AI-BASED CYBERSECURITY

# AI-Based Cybersecurity: A Review of Machine Learning Techniques for Network Intrusion Detection, Emerging Threats, and Open Challenges

Shweta Tarade, Archana Nilesh Pawar

Department of Artificial Intelligence and Machine Learning Engineering, Navsahyadri Group of Institutions, Pune, Maharashtra, India; Savitribai Phule Pune University, Pune, Maharashtra, India

Published November 2026

**Abstract**

Growing reliance on interconnected systems has raised the demand for dependable detection of malicious network activity. Signature-based intrusion detection systems recognize known attacks well but struggle with unseen attacks and shifting traffic behavior. Machine learning (ML) and deep learning (DL) can learn traffic patterns directly from data, yet their practical value depends on dataset quality, class imbalance, feature selection, false-positive rates, computational cost, and generalization to unfamiliar networks. This paper reviews AI-based network intrusion detection, covering conventional ML algorithms, deep neural architectures, hybrid approaches, benchmark datasets, and evaluation metrics. It compares the reported strengths and limitations of each model family and examines emerging challenges, including unseen attacks, adversarial robustness, explainability, privacy-preserving learning, and deployment under resource constraints. The review protocol defines research questions, search procedures, eligibility criteria, and evidence extraction. By linking model performance to data quality and deployment requirements, it aims to help researchers judge intrusion detection approaches beyond isolated accuracy figures.

**Keywords:** adversarial robustness; anomaly detection; artificial intelligence; cybersecurity; deep learning; machine learning; network intrusion detection

## 1 Introduction

### 1.1 Background

The spread of internet-connected devices, cloud platforms, and online services has made network protection a pressing concern. Networked systems hold sensitive information and support critical operations, yet they remain exposed to denial-of-service attacks, malware communication, unauthorized access, reconnaissance, and data exfiltration. Security mechanisms must therefore flag suspicious activity without disrupting legitimate traffic.

An Intrusion Detection System (IDS) monitors system events or network traffic for signs of a security violation. Denning's model of detecting abnormal patterns in audit records laid the groundwork for behavior-based detection [1], and modern network IDSs apply the same idea to packets, flows, and connection statistics. Signature-based systems match activity against known attack patterns and depend on current rules, whereas anomaly-based systems flag deviations from expected behavior but may mistake legitimate changes in usage for attacks [6].

ML learns patterns from data to separate benign from malicious activity. Supervised methods such as Decision Trees, Random Forest, Support Vector Machines (SVM), and gradient boosting rely on labeled examples, while unsupervised and semi-supervised methods help when labeled attacks are scarce. DL extends this with multilayer networks that learn richer representations [16]. High experimental accuracy, however, does not guarantee operational value: Sommer and Paxson identified the cost of errors, scarce training data, hard-to-interpret alerts, and unrealistic evaluation as persistent obstacles [2]. Earlier surveys have examined supervised ML [7], ML and DL approaches [4], [8], and deep learning specifically [5].

### 1.2 Problem statement, aim, and research questions

Three problems motivate this review. First, model performance depends on the quality, representativeness, and labeling of training data, so a model validated on one benchmark may not hold when traffic or attack distributions change. Second, attack classes are heavily imbalanced, and high overall accuracy can hide missed rare attacks; class-wise precision, recall, F1-score, and false-positive rate are needed. Third, adversaries adapt, traffic drifts, raw traffic is hard to share for privacy reasons, resources are limited, and analysts need explanations for alerts.

The aim is to examine ML techniques for network intrusion detection and assess their capabilities, limitations, and suitability for emerging requirements. The review classifies ML and DL approaches, examines datasets and metrics, compares conventional, deep, and hybrid methods, and identifies research gaps. It is guided by five questions:

**RQ1:** Which ML and DL techniques are used for network intrusion detection?

**RQ2:** Which datasets, features, and metrics are most common?

**RQ3:** What strengths and limitations are reported for each model family?

**RQ4:** How do data quality, class imbalance, and changing traffic affect reliability and generalization?

**RQ5:** What challenges arise from unseen attacks, evasion, explainability, privacy, and real-time deployment?

The following sections cover background concepts and the review methodology, examine ML and DL techniques, discuss datasets and metrics, and give a comparative synthesis. Emerging threats and open challenges are then addressed, research gaps and future directions outlined, and the paper concludes.

## 2 Background and related concepts

### 2.1 Network intrusion detection systems

A Network Intrusion Detection System (NIDS) analyzes traffic at a monitoring point to detect attacks, unauthorized access, or policy violations. An ML-based NIDS typically moves through five stages: traffic collection (packets, flows, or logs); preprocessing and feature extraction (handling missing values, encoding categorical attributes, normalizing, and selecting features such as duration, protocol, and packet and byte counts); training and validation; detection, in which the model classifies records or scores anomalies; and alerting for analyst follow-up. An IDS detects and reports, whereas an Intrusion Prevention System (IPS) can also block traffic or end connections [6]. Host-based systems watch logs, files, and processes on one machine, while network-based systems observe traffic but lose

visibility when payloads are encrypted. These categories complement one another, and practical deployments often combine signature, statistical, and ML-based detection [1].

### 2.2 Machine learning paradigms

Supervised learning (Decision Trees, Random Forest, SVM, k-Nearest Neighbors, gradient boosting) trains on labeled benign and attack records for binary or multiclass classification, but may miss attack types absent from training data. Unsupervised methods such as k-means, Isolation Forest, and autoencoders model normal behavior without labels; unusual traffic is not necessarily malicious, so false positives must be managed. Semi-supervised learning pairs a few labeled records with many unlabeled ones when labeling is costly. Reinforcement learning suits adaptive response selection better than direct classification. Hybrid designs may chain paradigms, for instance anomaly screening followed by supervised classification [2].

### 2.3 Evaluation metrics

Evaluation must avoid data leakage: normalization parameters and feature selection should be fitted on training data only, and time-based splits or cross-dataset tests give a more realistic picture than random splits [2]. Accuracy alone misleads on imbalanced data, so results are also reported through the following metrics, where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

$$\text{FPR} = \frac{FP}{FP + TN} \tag{5}$$

Low precision means analysts face many false alarms, and low recall means attacks go unnoticed. A confusion matrix, extended to one row per class in multiclass settings, shows which attack types are most often mistaken for benign traffic or for each other. For multiclass problems the averaging method (macro, micro, or weighted) should be stated. Inference latency, throughput, and memory use also matter, because a gateway-grade detector has different requirements from an offline forensic tool.

## 3 Systematic literature review methodology

This review examines ML and DL techniques for network intrusion detection through a structured assessment of publicly accessible scholarly literature. PRISMA 2020 [3] serves as the reporting framework for documenting sources, eligibility criteria, selection, and synthesis. Primary experimental studies provide the main evidence for comparing methods, whereas existing surveys [4], [5], [7], [8] supply context and related work.

An initial search of public scholarly sources was run on 27 October 2026, targeting publisher-hosted records on ML-based intrusion detection, benchmark datasets, class imbalance, out-of-distribution

detection, uncertainty quantification, and adversarial robustness. Search terms combined concepts such as “machine learning,” “deep learning,” “network intrusion detection,” “cross-dataset evaluation,” “zero-day attack,” and “adversarial robustness.” This initial search does not show that every planned database was searched exhaustively; exact search strings, dates, filters, and screening decisions must be documented for each source in the completed review.

Publications are eligible when they apply ML, DL, hybrid, or anomaly-detection techniques to network intrusion detection and provide methodological or experimental evidence. Unrelated work, duplicates, editorials, and secondary reviews without original experiments are excluded from the primary synthesis, although reviews are kept for background. For each eligible study, the extraction framework records the year, model family, dataset, preprocessing, validation design, metrics, findings, and limitations, together with evidence on cross-dataset generalization, unknown-attack detection, robustness, interpretability, efficiency, and deployment readiness. Results are synthesized thematically rather than ranked by accuracy, and are reported descriptively where experimental settings are not comparable.

Study quality is assessed with a predefined eight-criterion checklist covering task definition, dataset description, preprocessing transparency, validation design, metric selection, baseline comparison, limitations, and reproducibility, each scored 1, 0.5, or 0. The score qualifies the strength of the synthesis; it does not imply that a high-scoring study is free of bias. A PRISMA flow diagram will be reported only when record counts and exclusion reasons are supported by the actual screening log.

## 4 Machine learning techniques for network intrusion detection

ML models learn relationships between traffic features and attack categories, and can support binary classification, multiclass attack recognition, or anomaly detection. Their performance depends on dataset quality, feature selection, class distribution, and evaluation design.

**Tree-based models:** A Decision Tree classifies records through a sequence of feature-based splits chosen by criteria such as information gain or Gini impurity. Its rules are easy to inspect, but an unconstrained tree overfits and is unstable; pruning and depth limits help. Random Forest averages many trees trained on random samples and feature subsets, which reduces instability and yields feature-importance scores, although correlated features can distort those scores and memory and inference costs grow with ensemble size.

**Margin- and distance-based models:** An SVM seeks a large-margin boundary and uses kernels for nonlinear separation, but it is sensitive to scaling, kernel choice, and regularization, and becomes costly on large datasets. k-Nearest Neighbors labels a record by its closest training examples; it is a simple baseline, yet it must store the training data, predicts slowly as data grow, and suffers from unscaled or high-dimensional features.

**Probabilistic and linear models:** Naive Bayes is computationally cheap but assumes conditionally independent features, which rarely holds for correlated flow statistics. Logistic Regression offers interpretable coefficients and easy regularization, but its basic form draws a linear boundary and misses nonlinear interactions.

**Boosting and ensembles:** Gradient boosting, including XGBoost, trains learners successively and performs strongly on tabular flow features, but needs careful tuning and is harder to interpret. Bagging, voting, and stacking combine models; the gain is not guaranteed when members share the same weaknesses or biased data, and cost rises accordingly.

Conventional models remain valuable baselines, yet they share recurring limits: training data may not represent real traffic, rare attacks are under-detected, feature engineering and leakage can distort results, and success on known classes does not show that novel attacks will be recognized [2], [8].

## 5 Deep learning techniques for network intrusion detection

DL uses multilayer neural networks to learn representations from data [16]. It can reduce manual feature design, although many NIDS implementations still depend on engineered flow attributes [5].

**CNNs:** Convolutional networks learn local patterns from structured representations of traffic. Reshaping tabular features into an artificial grid does not create genuine spatial relationships, so the chosen representation must be justified and tested independently.

**RNN, LSTM, and GRU:** Recurrent models handle sequences in which ordering carries information. LSTM and GRU gating mitigates vanishing gradients, and GRUs use fewer parameters. Their value depends on constructing sequences that reflect real temporal structure, since arbitrarily ordered records create none, and on keeping inference latency acceptable.

**Autoencoders:** An autoencoder trained mainly on benign traffic reconstructs normal records well, so a large reconstruction error flags a possible anomaly. For one observation, the error can be taken as the mean squared difference between the n input features $x_i$ and their reconstructions $\hat{x}_i$:

$$\text{Reconstruction error} = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \hat{x}_i\right)^2 \tag{6}$$

The threshold should be set on validation data rather than test results, and a high error is not proof of malice, because legitimate behavior changes can trigger it too.

**Transformers:** Attention mechanisms model relationships among features or events and suit complex dependencies, but they demand substantial computation. Attention weights alone should not be treated as a full explanation of a prediction.

**Hybrid and other architectures:** Hybrids such as CNN-LSTM combine local feature extraction with sequence modeling, and other options include Deep Belief Networks and graph neural networks. Extra complexity does not by itself improve detection, so hybrids should be compared against single-model baselines.

Training and evaluation deserve as much care as architecture. Preprocessing parameters should be estimated on the training subset only, and class imbalance can be handled with weighting or resampling provided the effect is evaluated without leakage. Hyperparameters should be tuned on validation data so the test set stays untouched, sequential traffic benefits from temporal separation, and cross-dataset testing checks whether a model generalizes beyond its original environment.

DL models generally need large, representative, well-labeled data; they cost more to train and run, are harder to explain, and can degrade when traffic distributions shift. Strong results on known attacks do not establish detection of novel ones, and the models themselves can be manipulated. DL is therefore not universally superior to conventional ML [8].

## 6 Datasets and evaluation practices

Reported performance depends on the dataset as much as on the algorithm. Attack diversity, benign realism, label accuracy, size, and feature documentation should be examined before results are interpreted. Table 1 summarizes widely used benchmarks; it is qualitative and not a quality ranking.

**Table 1.** Qualitative comparison of intrusion detection datasets.

| Dataset | Characteristics | Main use | Limitation |
|---|---|---|---|
| KDD Cup 99 | Early benchmark built from simulated network connections. | Historical algorithm comparison and foundational experiments. | Redundant records and dated traffic [11]. |
| NSL-KDD | Revised KDD Cup 99 with reduced redundancy; common in baseline studies. | Educational studies and baseline classification. | Limited representation of modern network behavior. |
| UNSW-NB15 | Normal traffic and several attack categories with engineered features [9]. | Binary and multiclass detection evaluation. | Generated in a controlled environment. |
| CICIDS2017 | Labeled benign and attack flows from a scheduled testbed [10], [12]. | Comparing ML and DL approaches on flow-based features. | Testbed-specific patterns and data-quality issues. |
| CSE-CIC-IDS2018 | Large-scale benign and attack traffic [13]. | Evaluation under varied controlled scenarios. | High storage and compute cost; limited generalization. |

Dataset choice should follow the intended application, including the traffic source, attack categories, record counts, label distribution, and available features. Preprocessing commonly handles missing values, removes invalid or duplicate records, encodes categorical variables, and normalizes numeric ones; each step should be documented so results can be reproduced, and data should be split before any data-dependent step is fitted. Group- or time-based splits suit records that are temporally or operationally related.

KDD Cup 99 contains substantial redundancy [11], so strong results on it or NSL-KDD say little about modern networks. CICIDS2017 calls for care with missing or infinite values and duplicate records. Cross-dataset and temporal testing show whether a model learned general indicators of malice or dataset-specific artifacts; they require documented feature alignment and label mapping, and neither guarantees generalization to every environment. Because studies differ in dataset versions, features, splits, and protocols, figures from separate papers are often not directly comparable.

## 7 Comparative analysis of ML and DL approaches

Model choice involves predictive performance, computational cost, interpretability, data requirements, and generalization. Table 2 summarizes typical design properties of the main model families; the entries are not measured performance rankings.

**Table 2.** Qualitative comparison of model families.

| Model family | Interpretability | Limitation |
|---|---|---|
| Decision Tree and Random Forest | High for small trees; moderate for forests. | Overfitting and instability (single tree); memory cost (forest). |

| Model family | Interpretability | Limitation |
|---|---|---|
| Gradient boosting | Moderate to limited, depending on the ensemble. | Tuning effort and overfitting risk when validation is weak. |
| Naive Bayes and Logistic Regression | Relatively high, especially for Logistic Regression coefficients. | Independence assumption or linear boundary. |
| SVM and k-NN | Depends on kernel; neighbor-based reasoning for k-NN. | Costly on large data; sensitive to feature scaling. |
| CNN | Usually limited without explanation methods. | Depends on a meaningful input representation. |
| LSTM and GRU | Usually limited. | Needs genuine sequences; sequential processing adds latency. |
| Autoencoder | Reconstruction error is a signal, not an explanation. | Threshold selection and false-positive control. |
| Transformer | Requires analysis beyond attention weights. | High resource demand and deployment complexity. |

Three issues cut across the families. First, evaluation protocol: random splits can place near-duplicate records in both training and test sets, tuning must not touch the final test set, and each study's dataset, features, metrics, and validation strategy should be recorded, with missing details noted as uncertainty rather than assumed. Second, class imbalance: class weighting, resampling, and synthetic samples may help rare attacks but cause leakage if applied before splitting, so per-class metrics and confusion matrices are essential. Third, explainability and efficiency: small trees and Logistic Regression are relatively transparent, while complex models may need SHAP, LIME, or similar attribution methods, whose explanations are not definitive causal evidence; computational claims likewise require a stated hardware and software environment.

Efficiency shapes deployability. Conventional models often need less computation than deep architectures, but this is not a universal rule, because cost depends on implementation, model size, feature processing, and hardware. A high-throughput gateway needs low-latency inference, whereas offline analysis tolerates slower processing, and resource-constrained devices may need compression or quantization. Comparative studies should therefore state training time, inference latency, memory, and throughput together with the environment in which they were measured.

## 8 Emerging threats and open challenges

Network traffic keeps changing, attackers adapt, and detectors must process large volumes with low latency. Practical intrusion detection therefore needs reliable generalization, resilience, interpretability, and efficient deployment in addition to accuracy [2], [7].

**Zero-day and unknown attacks:** Supervised models learn boundaries from labeled examples and may label a novel attack as normal. Anomaly detection can expose deviations without labels for every attack, at the price of more false positives. Systems should combine both, route uncertain predictions for investigation, and be tested on held-out attack categories and temporally separated data.

**Adversarial evasion and data poisoning:** Attackers may alter packet timing, volume, or connection patterns so that malicious traffic resembles legitimate traffic, and small input perturbations are known to mislead neural networks [14]. Countermeasures include adversarial training, robust features, monitoring, and ensembles, though none is complete. Poisoning, in which misleading or

mislabeled samples enter training, matters most in pipelines that retrain on new traffic; source validation, label checks, versioned datasets, and audited updates are needed.

**Concept drift and class imbalance:** New applications, software updates, and evolving attacks shift traffic statistics. Drift detection, periodic retraining, and temporal evaluation help, but drift does not necessarily signal an attack. Rare attack classes call for class weighting, cost-sensitive learning, or threshold adjustment, judged through per-class precision and recall rather than accuracy.

**Encryption, explainability, and privacy:** Encryption limits payload inspection, so detectors lean on metadata such as packet sizes and timing, which may not separate benign from malicious flows. Analysts also need to know why an alert fired; explanation methods help but can be unstable or misleading. Federated learning [15] lets organizations train a shared model without pooling raw traffic, yet it adds communication overhead, data heterogeneity, and exposure to poisoning and inference attacks, so it should not be assumed private or secure.

**Real-time constraints and pipeline security:** Complex DL models can strain memory and latency budgets on gateways, edge devices, and industrial systems. Compression, pruning, quantization, and lightweight designs reduce cost but may lose sensitivity to some attacks. The detection pipeline is itself an attack surface, covering data collection, model storage, training, inference, and alerting, so access control, integrity checks, authenticated model updates, and logging are required.

## 9 Research gaps and future directions

Many studies report promising benchmark results that do not demonstrate reliable performance in changing environments, and differing datasets, preprocessing, and metrics make comparison difficult. Five priorities follow from the preceding analysis:

- **Generalization and unknown attacks:** Report within-dataset and cross-dataset results, use temporal and held-out-attack tests, and explore hybrids of supervised classification, anomaly detection, and uncertainty estimation.
- **Reproducible evaluation:** Document dataset versions, class distributions, preprocessing, hyperparameters, and hardware; prevent leakage; and report per-class metrics.
- **Robust, continual learning:** Evaluate models under explicit adversarial threat models and design controlled updates that resist poisoning and drift.
- **Explainable, privacy-aware, efficient detection:** Measure explanation stability and analyst usefulness, test federated learning against centralized baselines, and report latency, throughput, and memory.
- **End-to-end evaluation:** Assess the full data-to-alert pipeline, including alert volume, prioritization, integration with security operations, and human oversight of automated responses.

No single metric captures the suitability of a detector. Future studies should therefore evaluate detection effectiveness, generalization, robustness, efficiency, interpretability, privacy and security, and reproducibility together rather than optimizing one measure.

## 10 Conclusion

This review examined conventional ML algorithms, DL architectures, benchmark datasets, evaluation metrics, emerging threats, and open challenges in AI-based network intrusion detection. Decision Trees, Random Forest, SVM, and gradient boosting remain valuable for their manageable cost and

strength on tabular flow features, while CNNs, recurrent networks, autoencoders, and transformers can learn richer representations. No algorithm is universally superior; effectiveness depends on data quality, feature representation, configuration, and evaluation protocol.

Dataset choice and evaluation design matter equally. High accuracy on a single benchmark does not show that a model will generalize to an unfamiliar network or detect unseen attacks, so class-wise metrics, temporal testing, cross-dataset validation, and transparent reporting are needed. Zero-day detection, adversarial evasion, poisoning, drift, imbalance, encrypted traffic, interpretability, privacy, and efficiency remain open, and they call for assessing the complete detection pipeline rather than the classifier alone.

Future work should favor generalizable, explainable, privacy-aware, and efficient architectures, such as hybrids of supervised and anomaly-based detection, with model updates protected against manipulation and with standardized, reproducible protocols. Addressing these gaps can bring intrusion detection systems closer to dependable operational use.

## References


[1] Denning DE. An intrusion-detection model. *IEEE Transactions on Software Engineering*. 1987;SE-13(2):222–232. https://doi.org/10.1109/TSE.1987.232894

[2] Sommer R, Paxson V. Outside the closed world: on using machine learning for network intrusion detection. In: *Proceedings of the 2010 IEEE Symposium on Security and Privacy*. 2010:305–316. https://doi.org/10.1109/SP.2010.25

[3] Page MJ, et al. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *BMJ*. 2021;372:n71. https://doi.org/10.1136/bmj.n71

[4] Rehman HMRU, Liaquat S, Gul MJ, Jhandir MZ, Gavilanes D, Masias Vergara M, et al. A systematic literature study of machine learning techniques based intrusion detection: datasets, models, challenges, and future directions. *Journal of Big Data*. 2025;12:264. https://doi.org/10.1186/s40537-025-01323-2

[5] Anis FM, Alabdullatif M, Aljbli S, Hammoudeh M. A survey on the applications of deep learning in network intrusion detection systems to enhance network security. *IEEE Access*. 2025;13:185357–185373. https://doi.org/10.1109/ACCESS.2025.3624952

[6] Scarfone K, Mell P. *Guide to intrusion detection and prevention systems (IDPS)*. NIST Special Publication 800-94. Gaithersburg, MD: National Institute of Standards and Technology; 2007. https://doi.org/10.6028/NIST.SP.800-94

[7] Intrusion detection systems using supervised machine learning techniques: a survey. Procedia Computer Science. 2022;201:205–212. https://doi.org/10.1016/j.procs.2022.03.029

[8] Ahmad Z, Khan AS, Shiang CW, Abdullah J, Ahmad F. Network intrusion detection system: a systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*. 2021;32(1):e4150. https://doi.org/10.1002/ett.4150

[9] Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*. 2015:1–6. https://doi.org/10.1109/MilCIS.2015.7348942

[10] Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*. 2018:108–116. https://doi.org/10.5220/0006639801080116

[11] Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In: *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. 2009:1–6. https://doi.org/10.1109/CISDA.2009.5356528

[12] Canadian Institute for Cybersecurity. CICIDS2017: intrusion detection evaluation dataset [Internet]. University of New Brunswick. Available from: https://www.unb.ca/cic/datasets/ids-2017.html

[13] Canadian Institute for Cybersecurity. CSE-CIC-IDS2018 on AWS [Internet]. University of New Brunswick. Available from: https://www.unb.ca/cic/datasets/ids-2018.html

[14] Goodfellow I, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In: Proceedings of the International Conference on Learning Representations (ICLR); 2015. Available from: https://arxiv.org/abs/1412.6572

[15] McMahan HB, et al. Communication-efficient learning of deep networks from decentralized data. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*; 2017. PMLR 54:1273–1282. Available from: https://proceedings.mlr.press/v54/mcmahan17a.html

[16] LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521:436–444. https://doi.org/10.1038/nature14539