Journal of Positive Sciences (JPS) eISSN: 2582-9351, https://positive-sciences.com JPS. 2026 Nov; 6(6): 23-39

Article info
Received on: 05.09.2026
Accepted on: 09.10.2026
Published on: 21.11.2026
doi: https://doi.org/10.52688/ASP192978

Research Article

Machine Learning-Based Prediction of Storage-Feasible Content-Defined Chunking Configurations for Data Deduplication

Hashem Bedr Jehlol [1,*]

[1] Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq

[*] hashemhbg@uomustansiriyah.edu.iq

ABSTRACT

Content-defined chunking (CDC) preserves reusable regions after file modifications, but the storage benefit and metadata cost depend strongly on the selected chunking parameters. This study evaluates whether a machine-learning model can predict, before a later file version is processed, whether a deterministic CDC configuration will remain close to a storage-oriented reference. Five Gear-based configurations (C0-C4) were evaluated, with C0 used as the reference. A candidate was labeled storage-feasible when its loss in Version-B storage saving did not exceed 1.0 percentage point relative to C0. The model used 33 inputs available from Version A only. Training used 632 changed pairs from Linux 3.16, Linux 5.5, and backup version families. Extra Trees was selected by grouped five-fold out-of-fold evaluation and then frozen. On unseen Linux 5.7, the model achieved ROC-AUC 0.9419, PR-AUC 0.9121, precision 0.9924, and MCC 0.8534. On the independent 3DRDF final test, ROC-AUC was 0.8005, PR-AUC 0.8710, precision 0.9356, and MCC 0.6060. The fixed configurations exposed a clear storage-metadata trade-off: on 3DRDF, C0 saved 63.3353% of Version-B bytes, whereas C4 reduced the A+B chunk count by 76.8461% but saved 41.7543%. The use of a non-deployable descriptive oracle yielded 11.5729% reduction in chunks while still achieving 63.5649% savings on the frozen 1.0-point tolerance. However, no zero-violation automatic switching rule generalized across all training domains. These findings show that machine learning can be an effective tool for prediction and decision-making signals, while switching is a different problem.

KEYWORDS: Data deduplication; content-defined chunking; machine learning; storage feasibility; Extra Trees; metadata reduction

INTRODUCTION

Backup systems, source-code archives, cloud storage, and versioned data sets often contain repeated content. Chunk-level deduplication reduces this repetition by splitting a file into chunks, hashing the chunks, and storing one copy of each identical chunk [1-4]. The result depends strongly on where the chunk boundaries are placed.

Fixed-size chunking is simple, but an insertion or deletion shifts later boundaries. CDC avoids this problem by choosing boundaries from file content. FastCDC made this approach practical with a Gear-style hash and controlled chunk sizes [3]. Other work improved speed or stability with asymmetric extrema, parallel processing, duplicate locality, frequency rules, dynamic masks, hashless scans, and vector instructions [5, 7, 8, 11, 12, 20, 23, 25, 26].

The parameter choice still matters. Small chunks can find more duplicate regions, but they create more fingerprints and index entries. Large chunks reduce metadata, but they may miss reusable content. The system therefore has two goals that can conflict: storage saving and metadata reduction.Machine learning

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

algorithms have been used in deduplication as a way to make decisions about modes of operation, indexing, prefetching, resemblance searches [6, 9, 13, 14, 18, 22]. The function that is being studied in this research project is narrow down to the prediction of probability. The model does not make byte boundaries by itself and it does not replace the functionality of the determinist engine of CDC. Prediction of whether this particular setup would be within budget limits is what is being predicted.This experiment is strictly experimental in nature. Feature and model selection takes place using training data set, the final model of Extra Trees is frozen, version of Linux 5.7 is used as unseen validation dataset, and 3DRDF is used as the final independent test dataset. The Version B is required to create training label and to measure results, however, it is never used as an input in prediction time.

These include: (1) a file-level storage feasibility target for the evaluation of deterministic CDC operational configurations; (2) the design of a 33-feature Version-A format which has no future-version data; (3) a reproducible Gear family of configurations with clear feasibility labels; (4) group-wise training which maintains file path association; (5) evaluation on a never-before-seen Linux series as well as the independent 3DRDF dataset; and (6) transparent reporting of both the positive finding of feasibility prediction as well as the negative finding of no zero-violating auto-switching strategy. This research thus analyzes the predictive accuracy and headroom for metadata without offering the descriptive oracle as a viable learned policy.

RELATED WORK

CDC AND DETERMINISTIC CHUNKING

LBFS showed that content-derived boundaries can preserve common regions after edits [1]. Eshghi and Tang later described a general framework for content-based chunking [2]. FastCDC [3] used a simple Gear hash, minimum and maximum chunk limits, and normalized chunk sizes. The authors reported about ten times the throughput of an open-source Rabin-based CDC and about three times the throughput of Gear/AE while keeping almost the same deduplication ratio.

AE used an asymmetric extremum rule and reported more than 2.3 times higher chunking throughput than the then state-of-the-art methods with comparable deduplication efficiency [5]. Recent work focuses strongly on fast boundary search. SeqCDC reported 1.5-3.1 times higher chunking throughput at larger chunk sizes with similar space savings [20]. VectorCDC reported 21-46 times higher throughput than earlier vector acceleration techniques without changing space savings [23]. The 2026 vectorized studies continue this direction [25, 26].

LEARNING IN DEDUPLICATION

Learning has usually been added outside the CDC parameter decision. SDM learns a deduplication mode for mobile storage [6]. LIPA learns indexing and prefetching decisions [9]. SegDup uses reinforcement learning at the deduplication-strategy level [13]. DeepSketch, Chunk2vec, and a context-aware neural method learn how to find similar chunks after chunking [14, 18, 22]. These methods show that learning can support data reduction, but their learned decision is not the same as the storage-feasibility target used here.

COMPARISON WITH PREVIOUS STUDIES

Table 1 summarizes representative CDC and learning-based directions. The reported speed and data-reduction numbers are taken from the original studies and were obtained on different platforms and workloads. They are included only to position the present contribution. The closest practical alternatives to the learned predictor are fixed deterministic configurations and transparent Version-A rules; the present study does not claim a same-hardware speed advantage over FastCDC, SeqCDC, VectorCDC, or other optimized CDC implementations.

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

Table 1. Comparison with representative previous studies. Reported speed values are taken from the cited papers and are not a same-platform benchmark.

Study	Year	Main goal	Method / learned part	Reported result or role	Relation to this study
FastCDC [3]	2016	Fast CDC	Gear-based deterministic CDC	~10x Rabin CDC; ~3x Gear/AE; similar deduplication ratio	Reference for practical deterministic CDC
AE [5]	2017	Fast/stable CDC	Asymmetric extrema	>2.3x chunking throughput with comparable deduplication efficiency	Deterministic boundary rule
DeepSketch [14]	2022	Resemblance search	Learning-to-hash after deduplication	Up to 33% higher data-reduction ratio; 21% average improvement over a state-of-the-art post-deduplication method	Learning is after chunks are created
SeqCDC [20]	2024	Fast hashless CDC	Content-based skipping	1.5-3.1x higher chunking throughput at larger chunk sizes with similar space savings	Focuses on speed, not parameter feasibility
VectorCDC [23]	2025	Vectorized CDC	SSE/AVX acceleration	21-46x higher throughput than earlier vector acceleration techniques; space savings unchanged	Hardware acceleration, not learned selection
CARD [22]	2025	Resemblance detection	Neural context after chunking	Up to 75.03% more data removal and 5.6x-86.7x faster resemblance detection (reported by the original study)	Different ML stage
Present work	2026	Feasibility prediction	Extra Trees on Version-A features	Validation ROC-AUC 0.9419; final-test PR-AUC 0.8710	Predicts whether a deterministic CDC configuration stays near C0 storage saving; no autonomous policy is claimed

METHOD

DETERMINISTIC CDC CONFIGURATION SPACE

The five configurations use the same deterministic Gear table, SHA-256 chunk fingerprints, and duplicate-detection procedure; only the mask and chunk-size limits change. The Gear table contains 256 unsigned 64-bit values generated once with Python random seed 2026. While scanning byte b, the rolling value is updated as h = ((h << 1) + T[b]) mod 2^64. A position is a candidate when h AND (2^m - 1) = 0, where m is the configuration mask width. After a cut, only candidates between the minimum and maximum chunk sizes are eligible.

The one among the candidate that has the minimum distance from the target chunk size is chosen; tie is broken in favor of the shorter distance from the current chunk point and then on the position being closer. In case there is no eligible candidate, a compulsory boundary is imposed at the maximum size limit. The remainder below the maximum size is outputted in a single chunk. Table 2 gives the five operation points. C0 had the most storage savings in training and is only used as a storage benchmark. C0 and C1 intentionally share the same 10-bit candidate mask but use different minimum, target, and maximum sizes. The mask controls candidate density, whereas the target controls which eligible candidate is preferred inside the admissible window. Therefore, equal mask widths do not imply equal chunk boundaries or equal average chunk sizes.

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

Table 2. Deterministic CDC configurations used in the experiments.

Config.	Mask bits	Minimum (KB)	Target (KB)	Maximum (KB)
C0	10	1	4	16
C1	10	2	6	24
C2	11	2	8	32
C3	12	4	12	48
C4	12	4	16	64

Figure 1 shows the key separation in the method.

Decision-time path (Version A)	Training / evaluation path (Version B)
Version A file ↓ 33 Version-A features + candidate configuration ↓ Frozen Extra Trees: p(feasible) ↓ Feasibility estimate / decision-support output	Version B file ↓ Exact chunk reuse under C0-C4 ↓ Feasibility label during training OR Storage outcome during validation / final test
Version B is used to create labels and evaluate outcomes. It is never used as a model input.	

Figure 1. Overall feasibility-prediction framework. Version B is used to create labels and evaluate outcomes, not as a model input.

The model uses Version A only. Version B is needed during training to say whether the candidate configuration was actually storage-feasible. At evaluation time, Version B is again used only to measure what happened after the model was already frozen.

VERSION-A FEATURES

Each file pair produces four candidate rows, one for C1-C4. The final model uses 33 inputs. Twenty-three are common Version-A descriptors. File size is measured in KiB and is also represented by log2(max(file size in bytes, 1)). For files larger than 96 KiB, a 32-KiB block is taken from the beginning, middle, and end and concatenated; smaller files are used in full. From this sample the implementation computes Shannon entropy, zero-byte ratio, printable-byte ratio (ASCII 32-126 plus tab, line feed, and carriage return), whitespace ratio (tab, line feed, carriage return, and space), high-byte ratio (byte values >=128), adjacent-byte repetition ratio, byte mean, byte standard deviation, and a zlib compression ratio using compression level 1. These definitions are fixed for all datasets.

The remaining Version-A descriptors characterize CDC candidate structure for mask widths 10, 11, and 12. For each mask, the implementation records candidate density per MiB and the mean, median, and population standard deviation of consecutive candidate gaps in KiB, giving 12 mask-dependent values. Together with the 11 sample/file descriptors, these form 23 common features. Ten additional inputs describe the evaluated configuration and its immediate Version-A effect: configuration index, mask bits, minimum/target/maximum size in KiB, C0 chunk count, candidate-configuration chunk count, absolute

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

chunk reduction, percentage chunk reduction, and the candidate-to-C0 chunk-count ratio. All 33 inputs are available before Version B is processed. The ratio of the mask matches and the file size, denoted as N/M, represents the candidate density with N being the mask matches and M being the file size in MiB. The gap statistics are calculated using the candidate offset values that have been converted into KiB, and if less than two candidate values exist, the three gap statistics values would be set to zero.

STORAGE-FEASIBILITY LABEL

For all the configurations, version A is chunked first and the chunk SHA-256 hashes for version A are populated. Version B is subsequently chunked using the same configuration. Version B chunk is considered to be saved if the SHA-256 hash of that chunk already exists in version A or if the same chunk was present earlier in version B itself. Therefore, the amount of saving is in terms of saved bytes of version B and it is not a dedupe elimination ratio for the whole repository.

$$\text{Saving}(c) = \text{SavedBytes_B}(c) / \text{Size_B} \times 100 \quad (1)$$

To preserve the direction of the comparison, the signed saving difference is defined first as DeltaS(c) = Saving(c) - Saving(C0). Positive values mean that the candidate saves more Version-B bytes than C0, zero means a tie, and negative values mean a storage penalty. The binary feasibility label uses only the adverse part of this difference.

$$\text{DeltaS}(c) = \text{Saving}(c) - \text{Saving}(C0) \quad (2)$$

$$L(c) = \max(0, -\text{DeltaS}(c)); \; y(c) = 1 \text{ if } L(c) <= 1.0, \text{ otherwise } 0 \quad (3)$$

The tolerance of 1.0 was set during the training phase and remained fixed even after opening Linux 5.7 and 3DRDF. It was possible to make improvements and ties, however, these two options were not considered equivalent for the descriptive oracle when choosing the candidate. The oracle first filtered out those candidates whose L(c) falls within the tolerance range and then chose the one with the lowest A + B chunks; ties were resolved according to the number of saved bytes of Version-B followed by a deterministic order of configurations.

LEARNING, GROUPING, AND POLICY INTERPRETATION

Tree classifiers included Random Forest, Extra Trees, HistGradientBoosting, and XGBoost classifiers [27]-[29]. Five Fold StratifiedGroupKFold, shuffle=True, seed=2026 was chosen as the cross-validation approach. Note that the true value of a group ID for Linux is "Linux|Relative_Path". Thus, it will not happen that the same relative path from Linux 3.16 and Linux 5.5 would be present in different folds since the same relative path for each pair of files will have identical ID values. For Backup, it will be "Backup|Family|Relative Path". Each file pair corresponds to four possible rows. Hence, they cannot be split into folds.

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

Changed version pairs A → B	Four candidate rows C1-C4 per pair
Extract 33 Version-A features	Build feasibility label from Version-B storage loss
Grouped 5-fold OOF by file path / family	Compare RF, ET, HGB, and XGBoost
Select Extra Trees using training PR-AUC	Fit on 632 training pairs and freeze model/tolerance
One-pass evaluation after freezing: Linux 5.7 validation → 3DRDF final test	

Figure 2. Model learning, selection, freezing, and one-pass evaluation.

Figure 2 shows the full learning path. Extra Trees was selected using training out-of-fold PR-AUC, then fitted on all expanded training rows and frozen. None of the model, features, parameters, configuration, nor probability cut-off value was optimized under Linux 5.7 or 3DRDF. The 0.5 probability cut-off value is used only for calculating diagnostic indicators. A two-phase deployment strategy was considered in training too: the prediction was required to be a feasible one and it should reduce Version-A chunks compared to C0. However, no cut-off value ensured neither violations in storage size nor any chunk reduction for all training domains. Therefore, no automated selector is offered as a full system in this work, but just as feasibility indicator.

NOTATION

Table 3. Main notation used in the paper.

Symbol	Meaning
A, B	Earlier and later file versions
C0	Storage-oriented reference configuration
c	Candidate configuration C1-C4
SavedBytes_B(c)	Version-B bytes removed by exact chunk reuse
L(c)	Storage loss versus C0 in percentage points
y(c)	Binary storage-feasibility label
Chunks_A(c)	Version-A chunk count under configuration c
p(c)	Predicted feasibility probability
DeltaS(c)	Signed storage-saving difference Saving(c) - Saving(C0)

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

EXPERIMENTAL SETUP

FULL DATASET POOL

The project began with five distinct workload pools. Table 4 shows the discovered inventory since it is the record of data accessible to the research project and not necessarily because all files have been used for the supervised learning process. The feasibility label needs an established version-to-version connection and therefore, Linux, Backup, and 3DRDF pools were used in the redesign of the supervised learning process. The SQLite and Email pools were only inventory pools since there was no trustful A-to-B connection.

Table 4. Full discovered dataset pool and its role in the final feasibility-prediction study.

Dataset	Discovered files	Size (GiB)	Short description	Role in final study
SQLite	191,697	6.528	Structured/database-oriented files	Not used for supervised final model; no reliable final A->B mapping
Linux_used_10.9	909,083	10.924	Large versioned Linux/source workload	Linux 3.16 and 5.5 for training; Linux 5.7 for validation
allData_3drdf	5,803	2.275	Small file count with relatively larger versioned objects	Independent final test
Backup	691,581	30.826	Largest workload by capacity; repeated application/version folders	Training expansion from real adjacent versions
Email	517,400	1.324	Very large number of small email-oriented files	Not used for supervised final model; no reliable final A->B mapping
Total	2,315,564	51.877	Complete discovered corpus	Inventory before final pair construction

Discovered corpus: 2,315,564 files / 51.877 GiB			
Training	Validation	Final test	Inventory only
Linux 3.16 + 5.5 407 pairs + Backup 225 pairs	Linux 5.7 166 changed pairs 664 candidate rows	3DRDF 350 changed pairs 1,400 candidate rows	SQLite + Email not used in supervised model (no reliable final A→B pairing)

Figure 3. Use of the available datasets in the final feasibility-prediction experiment.

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

The reason for not using every available file in supervised learning is methodological, not computational. A classifier should be trained from pairs where the true future storage result can be measured. Adding unrelated files would increase sample count but would not produce a valid feasibility label.

FINAL PAIR SPLIT

There are 407 changed pairs in the frozen Linux training manifest, which consists of 213 pairs from Linux 3.16 and 194 pairs from Linux 5.5. This training expansion was generated using the folders of the backup application in an ordered manner. The adjacent pairs were selected based on identical paths. All the files ranging from 2 KiB to 4 MiB were selected as part of the training expansion. A common-path pair was marked changed when file sizes differed or, for equal sizes, when SHA-256 hashes differed. Eight Backup families contributed the final 225 changed expansion pairs: adminpharm, elearn, evening, journal, postgrad, rpcrconf, sarmad, and snfpaconf. The same relative path across several adjacent versions of one family remained in a single cross-validation group.

Linux 5.7 was used only for frozen validation and supplied 166 changed pairs, or 664 candidate rows. The 3DRDF corpus was paired across seven adjacent version transitions. Across these transitions, 1,276 relative paths were common to both sides of a pair; 350 were changed and 926 were identical. All 350 changed 3DRDF pairs were used in the final classifier test; they were not sampled from 1,276 candidates. The 926 identical pairs were retained in the inventory but excluded from classifier metrics because their unchanged outcome would make the feasibility task artificially easy. The final test therefore contains 1,400 candidate rows.

Table 5. Final training, validation, and test allocation.

Split	Source	Changed pairs	Candidate rows	Use
Training	Linux 3.16 + Linux 5.5	407	1,628	Model selection
Training expansion	Eight contributing Backup families	225	900	Increase domain diversity
Validation	Linux 5.7	166	664	Frozen validation
Final test	3DRDF	350	1,400	Independent final evaluation

IMPLEMENTATION AND REPRODUCIBILITY

The experiments were run on a Windows 64-bit workstation with 22 logical CPU cores, 31.47 GB RAM, and SSD storage. The environment used Python 3.11.0, NumPy 2.4.6, Pandas 3.0.6, scikit-learn 1.9.1, SciPy 1.17.1, XGBoost 3.2.0, and joblib 1.6.0. The Gear table was generated once with random seed 2026 and then reused. SHA-256 was used for exact chunk fingerprints.

Random Forest and Extra Trees used 600 trees, minimum leaf size 4, max_features=0.8, and class balancing. HistGradientBoosting used learning_rate=0.04, max_iter=350, max_leaf_nodes=15, min_samples_leaf=15, balanced class weights, and L2 regularization 1.0. XGBoost used 500 trees, depth 4, learning rate 0.03, subsample 0.85, column subsample 0.85, min_child_weight 3, L2 regularization 2.0, and histogram tree construction. The random seed was 2026 throughout.

Output files for pairs have been generated using the experiment code, while the aggregate tables and figures have been generated using these results only. Thus, the published aggregations will be consistent with the

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

same pair-level output files, and there will be no errors due to copying manually. Deterministic Gear table has been calculated from seed 2026, and configuration details and versions are listed below.

EVALUATION MEASURES

Both ROC-AUC and PR-AUC have been selected to measure the rankings since the feasibility prevalence varies for different datasets. Precision, Recall, F1 Score, Balanced Accuracy, and Matthews Correlation Coefficient (MCC) have been stated with respect to the fixed diagnostic threshold of 0.5. Storage saving is the proportion of version B byte size saved using exact chunk re-use. Metadata proxy is the total number of chunks for both A and B versions for a certain candidate configuration.

A+B chunk count is a controlled experiment and a proxy, not a model of persistent repository metadata. A-B pairs are reprocessed independently under each configuration candidate; the experiment does not move an existing Version A from one configuration to another, nor does it include any re-chunking, migration, index conversion, or repository sequence. The descriptive oracle picks for each A-B pair the feasible configuration with minimum A+B chunk count, based on the actual result of Version B. Hence, it assesses only headroom and is not an actionable AI result.

RESULTS

TRAINING MODEL SELECTION

The expanded training set contained 2,528 candidate rows. Of these, 1,295 rows, or 51.23%, were feasible at the 1.0-point tolerance. Table 6 shows the grouped out-of-fold results. All four tree models learned a strong signal. Extra Trees had the highest PR-AUC and was selected before validation.

Table 6. Grouped five-fold out-of-fold performance on the expanded training set.

Model	ROC-AUC	PR-AUC	MCC at 0.5
Extra Trees	0.9471	0.9636	0.8246
XGBoost	0.9463	0.9628	0.8045
Random Forest	0.9444	0.9623	0.8106
HistGradientBoosting	0.9407	0.9596	0.8157

The four tree-based models give similar ranking results. This suggests that the predictive signal is not specific to one estimator. Extra Trees had the highest PR-AUC and was selected before validation; after that point, the model and the 1.0-point tolerance were frozen.

LINUX 5.7 VALIDATION

Linux 5.7 was the first unseen series in the frozen evaluation pipeline. Only 24.85% of its 664 candidate rows were feasible. Extra Trees achieved ROC-AUC 0.9419 and PR-AUC 0.9121. At threshold 0.5 it produced 130 true positives, one false positive, 35 false negatives, and 498 true negatives; precision was 0.9924, recall 0.7879, F1-score 0.8784, balanced accuracy 0.8929, and MCC 0.8534. The single false positive is important because a feasibility false positive is the error type that can expose a storage controller to an excessive loss.

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

DRDF FINAL TEST

The 3DRDF final-test in the independent version was more different from the training dataset, and the domain shift is obvious. With 1,400 candidate rows, the frozen model managed to obtain ROC-AUC 0.8005 and PR-AUC 0.8710. With threshold 0.5, precision was 0.9356, recall 0.6292, F1-Score 0.7524, balanced accuracy 0.7904, and MCC 0.6060, with 465 true positives, 32 false positives, 274 false negatives, and 629 true negatives. Rank of configurations improved with increase of size of the configuration: Final-test ROC-AUC went from 0.7473 in C1 up to 0.8401 in C4, and PR-AUC from 0.8569 to 0.8892. Hence, the lower 3DRDF performance does not support a universal transfer claim. Table 7 gives the main unseen-data results.

Table 7. Frozen Extra Trees performance on unseen data.

Split	Rows	Feasible prevalence	ROC-AUC	PR-AUC	Precision	Recall	F1	MCC
Linux 5.7 validation	664	0.2485	0.9419	0.9121	0.9924	0.7879	0.8784	0.8534
3DRDF final test	1,400	0.5279	0.8005	0.8710	0.9356	0.6292	0.7524	0.6060

Figure 4 shows the change from training out-of-fold evaluation to validation and final testing. The drop on 3DRDF is important and is reported directly. The model still ranks feasible configurations better than chance, but the domain change is visible.

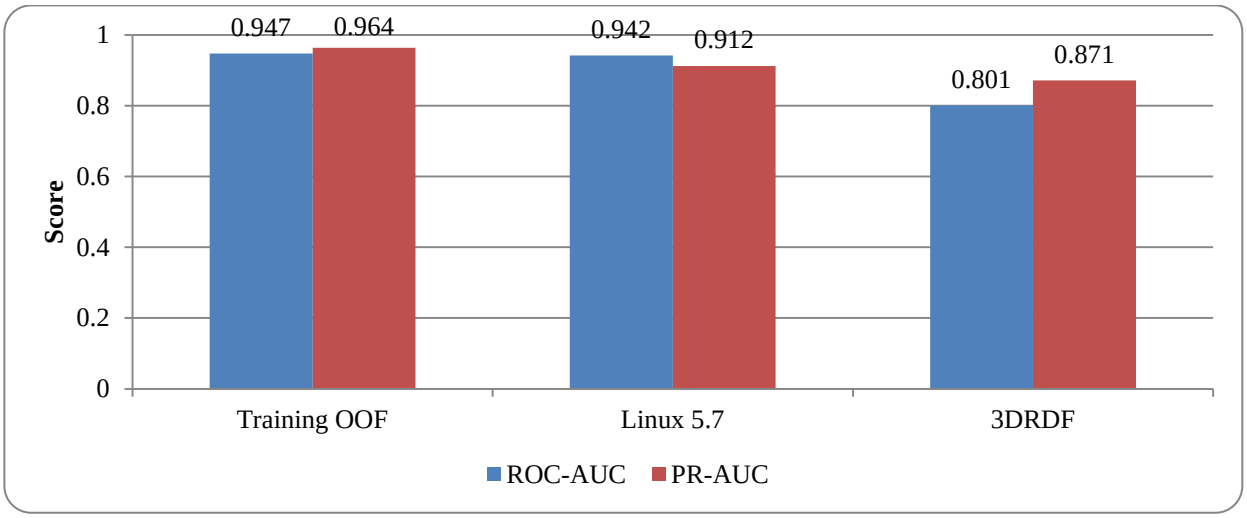


Figure 4. Ranking performance of Extra Trees on Training OOF, Linux 5.7 validation, and 3DRDF final test.

FIXED CONFIGURATION TRADE-OFF

It can be seen from the fixed configurations how important it is to choose a configuration even before the consideration of any learned policy. C0 on Linux 5.7 saved 70.0543% of the Version B memory bytes. C4 decreased the amount of chunks of A+B by 80.5007%, but the memory savings became only 22.4951%. C0 on 3DRDF saved 63.3353%; C4 decreased the chunks by 76.8461%, but memory savings became only 41.7543%. C1-C3 fall somewhere in between. This calls for feasibility predictions but does not prove that automatic switching is useful.

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

Table 8. Fixed CDC results on Validation and Final Test.

Split	Config.	Storage saving (%)	A+B chunks	Chunk reduction vs C0 (%)
Linux 5.7	C0	70.0543	3,036	0.0000
Linux 5.7	C1	57.4281	1,877	38.1752
Linux 5.7	C2	48.0605	1,298	57.2464
Linux 5.7	C3	34.2225	806	73.4519
Linux 5.7	C4	22.4951	592	80.5007
3DRDF	C0	63.3353	12,296	0.0000
3DRDF	C1	56.7268	8,106	34.0761
3DRDF	C2	52.6806	5,774	53.0416
3DRDF	C3	46.6972	3,681	70.0634
3DRDF	C4	41.7543	2,847	76.8461

Figure 5 plots the 3DRDF trade-off. The curve makes the main engineering problem easy to see: fewer chunks are possible, but the storage cost can become too large if the configuration is chosen without checking feasibility.

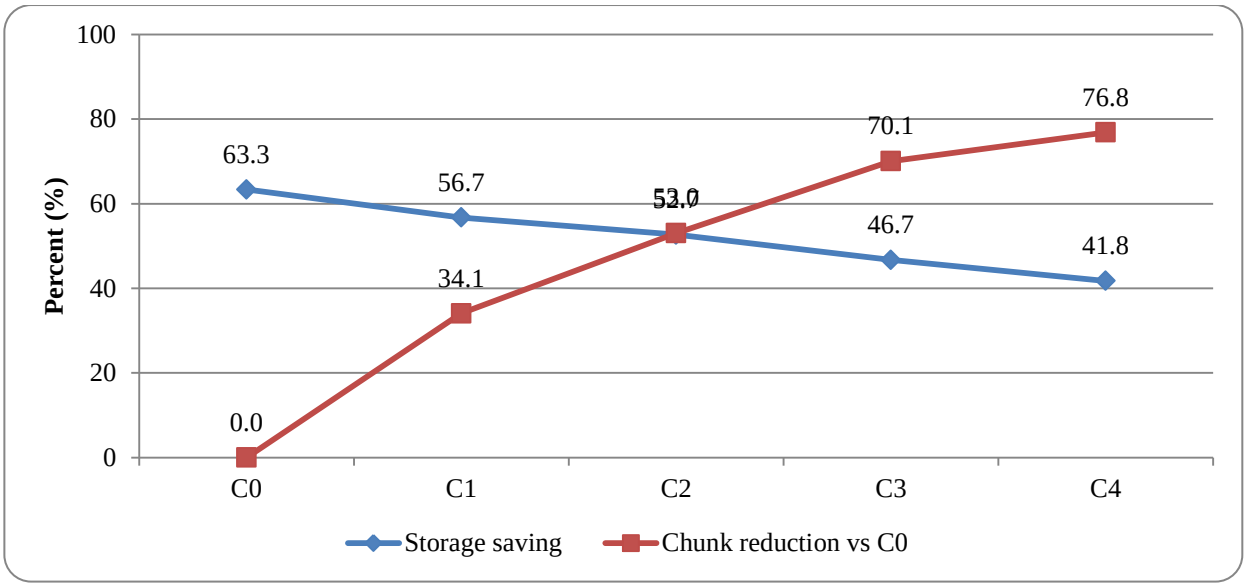


Figure 5. Storage-metadata trade-off of C0-C4 on the 3DRDF final test.

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

DESCRIPTIVE ORACLE AND POLICY LIMITATION

The oracle is used only to ask whether useful pair-specific headroom exists. On Linux 5.7, the zero-loss oracle switched 15 of 166 pairs, reduced A+B chunks by 4.8748%, and improved aggregate storage saving by 0.2695 percentage point. At the frozen 1.0-point tolerance, it switched 20 pairs and reduced chunks by 9.9473%, while aggregate storage saving remained 0.2158 point above C0. Because these choices use the actual Version-B outcome, they are not achievable predictions and must not be interpreted as model performance.

The independent 3DRDF test showed the same descriptive headroom. The zero-loss oracle reduced chunks by 7.9131% and improved aggregate storage saving by 0.3080 point. At 1.0 point it reduced chunks by 11.5729%, with aggregate storage saving still 0.2296 point above C0. These values establish that better per-file operating points exist in hindsight; they do not establish that the frozen classifier can select them without risk.

Table 9. Descriptive storage-metadata oracle on unseen splits.

Split	Tolerance (pp)	C0 storage (%)	Oracle storage (%)	Chunk reduction (%)	Switched pairs
Linux 5.7	0.0	70.0543	70.3239	4.8748	15
Linux 5.7	1.0	70.0543	70.2702	9.9473	20
3DRDF	0.0	63.3353	63.6433	7.9131	99
3DRDF	1.0	63.3353	63.5649	11.5729	103

It has been shown that there are indeed practical file-level decisions to be made, and the learned policy could not get to that level of performance. When training was increased from 407 to 632 data pairs, none of the models and thresholds evaluated managed to satisfy the criteria of no violations and positive A+B chunk compression at once in Linux 3.16, Linux 5.5, and Backup environments. Thus, the practical output of this work is feasibility assessment and risk-sensitive decision-making support.

QUANTITATIVE COMPARISON AND BASELINE CONTEXT

Table 10 provides quantitative context rather than a head-to-head ranking. The external numbers are those reported by the original papers and were obtained on different datasets, hardware, and tasks. The same-platform baselines available in this experiment are C0-C4 and the no-skill ranking baseline. A simple hand-built size/entropy switching rule was not evaluated under the frozen protocol; this is therefore reported as a limitation rather than replaced by an unfair post-hoc rule tuned on the unseen sets.

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

Table 10. Quantitative context with representative CDC and learning-based data-reduction studies. Published values are reported by the cited papers and are not a same-platform benchmark.

Study	Stage / main purpose	Published quantitative result	Directly comparable?	Position relative to this work
FastCDC [3] (2016)	Deterministic CDC	~10× Rabin CDC and ~3× Gear/AE, with nearly the same deduplication ratio	No - different platform and objective	Strong practical CDC speed reference; no learned configuration feasibility
DeepSketch [14] (2022)	ML reference search after deduplication	Data-reduction ratio improved by up to 33% (21% average) over a state-of-the-art delta-compression reference search	No - post-deduplication task	Shows that learning can improve data reduction after chunk creation
SeqCDC [20] (2024)	Hashless CDC throughput	1.5×-3.1× higher chunking throughput at larger chunk sizes with similar space savings	No - different platform and metric	Speed-oriented CDC; uses deterministic boundary logic
CARD [22] (2025)	Neural resemblance detection after chunking	Up to 75.03% more data removal and 5.6×-86.7× faster resemblance detection	No - different task and platform	Learning uses chunk context after chunks already exist
VectorCDC [23] (2025)	SIMD acceleration of hashless CDC	21×-46× higher throughput than earlier vector acceleration techniques, without changing space savings	No - hardware acceleration result	Orthogonal to configuration feasibility prediction
Vectorized SeqCDC [25] (2026)	Vectorized sequence-based CDC	10× higher throughput than unaccelerated and 1.2×-1.35× higher than vector-accelerated CDC, with minimal space-savings change	No - different platform and objective	Recent throughput reference; parameters remain deterministic
Present work (2026)	Pre-application storage-feasibility prediction for C1-C4	Linux 5.7: ROC-AUC 0.9419, PR-AUC 0.9121. 3DRDF: ROC-AUC 0.8005, PR-AUC 0.8710. Final-test zero-loss oracle: 7.9131% fewer chunks with +0.3080 pp storage saving.	Yes for internal baselines in Tables 7-9; not for external speedups	Adds a learned feasibility signal before Version B is known; no zero-violation automatic switching policy is claimed

The comparison shows that recent CDC work mainly improves chunking speed, while DeepSketch and CARD use learning after chunk creation. The present model addresses a different decision: whether a larger deterministic CDC configuration is likely to remain close to the storage saving of C0. Because these objectives are different, the strongest evidence for the AI component is the same-dataset comparison with a no-skill baseline shown in Figure 6.

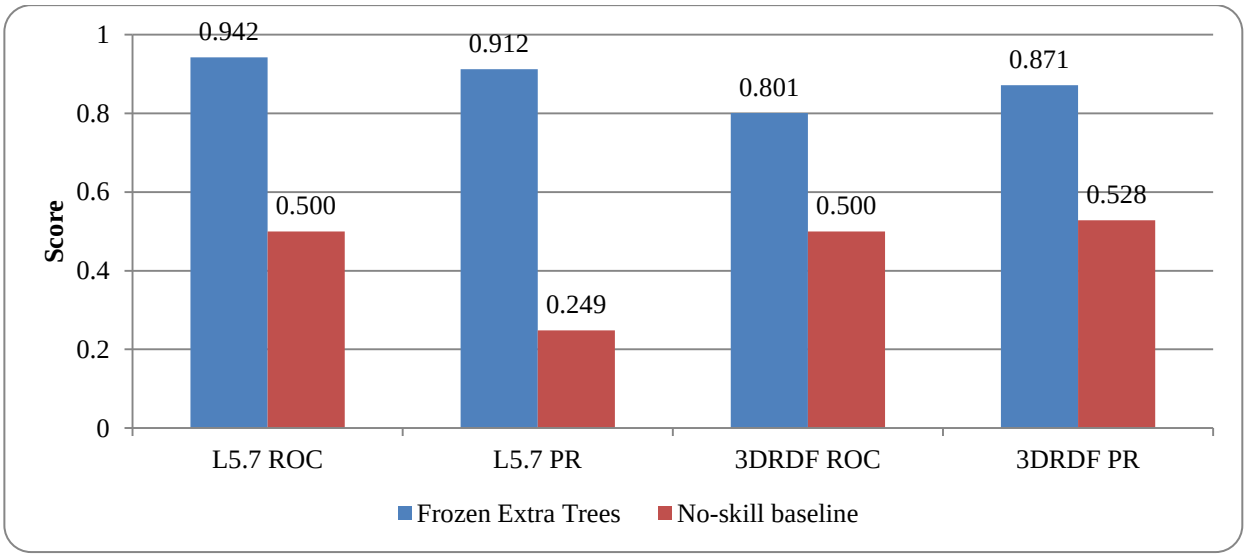


Figure 6. Frozen Extra Trees versus a no-skill ranking baseline on the same unseen data. For PR-AUC, the no-skill baseline equals the positive-class prevalence.

A random ranking has ROC-AUC 0.5. Its expected PR-AUC equals the fraction of feasible rows. On Linux 5.7, the frozen model reached ROC-AUC 0.9419 and PR-AUC 0.9121, compared with no-skill baselines of 0.5 and 0.2485. On 3DRDF, the corresponding values were 0.8005 and 0.8710, compared with 0.5 and

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

0.5279. This confirms that Version-A features contain useful predictive information. It does not remove the separate requirement for a zero-violation deployment policy.

DISCUSSION

WHAT WAS LEARNED FROM THE AI MODEL

The primary result from the AI is specifically limited: Information in Version-A has some useful signal regarding whether the deterministic CDC operating point will stay close to C0 during future storage savings. This is a good Linux 5.7 result, as the model was frozen prior to this evaluation suite. The low 3DRDF values mean that the signal is non-invariant under domain shift, but PR-AUC 0.8710 and precision 0.9356 mean that there is some ranking signal on a very different domain.

It does not make chunk boundaries. It decides if a specific deterministic configuration is feasible. The consequence of this is that the chunking algorithm is predictable and failure analysis is easy: candidate chunk numbers give us the initial Version-A metadata signal, and the content statistics and candidate gap features tell us if that potential gain is too costly in storage for the next version.

WHY GOOD CLASSIFICATION IS NOT SUFFICIENT FOR SWITCHING

Classification algorithm can make several errors but have high ROC-AUC and PR-AUC metrics at the same time. As for a storage controller, the cost imbalance is even more severe since one wrong prediction may lead to going beyond storage loss limits. Therefore, good ranking and absence of violation-free switching rule are not mutually exclusive for this reason. The prevalence ranges from 0.2485 for Linux 5.7 to 0.5279 for 3DRDF while false positives increase from 1 to 32. Thus, the controller in use will have to be adjusted and handle uncertainty with respect to the domain and prevalence change validation.

In the practical use case, the current classification algorithm will be used solely as a suggestion. In the future, the controller can use calibrated probabilities, abstention regions, deterministic validation, and safe checks before reconfiguration. Current experiments cannot be regarded as validation for the controller, therefore, zero-risk statements in the paper are excluded.

RELATION TO PREVIOUS STUDIES

In contrast, the speed-centric CDC algorithms in Table 1 are attempting to address a distinct issue. FastCDC, SeqCDC, and VectorCDC increase the speed of deterministic boundaries discovery while attempting to retain deduplication behavior [3, 20, 23]. The current research explores the possibility that characteristics of Version A can predict the storability of a number of deterministic boundaries. Both approaches complement each other in the sense that an optimized CDC core could be used subsequently under a feasibility predictor.

The learning-based similarity methods of DeepSketch, Chunk2vec, and CARD are applied when chunks have been discovered [14, 18, 22]. These approaches attempt to detect similar chunks for delta compression. The model in the paper is done earlier in time and attempts to predict the feasibility of a particular CDC configuration. Both approaches complement each other as opposed to being direct competitors.

As seen in Table 10, the quantitative context makes the distinction clear. External literature cites speed improvement or data reduction ratio achieved post-deduplication on their respective platforms, as opposed to feasibility prediction and same-platform metadata measurements reported in this paper.

LIMITATIONS

There are numerous issues with the study. First, the study has used only five controlled configurations of Gear, which means that C_0 is an optimum for this family but not universally so. Second, the 1.0-point

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

tolerance is an engineering constant that cannot be changed during training. The zero loss analysis is provided as a sensitivity check. Third, the model is coded in Python and does not make any claims regarding feature extraction performance and throughput as compared to an optimized C/C++ or SIMD-based approach. Fourth, the final classifier does not provide an automatic switching rule without violations for all the training domains. Fifth, no size/entropy post-hoc rule, calibration, feature groups ablation or pair-level bootstrap CI was performed, since doing so would defeat the purpose of having the evaluation separate from the tuning phase.

There are limitations to the use of the dataset and its metadata. Neither SQLite nor Email could be utilized in the supervised model due to the fact that the feasibility label demands reliable version pairs. A+B chunk count serves as a controlled proxy for fingerprint/index pressure and not an accurate byte representation of the persistent repository metadata; neither is there a model for the migration in case the repository changes chunking strategy across consecutive versions.

CONCLUSION

Machine-learning based feasibility predictions for content-defined chunking have been examined in this study. Five deterministic Gear configurations were assessed using an identical chunking and exact reuse process. The learning algorithm has been applied on The model used 33 Version-A inputs and was trained on 632 changed pairs. The performance of Extra Trees after freezing was ROC-AUC 0.9419 and PR-AUC 0.9121 on Linux 5.7, and ROC-AUC 0.8005 and PR-AUC 0.8710 on 3DRDF final test set. The above findings indicate that future storage feasibility can be predicted with some degree of accuracy by the previous version.

The fixed experiments exposed a clear storage-metadata trade-off, and the descriptive oracle showed that pair-specific configurations can reduce chunk count while remaining close to C0 in hindsight. The oracle is not a learned result, and no zero-violation automatic switching policy was demonstrated. The practical outcome of the present work is therefore a reproducible feasibility-prediction signal and a quantified description of the available trade-off. A deployable adaptive controller would require a separately validated risk-control layer, uncertainty handling, and repository-level transition accounting.

CONFLICTS OF INTEREST

The author declares no conflict of interest.

AUTHOR CONTRIBUTIONS

Hashem Bedr Jehlol was responsible for the conceptualization, methodology, software development, experiments, validation, formal analysis, data curation, visualization, and preparation and revision of the manuscript.

DATA AND CODE AVAILABILITY

The reproducibility package consists of the deterministic Gear-table seed and construction rule, feature-extraction and chunking scripts, configuration/dependency information, pair and split manifests, grouped-fold assignments, candidate-level labels and storage/chunk outcomes, the frozen model, and unseen-set predictions. These materials can be provided by the corresponding author and are intended for deposit in a public research repository when licensing of third-party datasets permits. Third-party data remain subject to their original licenses and distribution conditions.

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

REFERENCES

[1] A. Muthitacharoen, B. Chen, and D. Mazieres, "A Low-Bandwidth Network File System," in Proc. 18th ACM Symp. Operating Systems Principles (SOSP), pp. 174-187, 2001. doi: https://doi.org/10.1145/502034.502052.
[2] K. Eshghi and H. K. Tang, "A Framework for Analyzing and Improving Content-Based Chunking Algorithms," HP Laboratories, Tech. Rep. HPL-2005-30(R.1), 2005. Available: https://shiftleft.com/mirrors/www.hpl.hp.com/techreports/2005/HPL-2005-30R1.pdf
[3] W. Xia, Y. Zhou, H. Jiang, D. Feng, Y. Hua, Y. Hu, Q. Liu, and Y. Zhang, "FastCDC: A Fast and Efficient Content-Defined Chunking Approach for Data Deduplication," in Proc. USENIX ATC, pp. 101-114, 2016. Available: https://www.usenix.org/conference/atc16/technical-sessions/presentation/xia
[4] W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang, and Y. Zhou, "A Comprehensive Study of the Past, Present, and Future of Data Deduplication," Proceedings of the IEEE, vol. 104, no. 9, pp. 1681-1710, 2016. doi: https://doi.org/10.1109/JPROC.2016.2571298
[5] Y. Zhang, H. Jiang, D. Feng, W. Xia, M. Fu, F. Huang, and Y. Zhou, "A Fast Asymmetric Extremum Content Defined Chunking Algorithm for Data Deduplication in Backup Storage Systems," IEEE Transactions on Computers, vol. 66, no. 2, pp. 199-211, 2017. doi: https://doi.org/10.1109/TC.2016.2595565
[6] R. N. S. Widodo, H. Lim, and M. Atiquzzaman, "SDM: Smart Deduplication for Mobile Cloud Storage," Future Generation Computer Systems, vol. 70, pp. 64-73, 2017. doi: https://doi.org/10.1016/j.future.2016.06.023
[7] W. Xia, D. Feng, H. Jiang, Y. Zhang, V. Chang, and X. Zou, "Accelerating Content-Defined-Chunking Based Data Deduplication by Exploiting Parallelism," Future Generation Computer Systems, vol. 98, pp. 406-418, 2019. doi: https://doi.org/10.1016/j.future.2019.02.008
[8] F. Ni and S. Jiang, "RapidCDC: Leveraging Duplicate Locality to Accelerate Chunking in CDC-Based Deduplication Systems," in Proc. ACM Symp. Cloud Computing, pp. 220-232, 2019. doi: https://doi.org/10.1145/3357223.3362731
[9] G. Xu, B. Tang, H. Lu, Q. Yu, and C. W. Sung, "LIPA: A Learning-Based Indexing and Prefetching Approach for Data Deduplication," in Proc. 35th IEEE MSST, pp. 299-310, 2019. doi: https://doi.org/10.1109/MSST.2019.00010
[10] Y. Zhang, W. Xia, D. Feng, H. Jiang, Y. Hua, and Q. Wang, "Finesse: Fine-Grained Feature Locality Based Fast Resemblance Detection for Post-Deduplication Delta Compression," in Proc. FAST, pp. 121-128, 2019. Available: https://www.usenix.org/conference/fast19/presentation/zhang
[11] A. S. M. Saeed and L. E. George, "Data Deduplication System Based on Content-Defined Chunking Using Bytes Pair Frequency Occurrence," Symmetry, vol. 12, no. 11, Article 1841, 2020. doi: https://doi.org/10.3390/sym12111841
[12] Z. Xu and W. Zhang, "QuickCDC: A Quick Content Defined Chunking Algorithm Based on Jumping and Dynamically Adjusting Mask Bits," in Proc. IEEE ISPA/BDCloud/SocialCom/SustainCom, pp. 288-299, 2021. doi: https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00049
[13] X. Yuan, M. Moh, and T.-S. Moh, "Whole-File Chunk-Based Deduplication Using Reinforcement Learning for Cloud Storage," in Proc. IEEE/ACM ASONAM, pp. 269-276, 2022. doi: https://doi.org/10.1109/ASONAM55673.2022.10068661
[14] J. Park, J. Kim, Y. Kim, S. Lee, and O. Mutlu, "DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression," in Proc. FAST, pp. 247-264, 2022. Available: https://www.usenix.org/conference/fast22/presentation/park
[15] P. Zhou, Z. Wang, W. Xia, and H. Zhang, "UltraCDC: A Fast and Stable Content-Defined Chunking Algorithm for Deduplication-Based Backup Storage Systems," in Proc. IEEE IPCCC, pp. 298-304, 2022. doi: https://doi.org/10.1109/IPCCC55026.2022.9894295
[16] H. B. Jehlol and L. E. George, "Enhancing Deduplication Efficiency Using Triple Bytes Cutters and Multi Hash Function," International Journal of Intelligent Engineering and Systems, vol. 16, no. 5, pp. 162-172, 2023. doi: https://doi.org/10.22266/ijies2023.1031.15

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq

[17] A. Liu, A. Baba, S. Udayashankar, and S. Al-Kiswany, "DedupBench: A Benchmarking Tool for Data Chunking Techniques," in Proc. IEEE CCECE, pp. 469-474, 2023. doi: https://doi.org/10.1109/CCECE58730.2023.10288834
[18] C. Wang, K. Wang, M. Li, F. Wei, and N. Xiong, "Chunk2vec: A Novel Resemblance Detection Scheme Based on Sentence-BERT for Post-Deduplication Delta Compression in Network Transmission," IET Communications, vol. 18, no. 2, pp. 145-159, 2024. doi: https://doi.org/10.1049/cmu2.12719
[19] M. Al Jarah, S. Udayashankar, A. Baba, and S. Al-Kiswany, "The Impact of Low-Entropy on Chunking Techniques for Data Deduplication," in Proc. IEEE CLOUD, pp. 134-140, 2024. doi: https://doi.org/10.1109/CLOUD62652.2024.00025
[20] S. Udayashankar, A. Baba, and S. Al-Kiswany, "SeqCDC: Hashless Content-Defined Chunking for Data Deduplication," in Proc. 25th International Middleware Conference, pp. 292-298, 2024. doi: https://doi.org/10.1145/3652892.3700766
[21] M. Gregoriadis, L. Balduf, B. Scheuermann, and J. Pouwelse, "A Thorough Investigation of Content-Defined Chunking Algorithms for Data Deduplication," arXiv:2409.06066, 2024. Available: https://arxiv.org/abs/2409.06066
[22] X. Ye, W. Tian, Y. Wan, R. Li, W. Xiao, and Z. Xu, "Context-Aware Resemblance Detection for Data Deduplication with Neural Network," Engineering Applications of Artificial Intelligence, vol. 144, Article 110116, 2025. doi: https://doi.org/10.1016/j.engappai.2025.110116
[23] S. Udayashankar, A. Baba, and S. Al-Kiswany, "VectorCDC: Accelerating Data Deduplication with Vector Instructions," in Proc. 23rd USENIX Conf. File and Storage Technologies (FAST), pp. 513-522, 2025. Available: https://www.usenix.org/conference/fast25/presentation/udayashankar
[24] K. T. Truong, S.-P. Merz, M. Scarlata, F. Gunther, and K. G. Paterson, "Breaking and Fixing Content-Defined Chunking," in Proc. ACM CCS, pp. 2294-2308, 2025. doi: https://doi.org/10.1145/3719027.3744870
[25] S. Udayashankar, A. A. Mahmoud, and S. Al-Kiswany, "Vectorized Sequence-Based Chunking for Data Deduplication," IEEE Transactions on Parallel and Distributed Systems, vol. 37, no. 4, pp. 934-947, 2026. doi: https://doi.org/10.1109/TPDS.2026.3660793
[26] S. Udayashankar, A. Baba, and S. Al-Kiswany, "Accelerating Data Chunking in Deduplication Systems Using Vector Instructions," ACM Transactions on Storage, 2026. doi: https://doi.org/10.1145/3797270
[27] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely Randomized Trees," Machine Learning, vol. 63, pp. 3-42, 2006. doi: https://doi.org/10.1007/s10994-006-6226-1
[28] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785-794, 2016. doi: https://doi.org/10.1145/2939672.2939785.
[29] L. Breiman, "Random Forests," Machine Learning, vol. 45, pp. 5-32, 2001. doi: https://doi.org/10.1023/A:1010933404324.

*Corresponding author
Hashem Bedr Jehlol,
Department of AI, College of Science, Mustansiriyah University, Baghdad, Iraq
e-mail: hashemhbg@uomustansiriyah.edu.iq