Archives of Computational Methods in Engineering (2024) 31:4209–4233
https://doi.org/10.1007/s11831-024-10110-w

REVIEW ARTICLE

Machine Learning Optimization Techniques: A Survey, Classification, Challenges, and Future Research Issues

Kewei Bian[1] · Rahul Priyadarshi[2]

Received: 25 December 2023 / Accepted: 18 March 2024 / Published online: 29 March 2024

Abstract
Optimization approaches in machine learning (ML) are essential for training models to obtain high performance across numerous domains. The article provides a comprehensive overview of ML optimization strategies, emphasizing their classification, obstacles, and potential areas for further study. We proceed with studying the historical progression of optimization methods, emphasizing significant developments and their influence on contemporary algorithms. We analyse the present research to identify widespread optimization algorithms and their uses in supervised learning, unsupervised learning, and reinforcement learning. Various common optimization constraints, including non-convexity, scalability issues, convergence problems, and concerns about robustness and generalization, are also explored. We suggest future research should focus on scalability problems, innovative optimization techniques, domain knowledge integration, and improving interpretability. The present study aims to provide an in-depth review of ML optimization by combining insights from historical advancements, literature evaluations, and current issues to guide future research efforts.

1 Introduction

In recent years, the wide range of data in various industries has brought about both benefits and difficulties. Modern data generation exceeds conventional techniques of analysis in terms of volume, velocity, and diversity, resulting in it becoming increasingly difficult for individuals to extract useful information on their individual. An effective remedy for this issue is machine learning (ML), a branch of artificial intelligence that provides automated techniques for getting beneficial knowledge from enormous quantities of data [1].

✉ Rahul Priyadarshi
rahul.glorious91@gmail.com

Kewei Bian
rebekahbian1@163.com

1 College of Department of Linguistics and Translation, City University of Hong Kong, 83 Tat Chee Ave, Kowloon Tong, Hong Kong 999077, China

2 Faculty of Engineering & Technology, ITER, Siksha 'O' Anusandhan University, Bhubaneswar 751030, India

ML algorithms have shown significant potential in various fields such as healthcare, finance, e-commerce, transportation, and autonomous systems. ML models in healthcare can accurately evaluate medical images to identify diseases such as cancer, aid in diagnosing illnesses, forecast patient outcomes, and suggest individualized treatment strategies. ML algorithms are used in finance for fraud detection, algorithmic trading, risk assessment, and client segmentation to enhance efficiency and profitability [2–3]. ML in autonomous systems, such self-driving vehicles and drones, allows for immediate decision-making using sensor data, resulting in safer and more dependable performance.

The optimization procedure plays an important role for the success of ML models. Optimization is the process of repeatedly modifying the parameters of a model to reduce or increase a preset objective function, such a loss function in supervised learning [4]. The objective of optimization is to identify the parameters that most accurately match the provided data and perform well on novel information, thereby enhancing predicted accuracy and performance.

Optimization approaches play an important role in training ML models for obtaining high accuracy, efficiency, and generalization abilities. Optimization algorithms allows models to learn from data, adapt to complicated patterns, and generate reliable forecasts by modifying model parameters. The selection of an optimization method can greatly impact the efficiency and convergence of a ML model, making it an essential decision throughout the model development process [5].

Gradient descent is a basic optimization procedure used in many ML methods, such as neural networks. Gradient descent variations including Stochastic Gradient Descent (SGD), mini-batch gradient descent, and Adam optimizer are effective at optimizing large-scale models with millions of parameters. Bayesian optimization is a widely used approach that probabilistically searches the parameter space to identify the best configuration for hyperparameters, hence improving model performance and generalization [6–7].

Regularization methods like L1 and L2 have become essential for reducing overfitting and enhancing the generalization abilities of ML models. Regularization, achieved by including a penalty term into the loss function, supports the acquisition of more straightforward representations by the model [8]. This serves to minimize the probability of recollecting irrelevant details in the training data and enhances the model's performance on novel, unknown data.

Hyperparameter optimization techniques like grid search, random search, and Bayesian optimization assist in determining the most suitable hyperparameters, improving ML model performance together with optimization algorithms. Hyperparameters like learning rate, batch size, and regularization strength have a substantial influence on the behaviour and effectiveness of models, necessitating meticulous adjustment for best results [9].

Optimization methods are essential in ML since they allow models to learn from data, adjust to intricate patterns, and provide precise predictions in different fields. The increasing amount and complexity of data necessitate the creation of efficient and scalable optimization algorithms, which are crucial for advancing ML and exploring new potential for innovation and discovery [10].

An in-depth explanation of optimization approaches used to a wide range of ML issues across several domains is outlined in Table 1. Each case study provides a detailed examination of a genuine problem, including the dataset, optimization method, and achieved performance metrics. The table presents valuable insights on the effectiveness and flexibility of optimization approaches in addressing intricate issues with a focus on real-world applications in many fields such as healthcare, finance, transportation, and environmental research. By using these case studies, experts and researchers may understand how optimization approaches can be tailored to particular domains, allowing for the creation of educated problem-solving strategies and decision-making processes in real-world scenarios.

1.1 Purpose and Structure of the Paper

This study aims to provide an extensive overview of ML optimization approaches, categorize them according to their goals and features, recognize typical obstacles, and suggest potential research paths in the domain. The presentation

Table 1 Case studies of optimization in real-world applications

Application Domain	Problem Description	Dataset	Optimization Technique Used	Performance Metrics Achieved
Healthcare	Predicting Disease Risk	Electronic Health Records (EHR)	Random Forest with Bayesian Optimization for Hyperparameter Tuning	AUC, Accuracy, Sensitivity, Specificity
Finance	Portfolio Optimization	Historical Stock Market Data	Markowitz Portfolio Optimization	Portfolio Return, Risk (Standard Deviation), Sharpe Ratio
Autonomous Systems	Path Planning for Autonomous Vehicles	Lidar and Camera Sensor Data	Genetic Algorithms for Path Planning	Path Completion Time, Collision Avoidance Rate
E-commerce	Recommender Systems	User Purchase History	Collaborative Filtering with Matrix Factorization	Recommendation Accuracy, Precision, Recall
Manufacturing	Production Scheduling	Factory Production Data	Genetic Algorithms for Scheduling Optimization	Production Efficiency, Time-to-Market
Energy	Power Grid Optimization	Smart Meter Data	Particle Swarm Optimization for Load Balancing	Energy Cost Reduction, Grid Stability
Agriculture	Crop Yield Prediction	Soil and Weather Data	Support Vector Machines with Grid Search Optimization	Crop Yield Prediction Accuracy, Crop Health Monitoring
Transportation	Vehicle Routing Optimization	Delivery Orders and Traffic Data	Simulated Annealing for Routing Optimization	Delivery Time, Fuel Consumption
Retail	Inventory Management	Sales Data and Supply Chain Information	Dynamic Programming for Inventory Optimization	Stockout Rate, Holding Costs
Telecommunications	Network Optimization	Call Records and Network Topology	Tabu Search for Network Routing Optimization	Network Latency, Call Dropping Rate
Environmental Science	Pollution Control	Air Quality Monitoring Data	Genetic Algorithms for Emission Reduction Strategies	Pollution Levels, Public Health Impact

will start by examining current literature on optimization methods in ML, emphasizing important advancements and patterns. It will then provide a structured categorization of optimization methods, including different algorithms, approaches, and methodologies. The paper will explore the problems of ML optimization, including complications in the optimization landscape and scalability concerns. The paper will explore prospective directions for future study, such as creating new optimization algorithms, tackling scaling issues, and improving the interpretability of optimization outcomes. The study seeks to provide significant insights into the current state of ML optimization and direct future research in this quickly changing exposed using a systematic method.

2 Background and Literature Review

The recent development of optimization strategies in ML has originated in the early stages of artificial intelligence research. The approaches have advanced considerably over time in response to the challenges of learning from data and developing intelligent systems. Here is an in-depth elaboration of the historical milestones mentioned:

Frank Rosenblatt's perceptron algorithm belongs to the first examples of optimization methods in the field of ML. The perceptron, introduced in the late 1950s, established the groundwork for learning systems based on neural networks. The system used a basic version of gradient descent to modify the weights of connections between neurons in order to reduce classification mistakes [11].

Paul Werbos introduced backpropagation in the 1970s, which was a major improvement in optimization methods for neural networks. The approach facilitated the effective training of multi-layer neural networks by backpropagating mistakes and changing weights by gradient descent. Backpropagation transformed neural network research and established the scene for the revival of deep learning several decades later [12].

Nouretdinov, Ilia, et al. introduced support vector machines (SVMs) in the 1990s, which marked a significant change in ML optimization. SVMs have shown the efficacy of convex optimization methods in classification applications by offering a strong structure for optimizing the margin between classes. SVMs emerged popular due to their capacity to manage high-dimensional data and generalize well to novel instances [13].

Convex optimization approaches became popular in ML in the early 2000s because of its theoretical assurances and computational efficiency. Stephen Boyd and Lieven Vandenberghe provided substantial contributions to the advancement of convex optimization techniques and their utilization in ML [14]. Convex optimization has established a strong mathematical basis for several ML models such as logistic regression, linear SVMs, and robust PCA.

The 2015s observed a renewed interest in neural networks and the emergence of deep learning. Advancements in optimization algorithms like SGD, Adam, and RMSprop were essential in facilitating the training of deep neural networks with multiple levels. Geoffrey Hinton, Yoshua Bengio, and Yann LeCun were important individuals in showcasing the effectiveness of deep learning in computer vision, natural language processing, and voice recognition across several fields [15].

In recent years, there has been a trend towards investigating new optimization strategies that expand beyond conventional gradient-based approaches. Evolutionary algorithms, metaheuristic optimization, and Bayesian optimization are effective methods for solving non-convex optimization problems and navigating intricate search spaces. Furthermore, progress in technology, such as the development of specialized accelerators like GPUs and TPUs, has hastened the training of extensive ML models.

The present state of research on optimization strategies in ML includes a diverse collection of studies that cover algorithmic advancements, theoretical understandings, and real-world implementations. Many works by well-known authors have greatly progressed the area, illuminating different elements of optimization. Here is an in-depth analysis of the literature and the significant contributions made by prominent authors.

Authors [16] are prominent characters who have made significant advancements in optimization approaches within the area of ML. They have developed important algorithms such as SGD and Adam, which are now essential for training neural networks. Authors study established the foundation for comprehending the significance of optimization in deep learning and its influence on model effectiveness.

Authors [17] work has significantly advanced our knowledge of optimization issues in training deep neural networks. His research has provided insights into problems like disappearing gradients and inflating gradients, which provide major obstacles in improving deep neural networks. His research has led to the development of methods such as batch normalization and skip connections that tackle these challenges and enhance the robustness of training.

Recent research has delved into the convergence of optimization with other disciplines including reinforcement learning, meta-learning, and federated learning. Researchers such as [18] have studied optimization methods for reinforcement learning algorithms to enhance sampling efficiency and convergence characteristics. Researchers like as [19] have investigated optimization techniques for meta-learning, which entails acquiring the ability to learn across several tasks and domains. Federated learning, developed by researchers such as [20], aims to enhance models collectively across decentralized devices while safeguarding data privacy, leading to distinctive optimization difficulties and possibilities.

The literature on optimization strategies includes both theoretical studies and actual implementations in many fields [21]. have advanced the theoretical underpinnings of convex optimization by establishing precise mathematical structures to elucidate optimization techniques and their characteristics. Researchers such as [22] have utilized optimization techniques in computer vision, natural language processing, and robotics to effectively solve complex tasks, showcasing the efficacy of optimization-driven approaches.

It is crucial to identify common issues and trends in optimization strategies in ML to comprehend the existing situation and direct future research efforts. Here is a detailed examination of common obstacles and upcoming patterns:

One of the main obstacles in optimizing ML is handling non-convex goal functions. Several practical optimization problems have non-convex surfaces, which may result in suboptimal solutions and convergence challenges. Researchers like [23] have tackled these obstacles by creating adaptive optimization algorithms such as Adam. Adam modifies learning rates and integrates momentum to better negotiate non-convex terrains. Non-convex optimization continues to be a focus of study, with efforts directed towards developing more resilient algorithms that can effectively deal with intricate optimization surfaces.

Optimization algorithms have a significant problem in terms of scalability, especially in the presence of massive data and intricate model designs. As datasets expand and models get more intricate, conventional optimization methods may face challenges in meeting computing requirements [24]. have investigated distributed training and model parallelism strategies to address scalability issues efficiently. These methods include spreading out computations over numerous devices or machines to facilitate effective training on large datasets and intricate models. Scalability is still a challenge, particularly with the emergence of new paradigms like federated learning and edge computing.

Efforts to develop more effective and resilient optimization algorithms are a prominent focus in ML research. Conventional optimization methods might experience problems including sluggish convergence, sensitivity to beginning conditions, and vulnerability to adversarial assaults. Researchers are investigating innovative methods including evolutionary algorithms, metaheuristic optimization, and quantum-inspired optimization to tackle these difficulties [25]. These methods use concepts from biology, nature, and quantum physics to create optimization algorithms that are more effective and resilient. Collaborative research combining optimization, computer science, and physics is advancing innovation in this field, resulting in the creation of advanced optimization methods that provide enhanced performance and dependability. The optimization methods utilized in ML are comprehensively summarized in Table 2. The article accentuates the vast array of research initiatives devoted to advancing the bleeding edge of optimization algorithms and overcoming common challenges. In addition to optimizing training for particular neural network and ML model types, the objectives pertain to scalability concerns, non-convex optimization challenges, and optimization challenges in deep learning, reinforcement learning, and natural language processing. In addition to investigating distributed training, model parallelism, regularization techniques, and specialized optimization strategies tailored to particular domains and model architectures, the proposed solutions include the development of adaptive optimization algorithms such as Adam and SGD.

Optimization strategies in ML have evolved throughout time due to the pursuit of effective, scalable, and reliable algorithms. Optimization has played a crucial role in shaping the discipline from early perceptron's to the deep learning revolution. Literature encompasses a wide range of works, from fundamental methods such as SGD to sophisticated implementations in reinforcement and federated learning [26]. Interdisciplinary techniques are necessary to address difficulties such as non-convexity and scalability. Collaboration will be essential for developing optimization methods and their practical applications as the subject progresses.

3 Survey of Machine Learning Optimization Techniques

ML optimization methods have significance for modifying model parameters to achieve optimal results in tasks including classification, regression, clustering, and reinforcement learning. This study intends to provide an in-depth review of the various optimization strategies used in ML, including various objectives, algorithms, and perspectives.

3.1 Classification of Optimization Techniques Based on Optimization Objectives

ML optimization strategies are often classified according to the specific objectives they aim to improve. These goals usually consist of accuracy, speed, and generalization performance, all of which are essential for the overall efficiency of ML models.

Table 2 Authors' contributions to advancing optimization in machine learning

Authors	Objective	Proposed Solution
[27]	Tackle non-convex optimization difficulties.	Create an adaptive optimization method called Adam that modifies learning rates and includes momentum.
[28]	Address issues related to scalability.	Investigate distributed training and model parallelism methods to enhance training efficiency on extensive datasets and intricate models.
[29]	Address problems such as disappearing and exploding gradients in deep neural networks.	Implement batch normalization and skip connections to enhance training stability.
[30]	Improve the theoretical comprehension of convex optimization.	Create precise mathematical foundations for convex optimization techniques.
[31]	Advanced optimization methods for deep learning.	Present SGD and illustrate its efficacy in training deep neural networks.
[31]	Enhance optimization for reinforcement learning algorithms.	Explore methods to enhance sampling efficiency and convergence qualities in reinforcement learning.
[32]	Create optimization methods for federated learning.	Create algorithms for collaborative training on decentralized devices while maintaining data confidentiality.
[33]	Investigate optimization techniques for meta-learning.	Explore methods for acquiring the ability to learn efficiently across many jobs and fields.
[34]	Explore optimization methods for training deep learning models.	Implement layer-wise pretraining and unsupervised pretraining approaches to enhance convergence and generalization.
[35]	Enhance optimization for deep neural networks.	Suggest use dropout regularization strategy to mitigate overfitting and enhance model generalization.
[36]	Tackle obstacles in maximizing the efficiency of convolutional neural networks (CNNs).	Create visualization methods, such as deconvolutional networks, to examine the activity of ML and provide guidance for improvement.
[37]	Explore the optimization difficulties encountered while training deep neural networks.	Implement strategies such as cyclical learning rates to enhance optimization convergence and resilience.
[38]	Enhance the training process of recurrent neural networks (RNNs).	Recommend using methods like as gradient trimming to address problems like expanding gradients that may occur during RNN training.
[39]	Investigate optimization difficulties in natural language processing (NLP).	Explore methods for customizing optimization tactics to suit the unique features of NLP tasks.
[40]	Enhance the training process of tree-based models.	Implement methods such as gradient boosting and early stopping to enhance optimization convergence and accuracy.
[41]	Explore optimization methods for support vector machines (SVMs).	Implement kernel approximation and online learning methods to improve the efficiency of SVM training.
[42]	Tackle obstacles in training generative adversarial networks (GANs).	Recommend using approaches such as Wasserstein GANs and spectral normalization to enhance the stability and convergence of GAN training.
[43]	Enhance the training process of extensive neural networks for audio manipulation.	Implement layer-wise optimization and model parallelism approaches to enhance optimization efficiency and scalability.

3.1.1 Accuracy-Oriented Optimization Techniques

Precision-focused optimization techniques like SGD, Adam, AdaGrad, RMSprop, Nesterov Accelerated Gradient (NAG), and Adadelta are crucial for refining machine learning models to get superior prediction accuracy and reduce error measures. The strategies concentrate on improving model parameters via the adjustment of learning rates, the use of momentum, and the efficient exploration of optimization landscapes. Adam and RMSprop algorithms are proficient in optimizing complex, non-convex objective functions in deep learning by adjusting learning rates dynamically and considering past gradients [44–45]. NAG is a technique that improves standard gradient descent by adding momentum, which accelerates convergence and enhances performance. Adadelta adjusts learning rates automatically by considering gradient magnitude and update history, removing the need for human modifications. These accuracy-focused optimization strategies allow practitioners to effectively train models and achieve higher performance in many machine learning applications.

3.1.2 Speed-Oriented Optimization Techniques

Optimization strategies focused on speed are essential for effectively training ML models, especially when utilizing large-scale datasets and computationally demanding tasks. The strategies aim to optimize the training process to decrease computing time and resource requirements, as well as resulting in quicker convergence and enhanced efficiency.

1) **Mini-Batch Gradient Descent**: Mini-batch gradient descent is an adaptation of the gradient descent optimization process that splits the training data into smaller batches. Mini-batch gradient descent estimates gradients by utilizing a subset of data samples, compared to batch gradient descent which uses the complete dataset or SGD which uses a single data point [46]. Mini-batch

gradient descent allows for more frequent updates as well as faster convergence by adjusting model parameters using gradients calculated from each mini-batch, as opposed to batch gradient descent. Practitioners may take use of both stochastic and batch gradient descent advantages by using this technique, achieving a compromise between computing economy and convergence speed.

2) **Parallelization and Distributed Training**: Parallelization approaches divide computing tasks across numerous processing units like GPUs, TPUs, or distributed clusters. Parallelization approaches expedite training durations and enhance computational efficiency by handling several data points concurrently across several processing units. Distributed training involves sharing both the model parameters and data over numerous devices or processors to parallelize the process [47]. This enables simultaneous calculation of gradients, changes to parameters, and assessments of the model, resulting in substantial reductions in training duration and resource needs. Frameworks like as TensorFlow, PyTorch, and Horovod provide strong support for distributed training on different hardware setups, enabling practitioners to use parallelization and distributed computing methods to accelerate the training process.
3) **Momentum Optimization**: Momentum optimization is a method that improves the conventional gradient descent by including momentum into the parameter updating process. Momentum optimization involves accumulating a moving average of prior gradients to guide parameter changes, instead than depending just on the present gradient. Momentum optimization may expedite convergence by using momentum, particularly when dealing with noisy or sparse gradients [48]. The momentum term reduces oscillations and aids in navigating steep optimization landscapes more efficiently, resulting in quicker convergence throughout the optimization process.
4) **Adaptive Learning Rate Scheduling**: Adaptive learning rate scheduling approaches modify the learning rate during training by considering parameters including gradient magnitudes, parameter updates, and convergence progress. Examples of learning rate schedules include the learning rate decay, which decreases the learning rate over time, and learning rate warm-up, which boosts the learning rate at the beginning of training [49]. Adaptive learning rate scheduling approaches optimize the training process by adjusting the learning rate based on the optimization landscape, leading to quicker convergence and enhanced stability.
5) **Model Quantization and Compression**: Model quantization and compression approaches seek to decrease the computational and memory requirements of ML models by expressing parameters and activations with a reduced number of bits. Quantizing model parameters and activations to lower precision, such as 8-bit integers, using model quantization and compression approaches may greatly decrease memory bandwidth needs and speed up inference on hardware systems with restricted computing capabilities [50]. Methods like as quantization-aware training and post-training quantization allow users to train models with lower accuracy without significant loss in performance, making them ideal for speed-focused applications with limited resources.
6) **Early Stopping and Model Checkpointing**: Early stopping and model checkpointing are methods used to supervise the training process and terminate training when certain convergence criteria are reached or performance indicators level off. Early stopping strategies enable practitioners to halt training prematurely by assessing the model's performance on a validation set at intervals, thereby averting overfitting and decreasing the consumption of training time and computing resources [51]. Model checkpointing strategies enhance early halting by intermittently storing copies of the model parameters during the training process. These checkpoints enable the continuation of training from intermediate stages, recovery from failures, and deployment of trained models for inference. They help optimize the use of computing resources and support repeatability and scalability in ML processes.

Speed-focused optimization methods like momentum optimization, adaptive learning rate scheduling, model quantization and compression, and early stopping/model checkpointing are crucial for speeding up the training process and cutting down on computational costs in ML tasks. By using these methods, professionals may maximize the use of computing resources, speed up model convergence, and expand their ML processes to manage bigger datasets and more intricate models efficiently.

3.1.3 Generalization-Oriented Optimization Techniques

Optimization strategies focused on generalization are crucial for ensuring ML models generalize well to novel data and prevent overfitting. These methods focus on enhancing model parameters to enhance the model's capacity to generalize, leading to more resilient and dependable performance on novel instances.

1) **Regularization**: Regularization methods like L1 and L2 have a tendency to avoid overfitting by punishing

high parameter values [52–53]. Regularization strategies enhance the learning of simpler models that can generalize more effectively to novel data by including regularization terms into the objective function.

- **L1 Regularization (Lasso)**: L1 regularization introduces a penalty term based on the absolute value of the model's parameters into the loss function. This penalty promotes sparsity in the parameter space, which decreases the model's complexity and removes unnecessary features.
- **L2 Regularization (Ridge)**: L2 regularization introduces a penalty term that is directly related to the squared size of the model's parameters into the loss function. This penalty promotes the use of lower parameter values and smoother decision boundaries, which aids in preventing overfitting and enhancing generalization performance.

2) **Dropout**: Dropout is a regularization method utilized in neural networks to inhibit the co-adaptation of neurons and enhance generalization performance. Dropout randomly deactivates a portion of neurons in each layer during training, which adds noise and redundancy to the network [54]. This randomness promotes the network to acquire more resilient characteristics and avoids over-reliance on particular neurons or features, leading to enhanced generalization of novel inputs.
3) **Data Augmentation**: Data augmentation is a method that artificially enhances the variety of the training data-set by implementing various modifications on the input data. Data augmentation involves applying transformations like rotations, translations, flips, and zooms to training samples [55]. This process exposes the model to a wider array of data variances, promoting the learning of more invariant and robust features. Augmentation may be especially advantageous in situations with little or unbalanced training data, allowing the model generalize more effectively to novel instances and enhance its overall performance.
4) **Early Stopping**: Early stopping is a regularization method that prevents overfitting by evaluating the model's performance on a separate validation set during training and interrupting the training process when performance begins to deteriorate. Early stopping prevents overfitting by ending training before the model memorizes noise or outliers in the data, ensuring better generalization to new samples [56]. This strategy promotes the model to acquire more generalizable characteristics and avoids it from becoming too intricate, thus enhancing its capacity to generalize to new data.
5) **Ensemble Learning**: Ensemble learning approaches amalgamate many base models to create a more resilient and precise composite model. Ensemble approaches combine predictions from numerous models to enhance generalization performance by reducing variation and bias of individual models [57]. Bagging, boosting, and stacking techniques use a variety of base models trained on distinct portions of the training data or using various methods to capture complementing characteristics of the underlying data distribution. Ensemble learning efficiently utilizes the combined knowledge of numerous models to provide more reliable predictions and improve generalization to new instances.
6) **Transfer Learning**: Transfer learning is a ML method that utilizes information acquired from a source domain or job to enhance performance on a target domain or task. Transfer learning allows models to improve their ability to generalize to new data and tasks with limited annotated training data by transferring representations learnt from pre-trained models trained on large-scale datasets [58]. Transfer learning is beneficial when the target task lacks adequate training data or when the source and target domains have comparable fundamental properties. Transfer learning improves generalization performance and speeds up model convergence by using pre-trained models as feature extractors or fine-tuning their parameters for the target task. This makes it a valuable tool for optimization focused on generalization.

Generalization-oriented optimization approaches including data augmentation, early stopping, ensemble learning, and transfer learning are crucial for enhancing the resilience and generalization capabilities of ML models. By integrating these methods throughout the training process, professionals may efficiently combat overfitting, promote the acquisition of more accurate features, and enhance the model's capacity to generalize to novel data and tasks.

3.2 Popular Optimization Algorithms

Optimization algorithms are essential in training ML models since they constantly update model parameters to minimize or maximize an objective function. This article presents an overview of popular optimization techniques in ML, each with distinct methods and compromises to achieve effective model optimization.

3.2.1 Gradient Descent and Its Variants

Gradient descent is an important optimization approach that reduces objective functions by repeatedly modifying model parameters in the direction opposite to the gradient

[59]. Different versions of gradient descent provide various approaches for adjusting model parameters and are designed to tackle distinct obstacles faced during optimization. Here, we explore several notable variations:

1) **Stochastic Gradient Descent (SGD)**: SGD improves the model assumptions by utilizing gradients calculated from a randomly chosen subset of training instances, known as mini-batches [60]. SGD provides randomness into the optimization process, resulting in quicker convergence and lower memory use compared to batch gradient descent. SGD rapidly navigates complicated optimization landscapes and handles large-scale datasets by updating parameters based on mini-batches. SGD's convergence may vary more than batch gradient descent because of the stochastic nature of gradient estimation.
2) **Adam (Adaptive Moment Estimation)**: Adam is a learning rate optimization technique that integrates principles from momentum and RMSprop [61]. It adapts learning rates for individual parameters by considering previous gradients and squared gradients, which is effective for optimizing intricate, non-convex objective functions often seen in deep learning applications. Adam can efficiently manage sparse gradients, noisy data, and non-stationary targets by adjusting learning rates according to gradient magnitudes and variance. Adam's adjustable learning rate approach allows for quicker convergence and improved performance across many optimization problems.
3) **RMSprop (Root Mean Square Propagation)**: RMSprop is an optimization approach that tackles the issue of decreasing learning rates in AdaGrad by using a moving average of squared gradients to dynamically adjust learning rates for each parameter [62]. RMSprop stabilizes the optimization process by leveling learning rates and limiting rapid diminishment. RMSprop effectively manages non-stationary specifications and improves optimization stability and efficiency compared to AdaGrad by using a leaky average of prior squared gradients. RMSprop is well-suited for training deep neural networks and other models with intricate, high-dimensional parameter spaces.
4) **Adagrad (Adaptive Gradient Algorithm)**: Adagrad is an optimization technique that optimizes the learning rates of individual parameters according on recent gradients. The system assigns higher learning rates to parameters that are updated seldom and lower learning rates to parameters that are updated regularly. Adagrad is especially useful for sparse data or features with varying scales since it adapts the learning rates according to the gradient magnitudes [63]. Adagrad's long-term accumulation of squared gradients might cause learning rates to decrease, leading to slower convergence during later stages of training.
5) **Adadelta**: Adadelta is a modification to Adagrad designed to resolve the problem of decreasing learning rate. Adadelta computes a moving average of squared gradients by using a sliding window of prior gradients instead of treating all historic gradients equally [64]. Adadelta adjusts learning rates depending on recent gradients to address the issue of declining learning rates. Adadelta is ideal for optimization problems that need a decreased sensitivity to the learning rate, such as in RNNs.
6) **Nadam (Nesterov-accelerated Adaptive Moment Estimation)**: Nadam is an extension of Adam that integrates Nesterov momentum into its update rule. Nesterov momentum calculates the gradient at a location slightly ahead in the direction of momentum, rather than at the present parameter values [65]. Nadam optimizes performance by using Nesterov momentum to predict parameter movements and modify update directions, resulting in quicker convergence and enhanced performance, particularly when dealing with noisy gradients.
7) **AdaMax**: AdaMax is a variation of the Adam optimization algorithm that incorporates the infinity norm in the update rule, resulting in improved convergence stability under certain circumstances [66]. AdaMax calculates the L-infinity norm of the exponential moving averages of the gradients, instead of the infinity norm of the gradient like Adam. AdaMax is modified to modify learning rates adaptively, independent of gradient magnitude, enhancing its resilience to noisy gradients and sparse data.
8) **AMSGrad**: AMSGrad is a modification of the Adam optimization method designed to resolve the problem of non-convergence that might occur in certain scenarios with the original Adam algorithm [67]. AMSGrad adjusts Adam's update algorithm to maintain the denominator term (squared gradients) as monotonically rising to avoid the learning rate from growing excessively. AMSGrad maintains the highest historical squared gradients to improve stability and convergence compared to Adam, especially in situations with non-convex optimization goals.

Gradient Descent and its variations in Fig. 1 provide a range of optimization methods designed to tackle various obstacles in ML applications. By adjusting learning rates, adding momentum, and addressing challenges including decreasing learning rates and erratic gradients, these versions enhance the optimization of model parameters and expedite the attainment of optimum solutions.

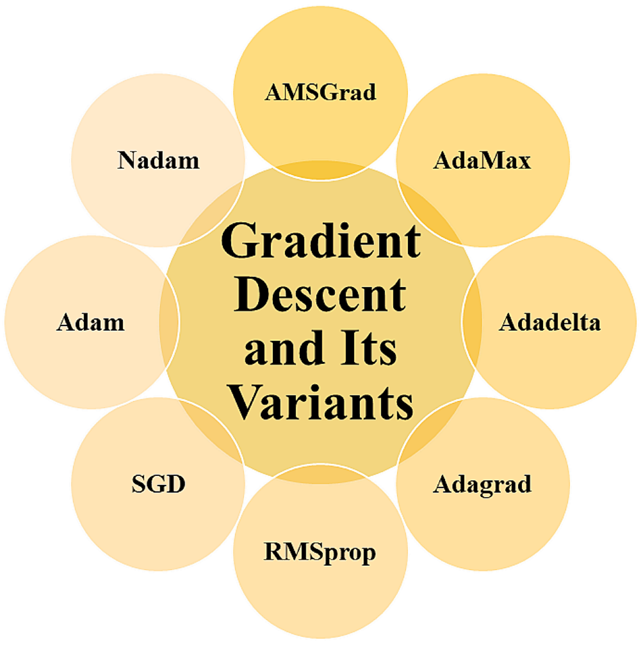

Fig. 1 Gradient descent and its variants in ML optimization techniques

3.2.2 Evolutionary Algorithms

Evolutionary algorithms are optimization methods that use populations and are influenced by the concepts of natural selection and evolution. The algorithms work on a group of potential solutions (individuals) and gradually improve them over several generations to discover the best or nearly best answers. Let's explore some instances of evolutionary algorithms as shown in Fig. 2 and details are also explained.

1) **Genetic Algorithms (GA)**: Genetic Algorithms (GA) imitate natural selection by progressively developing a group of potential solutions via selection, crossover, and mutation operations. The procedure starts by creating a population of people that represent possible solutions to the optimization issue [68–69]. Each generation involves selecting people based on their fitness, which is their capacity to solve the challenge, and then subjecting them to genetic operations like crossover and mutation to create offspring. The children replace fewer fit people in the population, resulting in the progressive development of improved solutions over subsequent

Fig. 2 Evolutionary algorithms of ML optimization techniques

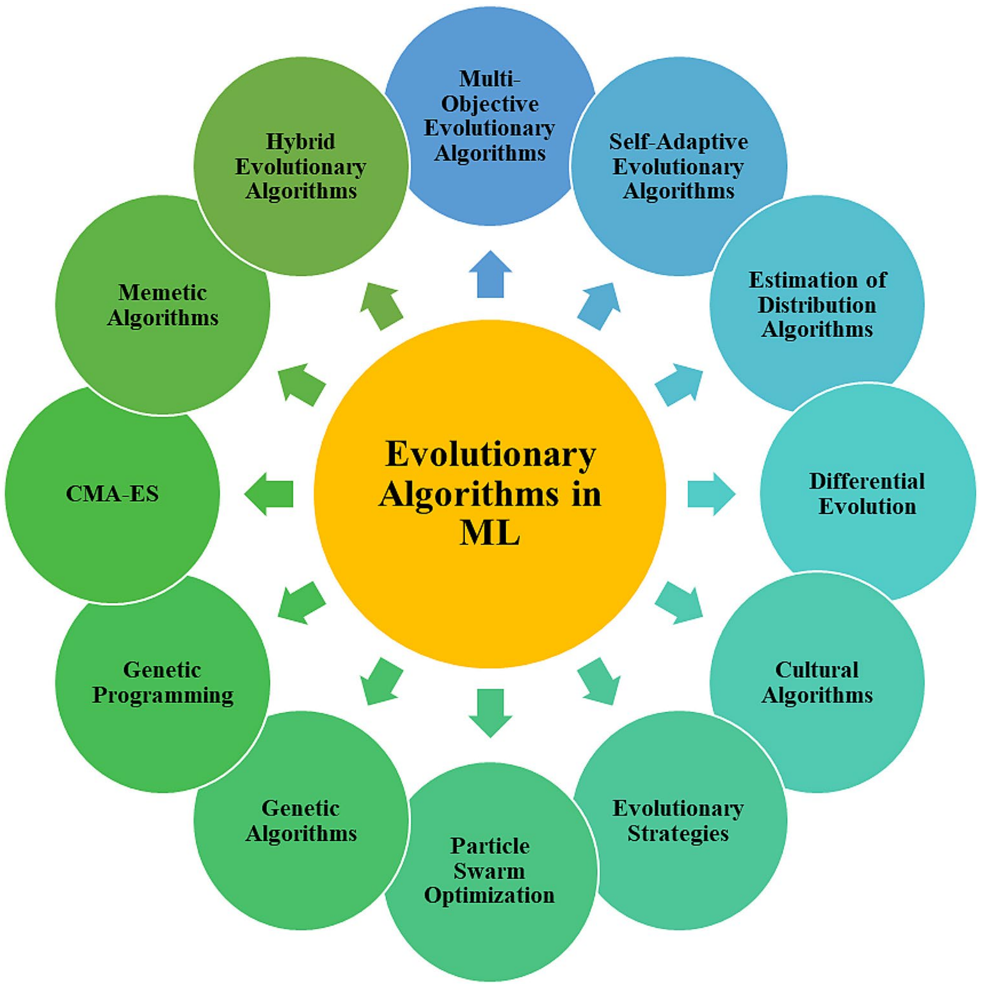
Evolutionary Algorithms in ML
Multi-Objective Evolutionary Algorithms
Self-Adaptive Evolutionary Algorithms
Estimation of Distribution Algorithms
Differential Evolution
Cultural Algorithms
Evolutionary Strategies
Particle Swarm Optimization
Genetic Algorithms
Genetic Programming
CMA-ES
Memetic Algorithms
Hybrid Evolutionary Algorithms

Table 3 Comparison of optimization algorithms

Optimization Algorithm	Convergence Speed	Memory Requirements	Scalability	Suitability for Different Problems	Robustness	Interpretability
SGD [133]	Moderate	Low	High	General-purpose optimization, large-scale datasets	Low	Moderate
Adam [134]	Fast	Moderate	Moderate	Deep learning, non-convex optimization	Moderate	Low
RMSprop [135]	Moderate	Low	High	Deep learning, non-convex optimization	Moderate	Low
Genetic Algorithms [136]	Slow	High	Low	Combinatorial optimization, parameter search	High	Low
Particle Swarm Optimization (PSO) [137]	Moderate	Low	Low	Continuous optimization, swarm intelligence	Low	Low
Simulated Annealing [138]	Slow	Low	High	Optimization with complex search spaces	Moderate	Low
Ant Colony Optimization (ACO) [139]	Moderate	Low	Moderate	Combinatorial optimization, graph problems	Moderate	Low
Differential Evolution [140]	Moderate	Moderate	Moderate	Continuous optimization, parameter search	Moderate	Low
Bayesian Optimization [141]	Moderate	Moderate	Low	Optimization with expensive evaluations, hyperparameter tuning	Moderate	Low
Evolutionary Strategies [142]	Moderate	Moderate	Moderate	Continuous optimization, population-based search	Moderate	Low

generations. Genetic algorithms use selection, crossover, and mutation to navigate the solution space, directing the search towards favorable areas and eventually reaching optimum or nearly optimal solutions.

2) **Evolutionary Strategies (ES)**: Evolutionary Strategies (ES) is a classification of evolutionary algorithms that concentrates on enhancing continuous parameters by using a Gaussian distribution to create offspring solutions. Evolutionary strategy (ES) functions on a group of potential solutions, with each option being represented by an array of continuous variables. Each generation produces offspring solutions by adjusting the characteristics of parent solutions based on a predetermined Gaussian distribution [70–71]. Offspring solutions are assessed according to their fitness, and those with superior fitness levels are chosen to generate the next generation. Evolutionary techniques iteratively adjust candidate solutions' characteristics according on their performance to explore the solution space and converge towards optimum or near-optimal solutions.
3) **Genetic Programming (GP)**: genetic Programming (GP) utilizes evolutionary algorithms to develop computer programs or models represented as trees. Genetic programming initializes populations of program structures, with each program serving as a possible solution to the optimization issue [72–73]. Genetic operators like mutation, crossover, and reproduction are used to develop these program structures throughout many generations. Subtrees from parent programs are swapped during crossover to generate offspring programs that may have varied architectures. Mutation brings about random alterations in particular programs, whereas reproduction enables certain programs to be passed on without modifications to the following generation. Each program's fitness is assessed according to its problem-solving capabilities, and selection processes decide which programs will be kept for generating children in the next generation. Genetic programming generates program structures by applying genetic operators and selection processes iteratively, leading to solutions that maximize the specified aim.
4) **Differential Evolution (DE)**: Differential Evolution (DE) is an optimization approach that enhances candidate solutions by modifying their parameter vectors via differential mutation and recombination operations within a population-based framework. DE maintains a population of possible solutions, each characterized by a vector of parameter values. DE creates trial solutions in each iteration by merging vectors from the current population via differential mutation and recombination [74–75]. The experimental solutions are evaluated against the current population, and individuals with

better fitness are selected for the following generation. DE's differential mutation technique promotes effective exploration of the solution space, while its recombination operations support the exploitation of attractive areas, resulting in quick convergence to optimum or near-optimal solutions.

5) **Particle Swarm Optimization (PSO)**: Particle Swarm Optimization (PSO) is an optimization technique based on the collective behaviour of bird flocks or fish schools. The PSO algorithm maintains a population of particles, with each particle representing a potential solution to the optimization problem. Particles travel around the solution space in each iteration depending on their individual best-known location and the best-known position of the overall population [76–77]. The movement is directed by velocity vectors that are adjusted according to each particle's past behaviour and the combined impact of nearby particles. The exploration-exploitation mechanism of PSO allows for effective investigation of the solution space while taking use of interesting locations discovered by the swarm. PSO uses repeated adjustments of particle locations and velocities to achieve convergence towards optimum or near-optimal solutions for various optimization problems.
6) **Cultural Algorithms (CA)**: Cultural Algorithms (CA) are evolutionary algorithms that include cultural evolution ideas into the optimization process. CA maintains a population of candidate solutions and a belief space that represents cultural knowledge or norms [78–79]. Each iteration involves evaluating prospective solutions according to their fitness and updating cultural knowledge depending on the population's performance. Cultural information impacts the development of potential solutions by favouring selection, crossover, and mutation operations in areas of the solution space that align with existing norms. By integrating genetic evolution with cultural development, CA efficiently reach optimum or almost optimal solutions while maintaining and spreading valuable information within the population.
7) **Covariance Matrix Adaptation Evolution Strategy (CMA-ES)**: Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is an evolutionary strategy that aims to optimize continuous parameters by using a multivariate normal distribution. CMA-ES uses a group of potential solutions and modifies the covariance matrix of the multivariate normal distribution to direct the search. CMA-ES efficiently explores and exploits the solution space by adapting the covariance matrix according to the success of prior candidate solutions, resulting in quick convergence to optimum or near-optimal solutions [80–81]. CMA-ES is suitable for optimizing positions involving high-dimensional parameter spaces and intricate, non-convex objective functions.
8) **Memetic Algorithms (MA)**: Memetic Algorithms (MA) combines evolutionary algorithms with local search techniques to enhance solution quality. MA maintains a population of potential solutions that are subjected to genetic operations like mutation, crossover, and selection [82–83]. Also, each potential solution undergoes local search techniques to investigate the surrounding area and enhance its quality. By combining worldwide exploration with nearby exploitation, MA can efficiently manage the balance between exploration and exploitation, resulting in enhanced convergence and solution quality.
9) **Estimation of Distribution Algorithms (EDA)**: Estimation of Distribution Algorithms (EDA) is a type of evolutionary algorithms that use statistical techniques to represent the probability distribution of potential solutions. EDA develops and improves a probabilistic model of the solution space using the observed candidate solutions. The probabilistic model is used to generate fresh potential solutions, which are assessed and added to the existing population [84–85]. EDA may effectively direct the search process towards favorable areas by explicitly modeling the probability distribution of the solution space, resulting in quicker convergence and enhanced solution quality.
10) **Hybrid Evolutionary Algorithms**: Hybrid Evolutionary Algorithms combines evolutionary optimization approaches with other optimization methods to use their complimentary capabilities. Hybrid algorithms combine evolutionary algorithms with gradient-based optimization methods, local search algorithms, or metaheuristic approaches to enhance convergence speed and solution quality [86]. Hybrid techniques combine evolutionary algorithms with other optimization methodologies to successfully solve complicated optimization problems with various features and constraints.
11) **Self-Adaptive Evolutionary Algorithms**: Self-Adaptive Evolutionary Algorithms modify their parameters and operations in response to the problem's traits and the algorithm's effectiveness. These algorithms adjust their mutation rates, crossover probabilities, population sizes, or other parameters automatically as they enhance their performance for particular issue scenarios [87–88]. Self-adaptive evolutionary algorithms may provide resilient and efficient optimization in many problem domains and situations by adjusting to changing circumstances and needs.
12) **Multi-Objective Evolutionary Algorithms (MOEA)**: Multi-Objective Evolutionary Algorithms (MOEA) are

specifically created to maximize many goals that are in competition with each other at the same time. MOEAs preserve groups of potential solutions, with each one embodying a balance between many goals [89–90]. The algorithms try to provide a collection of answers called the Pareto front, which represents the best possible compromises between competing goals. MOEAs use specific selection processes and tactics to maintain variety in order to effectively explore the Pareto front and find high-quality solutions that meet several optimization requirements.

Evolutionary Algorithms consist of several optimization approaches that are influenced by ideas of natural selection and evolution. These algorithms manage groups of potential solutions and gradually improve them using evolutionary processes, social exchanges, or adaptive methods to discover the best or nearly best answers for intricate optimization issues. Evolutionary Algorithms use the combined knowledge of populations and adjust to different conditions to provide strong and efficient methods for tackling a variety of optimization problems in different fields.

3.2.3 Bayesian Optimization

Bayesian optimization is an approach of optimizing black-box objective functions with prohibitive evaluations using a probabilistic framework [91–92]. Bayesian optimization represents the objective function as a Gaussian process, providing a flexible and probabilistic representation of uncertainty without needing explicit gradients or derivatives of the objective function like classic optimization approaches.

1) Key Components of Bayesian Optimization:

- **Surrogate Model**: Bayesian optimization provides a predictive probabilistic model of the desired function by using the evaluations that have been observed [93]. The surrogate model, typically a Gaussian process, represents the fundamental patterns and uncertainties in the objective function space.
- **Acquisition Function**: The acquisition function promotes the pursuit of optimum solutions by managing the trade-off between exploration and exploitation. The usefulness of picking a candidate point is determined by considering both the predictions of the surrogate model and the uncertainty estimations. Common acquisition functions include Expected Improvement (EI), Probability of Improvement (PI), and Upper Confidence Bound (UCB) [94].
- **Sequential Sampling**: Bayesian optimization involves selecting potential sites based on the acquisition function's suggestions, evaluating the objective function at these points, and updating the surrogate model with the observed data in an iterative manner [95]. The sequential sampling approach effectively navigates the objective function space and concentrates the search on favourable areas.

2) Advantages of Bayesian Optimization:

- **Efficient Exploration**: Bayesian optimization effectively navigates the objective function space by iteratively updating a probabilistic surrogate model and concentrating the search on areas with high potential for improvement [96]. This allows for efficient worldwide search even in complex or noisy optimization situations.
- **Sample Efficiency**: Bayesian optimization uses probabilistic models and acquisition functions to efficiently optimize by needing fewer objective function evaluations than standard approaches. This is especially beneficial in situations when objective function evaluations are resource-intensive or time-consuming [97].
- **Adaptive Trade-off**: Bayesian optimization uses the acquisition function to balance exploration and exploitation, enabling it to adjust between exploring new parts of the objective function space and exploiting already promising locations. This adaptive behavior allows for strong and effective optimization in several issue areas [98].

3) Applications of Bayesian Optimization:

- **Hyperparameter Tuning**: Bayesian optimization has become popular for optimizing hyperparameters in ML models, with the objective function reflecting model performance indicators like accuracy or validation loss. Bayesian optimization effectively searches the hyperparameter space to find the best situations that optimize model performance and minimize computing resources [99].
- **Experimental Design**: Bayesian optimization is utilized in experimental design to optimize parameters and situations, including medication dose in pharmaceutical research or experimental setup in scientific trials. Bayesian optimization speeds up the process of finding the best solutions by choosing trials that maximize predicted information gain in an iterative manner [100].
- **Automated Machine Learning (AutoML)**: Bayesian optimization is a fundamental element of AutoML systems, streamlining the tasks of model selection, feature engineering, and hyperparameter optimization. AutoML frameworks effectively explore the range of potential

models and settings using Bayesian optimization to find high-performing ML processes [101].

Bayesian optimization provides a systematic and effective method for optimizing black-box objective functions that need costly assessments. Bayesian optimization is a powerful tool for optimizing ML models and other complex systems by efficiently exploring and exploiting the objective function space through probabilistic models, adaptive acquisition functions, and sequential sampling strategies.

3.3 Optimization Techniques for Various Machine Learning Technologies

Optimization strategies in ML are used in supervised learning, unsupervised learning, Semi-Supervised Learning, Transfer Learning, Metaheuristic Optimization, and reinforcement learning methodologies.

3.3.1 Supervised Learning

Supervised learning optimization strategies focus on minimizing a predetermined loss function that measures the difference between predicted and real labels. The goal is to develop a model that can precisely correlate input properties with their respective labels. Gradient-based optimization methods like SGD, Adam, and RMSprop are often used to train supervised learning models [102]. The methods update model parameters repeatedly by calculating gradients of the loss function with respect to the model parameters and changing them to minimize the loss.

3.3.2 Unsupervised Learning

Unsupervised learning tasks including clustering and dimensionality reduction involve maximizing objective functions using data structure and similarity measurements, without the use of labeled data. Clustering methods such as k-means and hierarchical clustering divide data into groups (clusters) where data points in the same cluster are more alike than those in other clusters [103]. Principal component analysis (PCA) is a dimensionality reduction approach that aims to decrease the number of variables in the data while retaining a significant portion of its variability. Optimization strategies often use iterative algorithms to update cluster assignments or alter data representations in order to minimize a given criteria, such as reducing intra-cluster variance or increasing between-cluster separation.

3.3.3 Reinforcement Learning

Reinforcement learning utilizes optimization methods to discover an optimum policy that maximizes cumulative rewards in a changing environment. Reinforcement learning algorithms use different optimization techniques to adjust the policy parameters using rewards and states observed during exploration and exploitation phases. Q-learning is a widely used reinforcement learning technique that updates the Q-values of state-action pairs repeatedly to estimate the ideal action-value function [104]. Policy gradient approaches, like REINFORCE, focus on optimizing policy parameters by calculating gradients of anticipated rewards in relation to the policy parameters and adjusting them via gradient ascent.

3.3.4 Semi-Supervised Learning

Semi-supervised learning is a ML approach that utilizes both labeled and unlabeled data to enhance model performance [105]. This method is beneficial in situations when acquiring labeled data is costly or time-intensive, but there is an abundance of unlabeled data. Here is a comprehensive review of the optimization components of semi-supervised learning.

1) **Joint Minimization of Loss Function**: Semi-supervised learning optimizes by simultaneously reducing the loss function for labeled and unlabeled input. The loss function usually has two parts: one for labeled data with ground truth labels and another for unlabeled data where model predictions are compared with pseudo-labels or inferred labels [106]. The goal is to determine model parameters that reduce the total loss for both identified and unlabeled occurrences.
2) **Incorporation of Constraints or Regularization Terms**: Semi-supervised learning approaches enhance the use of unlabeled data by integrating extra restrictions or regularization terms throughout the optimization phase. These constraints aim to enhance smoothness or consistency in predictions for comparable instances, use cluster structure or manifold shape in the data, and ensure agreement between labeled and unlabeled cases [107]. Regularization methods including entropy minimization, consistency regularization, and manifold regularization are often used to promote the model in generating more resilient and generalizable predictions.
3) **Utilization of Semi-Supervised Learning Techniques**: Semi-supervised learning approaches may be used to efficiently use the intrinsic structure or correlations present in the data. Self-training is the process of training a model on labeled data and then utilizing the

model's predictions to assign pseudo-labels to unlabeled occurrences in an iterative process [108]. Co-training involves training several models on distinct subsets of characteristics or representations and sharing reliable predictions between them. Graph-based regularization methods use graph structures derived from the data to transmit information and impose smoothness or consistency restrictions on the predictions.

4) **Combination of Supervised and Unsupervised Optimization Techniques**: Optimization methods in semi-supervised learning combine supervised and unsupervised strategies to use labeled and unlabeled data efficiently. Gradient-based optimization methods like SGD and its variations are often used to train semi-supervised learning models [109–110]. The techniques optimize model parameters by calculating gradients of the joint loss function with respect to the parameters and repeatedly updating them to minimize the total loss across labeled and unlabeled data.

Optimization in semi-supervised learning involves reducing the loss function for both labeled and unlabeled data simultaneously, while integrating constraints or regularization terms to make use of the structure of unlabeled data efficiently. Methods like self-training, co-training, and graph-based regularization have a tendency to leverage the intrinsic structure or connections in the data to enhance model performance. Semi-supervised learning combines supervised and unsupervised optimization approaches to improve model generalization and performance by making more efficient use of existing data.

3.3.5 Transfer Learning

Transfer learning is a ML technique that uses information gained from a specific position or domain to improve performance on a related but distinct task or area [111]. Transfer learning optimization includes adjusting pre-trained models or representations for new tasks or domains while reducing the difference between the source and target data distributions. Here is an in-depth analysis of the optimization components of transfer learning:

1) **Adaptation of Pre-trained Models**: Transfer learning usually starts with a pre-trained model that has been trained on a source task or domain using a substantial quantity of labeled data. The optimization phase entails customizing the pre-trained model for the particular position or area, especially when there is a scarcity of labeled data. The objective is to fine-tune model parameters to get high performance on the specific task by effectively using the information stored in the pre-trained model [112].

2) **Fine-tuning**: Fine-tuning is a prevalent method in transfer learning that involves adjusting pre-trained models to suit new tasks or domains. The pre-trained model's parameters are adjusted by gradient-based optimization methods like SGD or its variations throughout the fine-tuning process. The optimization procedure includes calculating gradients of the loss function of the goal task in relation to the model parameters and then updating them gradually to reduce the loss. Fine-tuning enables the model to modify its representations and acquire task-specific characteristics while preserving important information from the original position [113].

3) **Feature Extraction**: Another method of transfer learning is feature extraction, in which the pre-trained model's parameters remain unchanged, and only the top layers (classifier) are substituted or retrained for the new position. Optimization in this scenario entails training the classifier using characteristics derived from the pre-trained model while leaving the bottom layers fixed. Feature extraction is beneficial when the source and target activities have comparable low-level properties but vary in higher-level representations or output spaces [114].

4) **Domain Adaptation**: Domain adaption strategies are used in transfer learning when the source and destination domains have dissimilar distributions. Domain adaptation optimization minimizes the distribution difference between the source and target domains to modify the model to the target domain. Methods like adversarial training, domain adversarial neural networks (DANN), and domain confusion loss can be used to harmonize feature distributions across domains during optimization [115].

5) **Partial Retraining**: During some transfer learning situations, it might be advantageous to selectively retrain certain components of the pre-trained model while leaving the rest unmodified. Partial retraining enhances adaptability to the target task or domain by concentrating optimization on the most pertinent components of the model. In CNNs, only the remaining layers may be trained again for a new classification task, while the convolutional layers remain unchanged to preserve low-level feature representations [116].

Optimization in transfer learning involves adapting pre-trained models or representations to new tasks or domains while reducing the difference between the source and target data distributions. Methods like fine-tuning, feature extraction, domain adaptation, and partial retraining have a tendency to effectively optimize model parameters for the specific task or domain. Transfer learning allows for more effective and efficient learning with limited labeled data by using knowledge transfer from pre-trained models and adjusting representations for new tasks.

3.3.6 Metaheuristic Optimization

Metaheuristic optimization methods are a variety of optimization algorithms that derive motivation from natural and social circumstances [117–118]. These algorithms provide potent resources for addressing intricate optimization issues in several fields. Here is an in-depth assessment of metaheuristic optimization:

1) Characteristics of Metaheuristic Optimization:

- **Inspiration from Natural and Social Phenomena**: Metaheuristic optimization is impacted by various natural and social phenomena such as evolutionary processes, swarm behaviour, and physical dynamics. Metaheuristic algorithms strive to effectively explore the solution space and identify near-optimal solutions by imitating the processes found in these systems [119].
- **Heuristic Search Strategies**: Metaheuristic algorithms use heuristic search methods to efficiently explore intricate solution spaces. The techniques manage the search process by systematically investigating exciting areas, using identified solutions, and maintaining a balance between exploration and exploitation to prevent premature reliance on impoverished responses [120].
- **Iterative Improvement**: Metaheuristic optimization techniques usually consist of iterative procedures that progressively enhance potential solutions over various generations or iterations. These algorithms improve potential solutions, adjust search methods, and move towards almost perfect solutions via repeated cycles [121].
- **Adaptation and Flexibility**: Metaheuristic algorithms demonstrate adaptability and flexibility by dynamically adjusting parameters, strategies, and search operators depending on issue features, solution quality, or convergence progress. This flexibility improves their strength and efficiency in various optimization tasks [122].

2) Examples of Metaheuristic Optimization Algorithms:

- **Firefly Algorithm (FA)**: The Firefly Algorithm (FA) is inspired by the bioluminescent flashing characteristic of fireflies, which they employ to attract mates or prey. Candidate solutions in Firefly Algorithm appear as fireflies throughout the solution space [123]. Fireflies are attracted to brighter ones, symbolizing superior solutions, and navigate the solution space by considering the appeal of nearby fireflies. The Firefly Algorithm (FA) tries to optimize solutions by repeatedly adjusting the placements of fireflies according to their attractiveness and the distance between them. Fireflies gather toward brighter sources, mimicking the synchronized flashing activity seen in nature.
- **Harmony Search (HS)**: Harmony Search (HS) is based on performers improvising tunes that develop harmony, generating new melodies by blending components from current ones. In high school, potential solutions are shown as musical harmonies. The program creates fresh solutions by merging components from current harmonies and modifies them to enhance harmony [124]. HS tries to identify the optimal solution by continuously improving potential harmonies according to their harmony values. The program navigates the solution space by modifying musical components and strives towards harmonic solutions.
- **Bat Algorithm (BA)**: The Bat Algorithm (BA) is based on the echolocation behaviour of bats, in which bats use ultrasonic pulses for navigation and prey detection. Candidate solutions in the field of business analysis are presented as bats that generate sonar pulses inside the solution space [125]. Bats use echolocation to explore the solution space and modify their locations according to the strength of echoes, which indicate solution quality. BA seeks to determine the optimal solution by continuously adjusting the placements of bats according to their echolocation input. Bats are attracted to areas with increased echo intensity, mimicking their natural foraging behaviour.
- **Cuckoo Search (CS)**: Cuckoo Search (CS) is based on the brood parasitism behaviour of some cuckoo species, in which cuckoos deposit their eggs in the nests of other bird species. In the field of computing, proposed solutions are symbolized as cuckoo eggs placed in the nests of host birds. Cuckoos deposit eggs in host nests according on their quality and will substitute current eggs if they are superior [126]. Computer Science seeks to discover the optimal solution by continuously updating cuckoo eggs in host nests. Cuckoos search for optimal solutions by depositing eggs in nests with superior solutions, mimicking the brood parasitism behaviour observed in nature.
- **Artificial Bee Colony (ABC) Algorithm**: The ABC Algorithm is based on honeybees' foraging behaviour, in which bees share information about food sources via waggle dances. In the Artificial Bee Colony (ABC) algorithm, proposed solutions are symbolized as food sources, and artificial bees navigate the solution space by inspecting these sources. Bees share details about food quality and choose better food sources by receiving

comments [127]. ABC seeks to determine the optimal solution by continuously adjusting the locations of food sources. Bees navigate the solution space by locating potential sources with increased nectar levels, mimicking the foraging behaviours of honeybees.

3) Applications of Metaheuristic Optimization:

- **Engineering Design**: Metaheuristic optimization methods tend to be utilized in engineering design tasks, including structural optimization, aerodynamic design, and parameter tuning in engineering systems [128].
- **Scheduling and Logistics**: Metaheuristic optimization techniques are used in solving scheduling and logistics issues such as task scheduling, automobile routing, and resource allocation [129].
- **Machine Learning**: Metaheuristic optimization techniques are used in a range of ML applications, including hyperparameter optimization, feature selection, and neural network design optimization [130].
- **Combinatorial Optimization**: Metaheuristic optimization is widely used in solving combinatorial optimization issues which include that found in graph theory, network optimization, and combinatorial game theory [131].

Metaheuristic optimization techniques provide flexible and effective methods for addressing intricate optimization issues in several fields. By using heuristic search tactics inspired by natural and social events, these approaches facilitate effective exploration of solution spaces and discovery of near-optimal solutions in cases where standard optimization methods may struggle [132]. Optimization strategies are essential in a wide range of ML approaches, such as supervised learning, unsupervised learning, reinforcement learning, semi-supervised learning, transfer learning, and meta-learning. These methods assist models in efficiently acquiring knowledge from data, adjusting to different tasks or domains, and extrapolating to unfamiliar contexts or situations. Practitioners may choose and modify optimization approaches that align with their ML tasks and goals by comprehending the distinct aims and difficulties of each pattern.

A comprehensive overview of the optimization techniques widely used in ML is presented in Table 3. The paper delineates essential attributes including but not limited to convergence speed, memory demands, scalability, and applicability for diverse problem domains. This review evaluates the merits and drawbacks of various optimization algorithms, ranging from conventional techniques such as SGD to metaheuristic algorithms like genetic algorithms and PSO [143]. The results support practitioners and researchers in determining which optimization approach is most suitable for their particular application domains. Also, qualities such as interpretability and robustness are taken into account, offering significant perspectives on the real implications of implementing each optimization method. In general, this table behaves as a highly beneficial reference instrument for comprehending and contrasting the heterogeneous realm of ML optimization algorithms.

4 Challenges in Machine Learning Optimization

ML optimization experiences several challenges that affect the efficiency, efficacy, and resilience of learning algorithms is shown in Fig. 3. The challenges consist comprising the following:

1) **Optimization Landscape**: The optimization terrain is intricate, including non-convexity, multimodality, and irregular geometries [144]. Non-convex objective functions provide challenges in locating global optima, while multimodal functions include numerous local optima. Complex geometries hinder optimization due to abrupt peaks, valleys, and discontinuities, which provide challenges for optimization algorithms to get the correct responses.
2) **Scalability Issues**: ML activities involve managing extensive datasets and complex parameter spaces, which might result in scalability issues. Handling large datasets may place a heavy load on processing resources and memory limitations [145]. Also, studying parameter spaces with huge dimensions increases the computational workload of optimization algorithms, because they need to search across a large search space to locate the most suitable responses.
3) **Optimization Convergence**: Optimization algorithms may face convergence problems such as premature convergence, local optima, and saddle points. Premature convergence is when the algorithm stops before finding the best answer because it did not fully explore the search space [146]. Local optima are areas where the objective function approaches a minimum, although it may not be the overall minimum. Saddle points provide difficulties by causing gradients to either disappear or diminish, impeding advancement towards the most effective solution.
4) **Robustness and Generalization**: Ensuring resilience and adaptability is essential in optimizing ML. Overfitting happens when the model incorporates noise or irrelevant patterns from the training data, resulting in worse performance on data that is novel [147]. Underfitting happens when the model is too basic to capture

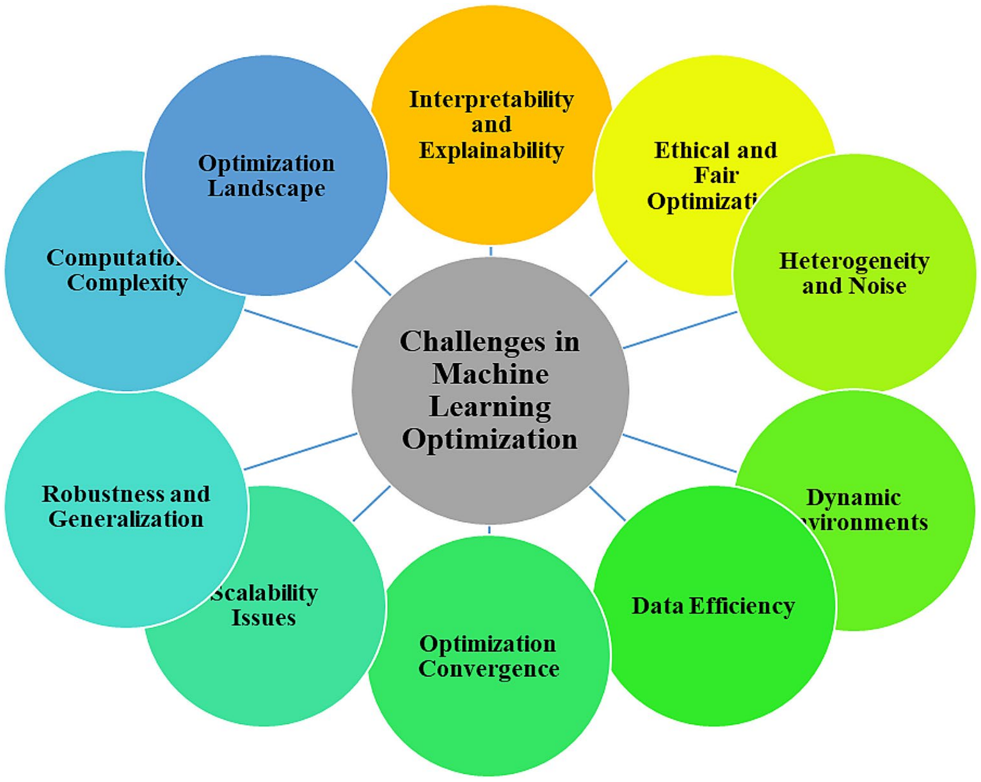

Fig. 3 Various challenges in machine learning optimization

Table 4 Challenges in machine learning optimization and associated optimization methods

Challenges in Machine Learning Optimization	Complex Parameters	Optimization Methods
Optimization Landscape	Non-convexity, Multimodality, Irregular Geometries	Gradient Descent Variants, Evolutionary Algorithms, Bayesian Optimization
Scalability Issues	Large Datasets, High-dimensional Parameter Spaces	Mini-batch Processing, Distributed Computing, Dimensionality Reduction
Optimization Convergence	Premature Convergence, Local Optima, Saddle Points	Learning Rate Scheduling, Momentum Techniques, Adaptive Optimization
Robustness and Generalization	Overfitting, Underfitting, Model Selection	Regularization Techniques, Cross-validation, Ensemble Learning
Computational Complexity	Time Constraints, Resource Constraints	Algorithmic Efficiency, Parallel Computing, Hardware Acceleration
Data Efficiency	Transfer Learning, Meta-learning, Active Learning	Few-shot Learning, Data Augmentation, Semi-supervised Learning
Heterogeneity and Noise	Robust Optimization Techniques, Data Preprocessing	Noise Robustness, Outlier Detection, Data Imputation
Interpretability and Explainability	Interpretable Model Structures, Explainable AI	Model Interpretation Methods, Attention Mechanisms, Rule-based Models
Ethical and Fair Optimization	Fairness-aware Algorithms, Bias Mitigation	Fairness Metrics, Bias Detection, Counterfactual Fairness
Dynamic Environments	Continuous Learning, Online Optimization	Adaptive Learning Rates, Lifelong Learning, Concept Drift Detection

the fundamental patterns in the data. Model selection involves identifying the suitable model structure and hyperparameters to achieve optimum performance while avoiding overfitting or underfitting.

5) **Computational Complexity**: Optimization techniques encounter time and resource limits due to their computational complexity, especially in large-scale or real-time applications. Iterative optimization techniques may demand significant computer resources and time to reach optimum solutions, restricting their practical use in situations with severe time restrictions or restricted computational resources [148].

6) **Data Efficiency**: Optimization approaches require substantial data quantities to efficiently learn model

Table 5 Future research directions in machine learning optimization

Future Research Issue	Description
Addressing scalability challenges	Establishing distributed and parallel optimization techniques to manage large data sets and intricate models is the focus of this research area.
Exploring novel optimization algorithms	The goal of research is to create novel optimization methods, like evolutionary algorithms, that are influenced by social and biological systems.
Incorporating domain knowledge and priors	In order to direct the search, future work will concentrate on incorporating domain-specific constraints and expert knowledge into optimization algorithms.
Developing adaptive and self-tuning techniques	This field of study focuses on developing optimization methods that can independently modify their parameters in response to input.
Enhancing interpretability and explainability	In order to foster understanding and trust, researchers work to increase the transparency and interpretability of optimization results.
Optimizing hyperparameters	This field focuses on creating effective techniques for enhancing performance by fine-tuning the hyperparameters of ML models.
Handling noisy and incomplete data	The main goal of research is to create reliable optimization strategies that can handle missing or noisy data without sacrificing performance.
Dealing with concept drift	Subsequent research endeavours to formulate optimization techniques that possess the ability to adjust to evolving data distributions and model assumptions.
Addressing ethical considerations	The goal of research is to create optimization strategies that take privacy, bias, and fairness into account, among other ethical considerations.
Leveraging meta-learning and transfer learning	Addressing the integration of meta-learning and transfer learning approaches into optimization processes is the focus of this line of inquiry.
Exploring quantum-inspired optimization techniques	By utilizing quantum phenomena, researchers hope to explore optimization methods motivated by ideas from quantum computing.
Developing robust optimization against adversarial attacks	This field focuses on developing optimization techniques that are resistant to adversarial attacks, guaranteeing the security and resilience of the model.
Addressing bias and fairness in optimization	The goal of future research is to provide optimization strategies that guarantee fairness and address bias in ML judgments and models.
Incorporating uncertainty modeling into optimization	In order to take uncertainty in data and models into account, researchers are attempting to include methodologies for uncertainty modeling into optimization algorithms.
Optimizing resource allocation in distributed systems	Creating optimization strategies for effective resource allocation in remote computing settings is the focus of this research area.
Incorporating privacy-preserving techniques	The goal of research is to create optimization techniques that put data privacy and confidentiality first, guaranteeing adherence to privacy laws and safeguarding private data.
Exploring neuro-symbolic optimization techniques	This field focuses on combining neural networks and symbolic reasoning in optimization procedures, utilizing the advantages of both paradigms for improved performance.
Developing energy-efficient optimization algorithms	In order to support sustainable ML practices, future work will focus on designing optimization algorithms that reduce energy usage during model training and inference.
Investigating reinforcement learning for optimization	The goal of research is to investigate how optimization tasks can be applied with reinforcement learning approaches, utilizing feedback and rewards to direct the search for the best answers.
Addressing interpretability in optimization	This line of inquiry aims to improve the interpretability of optimization techniques so that interested parties may comprehend and have confidence in the way optimization algorithms make decisions.
Handling imbalanced datasets	In order to increase model performance, researchers are working to build optimization algorithms that can handle imbalanced datasets, which are datasets with uneven class distributions.
Exploring federated learning optimization techniques	The goal of this field is to create optimization techniques specifically for federated learning environments, in which data is dispersed among several devices or locations.
Incorporating domain-specific constraints	Subsequent research endeavours to include domain-specific limitations, such equitable constraints or physical laws, into optimization algorithms to guarantee adherence to domain prerequisites.
Addressing catastrophic forgetting in continual learning	The goal of research is to create optimization strategies that can lessen catastrophic forgetting, which occurs when a model is trained on fresh data and then forgets what it has previously learned.
Optimizing for interpretability and decision-making	This field of research focuses on optimizing ML models for interpretability and decision-making in practical applications, in addition to their prediction accuracy.

Table 5 (continued)

Future Research Issue	Description
Exploring differentiable programming for optimization	The goal of this field is to identify optimization strategies that work well with end-to-end differentiable model architectures by utilizing differentiable programming paradigms.
Developing self-supervised learning optimization	In the future, research will focus on optimization strategies for self-supervised learning, in which models are trained to forecast specific features of the input data without the need for direct supervision.
Investigating optimization techniques for automated ML	The goal of research is to provide optimization techniques specifically for automated ML systems, which automatically choose and train models and hyperparameters.
Optimizing for adversarial robustness	This field of analysis focuses on creating optimization strategies that make ML models more resilient to hostile attacks, guaranteeing the security and dependability of the models.
Addressing uncertainty quantification in optimization	In order to measure and spread uncertainty throughout the optimization process, future research will integrate uncertainty quantification approaches into optimization algorithms.
Optimizing for real-time and low-latency applications	To ensure quick decision-making and response times, researchers are working to create optimization strategies that are suited for real-time and low-latency applications.
Addressing sample inefficiency in reinforcement learning	This field focuses on creating optimization techniques that increase reinforcement learning systems' sample efficiency, allowing for quicker learning with less samples.
Exploring multi-objective optimization techniques	Future research attempts to look at optimization techniques that can optimize several competing goals at once, enabling more adaptable and flexible model behaviour.
Incorporating human preferences into optimization	In order to ensure alignment with user preferences and values, researchers are working to develop optimization approaches that take human preferences and input into account during the optimization process.
Optimizing for heterogeneous and multimodal data	In order to enable models to learn from a variety of data sources and modalities, this research direction entails developing optimization approaches capable of managing heterogeneous and multimodal data successfully.
Scalable optimization for edge computing	In order to optimize models for deployment on devices with limited resources at the network edge, researchers are working to develop optimization approaches specifically suited for edge computing environments.
Adaptive optimization for non-stationary environments	The goal of this field is to create optimization techniques that can adjust to non-stationary settings, where goals and data distributions fluctuate over time.
Robust optimization against model perturbations	The goal of future research is to create optimization strategies that improve ML models' resistance to changes in input data, model parameters, or architectures.
Addressing ethical considerations in optimization	In order to ensure the ethical and responsible usage of ML models, researchers are working to build optimization techniques that take into account ethical aspects including accountability, privacy, and fairness.
Optimizing for resource-constrained environments	Creating optimization strategies that maximize model performance under resource limitations—like memory, processing, or energy limitations—is the focus of this line of inquiry.
Exploring optimization techniques for graph data	In order to facilitate effective learning and inference on graph-based representations, future research will look into optimization techniques designed specifically for graph-structured data.
Integrating uncertainty estimation into optimization	In order to enable models to measure and propagate uncertainty throughout the optimization process, researchers are working to develop optimization strategies that incorporate uncertainty estimating methodologies.
Addressing ethical considerations in optimization	In order to ensure the ethical and responsible usage of ML models, researchers are working to build optimization techniques that take into account ethical aspects including accountability, privacy, and fairness.
Optimizing for resource-constrained environments	Creating optimization strategies that maximize model performance under resource limitations—like memory, processing, or energy limitations—is the focus of this line of inquiry.
Exploring optimization techniques for graph data	In order to facilitate effective learning and inference on graph-based representations, future research will look into optimization techniques designed specifically for graph-structured data.

parameters, posing difficulties in situations when data is limited or costly to get. Data efficiency deals to the capacity of optimization algorithms to learn well from a little amount of data samples while maintaining high performance levels. Transfer learning, meta-learning, and active learning are methods used to enhance data efficiency by using knowledge from similar activities, adjusting to new tasks with less data, and selecting obtaining valuable data points during training [149].

7) **Heterogeneity and Noise**: Real-world datasets can be varied and noisy, consisting of a variety of types, sources, and quality levels. Interacting with various data types and distributions poses issues when creating optimization algorithms due to heterogeneity. Data noise may conceal genuine patterns and correlations, resulting in subpar model performance. Utilizing robust optimization techniques, robust loss functions, and data pretreatment approaches is crucial for reducing the influence of heterogeneity and noise on optimization solutions [150].
8) **Interpretability and Explainability**: As ML models are becoming increasingly used in important fields like healthcare and finance, the simplicity and comprehensibility of optimization results become crucial. Modern optimization methods may provide precise models but lack clarity in their decision-making process [151]. Comprehensible optimization algorithms and model frameworks enable stakeholders to comprehend and have confidence in model predictions, which aids in model deployment and decision-making in real-world situations.
9) **Ethical and Fair Optimization**: Optimization methods need to include ethical considerations and fairness issues in ML applications. Biases in data or optimization algorithms may result in unjust treatment of individuals or communities, which can perpetuate social imbalances. Fair optimization strategies strive to reduce biases and achieve fair results across various demographic groups, emphasizing justice, transparency, and responsibility in ML systems [152].
10) **Dynamic Environments**: Optimization strategies need to adapt and develop in circumstances where data distributions, aims, or limitations vary over time [153]. Continuous learning, online optimization, and adaptive algorithms allow models to adapt and remain effective and relevant in dynamic environments by learning and adjusting to changing issues.

A methodical overview of the primary challenges in ML optimization is provided in Table 4, along with optimization techniques that are aimed to solve these difficulties. It provides a methodical framework for comprehending the main challenges encountered in ML model optimization and the associated strategies used to overcome these difficulties. This table's objective is to make it easier for readers to comprehend the wide range of difficulties that come with improving ML algorithms as well as the numerous optimization techniques that may be used to address these difficulties. Researchers, practitioners, and enthusiasts in the area of ML and optimization may benefit greatly from the table's classification of issues and associated techniques.

5 Future Research Directions

Future prospects for ML optimization research include a broad spectrum of areas with the objective of resolving present challenges and expanding the potential of optimization methods. Below are some essential recommendations:

An overview of the emerging approaches for ML optimization research is presented in Table 5. In order overcome problems, maximize the use of opportunities, and improve the state-of-the-art in ML model optimization for many applications and domains, it identifies important areas where more study needs to be conducted.

6 Conclusion

In summary, this paper explored a variety of aspects associated with ML optimization, from previous developments to present difficulties and future potential areas for research. The importance of optimization in ML and its critical role in training models to achieve high accuracy, efficiency, and generalization capabilities were one of the initial issues we discussed. We surveyed optimization strategies and classified them according to goals including speed, accuracy, and generalization. The result provided us knowledge about widely used algorithms and how they are used in various ML scenarios. Based on a review of the literature, we were able to identify important publications and contributions by well-known scholars that provided knowledge about algorithmic developments, theoretical underpinnings, and real-world optimization applications. We identified typical optimization obstacles, such as the non-convex nature of objective functions, convergence problems, scalability issues, and resilience, generalization, and computational complexity concerns. We concluded by presenting several of proposals for potential fields of research in ML optimization. These involve studying novel optimization algorithms influenced by biological and social systems, integrating domain knowledge and priors into optimization processes, creating adaptive and self-tuning optimization techniques, addressing scalability challenges through parallel and distributed optimization, and improving the interpretability and explainability of optimization results. In the final analysis, the research and invention reported in this work emphasize the significance of continuous research and development in ML optimization. Researchers may progress in the field and develop more effective, reliable, and accessible ML models that can successfully handle challenging real-world situations by tackling present issues and investigating novel optimization paths.

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

References

1. Adams R (2013) Active Queue Management: a Survey. IEEE Commun Surv Tutorials 15(3):1425–1476. https://doi.org/10.1109/SURV.2012.082212.00018
2. Alsheikh M, Abu S, Lin D, Niyato, and Hwee Pink Tan (2014) Machine learning in Wireless Sensor networks: algorithms, strategies, and applications. IEEE Commun Surv Tutorials 16(4):1996–2018. https://doi.org/10.1109/COMST.2014.2320099
3. Anurag A, Priyadarshi R, Goel A, Gupta B (2020) 2-D Coverage Optimization in WSN Using a Novel Variant of Particle Swarm Optimisation. In 2020 7th International Conference on Signal Processing and Integrated Networks, SPIN 2020, 663–68. https://doi.org/10.1109/SPIN48934.2020.9070978
4. Badarla V, Siva Ram Murthy C (2010) A novel learning based solution for Efficient Data Transport in Heterogeneous Wireless Networks. Wireless Netw 16(6):1777–1798. https://doi.org/10.1007/s11276-009-0228-4
5. Priyadarshi R, Gupta B, and Amulya Anurag (2020) Wireless Sensor Networks Deployment: a result oriented analysis. Wireless Pers Commun 113(2):843–866. https://doi.org/10.1007/s11277-020-07255-9
6. Auld T, Moore AW, Gull SF (2007) Bayesian neural networks for internet traffic classification. IEEE Trans Neural Networks 18(1):223–239. https://doi.org/10.1109/TNN.2006.883010
7. Priyadarshi R, Gupta B, and Amulya Anurag (2020) Deployment techniques in Wireless Sensor networks: a Survey, classification, challenges, and Future Research Issues. J Supercomputing 76(9):7333–7373. https://doi.org/10.1007/s11227-020-03166-5
8. Priyadarshi R (2021) and Ravi Ranjan Kumar. An Energy-Efficient LEACH Routing Protocol for Wireless Sensor Networks. In Lecture Notes in Electrical Engineering, edited by Vijay Nath and J K Mandal, 673:423–30. Singapore: Springer Singapore. https://doi.org/10.1007/978-981-15-5546-6_35
9. Ayoubi S, Limam N, Salahuddin MA, Shahriar N, Boutaba R, Estrada-Solano F, Caicedo OM (2018) Machine Learning for Cognitive Network Management. IEEE Commun Mag 56(1):158–165. https://doi.org/10.1109/MCOM.2018.1700560
10. Priyadarshi R, Nath V (2019) A Novel Diamond–Hexagon Search Algorithm for Motion Estimation. Microsyst Technol 25(12):4587–4591. https://doi.org/10.1007/s00542-019-04376-5
11. Rosenblatt F (1960) Perceptron simulation experiments. Proceedings of the IRE 48.3:301–309
12. Werbos PJ (1994) The roots of backpropagation: from ordered derivatives to neural networks and political forecasting, vol 1. Wiley
13. Nouretdinov I et al (2011) Machine learning classification with confidence: application of transductive conformal predictors to MRI-based diagnostic and prognostic markers in depression. NeuroImage 56(2):809–813
14. Vandenberghe L, Boyd S (1996) Semidefinite programming. SIAM Rev 38(1):49–95
15. LeCun Y, Bengio Y, Hinton G (2015) Deep Learn Nat 521:436–444
16. Priyadarshi R, Rana H, Srivastava A, Nath V (2023) A Novel Approach for Sink Route in Wireless Sensor Network. In Lecture Notes in Electrical Engineering, edited by Vijay Nath and Jyotsna Kumar Mandal, 887:695–703. Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-19-1906-0_58
17. Bkassiny M, Li Y, Jayaweera SK (2013) A Survey on Machine-Learning techniques in Cognitive Radios. IEEE Commun Surv Tutorials 15(3):1136–1159. https://doi.org/10.1109/SURV.2012.100412.00017
18. Qiu Y, Ma L, and Rahul Priyadarshi (2024) Deep Learning challenges and prospects in Wireless Sensor Network Deployment. Arch Comput Methods Eng. https://doi.org/10.1007/s11831-024-10079-6
19. Chabaa S, Zeroual A, and Jilali Antari (2010) Identification and prediction of internet traffic using Artificial neural networks. J Intell Learn Syst Appl 02(03):147–155. https://doi.org/10.4236/jilsa.2010.23018
20. Chang C, Chung, Chih Jen Lin (2011) LIBSVM: a Library for Support Vector machines. ACM Trans Intell Syst Technol 2(3). https://doi.org/10.1145/1961189.1961199
21. Claeys M, Latre S, Famaey J, and Filip De Turck (2014) Design and evaluation of a self-learning http adaptive video streaming client. IEEE Commun Lett 18(4):716–719. https://doi.org/10.1109/LCOMM.2014.020414.132649
22. Claeys M, Latré S, Famaey J, Wu T, Van Leekwijck W, and Filip De Turck (2014) Design and optimisation of a (FA)Q-Learning-based HTTP adaptive streaming client. Connection Sci 26(1):25–43. https://doi.org/10.1080/09540091.2014.885273
23. Randheer SK, Soni S, Kumar, and Rahul Priyadarshi (2020). Energy-Aware Clustering in Wireless Sensor Networks BT - Nanoelectronics, Circuits and Communication Systems. In, edited by Vijay Nath and J K, Mandal 453–61. Singapore: Springer Singapore
24. Dowling J, Curran E, Cunningham R, and Vinny Cahill (2005) Using feedback in collaborative reinforcement learning to adaptively optimize MANET Routing. IEEE Trans Syst Man Cybernetics Part A:Systems Hum 35(3):360–372. https://doi.org/10.1109/TSMCA.2005.846390
25. Priyadarshi R, Gupta B (2023) 2-D Coverage optimization in obstacle-based FOI in WSN using modified PSO. J Supercomputing 79(5):4847–4869. https://doi.org/10.1007/s11227-022-04832-6
26. Edalat Y, Ahn JS, and Katia Obraczka (2016) Smart experts for Network State Estimation. IEEE Trans Netw Serv Manage 13(3):622–635. https://doi.org/10.1109/TNSM.2016.2586506
27. Este A, Gringoli F, and Luca Salgarelli (2009) Support Vector machines for TCP Traffic classification. Comput Netw 53(14):2476–2490. https://doi.org/10.1016/j.comnet.2009.05.003
28. Rawat P, Chauhan S, Priyadarshi R (2021) A novel heterogeneous clustering protocol for lifetime maximization of Wireless Sensor Network. Wireless Pers Commun 117(2):825–841. https://doi.org/10.1007/s11277-020-07898-8
29. García-Teodoro P, Díaz-Verdejo J, Maciá-Fernández G, and E. Vázquez (2009) Anomaly-based network intrusion detection: techniques, systems and challenges. Computers Secur 28(1–2):18–28. https://doi.org/10.1016/j.cose.2008.08.003

30. Priyadarshi R, and Bharat Gupta (2021) Area Coverage optimization in three-Dimensional Wireless Sensor Network. Wireless Pers Commun 117(2):843–865. https://doi.org/10.1007/s11277-020-07899-7
31. Yin, F., Lin, Z., Kong, Q., Xu, Y., Li, D., Theodoridis, S.,… Cui, S. R. (2020). FedLoc:Federated Learning Framework for Data-Driven Cooperative Localization and Location Data Processing. IEEE Open Journal of Signal Processing, 1:187–215. https://doi.org/10.1109/OJSP.2020.3036276
32. Yin F, Fritsche C, Jin D, Gustafsson F, Zoubir AM (2015) Cooperative localization in WSNs using Gaussian Mixture modeling: distributed ECM algorithms. IEEE Trans Signal Process 63(6):1448–1463. https://doi.org/10.1109/TSP.2015.2394300
33. Xu G, Zhang Q, Song Z, Ai B (2023) Relay-assisted Deep Space Optical Communication System over coronal fading channels. IEEE Trans Aerosp Electron Syst 59(6):8297–8312. https://doi.org/10.1109/TAES.2023.3301463
34. Yan, A., Li, Z., Gao, Z., Zhang, J., Huang, Z., Ni, T.,… Wen, X. (2024). MURLAV: A Multiple-Node-Upset Recovery Latch and Algorithm-Based Verification Method. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. https://doi.org/10.1109/TCAD.2024.3357593
35. Yan, A., Cao, A., Huang, Z., Cui, J., Ni, T., Girard, P.,… Zhang, J. (2023). Two Double-Node-Upset-Hardened Flip-Flop Designs for High-Performance Applications. IEEE Transactions on Emerging Topics in Computing, 11(4):1070–1081. https://doi.org/10.1109/TETC.2023.3317070
36. Dai M, Luo L, Ren J, Yu H, Sun G (2022) PSACCF: prioritized online slice Admission Control considering Fairness in 5G/B5G networks. IEEE Trans Netw Sci Eng 9(6):4101–4114. https://doi.org/10.1109/TNSE.2022.3195862
37. Sun G, Xu Z, Yu H, Chang V (2021) Dynamic network function provisioning to Enable Network in Box for Industrial Applications. IEEE Trans Industr Inf 17(10):7155–7164. https://doi.org/10.1109/TII.2020.3042872
38. Sun, G., Zhu, G., Liao, D., Yu, H., Du, X.,… Guizani, M. (2019). Cost-Efficient Service Function Chain Orchestration for Low-Latency Applications in NFV Networks. IEEE Systems Journal, 13(4):3877–3888. https://doi.org/10.1109/JSYST.2018.2879883
39. Ma X, Dong Z, Quan W, Dong Y, Tan Y (2023) Real-time assessment of asphalt pavement moduli and traffic loads using monitoring data from Built-in Sensors: Optimal sensor placement and identification algorithm.Mech Syst Signal Process 187:109930. https://doi.org/10.1016/j.ymssp.2022.109930
40. Qu J, Mao B, Li Z, Xu Y, Zhou K, Cao X, Wang X (2023) Recent progress in Advanced Tactile Sensing technologies for Soft Grippers. Adv Funct Mater 33(41):2306249. https://doi.org/10.1002/adfm.202306249
41. Priyadarshi R, Bhardwaj P, Gupta P, and Vijay Nath (2023) Utilization of smartphone-based Wireless sensors in Agricultural Science: a state of art. Lecture Notes Electr Eng 887:681–688. https://doi.org/10.1007/978-981-19-1906-0_56
42. Li R, Peng B (2022) Implementing Monocular Visual-Tactile sensors for Robust Manipulation. Cyborg Bionic Syst 2022. https://doi.org/10.34133/2022/9797562
43. Aibin Y, Feng X, Zhao X, Zhou H, Cui J, Ying Z, Girard P, Wen X HITTSFL: Design of a Cost-Effective HIS-Insensitive TNU-Tolerant and SET-Filtering Latch for Safety-Critical Applications, IEEE/ACM Design Automation Conference (DAC2020), Oral, pp. 1–6, 2020/7/19–23, San Francisco, USA
44. J., X., S., H. P., X., Z., & J., H. (2022) The improvement of Road Driving Safety guided by visual Inattentional blindness. IEEE Trans Intell Transp Syst, 23(6):4972–4981. https://doi.org/10.1109/TITS.2020.3044927
45. Priyadarshi R, and Bharat Gupta (2020) Coverage Area Enhancement in Wireless Sensor Network. Microsyst Technol 26(5):1417–1426. https://doi.org/10.1007/s00542-019-04674-y
46. Dai X, Xiao Z, Jiang H, Alazab M, Lui JCS, Dustdar S, Liu J (2023) Task Co-offloading for D2D-Assisted Mobile Edge Computing in Industrial Internet of things. IEEE Trans Industr Inf 19(1):480–490. https://doi.org/10.1109/TII.2022.3158974
47. Jiang H, Dai X, Xiao Z, Iyengar AK (2022) Joint Task Offloading and Resource Allocation for Energy-Constrained Mobile Edge Computing. IEEE Trans Mob Comput. https://doi.org/10.1109/TMC.2022.3150432
48. Dai X, Xiao Z, Jiang H, Lui JCS (2023) UAV-Assisted Task Offloading in Vehicular Edge Computing Networks. IEEE Trans Mob Comput. https://doi.org/10.1109/TMC.2023.3259394
49. Sun L, Liang J, Zhang C, Wu D, Zhang Y (2023) Meta-transfer Metric Learning for Time Series classification in 6G-Supported Intelligent Transportation systems. IEEE Trans Intell Transp Syst. https://doi.org/10.1109/TITS.2023.3250962
50. Mao Y, Sun R, Wang J, Cheng Q, Kiong L, Ochieng C, Y. W (2022) New time-differenced carrier phase approach to GNSS/INS integration. GPS Solutions 26(4):122. https://doi.org/10.1007/s10291-022-01314-3
51. Mao Y, Zhu Y, Tang Z, Chen Z (2022) A Novel Airspace Planning Algorithm for Cooperative Target localization. Electronics 11(18):2950. https://doi.org/10.3390/electronics11182950
52. Xie Y, Wang X, Shen Z, Sheng Y, Wu G (2023) A two-stage estimation of distribution Algorithm with Heuristics for Energy-Aware Cloud Workflow Scheduling. IEEE Trans Serv Comput 16(6):4183–4197. https://doi.org/10.1109/TSC.2023.3311785
53. Shang M, Luo J (2021) The Tapio Decoupling Principle and Key strategies for changing factors of Chinese urban Carbon Footprint based on Cloud Computing. Int J Environ Res Public Health 18(4):2101. https://doi.org/10.3390/ijerph18042101
54. Luo J, Zhao C, Chen Q, Li G (2022) Using deep belief network to construct the agricultural information system based on internet of things. J Supercomputing 78(1):379–405. https://doi.org/10.1007/s11227-021-03898-y
55. Cao B, Zhao J, Yang P, Gu Y, Muhammad K, Rodrigues J, C P J, V de Albuquerque, C H (2020) Multiobjective 3-D Topology Optimization of Next-Generation Wireless Data Center Network. IEEE Trans Industr Inf 16(5):3597–3605. https://doi.org/10.1109/TII.2019.2952565
56. Yu J, Lu L, Chen Y, Zhu Y, Kong L (2021) An indirect eavesdropping attack of keystrokes on Touch screen through Acoustic Sensing. IEEE Trans Mob Comput 20(2):337–351. https://doi.org/10.1109/TMC.2019.2947468
57. Li K, Ji L, Yang S, Li H, Liao X (2022) Couple-Group Consensus of Cooperative–competitive heterogeneous Multiagent systems: a fully distributed event-triggered and Pinning Control Method. IEEE Trans Cybernetics 52(6):4907–4915. https://doi.org/10.1109/TCYB.2020.3024551
58. Min H, Lei X, Wu X, Fang Y, Chen S, Wang W, Zhao X (2024) Toward interpretable anomaly detection for autonomous vehicles with denoising variational transformer. Eng Appl Artif Intell 129:107601. https://doi.org/10.1016/j.engappai.2023.107601
59. Hou X, Zhang L, Su Y, Gao G, Liu Y, Na Z, Chen T (2023) A space crawling robotic bio-paw (SCRBP) enabled by triboelectric sensors for surface identification. Nano Energy 105:108013. https://doi.org/10.1016/j.nanoen.2022.108013
60. Hou X, Xin L, Fu Y, Na Z, Gao G, Liu Y, Chen T (2023) A self-powered biomimetic mouse whisker sensor (BMWS) aiming at terrestrial and space objects perception. Nano Energy 118:109034. https://doi.org/10.1016/j.nanoen.2023.109034
61. Liang X, Chen Z, Deng Y, Liu D, Liu X, Huang Q, Arai T (2023) Field-controlled microrobots fabricated by Photopolymerization. Cyborg Bionic Syst 4:9. https://doi.org/10.34133/cbsystems.0009

62. Ma S, Chen Y, Yang S, Liu S, Tang L, Li B, Li Y (2023) The Autonomous Pipeline Navigation of a Cockroach Bio-robot with enhanced walking stimuli. Cyborg Bionic Syst 4:67. https://doi.org/10.34133/cbsystems.0067
63. Cai Z, Zhu X, Gergondet P, Chen X, Yu Z (2023) A friction-driven strategy for Agile Steering Wheel Manipulation by Humanoid Robots. Cyborg Bionic Syst 4:64. https://doi.org/10.34133/cbsystems.0064
64. Li X, Sun Y (2021) Application of RBF neural network optimal segmentation algorithm in credit rating. Neural Comput Appl 33(14):8227–8235. https://doi.org/10.1007/s00521-020-04958-9
65. Long X, Mao M, Su T, Su Y, Tian M (2023) Machine learning method to predict dynamic compressive response of concrete-like material at high strain rates. Def Technol 23:100–111. https://doi.org/10.1016/j.dt.2022.02.003
66. Long X, Lu C, Su Y, Dai Y (2023) Machine learning framework for predicting the low cycle fatigue life of lead-free solders. Eng Fail Anal 148:107228. https://doi.org/10.1016/j.engfailanal.2023.107228
67. Hu J, Wu Y, Li T, Ghosh BK (2019) Consensus Control of General Linear Multiagent Systems with Antagonistic Interactions and communication noises. IEEE Trans Autom Control 64(5):2122–2127. https://doi.org/10.1109/TAC.2018.2872197
68. Chen B, Hu J, Zhao Y, Ghosh BK (2022) Finite-Time velocity-free Rendezvous Control of multiple AUV Systems with Intermittent Communication. IEEE Trans Syst Man Cybernetics: Syst 52(10):6618–6629. https://doi.org/10.1109/TSMC.2022.3148295
69. Bo C, Jiangping H, Bijoy G (2023) Finite-Time Observer Based Tracking Control of Heterogeneous Multi-AUV Systems with Partial Measurements and Intermittent Communication. Science China Information Sciences. https://doi.org/10.1007/s11432-023-3903-6
70. Jiang Y, Li X (2022) Broadband cancellation method in an adaptive co-site interference cancellation system. Int J Electron 109(5):854–874. https://doi.org/10.1080/00207217.2021.1941295
71. Zhang, X., Deng, H., Xiong, Z., Liu, Y., Rao, Y., Lyu, Y.,… Li, Y. (2024). Secure Routing Strategy Based on Attribute-Based Trust Access Control in Social-Aware Networks.Journal of Signal Processing Systems. https://doi.org/10.1007/s11265-023-01908-1
72. Lyu T, Xu H, Zhang L, Han Z (2024) Source selection and resource allocation in Wireless-Powered Relay networks: an adaptive dynamic programming-based Approach. IEEE Internet Things J 11(5):8973–8988. https://doi.org/10.1109/JIOT.2023.3321673
73. Liu G (April 2021) Data Collection in MI-Assisted Wireless Powered Underground Sensor networks: directions, recent advances, and challenges. IEEE Commun Mag 59(4):132–138. https://doi.org/10.1109/MCOM.001.2000921
74. Zhao L, Qu S, Xu H, Wei Z, Zhang C (2024) Energy-efficient trajectory design for secure SWIPT systems assisted by UAV-IRS. Veh Commun 45:100725. https://doi.org/10.1016/j.vehcom.2023.100725
75. Hou M, Zhao Y, Ge X (2017) Optimal scheduling of the plug-in electric vehicles aggregator energy and regulation services based on grid to vehicle. Int Trans Electr Energy Syst 27(6):e2364. https://doi.org/10.1002/etep.2364
76. Lei Y, Yanrong C, Hai T, Ren G, Wenhuan W (2023) DGNet: an adaptive lightweight defect detection model for New Energy Vehicle Battery Current Collector. IEEE Sens J 23(23):29815–29830. https://doi.org/10.1109/JSEN.2023.3324441
77. Xu Y, Wang E, Yang Y, Chang Y (2022) A unified collaborative representation learning for neural-network based Recommender systems. IEEE Trans Knowl Data Eng 34(11):5126–5139. https://doi.org/10.1109/TKDE.2021.3054782
78. Liu X, Lou S, Dai W (2023) Further results on System identification of nonlinear state-space models. Automatica 148:110760. https://doi.org/10.1016/j.automatica.2022.110760
79. Wang Q, Dai W, Zhang C, Zhu J, Ma X (2023) A Compact Constraint Incremental Method for Random Weight Networks and its application. IEEE transactions on neural networks and Learning systems. https://doi.org/10.1109/TNNLS.2023.3289798
80. Zhang, H., Mi, Y., Liu, X., Zhang, Y., Wang, J.,… Tan, J. (2023). A differential game approach for real-time security defense decision in scale-free networks. Computer Networks, 224, 109635. https://doi.org/10.1016/j.comnet.2023.109635
81. Cao K, Ding H, Li W, Lv L, Gao M, Gong F, Wang B (2022) On the Ergodic Secrecy Capacity of Intelligent reflecting surface aided Wireless Powered Communication systems. IEEE Wirel Commun Lett PP(1). https://doi.org/10.1109/LWC.2022.3199593
82. Cheng, B., Wang, M., Zhao, S., Zhai, Z., Zhu, D.,… Chen, J. (2017). Situation-Aware Dynamic Service Coordination in an IoT Environment. IEEE/ACM Transactions on Networking,25(4), 2082–2095. https://doi.org/10.1109/TNET.2017.2705239
83. Zheng, W., Lu, S., Cai, Z., Wang, R., Wang, L.,… Yin, L. (2023). PAL-BERT: An Improved Question Answering Model. Computer Modeling in Engineering & Sciences. https://doi.org/10.32604/cmes.2023.046692
84. Cao B, Li Z, Liu X, Lv Z, He H (2023) Mobility-aware Multiobjective Task Offloading for Vehicular Edge Computing in Digital Twin Environment. IEEE J Sel Areas Commun 41(10):3046–3055. https://doi.org/10.1109/JSAC.2023.3310100
85. Geurts P, Ernst D, and Louis Wehenkel (2006) Extremely randomized trees. Mach Learn 63(1):3–42. https://doi.org/10.1007/s10994-006-6226-1
86. Giacinto G, Perdisci R, Rio MD, and Fabio Roli (2008) Intrusion detection in computer networks by a modular ensemble of one-class classifiers. Inform Fusion 9(1):69–82. https://doi.org/10.1016/j.inffus.2006.10.002
87. Goldberger AS (2004) Econometric Computing by Hand. J Econ Soc Meas 29(1–3):115–117. https://doi.org/10.3233/jem-2004-0213
88. Ha S, Rhee I, Xu L (2008) CUBIC: a new TCP-Friendly high-speed TCP variant. Operating Syst Rev (ACM) 42(5):64–74. https://doi.org/10.1145/1400097.1400105
89. Hajji H (2005) Statistical Analysis of Network Traffic for Adaptive Faults Detection. IEEE Trans Neural Networks 16(5):1053–1063. https://doi.org/10.1109/TNN.2005.853414
90. Hariri B, Sadati N (2007) NN-RED: an AQM mechanism based on neural networks. Electron Lett 43(19):1053–1055. https://doi.org/10.1049/el:20071791
91. Hu T, and Yunsi Fei (2010) QELAR: a machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater Sensor Networks. IEEE Trans Mob Comput 9(6):796–809. https://doi.org/10.1109/TMC.2010.28
92. Hu W, Wei Hu, and Steve Maybank (2008) AdaBoost-Based algorithm for Network Intrusion Detection. IEEE Trans Syst Man Cybernetics Part B: Cybernetics 38(2):577–583. https://doi.org/10.1109/TSMCB.2007.914695
93. Jain V, Randheer R, Priyadarshi, and Ankush Thakur (2019) Performance analysis of Block Matching algorithms. Lecture Notes Electr Eng 556:73–82 Springer Singapore. https://doi.org/10.1007/978-981-13-7091-5_7
94. Jayaraj A, Venkatesh T, Siva Ram C Murthy (2008) Loss classification in Optical Burst switching networks using machine learning techniques: improving the performance of TCP. IEEE J Sel Areas Commun 26(6):45–54. https://doi.org/10.1109/JSACOCN.2008.033508
95. Khanafer RM, Solana B, Triola J, Barco R, Moltsen L, Altman Z, Lázaro P (2008) Automated diagnosis for UMTS Networks

using bayesian Network Approach. IEEE Trans Veh Technol 57(4):2451–2461. https://doi.org/10.1109/TVT.2007.912610

96. Kiciman E, and Armando Fox (2005) Detecting application-level failures in component-based internet services. IEEE Trans Neural Networks 16(5):1027–1041. https://doi.org/10.1109/TNN.2005.853411
97. Klaine P, Valente MA, Imran O, Onireti, and Richard Demo Souza (2017) A Survey of Machine Learning techniques Applied to Self-Organizing Cellular Networks. IEEE Commun Surv Tutorials 19(4):2392–2431. https://doi.org/10.1109/COMST.2017.2727878
98. Kumar S, Soni SK, Randheer, and Rahul Priyadarshi (2020) Performance Analysis of Novel Energy Aware Routing in Wireless Sensor Network. Lecture Notes Electr Eng 642:503–511 Springer Singapore. https://doi.org/10.1007/978-981-15-2854-5_44
99. Kumar S, Soni SK, Randheer (2020) and Rahul Priyadarshi. Performance Analysis of Novel Energy Aware Routing in Wireless Sensor Network. In Lecture Notes in Electrical Engineering, edited by Vijay Nath and J K Mandal, 642:503–11. Singapore: Springer Singapore. https://doi.org/10.1007/978-981-15-2854-5_44
100. Lemaître G, Nogueira F, Aridas CK (2017) Imbalanced-Learn: a Python Toolbox to tackle the curse of Imbalanced datasets in Machine Learning. J Mach Learn Res 18:1–5
101. Mirza M, Sommers J, Barford P, Zhu X (2010) A Machine Learning Approach to TCP Throughput Prediction. IEEE/ACM Trans Networking 18(4):1026–1039. https://doi.org/10.1109/TNET.2009.2037812
102. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533. https://doi.org/10.1038/nature14236
103. Priyadarshi R, Yadav S, and Deepika Bilyan (2019) Performance and comparison analysis of MIEEP Routing Protocol over adapted LEACH Protocol. Smart Comput Strategies: Theoretical Practical Aspects 237–245. https://doi.org/10.1007/978-981-13-6295-8_20
104. Moustapha AI, and Rastko R. Selmic (2008) Wireless Sensor Network modeling using modified recurrent neural networks: application to Fault Detection. IEEE Trans Instrum Meas 57(5):981–988. https://doi.org/10.1109/TIM.2007.913803
105. Muniyandi A, Prabakar R, Rajeswari, Rajaram R (2012) Network Anomaly detection by cascading K-Means clustering and C4.5 decision Tree Algorithm. Procedia Eng 30:174–182. https://doi.org/10.1016/j.proeng.2012.01.849
106. Nguyen TTT, Armitage G, Philip Branch, and Sebastian Zander (2012) Timely and continuous machine-learning-based classification for interactive IP traffic. IEEE/ACM Trans Networking 20(6):1880–1894. https://doi.org/10.1109/tnet.2012.2187305
107. Nguyen TTT, and Grenville Armitage (2008) A survey of techniques for internet traffic classification using machine learning. IEEE Commun Surv Tutorials 10(4):56–76. https://doi.org/10.1109/SURV.2008.080406
108. Nichols K, and Van Jacobson (2012) Controlling Queue Delay. Queue 10(5):20–34. https://doi.org/10.1145/2208917.2209336
109. Nunes BA, Arouche K, Veenstra W, Ballenthin S, Lukin, Obraczka K (2014) A Machine Learning Framework for TCP Round-Trip Time Estimation. Eurasip Journal on Wireless Communications and Networking 2014. https://doi.org/10.1186/1687-1499-2014-47
110. Panda M, Abraham A, and Manas Ranjan Patra (2012) A hybrid Intelligent Approach for Network Intrusion Detection. Procedia Eng 30:1–9. https://doi.org/10.1016/j.proeng.2012.01.827
111. Pandey A, Kumar D, Priyadarshi R (2023) and Vijay Nath. Development of Smart Village for Better Lifestyle of Farmers by Crop and Health Monitoring System. In Lecture Notes in Electrical Engineering, edited by Vijay Nath and Jyotsna Kumar Mandal, 887:689–94. Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-19-1906-0_57
112. Pandey A, Kumar D, Priyadarshi R, and Vijay Nath (2023) Development of Smart Village for Better Lifestyle of Farmers by Crop and Health Monitoring System. Lecture Notes Electr Eng 887:689–694. https://doi.org/10.1007/978-981-19-1906-0_57. Springer Nature Singapore Singapore
113. Peddabachigari S, Abraham A, Grosan C, and Johnson Thomas (2007) Modeling intrusion detection system using hybrid Intelligent systems. J Netw Comput Appl 30(1):114–132. https://doi.org/10.1016/j.jnca.2005.06.003
114. Pinson MH, Wolf S (2004) A new standardized method for objectively measuring Video Quality. IEEE Trans Broadcast 50(3):312–322. https://doi.org/10.1109/TBC.2004.834028
115. Priyadarshi R, Rawat P, and Vijay Nath (2019) Energy dependent cluster formation in heterogeneous Wireless Sensor Network. Microsyst Technol 25(6):2313–2321. https://doi.org/10.1007/s00542-018-4116-7
116. Jiang H, Luo Y, Zhang QY, Yin MY, and Chun Wu (2017) TCP-Gvegas with prediction and adaptation in Multi-hop Ad Hoc Networks. Wireless Netw 23(5):1535–1548. https://doi.org/10.1007/s11276-016-1242-y
117. Priyadarshi R, Rawat P, Nath V, Acharya B, Shylashree N (2020) Three Level Heterogeneous Clustering Protocol for Wireless Sensor Network. Microsyst Technol 26(12):3855–3864. https://doi.org/10.1007/s00542-020-04874-x
118. Jiang S, Song X, Wang H, Han JJ, Li QH (2006) A clustering-based method for unsupervised intrusion detections. Pattern Recognit Lett 27(7):802–810. https://doi.org/10.1016/j.patrec.2005.11.007
119. Priyadarshi R, Singh L, Kumar S, Sharma I (2018) A Hexagonal Network Division Approach for Reducing Energy Hole Issue in WSN. Eur J Pure Appl Math 118 (March)
120. Jin Y, Duffield N, Erman J, Haffner P, Sen S, and Zhi Li Zhang (2012) A modular machine Learning System for Flow-Level Traffic classification in large networks. ACM Trans Knowl Discovery Data 6(1). https://doi.org/10.1145/2133360.2133364
121. Karagiannis T, Papagiannaki K, Faloutsos M (2005) BLINC: Multilevel Traffic classification in the Dark. Comput Communication Rev 35(4):229–240. https://doi.org/10.1145/1090191.1080119
122. Karami A (2015) ACCPndn: adaptive congestion control protocol in named data networking by learning capacities using optimized time-lagged feedforward neural network. J Netw Comput Appl 56:1–18. https://doi.org/10.1016/j.jnca.2015.05.017
123. Priyadarshi R, Soni SK, and Prashant Sharma (2019) An enhanced GEAR Protocol for Wireless Sensor Networks. Lecture Notes Electr Eng 511:289–297 Springer Singapore. https://doi.org/10.1007/978-981-13-0776-8_27
124. Rao S (2006) Operational Fault detection in Cellular Wireless Base-stations. IEEE Trans Netw Serv Manage 3(2):1–11. https://doi.org/10.1109/TNSM.2006.4798311
125. Rawat P, Chauhan S, and Rahul Priyadarshi (2020) Energy-efficient clusterhead selection Scheme in Heterogeneous Wireless Sensor Network. J Circuits Syst Computers 29(13):2050204. https://doi.org/10.1142/S0218126620502047
126. Reddy EK (2017) Comparative Analysis of Clustering Techniques in Data Mining. Int J Adv Sci Technol Eng Manage Sci 9028(1):2454–2356. www.ijastems.org
127. Ross DA, Lim J, Lin RS, Ming HY (2008) Incremental learning for Robust Visual Tracking. Int J Comput Vision 77(1–3):125–141. https://doi.org/10.1007/s11263-007-0075-7
128. Sateesh V, Anugrahith A, Kumar R, Priyadarshi, Nath V (2021) A Novel Deployment Scheme to Enhance the Coverage in Wireless Sensor Network. In Lecture Notes in Electrical Engineering, edited by Vijay Nath and J K Mandal, 673:985–93. Singapore: Springer Singapore. https://doi.org/10.1007/978-981-15-5546-6_82

129. Shon T, and Jongsub Moon (2007) A Hybrid Machine Learning Approach to Network Anomaly Detection. Inf Sci 177(18):3799–3821. https://doi.org/10.1016/j.ins.2007.03.025
130. Singh L, Kumar A (2020) and Rahul Priyadarshi. Performance and Comparison Analysis of Image Processing Based Forest Fire Detection. In Lecture Notes in Electrical Engineering, edited by Vijay Nath and J K Mandal, 642:473–79. Singapore: Springer Singapore. https://doi.org/10.1007/978-981-15-2854-5_41
131. Sun J, Chan S, Zukerman M (2012) IAPI: An Intelligent adaptive PI active Queue Management Scheme. Comput Commun 35(18):2281–2293. https://doi.org/10.1016/j.comcom.2012.07.013
132. Priyadarshi R, and Raj Vikram (2023) A triangle-based localization Scheme in Wireless Multimedia Sensor Network. Wireless Pers Commun 133(1):525–546. https://doi.org/10.1007/s11277-023-10777-7
133. Tesauro G (2007) Reinforcement learning in Autonomic Computing: a Manifesto and Case studies. IEEE Internet Comput 11(1):22–30. https://doi.org/10.1109/MIC.2007.21
134. Tsai C, Fong YF, Hsu CY, Lin, Wei YL (2009) Intrusion detection by machine learning: a review. Expert Syst Appl 36(10):11994–11990. https://doi.org/10.1016/j.eswa.2009.05.029
135. Priyadarshi R, Yadav S (2019) and Deepika Bilyan. Performance Analysis of Adapted Selection Based Protocol over LEACH Protocol. In Smart Computational Strategies: Theoretical and Practical Aspects, edited by Ashish Kumar Luhach, Kamarul Bin Ghazali Hawari, Ioan Cosmin Mihai, Pao-Ann Hsiung, and Ravi Bhushan Mishra, 247–56. Singapore: Springer Singapore. https://doi.org/10.1007/978-981-13-6295-8_21
136. Wang M, Cui Y, Wang X, Shihan Xiao, and Junchen Jiang (2018) Machine learning for networking: Workflow, advances and opportunities. IEEE Network 32(2):92–99. https://doi.org/10.1109/MNET.2017.1700200
137. Priyadarshi R (2024) Energy-efficient routing in Wireless Sensor networks: a Meta-heuristic and Artificial Intelligence-Based Approach: a Comprehensive Review. Arch Comput Methods Eng. https://doi.org/10.1007/s11831-023-10039-6
138. Stigler SM (2007) Gauss and the invention of least squares. Annals Stat 9(3). https://doi.org/10.1214/aos/1176345451
139. Priyadarshi R (2024) Exploring machine learning solutions for overcoming challenges in IoT-Based Wireless Sensor Network Routing: a Comprehensive Review. Wireless Netw. https://doi.org/10.1007/s11276-024-03697-2
140. Thakkar Mansi K, Patel MM (2018) Energy Efficient Routing in Wireless Sensor Network. Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA 2018 118(20):264–68. https://doi.org/10.1109/ICIRCA.2018.8597353
141. Priyadarshi R (2017) and Abhyuday Bhardwaj. Node Non-Uniformity for Energy Effectual Coordination in Wsn. International Journal on Information Technologies & Security, № 4(4):2017. https://ijits-bg.com/contents/IJITS-No4-2017/2017-N4-01.pdf
142. Wang Y, Martonosi M, and Li-Shiuan Peh (2007) Predicting Link Quality using supervised learning in Wireless Sensor Networks. ACM SIGMOBILE Mob Comput Commun Rev 11(3):71–83. https://doi.org/10.1145/1317425.1317434
143. Priyadarshi R, Bhardwaj P, Gupta P (2023) and Vijay Nath. Utilization of Smartphone-Based Wireless Sensors in Agricultural Science: A State of Art. In Lecture Notes in Electrical Engineering, edited by Vijay Nath and Jyotsna Kumar Mandal, 887:681–88. Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-19-1906-0_56
144. Xu K, Tian Y, and Nirwan Ansari (2004) TCP-Jersey for Wireless IP communications. IEEE J Sel Areas Commun 22(4):747–756. https://doi.org/10.1109/JSAC.2004.825989
145. Zhang C, Jiang J, and Mohamed Kamel (2005) Intrusion detection using hierarchical neural networks. Pattern Recognit Lett 26(6):779–791. https://doi.org/10.1016/j.patrec.2004.09.045
146. Priyadarshi R, Singh L, Randheer, Singh A (2018) A Novel HEED Protocol for Wireless Sensor Networks. In 2018 5th International Conference on Signal Processing and Integrated Networks, SPIN 2018, 296–300. https://doi.org/10.1109/SPIN.2018.8474286
147. Yi C, Afanasyev A, Moiseenko I, Wang L, Zhang B, Zhang L (2013) A case for Stateful Forwarding Plane. Comput Commun 36(7):779–791. https://doi.org/10.1016/j.comcom.2013.01.005
148. Priyadarshi R, Singh L, Singh A, Thakur A (2018) SEEN: Stable Energy Efficient Network for Wireless Sensor Network. In 2018 5th International Conference on Signal Processing and Integrated Networks, SPIN 2018, 338–42. https://doi.org/10.1109/SPIN.2018.8474228
149. Williams N, Zander S, Armitage G (2006) A Preliminary Performance Comparison of Five Machine Learning Algorithms for practical IP Traffic Flow classification. Comput Communication Rev 36(5):7–15. https://doi.org/10.1145/1163593.1163596
150. Priyadarshi R, Soni SK, Bhadu R, Nath V (2018) Performance Analysis of Diamond Search Algorithm over full search algorithm. Microsyst Technol 24(6):2529–2537. https://doi.org/10.1007/s00542-017-3625-0
151. Wang Z, Zhang M, Wang D, Song C, Liu M, Li J, Lou L, and Zhuo Liu (2017) Failure prediction using machine learning and Time Series in Optical Network. Opt Express 25(16):18553. https://doi.org/10.1364/oe.25.018553
152. Priyadarshi R, Soni SK, and Vijay Nath (2018) Energy efficient cluster head formation in Wireless Sensor Network. Microsyst Technol 24(12):4775–4784. https://doi.org/10.1007/s00542-018-3873-7
153. Zhang J, Chen C, Xiang Y, Wanlei Zhou, and Yong Xiang (2013) Internet traffic classification by aggregating correlated naive bayes predictions. IEEE Trans Inf Forensics Secur 8(1):5–15. https://doi.org/10.1109/TIFS.2012.2223675