Wireless Networks
https://doi.org/10.1007/s11276-024-03760-y

ORIGINAL PAPER

Distributed file systembased optimization algorithm

Uppuluri Lakshmi Soundharya[1] · G Vadivu[2] · Gogineni Krishna Chaitanya[3]

Accepted: 25 April 2024

Abstract
Database engines and file systems have been using prefetching and caching technologies for decades to enhance the performance of I/O-intensive applications. When future data access needs to be accelerated, prefetching methods often provide gains depending on the latency of the entire system by loading primary memory elements. Its execution time, where the data level prefetching rules are set, has to be much improved, as they are challenging to optimize, comprehend, and manage. This paper aims to introduce a novel distributed file system (DFS) model through dynamic prefetching, that includes four processes such as (1) Identification of popular files, (2) Estimation of support value for a file block, (3) Extraction of frequent block access patterns, and (4) Matching algorithm. At first, the input files are given to the first phase (i.e.), identification of popular sizes, where the popular files are identified. The support value of the file blocks that correspond to popular files is calculated in the second stage. Then, the extraction of frequent block access patterns is done in the third phase. At last, in the matching algorithm, the identification or prediction of frequent access pattern of the query is done by the optimized Neural Network (NN). Here, the weight of NN is optimally tuned by the Harmonic Mean based Grey Wolf Optimization (HMGWO) Algorithm.The proposed NN + HMGWO model produces reduced FPR values with good quality, which are 70.84%, 73.86%, 70.51%, 62.90%, 55.76%, 78.63%, and 73.86%, respectively, in comparison to other standard models like NN + WOA, NN + GWO, NN + PSO, NN + FF, FBAP, NN, and SVM. Lastly, the effectiveness of a chosen scheme is compared to other current methods in terms of delay analysis, latency analysis, hit ratio analysis, and correspondingly.

Keywords Prefetching · Caching · Distributed file system · Neural network · Optimization

Nomenclature

AFS	Andrew file system
APS	Daptable prefetching scheme
AFPL	Appointed file-prefetching language
ARIMA	Autoregressive integrated moving average
BLC	Block locality caching
BIO	Block I/O layer
BAP	Block access pattern
CSS	Cloud storage system
DNs	Data nodes
DNN	Deep neural network
DFS	Distributed file systems
EMF	Eclipse modeling framework
FTL	Flash translation layer
FANB	Fair allocation of network bandwidth
GS	Global support
IPODS	Informed prefetching for distributed multi-level storage
LBA	Logical block addresses
LS	Local support
LAN	Local area network
NN	Neural network
NFS	Network file system
MPJUP	Mobility prediction and joint user prefetching
RKs	Racks
RAID	Redundant array of independent disks
SNN	Secondary name node
SAP	Strip-aware prefetching

✉ Uppuluri Lakshmi Soundharya
uppulurilakshmisoundharya@gmail.com

[1] Research Scholar, Department of data science and business systems, SRM Institute of Science and Technology, SRM Nagar, Chengalpattu District, Kattankulathur 603203, Tamil Nadu, India

[2] Professor, Department of Data Science and Business Systems, School of Computing, Faculty of Engineering and Technology, SRM Institute of Science & Technology, SRM Nagar, Kattankulathur, Chengalpattu 603203, Tamil Nadu, India

[3] Associate Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, 522502, Andhra Pradesh, India

Published online: 28 May 2024

TC/S Thin-client/server
GWO Grey wolf optimization

1 Introduction

TC/S computing method is becoming popular due to its less cost and the rapid deployment of applications running at the server side, or known as server-based computing [1, 2]. Moreover, the server-based computing allows the corporations in the server infrastructure with more control by managing at the desktops. The transparent working devices are given based on the types of clients [3, 4]. To provide a transparent work environment, the conventional approach is to use a single application server design. All client data and applications are collected on a single server, and each client device is connected to the same application server. This is by efficient transfer of keyboard inputs, mouse clicks, and screen updated through a display protocol as possible [5, 6]. The display updates needmore bandwidth than the keyboard inputs and user mouse, and thus it is the technology used for the display updates thatrestricts the network area [7].

Automatic prefetching is attained in the application server for predicting the accessed files by a user, and the files that are prefetching in parallel within the user's employment [8]. Moreover, the automatic prefetching uses the past file access records for predicting the upcoming file system requirements. The main objective is to offer the data in the request progress, and removing the access latencies effectively [9]. The performance of automatic prefetching is slightly superior to other existing demand prefetching as determined in DFS including the AFS and the Sun NFS [10, 11]. However, as these schemes can derive only an approximation prediction and some of the prefetched files will not be accessed by the user. All pertinent facts and strategies related to sustainable road transportation in order to contribute to the existing body of knowledge [12, 13].This may affect the performance by increasing the network traffic.The effectiveness of the services offered is closely correlated with the system performance of the network [14].

The use of an appointed file-prefetching method [15, 16, 17, 18] is implemented for DFS, where the major idea is to facilitate the system administrator or user for specifying how the system performed the prefetching. The AFPL is executed on JAVA [19]. The system administrator or user could write the AFPL programs to prefetch the necessary files for instructing the system from the file server. Two prefetching models are determined in the AFPL: event prefetching and time prefetching [20]. The system prefetched the selected files in time prefetching at specific times, while the prefetching is performed in the event prefetching when the specific events occur. Both systems could accessed the past file access records and the user's schedule to specify the prefetching.

Although, the behavior-based prefetchers are efficient at the system-level for predicting the upcoming accesses; however, the behavior-based prefetchers would not convincing the LBA access patterns at the storage level due to the weak-locality for prefetching [21]. Several machine learning based approaches [22, 23] are implemented like C-miner, Diskseen, and ARIMA model. Likewise, the prefetching approaches such as the ARIMA model needs a long sequence to make a accurate prediction with no structural change, while the frequent pattern mining method involves recurring access patterns for building the association rule, that could be reused to decide as the prefetched data [24, 25, 26]. However, putting in place a learning-based prefetcher for the storage systems has two significant drawbacks. Initial, there exist a semantic gap among the storage devices and storage systems (i.e., file system) that makes the learning-based prefetchers more difficult to study the beneficial patternsin the storage devices [27, 28]. Numerous abstraction layers include the FTL and Linux BIO is applied for simplifying the communication through the LBAs since to make certain compatibility among different storage devices and systems [29, 30, 31, 32]. Short-term confusion and Fourier transform functionality are extracted and analyzed using neural network topologies [33]. Many significant mathematical science phenomena are described by optimal control issues [34].Subsequently, the use of multi-threading outcomes makes the prefetching at the storage level in an interleaved access request (in traditional prefetching models). Moreover, the training machine learning scheme includes the frequent pattern mining with superior variance degrees of LBA provides the system studied with some rules and limited coverage of the prefetcher to overcome the miss latency [35]. Furthermore, these restrictions lead to learning complexity issues, which get worse as storage device sizes increase.The main contributions are:

- Introduces a new DFS model that uses a new threshold computation to identify popular input file sizes.
- Computes the support value of the file blocks belongs to popular files.
- Proposes a novel HMGWO model for tuning the NN model via selecting the optimal weights.

The structure of the paper is;Sect. 2 discusses the related works on prefetching and caching in DFS. Section 3 determines the overall explanation of proposed framework: a machine learning based approach. Section 4 illustrates the steps involved in prefetching and caching of DFS. Section 5 explains the proposed harmonic mean based grey wolf optimization. Section 6 contains the outputs and their explanations. Finally, Sect. 7 represents the work's conclusion.

2 Literature review

2.1 Related works

In 2019, Chen et al. [36] have developed Smeta known as the metadata prefetching approaches that was integrated seamlessly into DFS and significantly scales the performance of metadata. Moreover, a new and entirely different system was used for discovering the correlations of explicit data through considering the reference associations among the files in several applications. SMeta explored the correlations upon the files that were written through a "light-weight pattern matching algorithm", which stored the correlations of file metadata in the reserved extensive attributes for avoiding the DFS APIs changes, and collapsed the multiple I/O rounds to access the target files metadata. Further, the cost-efficient adaptive feedback method was designed for enhancing the prefetching accuracy and when compared to baselines, the SMeta has shown improved metadata performance with respect to scalability, latency, and throughput.

In 2020, Ganfureet et al. [37] has proposed DeepPrefetcher, a new DNN motivated the context-aware prefetching model, which was adapted to arbitrary memory access patterns. In addition, the BAP contexts were captured under DeepPrefetcher by the leverage LSTM – DL model and distributed representation as a prefetched data for the context-aware prediction. The difference among successive access requests was modeled that included more patterns than LBA value. Finally, the simulation outcomes have revealed that the DeepPrefetcher has shown enhanced speedup, coverage, and average prefetch accuracy as 17.2%, 19.5%, and 21.5%, correspondingly than the baseline prefetching strategy.

In 2018, Lee, et al. [38] have revealed the most traditional DFSs with extremely degraded outcomes in various running environments for simultaneous read streams. Furthermore, the cause behind the degraded performance was determined through the 2 DFS issues. The major cause of the degradation performance was that DFSs perform the prefetching in a better manner with no regard to the network delay time and the storage device types. An APS model was implemented for various running environments. Furthermore, an inverse link was created between a lower sensitivity to greater network delays in DFSs and the superior performance of random reads, which was verified by calculating the efficiency evaluation for increasing network delays and randomized reads. The APS has conducted the prefetching in a suitable way based on the exact running environment with an adaptable model. Finally, the proposed model has shown maximum network delays performance and small degradation performance of the random reads.

In 2020, Yu et al. [39] have adopted a content caching approach on the basics of MPJUP model. By using the user's movement to anticipate their location with regard to of mobility, the selected policy has predicted the prefetching device data and divided the partial cache space for the purpose of prefetching the data. In addition, the presented approach was executed for minimizing the backhaul load through lowering the content backhauls count via cooperating prefetch data among the edge cache device and the user. The evaluation have proven that the adopted method attains minimum backhaul load and average delay, and thus the prefetch scheme was also suitable for more widespread networks.

In 2018, Assaf et al. [40] have presented an informed prefetching model known as IPODS, in which the application was used the closed access patterns in distributed multi-level storage schemes for prefetching the hinted blocks. The IPODS technique was fluctuated from the traditional prefetching models in 2 manners. Primarily, it minimizes the I/O stall applications via the hinted data in storage servers' fast buffers and clients' local caches. Next, multiple informed prefetching systems organized semi dependent for fetching the blocks in a prefetching pipeline, "(1) from low-level to high-level storage devices in servers and (2) from high-level devices in servers to the clients' local cache".

In 2018, Hyun et al. [41] have proposed a FANB for the simultaneous read streams and to determine the discovered issues. The proposed model achieved better performance with a pair of SAP and FANB than all the traditional DFSs regardless of (1) strip size, (2) the number of striped disks, (3), the amount of read streams and (4) the kind of a striped RAID. Further, the adopted method has shown superior outcomes when compared to other conventional DFSs at least two times for all striped RAIDs configurations and its types. Additionally, the performance gap among the adopted model and the traditional DFSs were wider with enhanced number of striped disks.

In 2019, Li et al. [42] have implemented a prefetch-aware fingerprint cache organization approach known as PreCache for data deduplication model for alleviating the prefetch-related cache pollution. Further, the fingerprint prefetching and PreCache could work together for enhancing the outcomes of data deduplication models. The PreCache on SiLo and BLC representative data deduplication models was executed. Finally, the simulation outcomes based on 3 real-world workloads (Homes, MacOS, and Kernel) has attained 32.22% maximized deduplication throughput and the PreCache would alleviate the prefetch-related cache pollution in the adopted approach.

In 2019, Daniel et al. [43] have presented an event-based language known as PrefetchML DSL which described the caching and prefetching rules over other schemes. A monitoring feature built into PrefetchML let modelers identify

unwanted circumstances and modify their conventional plans, which largely resolved the problems. The PrefetchML architecture was designed on NeoEMF/Graph and on top of the EMF, and simulation outcomes has shown an enhancement in important execution time than the non-prefetching used cases. At the end, a monitoring component in the proposed system has provided a group of measures that allows the modelers to optimize the PrefetchML plans.

2.2 Review

Table 1 displays the existing techniques on Prefetching and caching in DFS. Initially, metadata prefetching method was deployed in [36], which presents better latency, higher throughput and maximum scalability; however, an advanced solution was not designed for data correlation extraction to be filed with more difficult formats. DNN inspired context-aware prefetching method were exploited in [37] that offer decrease increase an average prefetch accuracy, better coverage, and higher speed up, but an in-depth study was not conducted for prefetching while applying the reinforcement assisted learning process. Moreover, APS was deployed in [38] that offer lowest throughput, low delay, and minimizes the degradation performance. Likewise, MPJUP scheme was exploited in [39], which offers reduce data transfer latency, reduces the average delay and minimize backhaul load. IPODS system was exploited in [40] that have reduced overall I/O access time, hides network latency, and improves I/O performance; however, a pipelining approach was not proposed for predictive prefetching in a distributed computing environment. In addition, the FANB model was introduced in [41], which offers low higher performance, better strip size, and increased number of striped disks. PreCache scheme was suggested in [42] that offer small LRU cache, improved deduplication throughput, and lowest fingerprint cache miss ratio. Finally, PrefetchML DSL model was implemented in [43], which offers improve query execution time, and reduces the concurrency bottlenecks; however, the automatic generations of PrefetchML scripts were not used based on transformations for the metamodel and static analysis of available queries. These restrictions must be considered in order to properly use prefetching and caching in DFS in the current work.

2.3 Objectives

- Analysis is done to minimize the error convergence.
- To reduce delays, analysis of them is conducted.
- To show the stability of the offered technique, a statistical study is conducted.

Table 1 Review on prefetching and caching in DFS: Benefits and drawbacks

Author [citation]	Proposed model	Benefits	Drawbacks
Chen et al. [36]	Metadata prefetching method	Improved delay, increased productivity, and maximal adaptability	An improved solution was not developed for data correlation extraction to be filed with increasingly sophisticated forms
Ganfureet al. [37]	DNN inspired context-aware prefetching method	Enhanced protection, greater precision on average prefetch, and faster speedup	It has not been thoroughly investigated how to prefetch using the reinforcement assisted learning method
Lee et al. [38]	APS	Minimum productivity, shortest delay, and least amount of performance loss	Not every I/O scheduler is taken into consideration by the suggested paradigm
Yu et al. [39]	MPJUP scheme	Minimize the backhaul load. Lower the average delay. Decrease data transmission latency	The suggested approach did not conduct more sophisticated mobility prediction algorithms or better predictions
Assaf et al. [40]	IPODS system	The efficiency of I/O operations is improved, network latency is hidden, and total I/O time to access is decreased	It was not suggested to use a pipelining method for predictive prefetching in a distributed computing setting
Hyun et al. [41]	FANB model	Improved strip size, more striped drives, and enhanced efficiency	It was not possible for the maximum readahead length to satisfy the PNR criteria
Li et al. [42]	PreCache scheme	Minimum fingerprint cache failure ratio; enhanced deduplication speed; minimal LRU cache	The approach that was proposed did not make use of machine learning techniques for the automatic modification of these parameters
Daniel et al. [43]	PrefetchML DSL model	minimizes bottlenecks caused by concurrent	Static query evaluation and met updates to models did not trigger the automatic generation of prefetch ML scripts

3 Overall description of proposed framework: a machine learning based approach

This paper presents a new DFS system that consists of four methods With the use of dynamic prefetching. (i) Identification of popular files, (ii) Estimation of support value for a file block, (iii) Extraction of frequent block access patterns, and (iv) Matching algorithms. Initially, the input files are given to the first phase (i.e.), identification of popular sizes, in which the popular files are identified from this process. The final part then involves extracting common blocks access patterns. Finally, in the matching algorithm, the prediction or identification of frequent access patterns is done using the machine learning techniques, where the optimized NN is exploited. Using the HMGWO Algorithms the weight of the NN is optimally tweaked for more accurate predictions. Finally, the predicted output is produced effectively. Figure 1 displays the design of the adopted method.

3.1 Architecture of cloud storage system (CSS)

The CSS includes several DNs that are ordered in multiple RKs ordered in the cluster structure. Additionally, the data is saved in DN in this kind of format, and the metadata is gathered in the name node. Under certain circumstances, the data is restored through SNN only if name node fails. Additionally, the name node and SNN are situated adjacent to one other in the RKs that make up CSS. In addition, the CSS is same as the Hadoop storage scheme. Figure 2 illustrates the architecture of a CSS framework.

Let assume that the name of CSS nodes keeps track of clients application code access information to the CSS file blocks in a log file kept on file within the system. Each entry also contains details taken from a system log file, including the DNS IP address where the file blocks are used. The user's application software and the file name are also run.

4 Steps involved in prefetching and caching of DFS

The prefetching and caching of DFS involves four major steps as follows.

- Identification of popular files
- Estimation of support value
- Extraction of frequent access block pattern
- Matching algorithm

4.1 Identification of popular files

Consider the number of entries A associated with the client sessions and it is observed in the log. If a file F emerges in L number of entries in the log then estimated the support for the file (FS)as L/A. The threshold for popular files will be FT. If $FS >= FT$, and F is regarded as a popular file. Moreover, the frequent BAPs are find only for the popular files and calculated the global support value belong to popular files for the file blocks.

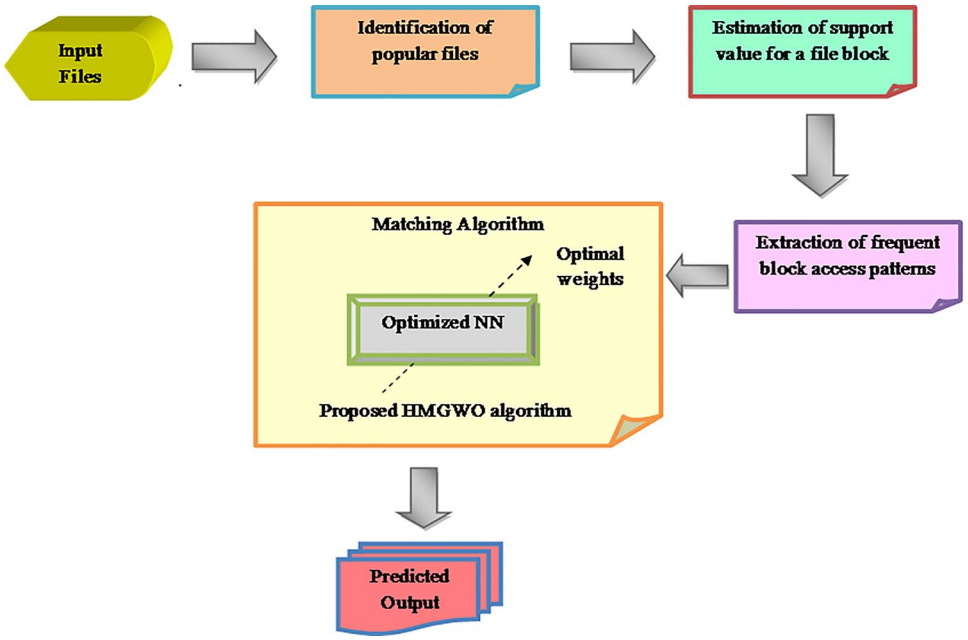

Fig. 1 Architecture of the adopted framework

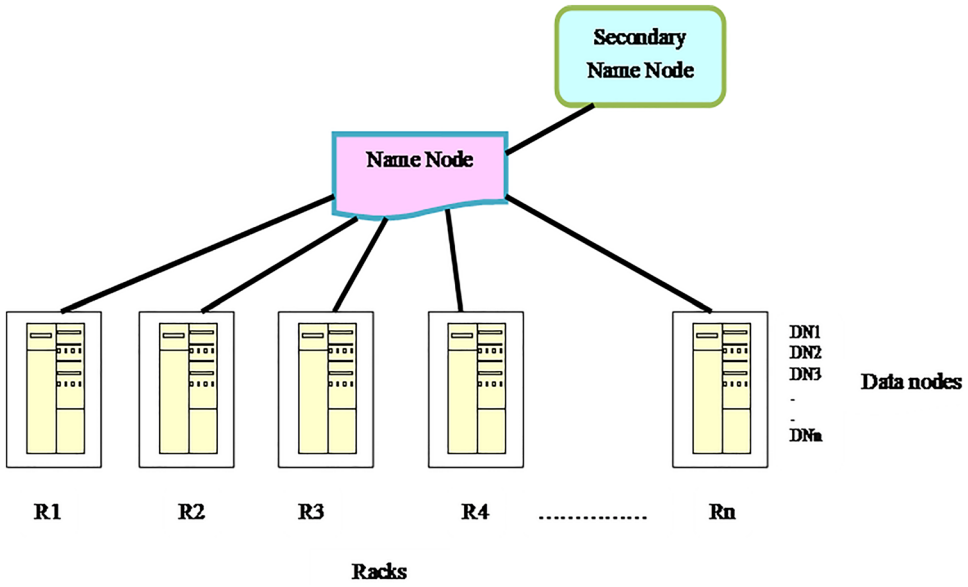

Fig. 2 Architecture of the CSS framework

4.2 Estimation of support value

The support value of a block file belongs to only popular files are estimated. There are 2 types of support used for file blocks such as.

- LS
- GS

The LS value is estimated for a file block via allowing the log entries of a particular DN. For every file block, a single GS level will be determined; the GS value is estimated by utilizing the log entries of every DN for that particular file block. Moreover, the popular files are represented as the files $F1$ and $F2$. Consider the requested file blocks during a session through the client application programs are determined as [$Fl\ Bu\ Bv$ ...] where,Bu denote the block number of the file Fl, and Fl indicate the file name. Moreover, the total number of sessions in the DN is created where the file block $Fl\ Bu$ shown as [$T(Fl\ Bu)$] and entire number of sessions produced in the DN and thus the file Fl emerges as QP. Subsequently, the support value is computed for a particular file block ($QS(Fl\ Bu)$) as $\left[T(Fl\ Bu)/\ QP\right]$.

4.3 Extraction of frequent access block pattern

For instance, the BAP [$F1\ B1\ B2$] is taken from the entry. Further, number of sessions in $F1\ B1\ B2$ is 8, and the total number of sessions in which $F1\ B1$ or $F1\ B2$ is 11. Subsequently, the support value for the BAP $F1\ B1\ B2$ is calculated as 8/11 (i.e.),72.7%. Likewise, the support value for all BAP is calculated. Then, the threshold (t) value is set and extracted the frequent BAPs based on the value of t.

4.3.1 Determining number of replicas

The replication number has an important impact on storage space used in the DNs. Let's examine the threshold amount that determines the sequence of frequently block access for each block. If ($BAP(blocks)$ >= min sup) then these blocks is the frequent BAP.

4.3.2 Placement of replicas:

Assume block B of a file F is formed first in DN1 that was placed in the $R1$ rack. Moreover, the B indicates the frequent BAP (FP) member that includes number of blocks n.

4.4 Matching algorithm via optimized neural network

The matching algorithm identifies the frequent block access patterns using optimized NN. The extracted frequent BAP is given as the input of NN to identify the frequent access pattern of query file. Using enhanced GWO, the weight of NN is adjusted to increase process precision.

(i) NN: "NN is a group of algorithms which endeavours for recognizing the fundamental relations in a group of data through a process that mimics the way the human brain operates". The input provided to the NN [44] is the extracted frequent BAP (FP) and it is determined in Eq. (1), here z represents the total number of frequent block access patterns.

$$FP = \left\{FP_1, FP_2, ...FP_z\right\} \tag{1}$$

The NN framework consists of hidden, output, and input layers, respectively. In addition, the output of the hidden

Fig. 3 Solution encoding

layer $H^{(D)}$ is specified in Eq. (2), where,g and $\hat{E}$ indicates the neurons in input layer and hidden layer correspondingly,$W^{(X)}_{(g\hat{E})}$ denotes the weight between the gth input neuron to $\hat{E}$th hidden neuron,a indicatesthe activation function, $W^{(X)}_{(t\hat{x})}$ indicates the bias weight with $\hat{x}$th hidden neuron, and $\hat{y}_{\hat{x}}$ referred as input neurons count. Further, the network output $\hat{J}_Y$ is given in Eq. (3), where, $W^{(J)}_{(t\hat{Y})}$ denotes theoutput bias weight of $\hat{Y}$th output layer,$\hat{o}ut$ portrays the output neurons,$W^{(J)}_{(\hat{E}\hat{Y})}$ refers to the weight among the $\hat{E}$th hidden layers to $\hat{Y}^{th}$ output layer and NH specifies the number of hidden neurons. Consequently, the error (Er^*) attained in both actual and predicted values should be low as given in Eq. (4). Here,$\hat{y}_{NC}$ portraysthe output neuron count,$J_{\hat{Y}}$ and $\widehat{J}_{\hat{Y}}$ specifies the actual and predicted output, correspondingly. The output of NN is denoted as MA.

$$H^{(D)} = a\left(W^{(X)}_{(t\hat{E})} + \sum_{g=1}^{y_E} W^{(X)}_{(\hat{g}\hat{E})} FP\right) \tag{2}$$

$$\widehat{J}_{\hat{Y}} = a\left(W^{(J)}_{(t\hat{Y})} + \sum_{E=1}^{y_x} W^{(J)}_{(\hat{E}\hat{Y})} H^{(D)}\right) \tag{3}$$

$$Er^* = \underset{\left\{W^{(y)}_{(t\hat{E})}, W^{(y)}_{(g\hat{E})}, W^{(J)}_{(t\hat{Y})}, w^{(J)}_{(\hat{E}\hat{Y})}\right\}}{\arg\min} \sum_{Y=1}^{\hat{y}_{NC}} \left| J_{\hat{Y}} - \widehat{J}_{\hat{Y}} \right| \tag{4}$$

5 Tuning of optimal weight via proposed harmonic mean based grey wolf optimization

5.1 Solution encoding and objective function

As previously indicated, superior results are guaranteed because the NN weights are fine-tuned. In this work, the HMGWO model is implemented for tuning purposes. Figure 3 illustrates the input solution given to the algorithm, where, M portrays the total amount of CNN weights. The adopted modelobjective function is given in Eq. (5).

$$Obj = \min\left(Er^*\right) \tag{5}$$

5.2 Proposed HMGWO algorithm

Although, the traditional GWOmodel [45] provides accurate assessments; still, it endure from poor ability in local searching, slow convergence, and lower precision. As a result, the selected IGWO algorithms has some changes to overcome the shortcomings of the current GWO. In the field of optimization, an optimization issue is solved when the optimum values for the choice factors are found to maximize or minimize a set of function objectives without violating any limitations. Most real-world optimization issues have a number of difficulties, including large solution spaces, dynamic/noisy objective operations, non-convex search landscapes, high processing costs, and non-linear restrictions. Generally, self-improvement is proved to be promising for solving many of the complex optimization problems [24], 25, 26, 22, 23. The hunting characteristics of grey wolves and their leadership hierarchy are described by the GWO algorithm. Hybrid technique optimizations select dynamically an optimization algorithm from a collection of various methods that all carry out the same optimization during compilation. For every code segment that requires optimization, they employ a heuristic to determine the best course of action. There are 4 grey wolf groups like $\lambda, \gamma, \psi, \zeta$ that are influenced for the operation of hierarchy of leadership. Moreover, the 3 major options are used in the hunting process such as encircling the prey, hunting and attacking the prey.

During the hunting process, the wolves such as λ, γ and ψ are considered as the foremost wolves. The leader λ present among such wolves involves in the resting location, hunting process, time to wake up, etc. Further, γ and ψ holds 2nd or 3rd level which assists λ in making the decisions. The final level of wolves is potrayed as ζ allocated for eating purpose only.

(a)Encircling prey The wolves' encircling nature is given in Eq. (6) and Eq. (7), where, $\vec{C}$ and $\vec{K}$ refers to the coefficient vectors, $\vec{Z}_{prey}$ indicates the position vectors of prey, i represents the current iteration and $\vec{Z}$ refers to the position vectors of the grey wolves'. Moreover, $\vec{C}$ and $\vec{K}$ representation is given in Eq. (8) and (9), correspondingly, where r_1 and r_2 denotes the arbitrary vectors presentconstantly between [0, 1].

$$\vec{S} = \left| \vec{K}.\vec{Z}_{prey}(i) - \vec{Z}(i) \right| \tag{6}$$

$$\vec{Z}(i+1) = \vec{Z}_{prey}(i) - \vec{C}.\vec{S} \tag{7}$$

$$\vec{C} = 2\vec{q}..\vec{r}_1 - \vec{q} \tag{8}$$

$$\vec{K} = 2\vec{r}_2 \tag{9}$$

As per the suggestedHMGWO logic, the value of q is expressedin Eq. (10). Where, q_j represents the maximum value, j represents the admissible maximum iteration number, and V denotes the total iteration.

$$q = q_j e^{-it/V} \tag{10}$$

Hunting: The wolve's hunting nature is represented in Eq. (11) to Eq. (16).

$$S_\lambda = \left|K_1.Z_\lambda - Z\right| \tag{11}$$

$$S_\gamma = \left|K_2.Z_\gamma - Z\right| \tag{12}$$

$$S_\psi = \left|K_3.Z_\psi - Z\right| \tag{13}$$

$$Z_1 = Z_\lambda - C_1.\left(S_\lambda\right) \tag{14}$$

$$Z_2 = Z_\gamma - C_2.\left(S_\gamma\right) \tag{15}$$

$$Z_3 = Z_\psi - C_3.\left(S_\psi\right) \tag{16}$$

Traditionally GWO is updated on the basis of positions of λ, γ and ψ. As per adopted HMGWO method, the final update of GWO is performed by the harmonic mean of the wolf and it is given in Eq. (17), where, $i = 1, 2, 3$.The merits of harmonic mean for final updating, the solution will be set inside the bounds without considering the limit conditions.The pseudo code for proposed HMGWO method is presented in Algorithm 1

$$\vec{Z}(i+1) = \frac{3}{\sum\left(\frac{1}{Z_i}\right)} \tag{17}$$

Algorithm 1 Pseudo code of Proposed HMGWO method

Initialize population $U_b\left(b = 1,2,\ldots N\right)$
Initialize q , K and C
The fitness are calculated for each search agent
Z_λ =1st best search agent
Z_γ =2nd best search agent
Z_ψ =3rd best search agent
While ($i < i_{\max}$)
for every search agent
The proposed position update of the current search agent is by Eq. (17).
End for
Update q , K and C
compute the fitness
Update Z_λ , Z_γ and Z_ψ
$i = i+1$
End while
Return Z_λ

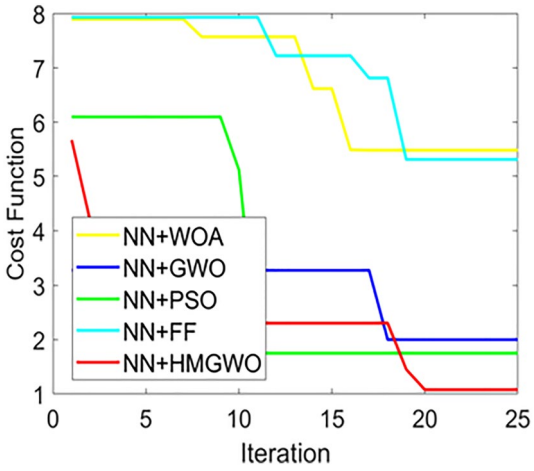

Fig. 4 Convergence analysis of suggested NN + HMGWOmodel over other traditional methods

6 Results and discussion

6.1 Simulation procedure

The suggested NN + HMGWO scheme for Prefetching and caching in DFS was executed in **MATLAB** and the outcomesare verified. Moreover, the dataset was gathered from: "https://github.com/shervinmin/DeepCovid". In this case, the gathered dataset consists of 5,000 annotated photos. This dataset was created from two sources. For COVID-19 X-ray samples, use the Covid-Chestxray-Dataset; for non-COVID samples, use the ChexPert dataset.The performance of the presented NN + HMGWO method with prefetching and caching in DFS was computed over the existing schemes such as NN + WOA [46], NN + GWO [45], NN + PSO [47], NN + FF [48], FBAP [49], NN [44], and SVM [50], correspondingly. Furthermore, by adjusting the block size to 10, 20, and 30 in terms of delay analysis, hit ratio analysis, and latency analysis, accordingly, the efficiency of the proposed and standard approaches was determined.

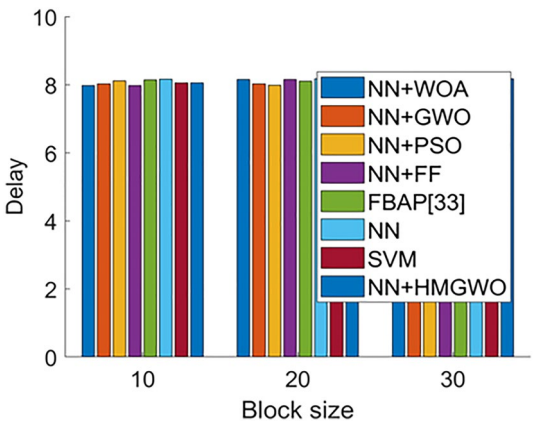

Fig. 5 Delay analysis of adopted NN + HMGWOmodel over other traditional schemes

6.2 Convergence analysis

Figure 4 illustrates the convergence evaluation of the proposed NN + HMGWO method over other existing techniques for a range of iterations, from 0 to 25. In addition, as the graphic explains, the cost function steadily gets less as the number of iterations rises. At the 20th iteration, the performance of the developed NN + HMGWO technique is 80%, 50%, 45%, and 78.84% greater than that of the existing techniques. This is sometimes investigated in both the suggested and traditional scenarios. However, when compared to other traditional approaches in the last repetition, the suggested approach still needs to be improved.

6.3 Analysis on delay

The developed NN + HMGWOmodel in terms of delay is demonstrated in Fig. 5. Initially, the prefetching and caching in DFS for delay analysis attains better outcomes of the developed method for block size 20. It is observed that the proposedNN + HMGWO method remains higher (~ 8.1742) values for block size 30; however, the developedNN + HMGWO method retains the minimum value (~ 8.0568) for block size 10. For example, the delay analysis of proposed NN + HMGWO method attains lower value (~ 7.8492) with superior performance for block size 20. The results therefore demonstrate the betterment of the suggested NN + HMGWO method.

6.4 Latency analysis

The evaluation of latency analysis of various methods is illustrated in Fig. 6. In this case, the simulation is run for

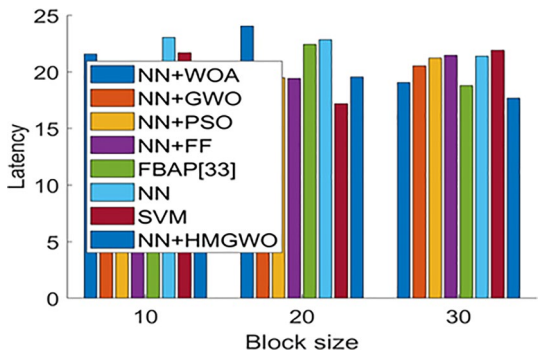

Fig. 6 Latency evalaution of suggestedNN + HMGWOmethods

block sizes of 10, 20, and 30, and the results are achieved. Based on the graph, it can be shown that, for block size 20, the suggested NN + HMGWO approach produces better results than other traditional systems. Furthermore, the proposed NN + HMGWO model attains minimum latency (~ 18.5) than other methodsincludesNN + WOA (~ 22), NN (~ 23), and SVM (~ 22.5), respectively for block size 10. Moreover, the proposed NN + HMGWO method holds reduced latency for both block size 20 when compared to block size 30. Thus, the adopted NN + HMGWO method has validated effectively with better outcomes.

6.5 Analysis on hit ratio

Table 2 presents the hit ratio effectiveness of the implemented NN + HMGWO algorithm compared with different current methods for block sizes 10, 20, and 30. The table shows that, for all block sizes, a developed NN + HMGWO approach achieves the highest hit ratio as compared to the other models. However, the hit ratio of developed NN + HMGWO scheme remains to be higher (~ 0.901) for block size 10 when compared to the block size 20. More particularly, for block size 30, the adopted NN + HMGWO model in terms of hit ratio remains superior value (~ 0.891) to existing methods.Thus, the suggested PDU-SLnO approach saw an improvement.

6.6 Statistical analysis

Table 3 illustrates the statistical evaluation of the developed approach for accuracy metric compared to other current approaches. The different methods are random in nature, it is necessary to run them multiple times in order to acquire the objective function's statistics and to ensure an accurate comparison.The evaluation of the adopted NN + HMGWO model shows maximum performance than

Table 2 Hit ratio analysis of adopted and existing schemes

Metrics	NN + WOA [46]	NN + GWO [45]	NN + PSO [47]	NN + FF [48]	FBAP [49]	NN [44]	SVM [50]	Proposed NN + HMGWO
Block size 10	0.802	0.761	0.125	0.79	0.59	0.35	0.32	0.901
Block size 20	0.45	0.72	0.42	0.71	0.52	0.61	0.39	0.881
Block size 30	0.78	0.59	0.56	0.157	0.015	0.805	0.42	0.891

Table 3 Statistical analysis: adopted vs. conventional schemes

Metrics	NN + WOA [46]	NN + GWO [45]	NN + PSO [47]	NN + FF [48]	Proposed NN + HMGWO
Standard deviation	1.1053	0.60731	2.1319	1.113	1.1178
Variance	1.2218	0.36883	4.545	1.2387	1.2494
Mean	6.7487	2.8674	3.4488	6.9646	2.3729
Best	7.8938	3.2756	6.0948	7.9285	5.6656
Worst	5.4805	2	1.75	5.3113	1.0769

Table 4 overall performance analysis of proposed and traditional models in terms of different measures

Metrics	NN + WOA [46]	NN + GWO [45]	NN + PSO [47]	NN + FF [48]	FBAP [49]	NN [44]	SVM [50]	Proposed NN + HMGWO
NPV	0.84829	0.83077	0.85	0.88077	0.9	0.79293	0.83077	0.95577
Specificity	0.84829	0.83077	0.85	0.88077	0.9	0.79293	0.83077	0.95577
MCC	0.56751	0.3914	0.52583	0.66744	0.72996	0.45881	0.3914	0.80714
F1-Score	0.62079	0.50764	0.39286	0.58333	0.75	0.48854	0.50764	0.82639
Precision	0.54167	0.43428	0.47176	0.55724	0.875	0.55961	0.43428	0.825
FPR	0.15171	0.16923	0.15	0.11923	0.1	0.20707	0.16923	0.044231
Accuracy	0.5	0.5	0.57143	0.64286	0.71429	0.42857	0.5	0.85714
FNR	0.375	0.5625	0.5	0.25	0.2	0.5	0.5625	0.1
Sensitivity	0.625	0.4375	0.5	0.75	0.8	0.5	0.4375	0.9
FDR	0.45833	0.53213	0.27554	0.38459	0.125	0.56638	0.53213	0.175

Table 5 Analysis of K-Fold

Methods	K = 2	K = 3	K = 4	K = 5
WOA	0.80769	0.82836	0.82549	0.83143
GWO	0.80769	0.82828	0.84604	0.81953
PSO	0.81953	0.79596	0.82847	0.85786
FF	0.84024	0.80191	0.81061	0.8313
CMP	0.83432	0.80181	0.81057	0.80465
NN	0.84024	0.82836	0.84016	0.81668
SVM	0.79586	0.82846	0.81656	0.80158
PROP	0.91941	0.92467	0.91124	0.91871

the conventional techniquesin Prefetching the frequent access. The suggestedNN + HMGWO model has attained better mean value (~ 2.3729) than other traditional scheme. For best case scenario, the proposedNN + HMGWO method hold is higher. Furthermore, the proposed NN + HMGWO model have proved betterment in DFS prefetching and catching than other existing techniques.

6.7 Overall performance analysis

Table 4 shows the overall performance evaluation of the suggested NN + HMGWO strategy compared to other conventional approaches for different metrics. According to the table, the NN + HMGWO method that was used has demonstrated its superiority over conventional approachesin terms of prefetching and detecting DFS. Moreover, the proposed NN + HMGWO model attains better accuracy value (~ 0.85714) than other existing techniques like NN + WOA, NN + GWO, NN + PSO, NN + FF, FBAP, NN, and SVM,accordingly. Further, the developed NN + HMGWO method achieve lower FPR values with greater values that is 70.84%, 73.86%, 70.51%, 62.90%, 55.76%, 78.63%, and 73.86% over the other modelsincludes NN + GWO, NN + PSO, NN + WOA, NN + FF, FBAP, NN, and SVM. According to the results, the developed NN + HMGWO method performs better for prefetching and catching in DFS than the traditional techniques.

6.8 K Fold validation analysis

Reproducibility is the extent to which a device can reliably produce an identical result under identical circumstances. Reliability, repeatability, and reproducibility are commonly used synonymously. Table 5 describes the validation analysis of the recommended approach techniques against well-known methods including the WOA, GWO, PSO, FF, CMP, NN, and SVM. The findings show that, with a greater value of 0.9246 in k = 3, the proposed USS-OS technique performs greater than the other algorithms, which include WOA, GWO, PSO, FF, CMP, NN, and SVM.

7 Discussion

This paper provides a new DFS approach by utilizing a collection of four steps termed dynamic prefetching. In the next step, new threshold-based FBAPs are obtained after the support value has been established. The effectiveness of the previously proposed NN + HMGWO technique in DFS, which includes prefetching and caching, was compared to the current methods. The suggested NN + HMGWO approach outperforms the traditional schemes such as NN + PSO, NN + WOA, NN + GWO,and NN + FF by 80%, 50%, 45%, and 78.84%, respectively, in the 20th iteration. The developed NN + HMGWO model achieves a higher accuracy value (~ 0.85714)for various schemes.Finally, the optimized Neural Network (NN) performs the identification or prediction of the query's frequent access pattern in the matching method.

8 Conclusion

Dynamic prefetching, which this work uses to develop a new DFS method, consists of four steps: (i) Identification of popular files, (ii) Estimation of support value for a file block, (iii) Extraction of frequent block access patterns, and (iv) Matching algorithm. At first, the input files were given to the first phase (i.e.), identification of popular sizes, where the popular files were identified. It calculates the support value of the data blocks that correspond to famous files in the second stage. Then, the extraction of frequent block access patterns was done in the third phase. At last, in the matching algorithm, the identification or prediction of frequent access pattern of the query was done by the optimized NN. In this case, the HMGWO Algorithm optimally tuned the weight of NN. Lastly, adeveloped method effectiveness was compared to other methods that were already in use in terms of hit ratio, convergence, delay, and latency analyses.Based on the analysis, the suggested NN + HMGWO technique outperformed by the results of 80%, 50%, 45%, and 78.84%, accordingly, at the 20th iteration. In addition, the proposed NN + HMGWO model attains minimum latency (~ 18.5) than other existing methods like NN + WOA (~ 22), NN (~ 23), and SVM (~ 22.5), respectively for block size 10. The suggested NN + HMGWO technique achieve greater values than other methods in the best-case situation. Similarly, the proposed NN + HMGWO model attains better accuracy value (~ 0.85714) than other comparisons.

Funding This research did not receive any specific funding.

Data availability No new data were generated or analysed in support of this research.

Declarations

Conflict of interest The authors declare no conflict of interest.

Ethical approval Not Applicable.

Informed consent Not Applicable.

References

1. Moon, Y., & Lim, M. (2019). Simulation analysis of prefetching image content for social networking service framework. *Multimed Tools Appl, 78*, 28435–28452. https://doi.org/10.1007/s11042-017-5492-1
2. Lee, S. M., Yoon, S. K., Kim, J. G., et al. (2018). Adaptive correlated prefetch with large-scale hybrid memory system for stream processing. *The Journal of Supercomputing, 74*, 4746–4770. https://doi.org/10.1007/s11227-018-2466-7
3. Yuan, P., Cai, Y., Liu, Y., et al. (2020). ProRec: A unified content caching and replacement framework for mobile edge computing. *Wireless Networks, 26*, 2929–2941. https://doi.org/10.1007/s11276-020-02248-9
4. Tang, Y., Guo, K., & Tian, B. (2018). "A block-level caching optimization method for mobile transparent computing. *Peer-to-Peer Netw. Appl., 11*, 711–722. https://doi.org/10.1007/s12083-017-0554-8
5. Rezvani, S., Parsaeefard, S., Mokari, N., Javan, M. R., & Yanikomeroglu, H. (2019). Cooperative multi-bitrate video caching and transcoding in multicarrier NOMA-assisted heterogeneous virtualized MEC networks. *IEEE Access, 7*, 93511–93536. https://doi.org/10.1109/ACCESS.2019.2927903
6. Kougkas, A., Devarajan, H., & Sun, X. H. (2020). I/O Acceleration via multi-tiered data buffering and prefetching. *Journal Computer Science Technology, 35*, 92–120. https://doi.org/10.1007/s11390-020-9781-1
7. Luo, C. L. S. (2020). Adaptive priority-based cache replacement and prediction-based cache prefetching in edge computing environment. *Journal of Network and Computer Applications, 165*, 102715.
8. Asim, M., Maria, M., Guy, M., & Gogniat. (2019). FLUSH + PREFETCH: A countermeasure against access-driven cache-based side-channel attacks". *Journal of Systems Architecture, 104*, 101698.
9. Fujita, Q. H. H. (2019). Adaptive resource prefetching with spatial–temporal and topic information for educational cloud storage systems. *Knowledge-Based Systems, 181*, 104791.
10. Wei, Y., Banawan, K., & Ulukus, S. (2019). Fundamental limits of cache-aided private information retrieval with unknown and uncoded prefetching. *IEEE Transactions on Information Theory, 65*(5), 3215–3232. https://doi.org/10.1109/TIT.2018.2883302
11. Wei, Y., Banawan, K., & Ulukus, S. (2018). Cache-aided private information retrieval with partially known uncoded prefetching: Fundamental limits. *IEEE Journal on Selected Areas in Communications, 36*(6), 1126–1139. https://doi.org/10.1109/JSAC.2018.2844940
12. Nabeeh, N. (2023). Assessment and Contrast the Sustainable Growth of Various Road Transport Systems using Intelligent Neutrosophic Multi-Criteria Decision-Making Model
13. Sallam, K., Mohamed, M., & Mohamed, A. W. (2023). Internet of things (IoT) in supply chain management: challenges, opportunities, and best practices. *Sustainable Machine Intelligence Journal, 2*, 1–3.
14. Alsaffar, M., et al. (2021). Network management system for IoT based on dynamic systems. *Computational and Mathematical Methods in Medicine*. https://doi.org/10.1155/2021/9102095
15. Nipanikar, S. I., & Hima Deepthi, V. (2019). Enhanced whale optimization algorithm and wavelet transform for image steganography. *Multimedia Research, 2*(3), 23–32.
16. Gayathri Devi, K. S. (2020). Optimal reactive power dispatch for voltage stability improvement using enhanced whale optimization algorithm. *Journal of Computational Mechanics Power System and Control*. https://doi.org/10.46253/jcmps.v3i4.a3
17. Rao, I. V., & Malleswara Rao, V. (2019). An enhanced whale optimization algorithm for massive MIMO system. *Journal of Networking and Communication Systems, 2*(4), 12–22.
18. Anand, S. (2020). Intrusion detection system for wireless mesh networks via improved whale optimization. *Journal of Networking and Communication Systems*. https://doi.org/10.46253/jnacs.v3i4.a2
19. Zhang, K., & Tian, C. (2018). Fundamental limits of coded caching: from uncoded prefetching to coded prefetching. *IEEE Journal on Selected Areas in Communications, 36*(6), 1153–1164. https://doi.org/10.1109/JSAC.2018.2844958
20. Zhang, W., et al. (2019). Design and analysis of an effective two-step clustering scheme to optimize prefetch cache technology. *IEEE Access, 7*, 176438–176447. https://doi.org/10.1109/ACCESS.2019.2943498
21. Xu, H., Gong, C., & Wang, X. (2019). Efficient file delivery for coded prefetching in shared cache networks with multiple requests per user. *IEEE Transactions on Communications, 67*(4), 2849–2865. https://doi.org/10.1109/TCOMM.2018.2890262
22. George, A., Rajakumar, B. R. (2013) APOGA: An Adaptive Population Pool Size based Genetic Algorithm. In: AASRI Procedia - 2013 AASRI Conference on Intelligent Systems and Control (ISC 2013), 4, pp, 288–296. https://doi.org/10.1016/j.aasri.2013.10.043
23. Rajakumar, B. R., George, A. (2012) A New Adaptive Mutation Technique for Genetic Algorithm. In: proceedings of IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pp, 1–7, 18–20, Coimbatore, India. https://doi.org/10.1109/ICCIC.2012.6510293
24. Rajakumar, B. R. (2013). Impact of static and adaptive mutation techniques on genetic algorithm. *International Journal of Hybrid Intelligent Systems, 10*(1), 11–22. https://doi.org/10.3233/HIS-120161
25. Rajakumar, B. R. (2013). Static and adaptive mutation techniques for genetic algorithm: A systematic comparative analysis. *International Journal of Computational Science and Engineering, 8*(2), 180–193. https://doi.org/10.1504/IJCSE.2013.053087
26. Swamy, S. M., Rajakumar, B. R., Valarmathi, I. R. (2013) Design of Hybrid Wind and Photovoltaic Power System using Opposition-based Genetic Algorithm with Cauchy Mutation. In: IET Chennai Fourth International Conference on Sustainable Energy and Intelligent Systems (SEISCON 2013), Chennai, India. https://doi.org/10.1049/ic.2013.0361
27. Oh, Y., et al. (2019). Adaptive cooperation of prefetching and warp scheduling on GPUs. *IEEE Transactions on Computers, 68*(4), 609–616. https://doi.org/10.1109/TC.2018.2878671

28. Chandanapalli, S. B., Sreenivasa Reddy, E., & Rajya Lakshmi, D. (2019). Convolutional neural network for water quality prediction in WSN. *Journal of Networking and Communication Systems, 2*(3), 40–47.
29. Rim, M., & Kang, C. G. (2020). Content prefetching of mobile caching devices in cooperative D2D communication systems. *IEEE Access, 8*, 141331–141341. https://doi.org/10.1109/ACCESS.2020.3012442
30. Srinivas, V., & Santhirani, Ch. (2020). Hybrid particle swarm optimization-deep neural network model for speaker recognition. *Multimedia Research, 3*(1), 1–10.
31. Chithra, R. S., & Jagatheeswari, P. (2019). Enhanced WOA and modular neural network for severity analysis of tuberculosis. *Multimedia Research, 2*(3), 43–55.
32. Shaik, J. B., & Ganesh, V. (2020). Deep neural network and social ski-driver optimization algorithm for power system restoration with VSC - HVDC technology. *Journal of Computational Mechanics, Power System and Control, 3*(1), 1–9.
33. Zhao, J., Li, D., Pu, J., Meng, Y., Sbeih, A., & Hamad, A. A. (2022). Human-computer interaction for augmentative communication using a visual feedback system. *Computers and Electrical Engineering, 100*, 107874.
34. Smeein, S. B., Shihab, S., & Delphi, M. (2023). Operational Spline Scaling Functions Method for Solving Optimal Control Problems. *Samarra Journal of Pure and Applied Science*, *5*(2)
35. Parrinello, E., Ünsal, A., & Elia, P. (2020). Fundamental limits of coded caching with multiple antennas, shared caches and uncoded prefetching. *IEEE Transactions on Information Theory, 66*(4), 2252–2268. https://doi.org/10.1109/TIT.2019.2955384
36. Chen, Y., Li, C., Lv, M., Shao, X., Li, Y., & Xu, Y. (2019). Explicit data correlations-directed metadata prefetching method in distributed file systems. *IEEE Transactions on Parallel and Distributed Systems, 30*(12), 2692–2705. https://doi.org/10.1109/TPDS.2019.2921760
37. Ganfure, G. O., Wu, C.-F., Chang, Y.-H., & Shih, W.-K. (2020). DeepPrefetcher: A deep learning framework for data prefetching in flash storage devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39*(11), 3311–3322. https://doi.org/10.1109/TCAD.2020.3012173
38. Lee, S., Hyun, S. J., Kim, H. Y., et al. (2018). APS: Adaptable prefetching scheme to different running environments for concurrent read streams in distributed file systems. *The Journal of Supercomputing, 74*, 2870–2902. https://doi.org/10.1007/s11227-018-2333-6
39. Yu, G., & Wu, J. (2020). Content caching based on mobility prediction and joint user Prefetch in Mobile edge networks. *Peer-to-Peer Network, Appl., 13*, 1839–1852. https://doi.org/10.1007/s12083-020-00954-x
40. Al Assaf, M. M., Jiang, X., Qin, X., et al. (2018). Informed prefetching for distributed multi-level storage systems. *J Sign Process Syst, 90*, 619–640. https://doi.org/10.1007/s11265-017-1277-z
41. Lee, S., Hyun, S. J., Kim, H. Y., et al. (2018). Fair bandwidth allocating and strip-aware prefetching for concurrent read streams and striped RAIDs in distributed file systems. *The Journal of Supercomputing, 74*, 3904–3932. https://doi.org/10.1007/s11227-018-2396-4
42. Li, M., Zhang, H., Wu, Y., et al. (2019). Prefetch-aware fingerprint cache management for data deduplication systems. *Front Computer Science, 13*, 500–515. https://doi.org/10.1007/s11704-017-7119-0
43. Daniel, G., Sunyé, G., & Cabot, J. (2019). Advanced prefetching and caching of models with PrefetchML. *Software & Systems Modeling, 18*, 1773–1794. https://doi.org/10.1007/s10270-018-0671-8
44. Mohan, Y., Chee, S. S., Pei Xin D. K., Foong, L. P. (2016) Artificial Neural Network for Classification of Depressive and Normal in EEG. In: 2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)
45. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.
46. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.
47. Pedersen, M. E. H., & Chipperfield, A. J. (2010). Simplifying particle swarm optimization. *Applied Soft Computing, 10*(2), 618–628.
48. Wang, H., Wang, W., Zhou, X., Sun, H., & Cui, Z. (2017). Firefly algorithm with neighborhood attraction. *Information Sciences, 382–383*, 374–387.
49. Ragunathan, T., Sharfuddin, M. (2015) Frequent block access pattern-based replication algorithm for cloud storage systems. In: 2015 Eighth International Conference on Contemporary Computing (IC3), pp 7–12. https://doi.org/10.1109/IC3.2015.7346644.
50. Avci, E. (2009). A new intelligent diagnosis system for the heart valve diseases by using genetic-SVM classifier. *Expert Systems with Applications, 36*(7), 10618–10626.

Uppuluri Lakshmi Soundharya is a Research Scholar in the Department of Data Science and Business Systems at SRM Institute of Science and Technology, India. Her area of interest includes Deep Learning, Machine Learning, and Distributed File Systems. Also she has contributed original research articles in IEEE Transactions, SCI, SCIE, and Scopus indexed journals. She has done Various Certification Courses. She has Current membership in Professional organizations in IANEG, Annual Member CSI.

G. Vadivu received Ph.D from SRM Institute of Science and Technoloy, India, and working as Professor in the Department of Data Science and Business Systems at SRM Institute of Science and Technology, India. Her area of interest includes Deep Learning, Machine Learning, Social Network Analytics. Served Guest editor in several journals of Elsevier Publishers, Also She has contributed original research articles in IEEE Transactions,

SCI, SCIE, and Scopus indexed journals. she presented various International Conference events. She has done Various Certification Courses. She has Current membership in Professional organizations in Annual Member IET, Annual Member in ISC, Annual Member in ACM-W, Annual Member CSI, She has received Awards in Research Grant by Dalhousie University, Canada-Dalhousie University, Canada 2012.

Gogineni Krishna Chaitanya Associate Professor Department of Computer Science and Engineering Koneru Lakshmaiah Education Foundation, Vaddeswaram, 522502 Andhra Pradesh, India. He has published 34 articles in various SCI and Scopus indexed Journals. He received several Excellence awards and several best Researcher awards. His research interests include digital forensics, Biometrics, Authentication and Machine Learning.