10, pp. 7852–7866, 2022.

[66] A. Aly, M. Fayez, M. M. Al-Qutt, and A. Hamad, "Navigating the Deception Stack: In-Depth Analysis and Application of Comprehensive Cyber Defense Solutions," *International Journal of Intelligent Computing and Information Sciences*, vol. 23, no. 4, pp. 50–65, 2023.

[67] A. M. Asad, T. Moustafa, and M. Ito, *Evaluating the communication effciency in federated learning algorithms*. 2020.

[68] M. K. Abdul-Hussein and H. T. Alrikabi, "Secured Transfer and Storage Image Data for Cloud Communications," *International Journal of Online & Biomedical Engineering*, vol. 19, no. 6, 2023.

[69] S. A. Jebur, K. A. Hussein, and H. K. Hoomod, "Abnormal Behavior Detection in Video Sur-veillance Using Inception-v3 Transfer Learning Approaches," *COMMUNICATIONS, CONTROL AND SYSTEMS ENGINEER-ING*, vol. 23, no. 2, pp. 210–221, 2023.

[70] H. Al-Ani, "Artificial neural network in the prediction of surface roughness: A comparative study", Sustainable Engineering and Innovation, vol. 5, no. 2, pp. 141-150, Dec. 2023.

[71] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020.