



Research paper

Semantic-driven seasonal data classification: An artificial intelligence-enabled cost-effective storage system

Zhu Yuan^{a,b}, Xueqiang Lv^{c,d}, Yunchao Gong^f, Ping Xie^d, Xiao Qin^e, Xindong You^{g,*}

^a Department of Information Management, The National Police University for Criminal Justice, Baoding 071000, China

^b Research Center for Smart Justice, The National Police University for Criminal Justice, Baoding 071000, China

^c Beijing Key Laboratory of Internet Culture and Digital Dissemination Research, Beijing Information Science and Technology University, Beijing 100101, China

^d Computer College, Qinghai Normal University, Xining 810008, China

^e The Department of Computer Science and Software Engineering, Auburn University, Auburn 36849-5347, USA

^f School of Information Engineering, Tianjin University of Commerce, Tianjin 300134, China

^g College of Information Engineering, Beijing Institute of Graphic Communication, Beijing 102600, China



ARTICLE INFO

Keywords:

Artificial intelligence

Transformer-based models

Semantic feature extraction

Seasonal access pattern

Data classification

Cost-effective storage optimization

ABSTRACT

Modern storage systems face significant challenges in balancing energy costs and performance, especially for text-rich workloads (e.g., e-commerce logs, social media archives). Traditional frequency-based classification methods fail to exploit semantic patterns in textual metadata, leading to suboptimal resource allocation. In this paper, we propose a novel deep learning-method based on Bidirectional Encoder Representations from Transformers-Recurrent Convolutional Neural Networks (BERT-RCNN) to extract seasonal features from data for classification, enabling cost-effective storage management. Our approach addresses the limitations of traditional frequency-based classification by incorporating semantic features from data text. Additionally, we explore the long-period seasonal features embedded within the text, offering a new approach for multi-feature classification. To evaluate the practical impact of our method, we developed a cost module within CloudSimDisk to simulate storage system operating costs. By constructing models for energy consumption and operating costs and generating real-world workloads from Baidu Index data, we implement and test our solution. Experimental results show that the BERT-RCNN model outperforms traditional K-means and other deep learning methods, reducing energy consumption by 3.90%–5.69%, saving operational costs by 3.62%–6.13%, and shortening response time by 15.12%–26.09%.

1. Introduction

Data centers and high-performance computing facilities are significant contributors to global greenhouse gas emissions, with annual CO₂ emissions reaching 100 megatonnes (Lannelongue et al., 2021). According to recent statistics, data centers account for approximately 1.5% of the world's total energy consumption, with 40% of this energy being consumed by storage systems (Arora and Bala, 2021). As the volume of data generated from both online and offline sources continues to grow, energy consumption and storage costs are rising exponentially, making it increasingly challenging for data center operators to maintain cost-effective and sustainable storage systems.

Classifying data based on access characteristics and storing it in appropriate areas has proven to be an effective strategy for cost-effective storage management (You et al., 2020). The fundamental idea is to store data in storage devices that offer varying levels of

energy consumption, cost, and performance depending on the category of data. In industrial storage systems, such as Baidu Cloud and Alibaba Cloud, data classification is typically rule-based or based on access frequency thresholds. However, these methods fail to consider the semantic meaning contained in textual metadata, which is a critical factor for improving storage efficiency. For example, product descriptions like 'winter jackets' may show seasonal access spikes. However, frequency-based models cannot associate these spikes with their underlying semantic meaning. This gap presents an opportunity for an AI-driven approach that combines Natural Language Processing (NLP) for semantic analysis with storage optimization to enhance the efficiency of data management.

Existing methods face two major shortcomings: (1) traditional data classification techniques primarily focus on access frequency and ignore important periodicity and seasonal patterns, and (2) statistical

* Corresponding author.

E-mail addresses: yuanzhu@cicp.edu.cn (Z. Yuan), lxq@bistu.edu.cn (X. Lv), gongyunchao@tjcu.edu.cn (Y. Gong), xieping@qhnu.edu.cn (P. Xie), xqin@auburn.edu (X. Qin), youxindong@bistu.edu.cn (X. You).

<https://doi.org/10.1016/j.engappai.2025.113004>

Received 11 February 2025; Received in revised form 18 October 2025; Accepted 27 October 2025

Available online 29 October 2025

0952-1976/© 2025 Elsevier Ltd. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

and traditional machine learning-based methods often lack the ability to extract rich semantic features from textual metadata. To address these limitations, we propose a novel method based on BERT-RCNN (BERT-Recurrent Convolutional Neural Networks), which is designed to extract seasonal features from data text for improved classification, enabling cost-effective storage management. First, we perform an in-depth analysis of the long-term seasonal semantic features present in data access patterns. Second, BERT-RCNN combines BERT's bidirectional contextual representation with RCNN's local and global feature extraction capabilities. BERT is utilized to obtain detailed word vector representations, which are then fed into RCNN. RCNN extracts both local and global context features, allowing for the extraction of deeper seasonal characteristics. Compared to conventional models, BERT-RCNN provides significant advantages in semantic feature extraction, thereby enabling a more complete understanding of the seasonal semantic patterns within the data.

To further enhance storage efficiency, we implement a data classification strategy that divides all data into hot data and cold data, according to their access patterns and seasonal relevance. Hot data refers to frequently accessed data that shows a strong correlation with the current season's contextual or semantic features (e.g., access for "winter coats" during winter). Such data requires high-performance storage due to intensive and time-sensitive access. Cold data, in contrast, denotes information whose access frequency declines as the season changes (e.g., "summer sandals" during winter). These datasets are suitable for low-performance, cost-efficient storage tiers. This distinction ensures that storage resources are dynamically aligned with user demand patterns across different seasons, enabling more balanced system utilization. Additionally, data migration between storage devices is conducted to align with the change in seasons, ensuring continuous optimization of storage resources.

The key contributions of this work are as follows:

- We propose a semantic-seasonal data classification framework that integrates semantic understanding of textual metadata with seasonal access trends. This framework enables intelligent, cost-aware data storage by aligning linguistic meaning with seasonal behavior patterns.
- We develop a BERT-RCNN hybrid model that jointly captures contextual semantics and sequential dependencies. Unlike traditional models that treat keywords such as "winter jackets" and "coats" as isolated tokens, our approach identifies their shared seasonal semantics, enabling accurate classification of season-related data.
- We perform a comprehensive experimental evaluation, including both qualitative and quantitative analyses, to validate the model's effectiveness. Results show that the proposed semantic-seasonal classification significantly reduces storage energy consumption and operational costs, while offering interpretable insights into long-term data access behaviors.

The remaining parts of this paper are organized as follows. Section 2 elaborates on related background. Section 3 introduces the data classification storage methods extracting seasonal features by BERT-RCNN. Section 4 describes the model of energy consumption and operating cost in detail. Section 5 thoroughly discusses the experimental results. Section 6 briefly reviews the related works centered around cost-effective storage strategies powered by deep-learning-based data classification and text classification. Finally, we conclude this paper in Section 8.

2. Background

This section provides an overview of the development of text classification methods, an analysis of the seasonal characteristics of data access, and an introduction to the extraction of seasonal features using the K-means clustering algorithm.

2.1. Text classification

Text classification aims to automatically assign text documents to one or more predefined categories, serving as a core task in natural language processing (NLP). Early rule-based methods, which relied on manually crafted linguistic patterns, achieved satisfactory results in narrow domains but were limited by scalability and adaptability as data volumes increased (Kowsari et al., 2019; Li et al., 2022). Subsequent research introduced traditional machine learning algorithms such as support vector machines (SVM), Naïve Bayes, and logistic regression (Razno, 2019), where performance largely depended on feature engineering techniques like TF-IDF and n-gram modeling. These methods improved efficiency but still struggled to capture deeper contextual or semantic relationships between words. With the advent of deep learning, neural architectures such as convolutional and recurrent networks enabled automatic feature extraction and context-aware representations, substantially improving classification accuracy (Minaee et al., 2021). More recently, transformer-based pretrained models (Devlin et al., 2018; Vaswani et al., 2017; Yang et al., 2019; Zhang et al., 2019) have established a new state of the art by leveraging self-attention mechanisms to capture long-range dependencies. Among them, BERT (Devlin et al., 2018) stands out for its bidirectional contextual understanding and adaptability to various downstream classification tasks. Meanwhile, graph-based representation learning has attracted growing attention for its capacity to model complex dependencies and multi-level relationships among data features (Gao et al., 2024b,a, 2025). These studies demonstrate that graph structures can effectively capture hierarchical feature correlations and relational semantics, offering new perspectives for text understanding beyond linear token sequences. Despite these advances, most existing works emphasize purely linguistic or semantic categorization without integrating temporal or domain-specific patterns. In contrast, our research extends semantic modeling into a semantic-seasonal context, aligning textual meaning with seasonal variations to support intelligent and cost-effective storage management.

2.2. Analysis of seasonal characteristics

The division of seasons based on the astronomical calendar is a widely accepted method for seasonal classification (Jennings et al., 2009; Smith, 2014). The four seasons cycle regularly throughout the year, with each season exhibiting distinct regional characteristics. For example, in the northern hemisphere (Orlove, 2021), spring spans from March to May, summer from June to August, autumn from September to November, and winter from December to February of the following year. This conventional division provides a clear temporal framework for analyzing the seasonal characteristics of data in subsequent sections.

Previous studies have shown that user data accesses exhibit substantial periodicity (Iodice et al., 2017; Fontugne et al., 2017), indicating that access patterns follow regular and repeating cycles. Seasonal features, though a common and significant type of periodicity, are often overlooked. To illustrate this, we present the access frequency of keywords obtained from the Baidu Index website. Fig. 1 displays the monthly access records for various clothing categories from January 1 to December 31 over the years 2018–2020. The X-axis represents months from 2018 to 2020, and the Y-axis shows the monthly access frequency for clothing categories. From Fig. 1, we observe that the access records for "Spring clothing", "Autumn clothing", and "Winter clothing" peak in spring, autumn, and winter, respectively. Interestingly, the peak for "Summer clothing" occurs earlier than expected, which can be attributed to the rapid rise in temperatures due to global warming, causing summer to start earlier each year. Nevertheless, the overall access peak still maintains seasonal regularity. Therefore, we conclude that leveraging these seasonal features to classify data could lead to cost-effective storage optimization.

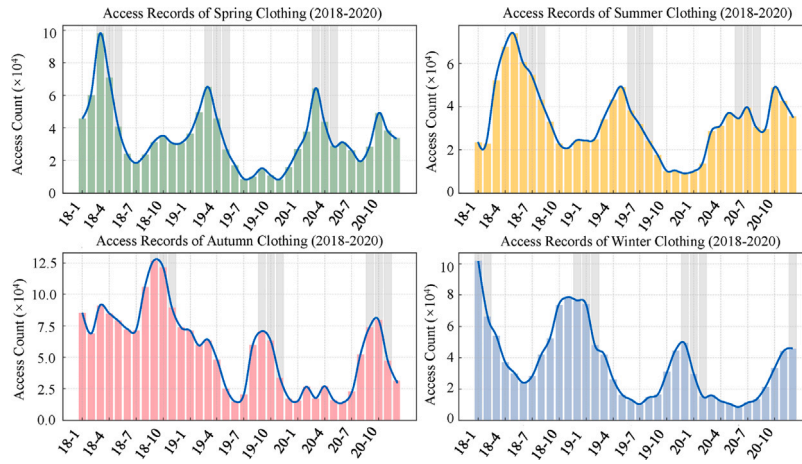


Fig. 1. The access records of different clothing from 2018 to 2020.

2.3. Extracting seasonal features based on K-means

Clustering techniques have been widely used to extract seasonal features in order to develop cost-effective storage solutions. In this section, we briefly introduce the clustering-based method, which will serve as the baseline for our comparative experiments.

Clustering is a technique used to identify inherent relationships in data by grouping similar data points into clusters or classes. Clustering algorithms (Xu and Wunsch, 2005; Fahad et al., 2014; Xu and Tian, 2015) can be categorized into five types: Partition-based clustering algorithms, Hierarchical clustering algorithms, Density-based clustering algorithms, Grid-based clustering algorithms, and Model-based clustering algorithms. Partition-based methods, such as K-means (Ahmed et al., 2020), K-means++ (Deshpande et al., 2020), and K-medoids (Park and Jun, 2009), are among the most commonly used algorithms. These methods divide a dataset of n objects into K categories, where $K \leq n$. The number of clusters is predetermined, and the algorithm iterates until the points within each cluster are sufficiently close and points between clusters are sufficiently distant. Previous studies have demonstrated that K-means is well-suited for problems with a fixed number of classifications and is known for its simplicity and fast convergence.

However, the performance of K-means depends significantly on the selection of the number of clusters (K). An excessively large value for K can lead to over-segmentation, increased complexity, and potential overfitting, while too small a value may result in information loss, reduced diversity, and increased sensitivity to outliers. For this work, we classify the data into five categories based on the seasonal characteristics: spring, summer, autumn, winter, and data with no seasonal features. We then use data access records from the Baidu Index website to calculate the percentage of records belonging to each season relative to the total access records for the year. By applying the K-means algorithm, we classify the seasonal features in the data to achieve our classification goals.

In conclusion, K-means is a simple, efficient, and widely used clustering method. However, its limitations in feature extraction—particularly when dealing with seasonal characteristics—highlight the need for more advanced methods, such as BERT-RCNN, to address the inherent periodicity and semantic patterns in the data.

3. BERT-RCNN-based features extraction for data classification storage

Fig. 2 illustrates the process of extracting seasonal features using BERT-RCNN to enable data classification storage. This approach consists of three main stages: data processing, seasonal feature extraction using BERT-RCNN, and data classification for storage.

3.1. Data processing

The primary data source for this study is publicly available datasets from Baidu. We obtain access frequency data for text from the Baidu Index website, which serves as the foundation for identifying whether a particular text exhibits seasonal features.

While previous research on classification storage has primarily relied on access frequency analysis, we investigate the relationship between data access frequency and seasonal periodic patterns. For this purpose, we use 1600 data texts covering various categories—such as clothing, food, shelter, and transportation—which encompass both seasonal and non-seasonal feature texts. These datasets are annotated based on their access frequency to identify seasonal characteristics.

3.2. Seasonal feature extraction based on BERT-RCNN

The process of extracting seasonal features is outlined in Algorithm 1. This process can be broken down into four steps: First, the algorithm identifies whether the text semantics indicate the presence of seasonal features based on the monthly distribution of seasons. Next, seasonal features are assigned labels according to the identified seasonal characteristics (spring: 0, summer: 1, autumn: 2, winter: 3). Non-seasonal data are assigned a distinct label (other: 4). Afterwards, The labeled training data are then fed into the *BERT – RCNN Training Model* to optimize the model parameters. Finally, test data are input into the trained BERT-RCNN Model, and a class label c is generated for each test data entry. Based on this label, data are stored in the appropriate storage area s , such as hot or cold storage, as determined by the seasonal feature. In this process, the model learns to associate text semantics with seasonal contexts. For example, metadata containing words such as “heater”, “snow”, or “down jacket” often indicate winter-related access patterns, while terms like “air conditioner”, “sunscreen”, or “beach” appear predominantly in summer-related data. By incorporating these semantic cues into the training process, the algorithm enables the system to automatically map linguistic expressions to their corresponding seasonal categories. This alignment between language and time allows for context-aware classification decisions that directly support intelligent data storage management.

3.2.1. BERT-RCNN overview

The core components of BERT encompass input representation and semantic extraction. Input representation has three parts: token embedding (dividing a sentence into subunits), segmentation embedding (distinguishing sentences to assess context), and position embedding (randomly initializing word vectors for sequential index preservation). The key symbols used throughout this section are summarized in Table 1.

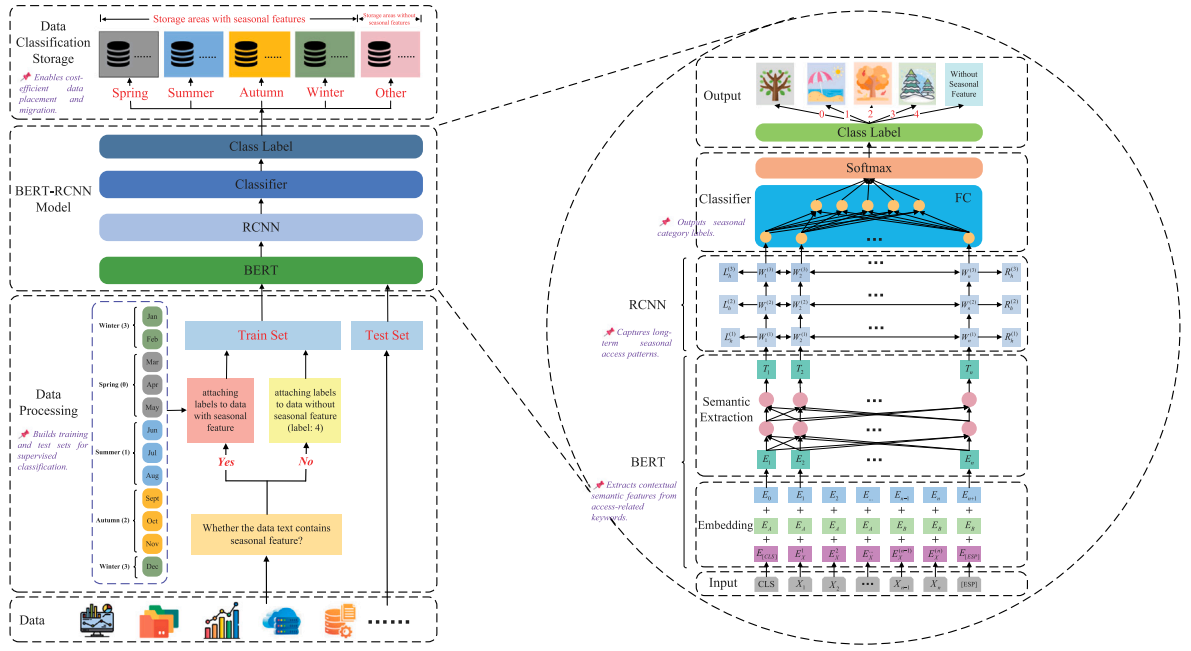


Fig. 2. Overview of the BERT-RCNN-based semantic seasonal classification framework. The system extracts contextual and sequential features from metadata to classify access records by season, enabling optimized data placement across seasonal storage zones.

Algorithm 1: Data Classification Storage based on BERT-RCNN

Input:
 L : A list contains all original train data texts
 L_a : A list contains all train data texts with labels
 H : A list contains all test data texts
 L_i : i presents the train data texts number ($0 \leq i < m$), which m is the total number of train data texts
 H_j : j presents the test data texts number ($0 \leq j < n$), which n is the total number of test data texts
 c : c indicates test data text H_j owns the seasonal feature class
Output:
 s : s means the storage area of test data text H_j

```

1 for  $i = 0; i < m; i++$  do
2   if data text  $L_i$  contains seasonal feature then
3     attaching labels to data with seasonal feature according
      to the monthly distribution of season;
      // spring:0;summer:1;autumn:2;winter:3
4   end
5   else
6     attaching label to data without seasonal feature;
      // other:4
7   end
8 end
9 BERT-RCNN Training Model( $L_a$ );
10 for  $j = 0; j < n; j++$  do
11    $c = \text{BERT-RCNN Model}(H_j)$ ;
12    $s = c$ 
      //  $H_j$  will be placed in the storage area  $s$ 
      indicated by  $c$ 
13 end
14 return  $s$ 

```

The semantic extraction layer of the BERT model, a multi-layer bidirectional decoder built on a transformer encoder, contains three parts: a multi-head attention mechanism, a layer normalization and residual connection, and a feedforward neural network.

Table 1

Symbol definitions for the BERT-RCNN architecture.

Symbol	Description
X	Input vector to a sublayer in the BERT encoder
$\text{SubLayer}(X)$	Output of a functional module (e.g., attention or FFN)
Q_i, K_i, V_i	Query, Key, and Value vectors for the i -th attention head
d_k	Dimension of key/query vectors used for attention scaling
μ_i, σ_i^2	Mean and variance of the i -th row of the hidden matrix
α, β	Learnable scale and shift parameters in LayerNorm
$\odot$	Element-wise (Hadamard) multiplication
$c_l(w_i), c_r(w_i)$	Left and right context vectors of word w_i in RCNN
$e(w_i)$	Embedding vector of word w_i
f	Non-linear activation function (e.g., ReLU or tanh)
x_i	Concatenation of context and word vector: $[c_l; e; c_r]$
q	Task-related query vector for attention calculation
$\text{score}(x_i, q)$	Similarity score between x_i and query vector q
α_i	Attention weight assigned to x_i
s	Final semantic vector computed as weighted sum of x_i

Multi-Head Attention Mechanism. By processing an input text, Q, K , and V vectors are originated with a length of 64. For each group of Q_i, K_i, V_i , the corresponding head $head_i$ can be obtained after $\text{Attention}(Q_i, K_i, V_i)$. Each vector is given a score through $Q_i K_i^T$. The transformer realizes the stable gradient by score normalization, which is divided by $\sqrt{d_k}$. Then, the result of the normalized score multiplies

V_i is supplied into softmax. The final result is registered as the weighted score of each vector. Finally, after all $head_i$ are spliced, the output of the multi-head attention mechanism is procured through a linear transformation.

$$head_i = \text{Attention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_K}}\right) V_i \quad (1)$$

$$\text{MultiHead}(Q, K, V) = [h_1 : \dots : h_{head}] \quad (2)$$

Layer Normalization and Residual Connection. Layer normalization normalizes the hidden layer in a neural network to the standard normal distribution, accelerating the model training and convergence.

$$\mu_i = \frac{1}{m} \sum_{i=1}^m x_{ij} \quad (3)$$

$$\sigma_i^2 = \frac{1}{m} \sum_{i=1}^m (x_{ij} - \mu_i)^2 \quad (4)$$

$$\text{LayerNorm}(X) = \alpha \odot \frac{x_{ij} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + \beta \quad (5)$$

μ_i is the average value of matrix rows derived from formula (3) and σ_i^2 is the variance of a matrix row gained by formula (4). In formula (5), $\odot$ is the multiplication of elements in the matrix, and ϵ is used to prevent the denominator from becoming 0. α and β are two learnable parameters to compensate for the information lost amid the normalization process.

Formula (6) signifies that the residual connection adds the input X to the output of a sublayer transformation, denoted as $\text{SubLayer}(X)$:

$$\text{ResidualConnection}(X) = X + \text{SubLayer}(X) \quad (6)$$

Here, $\text{SubLayer}()$ refers to one of the two core components in a BERT encoder layer—either a multi-head self-attention module or a position-wise feed-forward network. Each of these sublayers is followed by layer normalization and wrapped in a residual connection. This architectural design stabilizes training by mitigating gradient vanishing and degradation problems in deep networks.

Feedforward Neural Network. Two linear layers and an activation function are integral elements of a feedforward neural network - an underpinning of BERT.

$$X_{hidden} = \text{ReLU}(\text{Linear}(X)). \quad (7)$$

where ReLU represents an activation function, and Linear denotes a linear mapping function.

Although BERT has the ability to extract text features, RCNN carries out the deeper feature extraction. RCNN (Ming and Hu, 2015) replaces the traditional convolutional layer in conventional convolutional neural networks with a circular convolution layer featuring recursive structures. It combines the advantages of RNN and CNN to revamp performance. RCNN mitigates the disadvantage that later words in RNN have more significant influence than the previous words and avoids setting the dependence length of context through the size of the window like CNN. The fundamental formula of the RCNN model is presented as follows:

$$c_l(w_i) = f(w^{(l)} c_l(w_{i-1}) + w^{(sl)} e(w_{i-1})) \quad (8)$$

$$c_r(w_i) = f(w^{(r)} c_r(w_{i+1}) + w^{(sr)} e(w_{i+1})) \quad (9)$$

where $c(w_i)$ is an output vector after the BERT processing. $c_l(w_i)$ and $c_r(w_i)$ represent respectively the left and right context of the target text. $c_l(w_{i-1})$ is the above of $c_l(w_i)$ and $c_r(w_{i+1})$ is the below of $c_r(w_i)$. $w^{(l)}$ denotes a matrix that converts context to the next hidden layer. $w^{(sl)}$ - a transformation matrix - combines the semantics of the current word with the left context of the next word. $w^{(r)}$ and $w^{(sr)}$ mean the same meaning on the right. f is a nonlinear activation function.

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)] \quad (10)$$

$$\text{score}(x_i, q) = x_i^T \cdot q \quad (11)$$

$$\alpha_i = \text{softmax}(\text{score}(x_i, q)) \quad (12)$$

$$s = \sum_{i=1}^n \alpha_i x_i \quad (13)$$

where x_i is a vector containing itself and context. After the $\tanh$ activation function, x_i is calculated through matrix multiplication with tasks related query vector q to obtain correlation score . Then, the attention distribution α_i can be achieved through softmax . Last, BERT connects x_i and α_i multiply to acquire new vector s as a result.

Algorithm 2: BERT-RCNN Training Model

Input:

L_a : all train data text with labels.

Output:

The model of BERT-RCNN after training.

```

1  $E \leftarrow$  Token embedding( $L_i$ )+Segment embedding( $L_i$ )+Position
  embedding( $L_i$ );
  // Word embedding
2 for  $i = 0; i < 12; i++$  do
3    $head_i = \text{Attention}(Q_i, K_i, V_i)$ 
4    $\text{MultiHead}(Q, K, V) = [h_1 : \dots : h_{head}]$ 
  // Multi-head attention mechanism
5    $\text{LayerNorm}(X) = \alpha \odot \frac{x_{ij} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + \beta$ 
  // Layer normalization
6    $\text{ResidualConnection}(X) = X + \text{subLayer}(X)$ 
  // Residual connection
7    $X_{hidden} = \text{ReLU}(\text{Linear}(X))$ 
  // Feedforward neural network
8 end
9  $x_i = [c_l(w_i); e(w_i); c_r(w_i)]$ 
10  $\text{score}(x_i, q) = x_i^T \cdot q$ 
11  $\alpha_i = \text{softmax}(\text{score}(x_i, q) = x_i^T \cdot q)$ 
12  $s = \sum_{i=1}^n \alpha_i x_i$ 
  // RCNN extracts feature
13  $p = \text{FullyConnected Layer}(s)$ 
```

Algorithm 3: BERT-RCNN Model

Input:

H_j : all train data text with labels.

Output:

c : c means the class of data text H_j

```

1  $e = \text{BERT-RCNN Training Model}(H_j)$ 
2  $c = \text{Softmax}(e)$ 
  // BERT-RCNN represents the trained model in
  here
3 return  $c$ 
```

From a domain perspective, the purpose of using the BERT-RCNN architecture is not only to extract general semantic representations but to capture seasonally influenced semantics embedded in access-related textual data. In practice, user-generated or metadata text often reflects implicit seasonal intent — for instance, “ski resort booking” or “winter sale” correspond to winter, whereas “camping gear” or “sunblock promotion” relate to summer. Modeling these correlations allows the classifier to recognize latent seasonal meaning that purely statistical models would overlook.

The BERT model effectively extracts text features by amalgamating context and semantic information. In addition, RCNN further gleans deeper and more comprehensive features to enhance classification accuracy. In our implementation, we adopt the pre-trained BERT-Base model (12 layers, 768 hidden units) as the semantic encoder.

The model is fine-tuned on our seasonal classification dataset. Input texts—short metadata such as access-related keywords—are tokenized and passed into BERT to obtain contextualized hidden states. The final-layer embeddings are then forwarded to the RCNN component, which extracts deeper sequential and structural features. This pipeline ensures that both semantic meaning and long-term seasonal dependencies are captured. The integration of BERT and RCNN enhances the model's ability to detect latent seasonal characteristics in access patterns and supports more precise classification for storage decisions. The BERT and RCNN modules are fine-tuned jointly in an end-to-end fashion. The fine-tuned model outputs a seasonal probability distribution that reflects how likely each text item belongs to spring, summer, autumn, winter, or non-seasonal categories. These probabilities are further used by the storage decision module to allocate resources dynamically — data corresponding to the current season are designated as hot data and placed in high-performance storage, whereas off-season data are marked as cold and moved to lower-cost storage. This integration tightly couples semantic interpretation with operational decision-making, achieving both contextual understanding and practical efficiency. Specifically, BERT's final-layer contextualized token embeddings are passed directly to the RCNN module, which further processes them to capture structural and sequential features. This unified pipeline enables simultaneous optimization of both semantic and temporal representations.

Algorithm 2 **BERT-RCNN Training Model** outlines the entire training process, which embraces **BERT** (line 1–8) and **RCNN** (line 9–12).

Algorithm 3 illustrates the procedure of attaining the test data labels with **softmax** through the trained model.

3.3. Data storage

The storage system is divided into five distinct storage areas based on seasonal characteristics: spring, summer, autumn, winter, and non-seasonal. Each storage area can operate in one of two performance modes: high-performance mode, characterized by fast data transmission and low latency but higher energy consumption and operational costs, or low-performance mode, with the opposite characteristics.

To optimize storage, the system assigns high-performance storage mode to the current season's storage area and the non-seasonal storage area, while the remaining seasonal storage areas are placed in low-performance mode. As seasons change, the system adjusts the storage modes accordingly. Data migration between storage devices is performed in response to the seasonal rotation.

Algorithm 4 details how disk performance modes are adjusted with seasonal changes. The current season is determined using the function $Season(t)$ based on the current time t . If the current season differs from the expected one, the system switches the storage mode to high-performance for the current season and low-performance for others. This ensures that seasonal data are stored efficiently while reducing energy consumption during off-seasons. Although data migration occurs only four times a year due to seasonal transitions, its overhead is minimal and can be neglected compared to the long-term operational costs of storage systems.

3.4. Summary and design rationale

The proposed BERT-RCNN-based method integrates semantic representation and sequential feature modeling to enhance seasonal data classification for storage optimization. BERT captures contextual relationships within textual access data, while RCNN complements it by modeling local patterns and long-term dependencies across time.

This architectural design is motivated by the need to jointly address the limitations of traditional frequency-based classification and shallow models, which often fail to capture complex seasonal semantics. By combining BERT's powerful context-aware embeddings with RCNN's capacity for structured sequential learning, the model is well-suited to identify long-period patterns and support accurate classification decisions for data placement. This foundation enables the observed improvements in classification accuracy and storage cost-effectiveness.

Algorithm 4: Disks Performance Modes Adjustment with Season Change

Input:

$Storage_g$: storage area for the current season

$Storage_h$: storage area except for the current season

$Storage_{other}$: storage area without seasonal feature

t : t means the current time

T : T represents the number of days in one year

$Season(t)$: determine the season of the current time

Output:

The disks performance mode that have been adjusted according to season change

```

1 The disks in  $Storage_{other}$  always run high performance mode;
2  $t = 1$ ;
3  $g = Season(t)$ ;
4 while true do
5   if  $g \neq Season(t)$  then
6      $g = Season(t)$ ;
7     The disks in  $Storage_g$  change to high performance mode;
8     The disks in  $Storage_h$  change to low performance mode;
9   end
10   $t = (t + 1) \bmod T$ 
11 end

```

4. Energy consumption and operating cost

4.1. Model parameters

We design energy consumption and operating cost models to quantitatively evaluate the effectiveness of various classification methods. Table 2 details all parameters utilized in the models. Given data set $D = D_1, D_2, \dots, D_i$, we classify the data set into five categories by experimental requirements. The annual running time of a storage system is denoted by T . Since disks have two performance modes and each mode owns active and idle statuses, we set these parameters separately. For the high performance mode, we configure the transfer rate as v^h (Mbit/s), the energy consumption per unit time in active status is E_a^h (J/s), and the energy consumption per unit time in idle status is denoted as E_i^h (J/s). For the low performance mode, the transfer rate is v^l (Mbit/s), the energy consumption per unit time in active status is E_a^l (J/s), and the energy consumption per unit time in the idle status is E_i^l (J/s). To reflect disk status information, we apply the disk capacity, data transfer rate, energy consumption in active status, and energy consumption in idle status to indicate the status information of the disk. Thus, we have $d_{status} = \{C, v, E_a, E_i\}$. The disk status information in the high performance mode and the low performance mode is expressed as $d_h = \{C, v^h, E_a^h, E_i^h\}$ and $d_l = \{C, v^l, E_a^l, E_i^l\}$, respectively.

There are two key points to emphasize in the model: (1) Disks in the current seasonal storage area operate in the high performance mode, while those in other seasonal storage areas run in the low performance mode. (2) Disks in storage areas without seasonal features consistently operate in the high performance mode, irrespective of the seasons.

4.2. Energy consumption model

As the storage system adjusts the performance modes of disks based on seasonal variations, the total energy consumption consumed within one year is the sum of the energy consumption across the four seasons.

$$E_{total} = E_{total_spring} + E_{total_summer} + E_{total_autumn} + E_{total_winter}$$

Due to space constraints, we use the energy consumption model for spring as an illustrative example. In spring, disks in the spring and seasonal-feature-free storage areas operate in high performance

Table 2
The parameters of models.

Parameters	Description
D	Total data volume
n	The number of total disks
E	Energy consumption
E_{total}	Total energy consumption of all disks over one year
E_{total_season}	Total energy consumption of all disks in one season (e.g., E_{total_spring})
$E_{season_active}^h$	Energy consumed by seasonal disks in high-performance mode during active state (e.g., $E_{spring_active}^h$)
$E_{season_active}^l$	Energy consumed by seasonal disks in low-performance mode during active state (e.g., $E_{spring_active}^l$)
$E_{season_idle}^h$	Energy consumed by seasonal disks in high-performance mode during idle state (e.g., $E_{spring_idle}^h$)
$E_{season_idle}^l$	Energy consumed by seasonal disks in low-performance mode during idle state (e.g., $E_{spring_idle}^l$)
t_{total_idle}	Percentage of time at idle status in a year
n_s, n_m, n_a, n_w	Number of disks in per season storage area
D_{season}	Data volume of per season storage area (e.g., D_{spring})
$\eta_{season_active}^h$	Proportion of per season storage area disks in high mode at active status (e.g., $\eta_{spring_active}^h$)
$\eta_{season_active}^l$	Proportion of per season storage area disks in low mode at active status (e.g., $\eta_{spring_active}^l$)
t_{season_idle}	Time of disk at idle status in each season storage area (e.g., t_{spring_idle})
$\delta_{season_active}^h$	Data volume proportion of per season in high mode at active status (e.g., $\delta_{spring_active}^h$)
$\delta_{season_active}^l$	Data volume proportion of per season in low mode at active status (e.g., $\delta_{spring_active}^l$)
p^h, p^l	The cost for processing data per MB in high and low mode
p^s	The cost of storing data per MB/day

mode, while the remaining disks are placed in low performance mode. The energy consumption is evaluated as follows: First, the number of active disks in the spring storage area is $n_s \times \eta_{spring_active}^h$. The total transfer speed of these disks is measured as $v^h \times n_s \times \eta_{spring_active}^h$. Then, we calculate the amount of active data in spring as $D_{spring} \times \delta_{spring_active}^h$. Next, we derive the time spent in transmitting these data as $\frac{D_{spring} \times \delta_{spring_active}^h}{v^h \times n_s \times \eta_{spring_active}^h}$. Finally, the total energy consumption of spring-feature data in the high performance mode under active status is expressed as $\frac{E_a^h \times D_{spring} \times \delta_{spring_active}^h}{v^h \times n_s \times \eta_{spring_active}^h}$. T denotes the running time of a storage system within one year. The total running time in the spring is $\frac{T}{4}$. Then, the ratio of total idle time in spring to the total running time of spring is written as t_{spring_idle} . We compute the energy consumption of spring feature data in the high performance mode under idle status as $E_i^h \times \frac{T}{4} \times t_{spring_idle}$. By the same token, we obtain the energy consumption of the storage areas for the other seasons.

$$E_{total_spring} = E_{spring_active}^h + E_{spring_idle}^h + E_{summer_active}^l + E_{summer_idle}^l + E_{autumn_active}^l + E_{autumn_idle}^l + E_{winter_active}^l + E_{winter_idle}^l + E_{other_active}^h + E_{other_idle}^h$$

$$= \frac{E_a^h \times D_{spring} \times \delta_{spring_active}^h}{v^h \times n_s \times \eta_{spring_active}^h} + E_i^h \times \frac{T}{4} \times t_{spring_idle} + \frac{E_a^l \times D_{summer} \times \delta_{summer_active}^l}{v^l \times n_m \times \eta_{summer_active}^l} + E_i^l \times \frac{T}{4} \times t_{summer_idle} + \frac{E_a^l \times D_{autumn} \times \delta_{autumn_active}^l}{v^l \times n_a \times \eta_{autumn_active}^l} + E_i^l \times \frac{T}{4} \times t_{autumn_idle} + \frac{E_a^l \times D_{winter} \times \delta_{winter_active}^l}{v^l \times n_w \times \eta_{winter_active}^l} + E_i^l \times \frac{T}{4} \times t_{winter_idle} + \frac{E_a^h \times D_{other} \times \delta_{other_active}^h}{v^h \times n_o \times \eta_{other_active}^h} + E_i^h \times \frac{T}{4} \times t_{other_idle}$$

4.3. Operating cost model

With the established energy consumption model, we quantify the total operating cost of a storage system over one year. Again, we take the spring operating cost as an example. The cost for processing data per megabyte (MB) in high and low performance modes is denoted as p^h and p^l , respectively. Additionally, the cost of disk storage per MB/day is represented as p^s . Now, we quantify the operating cost in the following steps. First, the data volume to be processed under the active status in spring is $D_{spring} \times \delta_{spring_active}^h$. Then, we obtain the cost in the high performance mode under the active status as $D_{spring} \times \delta_{spring_active}^h \times p^h$. Next, the data volume under the idle status is $D_{spring} \times \delta_{spring_idle}^h$. T denotes the total running time for a year, and the running time in spring is $\frac{T}{4}$. Then, the running time under the idle status is expressed as $\frac{T}{4} \times t_{spring_idle}$. Finally, we obtain the total storage cost under the idle status as $D_{spring} \times \delta_{spring_idle}^h \times p^s \times \frac{T}{4} \times t_{spring_idle}$. The operating cost of the other seasons can be derived in the same fashion.

$$Cost_{total} = Cost_{process} + Cost_{storage}$$

$$Cost_{total} = Cost_{spring} + Cost_{summer} + Cost_{autumn} + Cost_{winter} \\ = Cost_{spring_process} + Cost_{spring_storage} + Cost_{summer_process} + Cost_{summer_storage} + Cost_{autumn_process} + Cost_{autumn_storage} + Cost_{winter_process} + Cost_{winter_storage}$$

$$Cost_{spring} = D_{spring} \times \delta_{spring_active}^h \times p^h + D_{spring} \times \delta_{spring_idle}^h \times p^s \times \frac{T}{4} \times t_{spring_idle} + D_{summer} \times \delta_{summer_active}^l \times p^l + D_{summer} \times \delta_{summer_idle}^l \times p^s \times \frac{T}{4} \times t_{summer_idle} + D_{autumn} \times \delta_{autumn_active}^l \times p^l + D_{autumn} \times \delta_{autumn_idle}^l \times p^s \times \frac{T}{4} \times t_{autumn_idle} + D_{winter} \times \delta_{winter_active}^l \times p^l + D_{winter} \times \delta_{winter_idle}^l \times p^s \times \frac{T}{4} \times t_{winter_idle} + D_{other} \times \delta_{other_active}^h \times p^h + D_{other} \times \delta_{other_idle}^h \times p^s \times \frac{T}{4} \times t_{other_idle}$$

5. Performance evaluation

We compare the performance of a handful of classification methods in realizing cost-effective storage from the perspectives of accuracy, energy consumption, operating cost, total processing time, and average response time.

5.1. Experimental environment

Table 3 lists the experimental parameters of the hardware and software configuration. While CloudSim (Calheiros et al., 2011), a widely used discrete-event simulator for cloud computing, falls short in simulating energy-aware storage in cloud systems, CloudSimDisk (Louis et al., 2015) addresses this gap. CloudSimDisk, a scalable module designed to simulate energy-aware cloud storage systems, incorporates HDD device models, HDD power models, and energy-aware data center persistent storage. To further evaluate the operating cost of cloud storage systems, we have developed a dedicated module within CloudSimDisk.

Table 3

Hardware and software configuration in the experiments.

Equipment/Software	Type/Version
CPU	Intel(R) Core(TM) i5-7200U CPU @ 2.50 GHz
Memory Size	12GB
Hard Disk	256GB(SSD)+1TB(HDD)
Network Card	Realtek PCIe FE Family Controller
Operating System	Windows 10
Cloud Environment Simulation	CloudSim 4.0
Energy Disk Simulation	CloudSimDisk 1.0
Programming Platform	IntelliJ IDEA

Table 4

HDD models characteristics of CloudSimDisk.

Key	Characteristics	Description
0	Manufacture Name	The name of Manufacture
1	Model Number	The unique manufacture reference
2	Capacity	The capacity of HDD (MB)
3	Average Rotation Latency	Rotational Latency is the amount of time it takes for the desired sector to rotate under the read-write heads of the disk drive. The average rotation latency of the disk is defined as half the amount of time it takes for the disk to make one full revolution.
4	Average Seek Time	Average Seek Time refers to the average time taken by the hard disk to move the head to the track where the data is located after receiving the system command. It reflects the ability of hard disk to read data to a certain extent.
5	Maximum Internal Data Transfer Rate	The maximum internal data transfer rate, also called Sustained Data Rate or Sustained Transfer Rate, is defined as the rate at which data is transferred physically from the disk to the internal buffer. The maximum internal data transfer rate is different in high performance and low performance modes.
6	Power Parameters	Power is the quantity of energy consumed in unit time, in W. Power parameters mainly include the power of active and idle status in different disk performance modes.
7	Cost Parameters	Cost is the overhead of processing and storing data per unit of time, in \$. Cost parameters contain the cost of processing data in high performance and low performance modes and storing data in unit time.

5.2. Disk models

CloudSimDisk provides a modeling framework for HDD components, encapsulating classes that represent various aspects of HDD storage components. We enhance the modeling by incorporating additional power and cost parameters. Refer to Table 4 for a comprehensive list of HDD characteristics implemented in CloudSimDisk. The parameter key is used to return the value of a specific characteristic of this HDD.

We implement three disk models—Western Digital HUC-109090CS S600, Seagate ST6000VN0001, and Toshiba MG-04SCA500E—based on performance parameters obtained from storage review reports (tim, 2018). For detailed parameter specifications, refer to Table 5.

Table 5

The parameters of different disk models.

HUC109090CSS600	
Capacity (GB)	900
Average rotation latency (ms)	3
Average seek time (ms)	4
Transfer rate at high/low mode (MB/s)	198/59
Active Power at high/low mode (W)	5.8/4.1
Idle Power at high/low mode (W)	3/1.3
Cost for processing data at high/low mode (\$/MB)	10/3.3
Seagate Nytro 1551 SATA	
Capacity (GB)	900
Average rotation latency (ms)	4.16
Average seek time (ms)	8.5
Transfer rate at high/low mode (MB/s)	216/64
Active Power at high/low mode (W)	11.27/8
Idle Power at high/low mode (W)	6.9/3
Cost for processing data at high/low mode (\$/MB)	12/4
Western Digital HUC109090CSS600	
Capacity (GB)	900
Average rotation latency (ms)	4.17
Average seek time (ms)	9
Transfer rate at high/low mode (MB/s)	215/64
Active Power at high/low mode (W)	11.3/8
Idle Power at high/low mode (W)	6.2/3
Cost for processing data at high/low mode (\$/MB)	11/3.8

5.3. Workload

Our workload is derived from daily access data on the Baidu index website. It is important to clarify that this dataset primarily consists of access records for textual metadata (keywords) related to various items (clothing, food, shelter, transportation), rather than raw file system I/O traces. Our approach focuses on leveraging the semantic information within these keywords to identify seasonal access patterns for classification, enabling cost-effective storage optimization. We select the access records of 200 data texts as a test set. These data texts are test data used in the latest related research K-ear (You et al., 2021) approach, ensuring the effectiveness of our semantic driven seasonal classification model in validation. Fig. 3 visually presents the total access records for these 200 test data texts over this period. Its primary purpose is to illustrate that the access frequency of our test data exhibits a long-tail distribution, where a small number of texts have extremely high access volumes while the majority have low access volumes. The X -axis indicates the order number of data text, such as Big data: 1, Cloud computing: 2, Cloud storage: 3, and so on, and the Y -axis represents the access records. To facilitate the experiment, we process the access records into a standard workload format by month.

In what follows, we articulate the relevant instructions of workload in detail.

1. **Workload Format.** Our workloads encompass essential information from access records, including *text*, *request size*, *storage area*, and *timestamp*.
2. **Workload Generation.** We transform access records into available I/O records. To facilitate seasonal feature extraction, we organize access records of data text by month.
3. **Number of I/O Records.** The disk load is closely related to the number of records. Through access analysis of data text, we find that the number of access records in each month is tremendous - in the order of millions (Yuan et al., 2022). The time taken to replay these accesses is computationally unacceptable. We cap the total monthly access records to optimize simulation time at 10,000. We narrow the daily access records using a scaling down method, ensuring that the seasonal features of

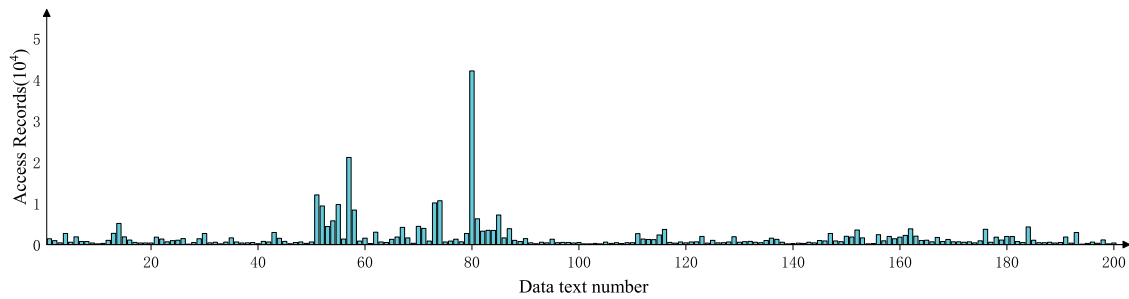


Fig. 3. Access records of data text (2018–2020).

accesses are preserved without dilution. This approach allows us to effectively validate the impact of semantic-driven seasonal classification on storage system performance and cost within a feasible simulation timeframe, while retaining the essential characteristics of the underlying data access patterns.

4. **Data Size Distribution.** Prior studies (Breslau et al., 1999; Riska and Riedel, 2006; Kavalanekar et al., 2008; Kotera et al., 2008; Yang and Zhu, 2016; Ma, 2021) have consistently shown a strong correlation between data size and access frequency. In the real world, abundant access focuses on a small amount of data and most data owns low access frequency - the Zipf-like distribution. To emulate this distribution, we arrange access frequencies in descending order and data sizes in ascending order.
5. **Request Size.** For the convenience and operability of the experiments, we assume that I/O accesses to the same text share an identical request size, which can be obtained via data size/frequency.
6. **Timestamp.** Since the number of I/Os is 10000, we proportionally scale down the time. We randomize the time into 10000 timestamps and separately assign each I/O access.

5.4. Classification accuracy

5.4.1. The competitor models

To highlight the strengths of BERT-RCNN, we benchmark it against K-means and six widely used deep-learning-based text classification models. The extraction of seasonal features for data classification using K-means has been detailed earlier, so we will focus on introducing the other methods below. Our inclusion of K-means as a comparative baseline is specifically motivated by its prior use in the K-ear approach for extracting periodic characteristics to enable cost-effective storage. K-ear is the latest solution in existing related research. While K-means is a traditional and simple clustering algorithm, its well-known limitations in capturing complex semantic features highlight the need for more advanced approaches like ours. Comparing our method with the k-means algorithm we use can fully validate the effectiveness of our approach.

MLP: MLP (Abdelwhab, 2022) transforms text data into vectors and constructs a multi-layer neural network with activation functions for non-linearity. It achieves text classification through feature extraction and non-linear mapping using forward and backward propagation.

TextRNN: TextRNN (Cai et al., 2018) represents text data as word or character vectors. It employs RNN structures to capture contextual information through sequential hidden states and utilizes activation functions and weight parameters for optimization. It facilitates effective text classification by learning sequential information within the text data.

BERT: The BERT (Devlin et al., 2018) is implemented through two main steps: pre-training and fine-tuning. Initially, pre-training is conducted on a large-scale text corpus, allowing BERT to acquire deep bidirectional contextual representations. Subsequently, during fine-tuning, the model is adjusted for specific text classification tasks using labeled data. BERT has emerged as a crucial technology in the field

of natural language processing, primarily due to its robust contextual understanding.

BERT-CNN: BERT-CNN (Wan and Li, 2022) combines BERT's output with Convolutional Neural Networks (CNN). By utilizing convolution and pooling on BERT's model output, it extracts local text information, thereby realizing efficient text classification.

BERT-DPCNN: BERT-DPCNN (Li et al., 2021) integrates BERT embeddings into a Deep Pyramid Convolutional Neural Network (DPCNN) for effective text representation. Leveraging a deep convolutional structure with shortcut connections, it captures hierarchical features for improved text understanding.

BERT-Bi-LSTM: By incorporating BERT's contextual embeddings and LSTM's sequential modeling, BERT-Bi-LSTM (Xue-mei et al., 2023) captures both contextual and sequential information for enhanced performance in text-related tasks. This hybrid model leverages the strengths of BERT and bidirectional LSTM to achieve comprehensive text understanding and effective data classification. Compared to a standard BERT-RNN architecture, the bidirectional structure of Bi-LSTM enables learning from both past and future contexts, offering richer temporal representation. Prior research has consistently shown that Bi-LSTM outperforms unidirectional RNNs in sequence modeling tasks, making BERT-Bi-LSTM a more representative and powerful baseline for evaluating BERT-based recurrent models.

The selected baseline models reflect a variety of architectural strengths: CNN is effective in learning local n-gram features, Bi-LSTM excels at capturing bidirectional sequential dependencies, and MLP offers simple nonlinear representations. Crucially, our selection includes several state-of-the-art deep learning models that represent current best practices in text classification. Specifically, BERT-CNN, BERT-Bi-LSTM, and BERT-DPCNN are widely adopted in contemporary literature as representative hybrid models that integrate BERT with structural encoders. These variants reflect key trends in modern deep learning research—namely, the fusion of contextual embeddings with convolutional or recurrent architectures.

However, these models often struggle to simultaneously capture semantic meaning and long-range temporal dependencies both of which are essential for classifying seasonal access patterns. Even BERT-based variants (e.g., BERT-CNN or BERT-Bi-LSTM) rely heavily on their downstream classifiers, which limits their ability to model both local and global structures. Our BERT-RCNN model addresses these limitations by combining BERT's contextual embedding with RCNN's ability to extract sequential and hierarchical features. This hybrid architecture enables more accurate classification of access logs with latent seasonal dynamics. Accordingly, our current selection of baselines provides a broad and representative comparison, covering diverse model families relevant to semantic-driven storage optimization.

5.4.2. Model training environment and parameters

To accomplish text classification based on seasonal features, we collect rich text corpus to train models. We construct a training set: a total of 1600 texts, including 1200 texts with seasonal features and 400 texts without seasonal features. Noteworthy, our approach applies

Table 6

The model parameter setting.

Models	MLP	Text-RNN	BERT (general parameter)	BERT-CNN	BERT-DPCNN	BERT-Bi-LSTM	BERT-RCNN
Parameter value	hidden_size=256 learning_rate=5e-5 dropout=0.5 vocabulary_size=250499	hidden_size = 128 learning_rate=1e-3 dropout=0.5	hidden_size =768 learning_rate=1e-5 batch_size=4	filter_size = (2,3,4) num_filters = 256 dropout = 0.1	num_filters = 250	rnn_hidden = 768 dropout = 0.1	filter_size = (2,3,4) num_filters = 256 rnn_hidden = 768 dropout=0.1

Table 7

Accuracy and overhead of all methods.

Methods	Accuracy(%)	Construction time (s)	Average GPU usage (MB)
K-means	60.00%	–	–
MLP	68.00%	303	0.36
TextRNN	77.00%	305	0.22
BERT	82.50%	349	0.18
BERT-CNN	82.50%	781	0.20
BERT-DPCNN	79.50%	497	3.71
BERT-Bi-LSTM	83.50%	356	1.20
BERT-RCNN	85.50%	354	0.29

to constructing any corpus because the deep-learning-based method exhibits the prediction ability - a performance edge in processing abundant texts or new texts over statistical and traditional machine learning methods.

The operating system of our testbed is Ubuntu 18.04, the hardware environment is Intel(R) Xeon(R) Gold 5218 CPU @2.30 GHz. The GPU is 4 * NVIDIA A40 (48 GB), the deep learning framework is Pytorch 1.7.0, and the Python version is 3.6.12. Table 6 summarizes the parameters of all the models.

5.4.3. Model accuracy and overhead

Table 7 tabulates the model accuracy and overhead results of BERT-RCNN and its competitors - K-means, MLP, TextRNN, BERT, BERT-CNN, BERT-DPCNN, and BERT-Bi-LSTM. From the perspective of accuracy, the BERT-based classification model outperforms K-means, MLP, and TextRNN by a significant margin, thanks to BERT's superior ability in text semantic extraction. Secondly, among the BERT-based models, BERT-DPCNN performs the worst, which may be attributed to the text length in our corpus. Our corpus primarily consists of short texts, which slashes the advantages of DPCNN, known for its network depth.

From the model overhead perspective, we thoroughly compare the model construction time and average GPU usage among the different models. Since the K-means method (You et al., 2021) is implemented through SPSS tools (Field, 2013), it is no longer involved in the comparison of model construction time and average GPU occupancy rate here. Our findings reveal that BERT-CNN has the longest construction time, indicating that it may require more computational resources during the training process. Additionally, the average GPU usage of BERT-DPCNN is observed to be the highest among all the models, suggesting that it may demand more GPU resources during inference. While the construction time and average GPU usage of BERT-RCNN are not optimal, they are still deemed acceptable and fall within an acceptable range for our experiment purposes. That is because it is worth trading a small amount of overhead for a significant improvement in accuracy.

Despite the relatively higher construction time and modest GPU consumption of BERT-RCNN (354 s and 0.29 MB respectively), the model achieves the highest accuracy (85.50%) among all compared methods. This increase in computation is acceptable in practical deployment for two key reasons. First, the model is trained offline and can be reused multiple times for large-scale seasonal data classification, thus amortizing the training cost over long-term usage. Second, from the perspective of storage system operators, the significant improvements in classification precision directly contribute to optimized data placement and migration policies, leading to energy and operational cost savings—as demonstrated in Section 5.5. Therefore, the moderate computational overhead of BERT-RCNN is justified by its substantial benefits in real-world storage cost efficiency.

The superior performance of BERT-RCNN stems from its architectural advantages in both semantic understanding and sequential modeling. As empirically verified through the performance comparison shown in Table 7, all BERT-based models—including BERT, BERT-CNN, BERT-Bi-LSTM, and BERT-RCNN—consistently outperform non-BERT baselines such as K-means, MLP, and TextRNN. This highlights the critical importance of BERT's advanced semantic contextualization in capturing the implicit meaning of access metadata, forming a powerful foundation for our classification task. BERT encodes contextual dependencies between tokens, enabling the model to capture nuanced meaning from access-related keywords. Building upon this strong semantic foundation, RCNN complements BERT by applying a recurrent structure over convolutional features, which helps model both local n-gram patterns and long-distance dependencies crucial for identifying periodicity and latent seasonal signals in time-evolving access patterns. This synergy, evident in the 3.00% accuracy improvement of BERT-RCNN (85.50%) over the standalone BERT model (82.50%), allows BERT-RCNN to deliver higher classification accuracy with better generalization across varying seasonal workloads. Among all models, BERT-RCNN achieves the highest accuracy, indicating that combining contextual embeddings with structured sequential modeling yields the most accurate classification for seasonal data placement.

5.5. Comparison across multiple years

Using the HUC109090CSS600 disk model, we compare all methods in terms of energy consumption, operating cost, total processing time, and average response time.

5.5.1. Total energy consumption

We evaluate the energy consumption implications of various data classification strategies. The workloads of tests are set to 2018, 2019, and 2020, respectively. Fig. 4 plots the yearly energy consumption comparison of all schemes from 2018 to 2020.

Fig. 4 demonstrates that K-means always has the highest energy consumption and BERT-RCNN is always the lowest. This discrepancy is attributed to the distinct advantage of deep learning over traditional machine learning in semantic mining within text. MLP always consumes the most energy among the deep-learning-based schemes. In 2018 and 2020, the energy consumption of BERT is the least. In 2019, BERT-RCNN owns the least energy consumption. Compared with BERT, BERT-RCNN applies 0.08% more energy consumption in 2018 and 0.92% more in 2020. Compared with K-means, the deep learning-enabled classification methods conserve annual energy consumption by 1.04%–5.21%, 1.89%–5.52%, and 2.66%–6.40%, respectively. Nevertheless, a single energy consumption indicator is not sufficient to reflect the advantages and disadvantages of the method. In what follows, we continue comprehensively evaluating performance using the other metrics — operating cost, processing time, and response time.

5.5.2. Total operating cost

We calculate the operating cost of all schemes from 2018 to 2020. The operating costs primarily consist of the expenses associated with data processing and data storage. Data processing costs encompass the costs of processing data across various storage devices. On the other hand, data storage costs refer to the differing expenses incurred by various devices due to data storage. This comprehensive assessment of

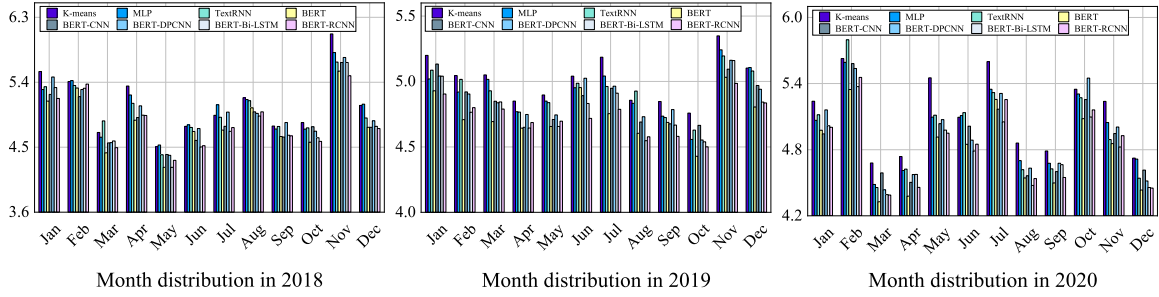


Fig. 4. Annual total energy consumption across different methods from 2018 to 2020 ($\times 10^4$ J).

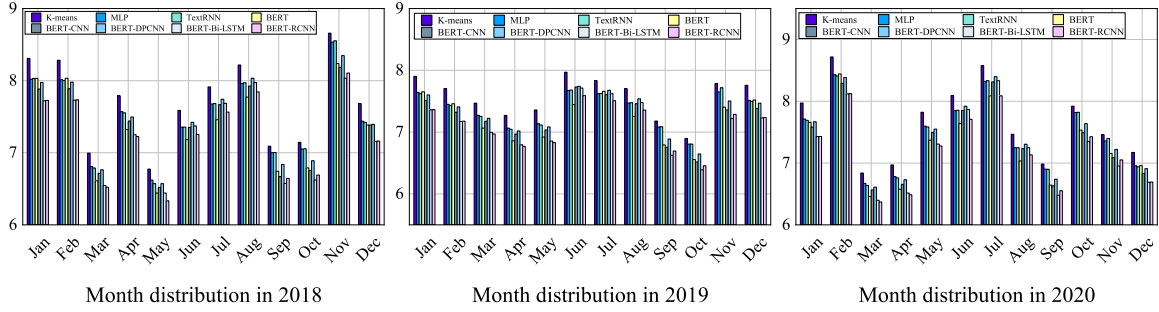


Fig. 5. Annual total operating cost across different methods from 2018 to 2020 ($\times 10^6$ \$).

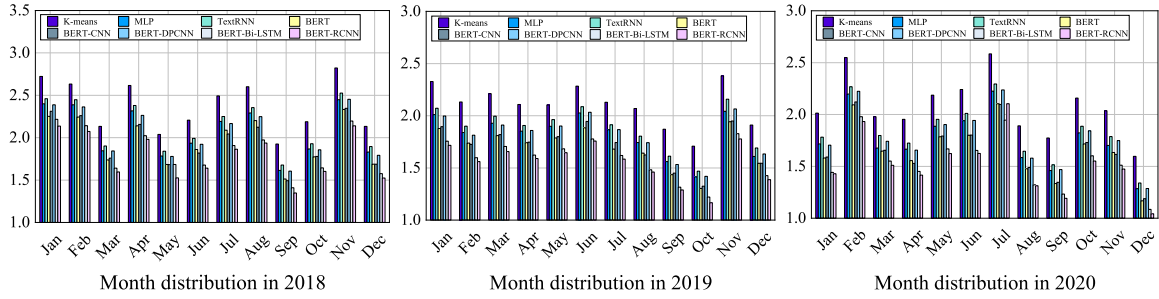


Fig. 6. Annual total operating cost across different methods from 2018 to 2020 ($\times 10^6$ s).

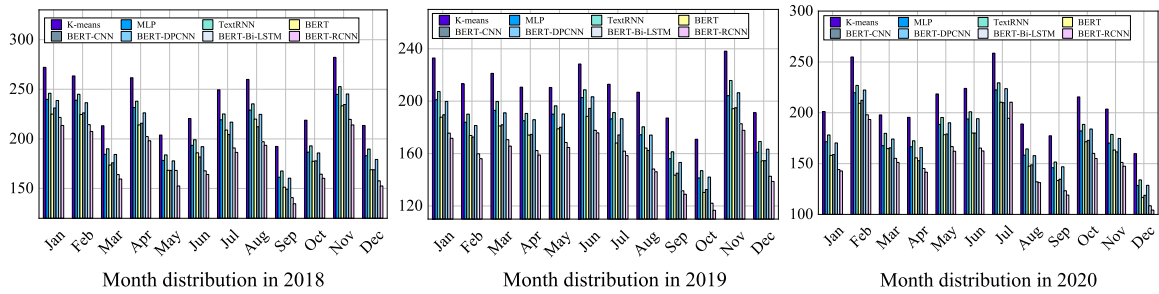


Fig. 7. Annual average response time across different methods from 2018 to 2020 (s).

operating costs enables a thorough analysis of the economic implications of different data classification strategies. Fig. 5 exhibits the annual operating cost of all methods from 2018 to 2020.

In Fig. 5, the operating cost of K-means is the highest every year among its peers. Regarding deep learning methods, MLP has the highest operating cost and BERT-RCNN has the lowest. Notably, BERT-RCNN achieves cost savings compared to K-means and MLP, ranging from 3.60%–6.10%, 3.57%–6.16%, and 3.69%–6.14% across the respective years. The main reason for cost reduction is the improvement in classification accuracy of BERT-RCNN, which enables accurate identification of data with seasonal features. On the contrary, the low classification

accuracy of the K-means method results in a mass of data being divided into the class without seasonal feature.

5.5.3. Total processing time

We evaluate the total processing time of all classification strategies. Total processing time, encompassing waiting and transaction time, reflects time spent processing all I/Os. Analyzing the total processing time provides valuable insights into the system's performance and resource utilization. It helps in understanding the effectiveness of different classification strategies in managing the entire I/O workload. Fig.

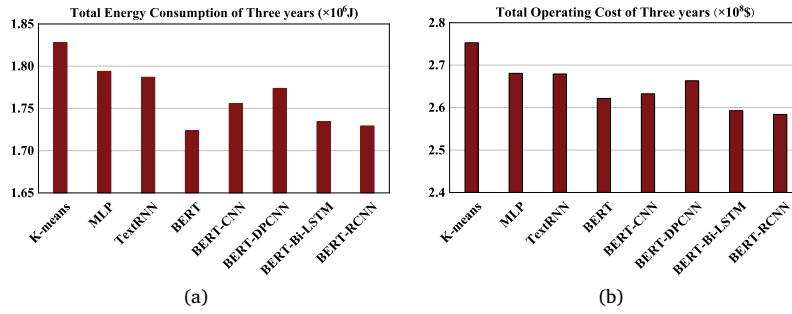


Fig. 8. Total energy consumption and operating cost of three years in different methods.

6 displays the annual total processing time of all tested schemes from 2018 to 2020.

Fig. 6 manifests the total processing time of K-means every month is the highest regardless of the years. Among the deep learning methods, the longest and shortest processing times go to TextRNN and BERT-RCNN, respectively. Compared against K-means and TextRNN, BERT-RCNN shortens the processing time by 16.71%–25.05%, 17.63%–26.32%, and 22.23%–27.04% across the respective years. This is inseparable from the improvement of model classification accuracy. Additionally, the specific characteristics of the dataset and the nature of the I/O operations also affect the performance of each classification strategy.

5.5.4. Average response time

We assess the average response time across all classification strategies. The average response time signifies the mean duration spent processing an I/O operation. It serves as a key metric reflecting the speed and efficiency with which a storage system handles requests, thereby directly influencing user experience and application performance. Fig. 7 exhibits the annual average response time of the evaluated techniques from 2018 to 2020.

Fig. 7 indicates the worst performer is K-means. It has the longest average response time among its peers. Specifically, the annual average response time of K-means is 237.61 s, 210.33 s, and 207.96 s separately. The response times of BERT-RCNN and TextRNN are the shortest and longest within the family of deep learning methods. The annual average response time of TextRNN is 213.81 s, 188.15 s, and 195.10 s, and the average response time of BERT-RCNN is 178.09 s, 154.97 s, and 151.73 s. The improvement in average response time raises the efficiency of I/O operation processing and enhances the overall response capability of the storage system.

5.5.5. Summary

Fig. 8 (a) demonstrates the total energy consumption from 2018 to 2020, during which K-means still has the highest energy consumption. Among the deep learning models, MLP energy consumption is the highest and BERT is the most energy efficient scheme. Compared with K-means and MLP, BERT conserves energy consumption by 5.69% and 3.90%, respectively.

Fig. 8 (b) manifests the total operating cost of all tested methods for three years, which shows that K-means has the highest total operating cost. Speaking of the deep learning methods, MLP and BERT-RCNN's operating cost measures are the highest and lowest. BERT-RCNN slashes the total operating cost of K-means and MLP by 6.13% and 3.62%.

Table 8 compares the total processing time and average response time of all schemes. Not surprisingly, the total processing times of TextRNN and BERT-RCNN are the longest and shortest. The average response time also follows the same trend. The average response time of K-means is 218.637 s. TextRNN and BERT-RCNN's average response times are 195.104 s and 161.597 s.

All the tested methods are assigned performance scores, ranging from 1 (worst) to 8 (best) as listed in Table 9. The score of K-means

Table 8

Comparison of time in different methods.

Methods	Waiting time (10^3 s)	Transaction time (10^3 s)	Total processing time (10^3 s)	Average response time (s)
K-means	7.850	2.114	7.871	218.637
MLP	6.763	2.272	6.786	188.502
TextRNN	7.001	2.244	7.024	195.104
BERT	6.356	2.347	6.380	177.218
BERT-CNN	6.375	2.328	6.398	177.728
BERT-DPCNN	6.756	2.273	6.779	188.299
BERT-Bi-LSTM	5.933	2.399	5.957	165.479
BERT-RCNN	5.793	2.414	5.817	161.597

Table 9

Comparison of score in different methods.

Methods	Model accuracy	Total energy consumption	Total operating cost	Total processing time	Average response time	Total score
K-means	1	1	1	1	1	5
MLP	2	2	2	3	3	12
TextRNN	3	3	3	2	2	13
BERT	6	8	6	6	6	32
BERT-CNN	6	5	5	5	5	26
BERT-DPCNN	4	4	4	4	4	20
BERT-Bi-LSTM	7	6	7	7	7	34
BERT-RCNN	8	7	8	8	8	39

is the lowest across the board. BERT-RCNN achieves the highest total score. Although BERT secures the highest score in terms of total energy consumption, BERT-RCNN is a runner up. However, the disadvantage in energy consumption does not affect BERT-RCNN's championship. MLP and TextRNN are the worst performers among the deep-learning-based methods, which is related to the model's characteristic. Compared with MLP and TextRNN, BERT typically performs better in text classification tasks thanks to its robust context modeling ability, benefits of pre-training and fine-tuning, excellent migration ability, flexible vocabulary representation, and multi-level abstract feature representation.

The aforementioned analysis reveals that extracting seasonal features of text for data classification using BERT-RCNN is a viable method to realize cost-effective storage. The improved performance of BERT-RCNN in terms of energy and cost efficiency results from its enhanced classification granularity, which enables more precise seasonal data scheduling. By accurately distinguishing between different access patterns, the system avoids unnecessary migrations and better aligns data with appropriate storage media over time.

5.6. Comparison across multiple disk models

5.6.1. Experimental results of various disks

To further illustrate the effectiveness and universality of BERT-RCNN, we present the experimental results in Table 10 for three disk models—HUC109090CSS600, ST6000VN0001, and MG04SCA500E—operating under the 2020 workload. When utilizing the HUC109090CSS600 disk, K-means has the worst effect in all the performance metrics. Regarding the energy efficiency of the deep learning methods, the best performer is BERT. Moreover, BERT-RCNN is a front runner in terms of the other metrics: the results are consistent with those reported in Section 5.5.

Table 10
Performance comparison in different methods under three disk models.

Methods	Energy consumption (10^5 J)			Operating cost (10^8 \$)			Total processing time (10^8 s)			Average response time (s)		
	A	B	C	A	B	C	A	B	C	A	B	C
K-means	6.139	8.872	8.976	9.199	11.126	10.208	2.496	2.184	2.349	207.963	181.998	195.677
MLP	5.975	8.871	8.972	8.965	10.792	9.976	2.116	1.837	1.860	176.288	153.035	154.978
TextRNN	5.952	8.871	8.965	8.958	10.772	9.955	2.201	1.914	1.938	183.348	159.508	161.491
BERT	5.746	8.789	8.841	8.756	10.538	9.739	1.989	1.718	1.741	165.755	143.168	145.070
BERT-CNN	5.882	8.864	8.947	8.803	10.587	9.783	1.995	1.724	1.747	166.235	143.659	145.541
BERT-DPCNN	5.926	8.870	8.963	8.907	10.712	9.895	2.133	1.851	1.875	177.740	154.222	156.204
BERT-Bi-LSTM	5.770	8.865	8.913	8.668	10.428	9.638	1.845	1.582	1.604	153.679	131.803	133.619
BERT-RCNN	5.799	8.867	8.957	8.634	10.385	9.601	1.821	1.563	1.584	151.729	130.214	132.003

A, B, and C represent disk models HUC109090CSS600, ST6000VN0001, and MG04SCA500E separately.

Table 11
Comparison of score in different methods under three disk models.

Methods	Model accuracy			Energy consumption			Operating cost			Total processing time			Average response time			Total score		
	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C
K-means	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	5	5
MLP	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	12	12	12
TextRNN	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	13	13	13
BERT	6	6	6	8	8	8	6	6	6	6	6	6	6	6	6	32	32	32
BERT-CNN	6	6	6	5	7	6	5	5	5	5	5	5	5	5	5	26	28	27
BERT-DPCNN	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	20	20
BERT-Bi-LSTM	7	7	7	7	6	7	7	7	7	7	7	7	7	7	7	35	34	35
BERT-RCNN	8	8	8	6	5	5	8	8	8	8	8	8	8	8	8	38	37	37

A, B, and C represent disk models HUC109090CSS600, ST6000VN0001, and MG04SCA500E separately.

Similarly, K-means is the worst across the board in cases of ST6000VN0001 and MG04SCA500E. The energy efficiency results of ST6000VN0001 and MG04SCA500E under the deep learning methods, the MLP's energy consumption is the highest among its peers and BERT is the most energy efficient one. Compared with K-means and MLP, BERT curbs the energy consumption by 0.93% and 0.92% under ST6000VN0001 and by 1.51% and 1.46% under MG04SCA500E. Not surprisingly, MLP's total operating cost is the highest and BERT-RCNN can save operating cost in the two disk models. BERT-RCNN conserves the total operating cost of K-means and MLP by 6.66%, 3.77%, 5.95%, and 3.76%. BERT-RCNN has a performance edge in terms of total processing time and average response time over the competitors: BERT-RCNN curtails the total processing time of K-means and TextRNN by 28.45% and 18.37% in the ST6000VN0001 case; the reductions become 32.54% and 18.26% under MG04SCA500E.

5.6.2. Summary

The experimental results of ST6000VN0001 and MG04-SCA500E agree with those for the HUC109090CSS600.

Table 11 summarizes the performance scores of all the methods containing the HUC109090CSS600, ST6000VN-0001, and MG04SCA500E disks, where K-means has the lowest scores. Among the deep learning techniques, BERT-RCNN takes the championship in total score. MLP and TextRNN have the lowest score. Although BERT-RCNN delivers dismal energy efficiency performance, BERT-RCNN curbs operating cost, total processing time, and average response time, making BERT-RCNN the best overall underpinning for cost-effective data storage.

6. Related work

6.1. Cost-effective storage strategy based data classification

After an extensive period of research and development, there emerged a bevy of approaches, each possessing distinct characteristics and advantages, to realize cost-effective storage through data classification (Yuan et al., 2022, 2024a,b; Potika et al., 2025). This section concisely introduces representative and promising works in this domain.

Striping-based Energy-Aware (SEA) (Xie, 2008) integrates an energy-conscious approach into data placement for RAID-structured storage. SEA places popular data on a set of disks (hot disk zone) and unpopular data on the remaining disks (cold disk zone). Compared to traditional

non-partitioned file assignments, SEA achieves energy savings without compromising system performance or disk reliability.

GreenHDFS (DelValle et al., 2017), a self-adaptive, energy-conserving variant of HDFS, achieves energy-proportional behavior in data centers with non-proportional components. It categorizes storage into hot and cold zones based on energy consumption and performance, employing an intelligent data classification strategy for data placement. In a three-month simulation within a Hadoop cluster, GreenHDFS realized a substantial 26% energy savings.

Lightning (Kaushik et al., 2010) configures servers in cloud storage into hot and cold zones. Servers in the cold zone are transitioned to inactive power modes, which results in energy and cost conservation for data centers.

Liao et al. (2015) built a cost-effective model for distributed storage systems in which racks are classified into active and sleep zones. Data are stored in distinct zones by analyzing different access frequency and regularity. Cost-effective storage systems were made possible by sleeping the servers in the sleep zone.

Zhang and Li (2014) categorized data into deduplication files, hot data blocks, and cold data blocks based on repetition and activity factors. Storage is organized into deduplication, hot, and cold zones. Experimental results show energy savings when the system load is below a specified threshold.

K-ear (You et al., 2021) categorizes data using the K-means clustering algorithm based on period characteristics by data frequency. The classified data is stored in various areas within a cloud storage system, enabling cost-effective storage through diverse energy and performance modes in distinct storage areas.

Table 12 summarizes typical data-classification-based cost-effective storage systems. Most methods only implement the single statistical method to count data access frequency without incorporating the period characteristic of data. Although K-ear concerns period characteristic, data classification based on traditional machine learning methods can merely extract shallow features, leaving out deep and comprehensive information. In contrast to these approaches, we adopt deep learning models to analyze seasonal cycle characteristics in data access. By extracting semantic features from the data text, our method achieves efficient classification, resulting in cost-effective storage.

Table 12
Summary of the reviewed related work compared with our work.

Works	Data classification criterion	Periodicity	Data classification algorithm
SEA	Frequency	–	Statistic
GreenHDFS	Frequency	–	Statistic
Lighting	Frequency	–	Statistic
Liao et al. (2015)	Frequency	–	Statistic
Zhang and Li (2014)	Frequency	–	Statistic
K-ear	Frequency	✓	Machine Learning (K-means)
Our work	Text semantics	✓	Deep Learning

6.2. Deep learning based text classification

Text classification powered by traditional machine learning (Razno, 2019) often has an insufficient understanding of text semantics, structure, sequence, and context, thereby leading to the limited expression ability of the model. As deep learning research progresses, deep learning-enabled schemes begin to shine in the text classification realm (Minaee et al., 2021).

Kim (Chen, 2015), utilizing CNN (Convolutional Neural Networks) to furnish text classification, vectorizes text into a matrix and maps original features to a semantic space of higher dimensions through convolution and pooling operations. Simultaneously, richer semantic information is extracted by setting different convolution kernels. Many researches (Zhang and Wallace, 2015; Conneau et al., 2016; Yao et al., 2019) either improve CNN or integrate CNN with other methods (Lai et al., 2015) to bolster the classification effect.

RNN (Recurrent Neural Network) (Liu et al., 2016) processes sequence data for context information, yet faces challenges with long-term dependencies, gradient exploding, and vanishing. LSTM (Long Short-Term Memory) (Zhou et al., 2016), a specialized RNN, addresses these issues by introducing cell states. In comparison, GRU (Gate Recurrent Unit) (Zulqarnain et al., 2019), a simpler structure with reset and update gates, has fewer parameters, facilitating easier convergence than LSTM.

Bahdanau et al. (2014) introduced the attention mechanism in NLP tasks to swiftly highlight valuable information from extensive data using limited attention resources.

With the application of deep learning in text classification, researchers have found that a single deep learning model is no longer sufficient to improve classification performance significantly. Recent evidence suggests (Zhang et al., 2018; She and Zhang, 2018; Liu and Guo, 2019; Yu et al., 2020; Deng et al., 2022) that fusing CNN, RNN, LSTM, GRU, and other models and combining attention mechanism can reach remarkable enhancement in model outcomes.

Pre-trained models are trained in advance, exhibiting excellent performance on datasets. The Transformer, proposed by Vaswani et al. (2017), employs self-attention mechanisms for parallel operations and concurrent capture of long-term dependencies, significantly improving performance. BERT (Devlin et al., 2018), XLNet (Yang et al., 2019), and ERNIE (Zhang et al., 2019) are transformer-based variants, with BERT standing out for its outstanding performance in NLP tasks like text classification and similarity. Research has demonstrated that integrating a pre-trained model with other deep learning models (such as BERT-CNN and BERT-RNN) for text classification tasks can effectively optimize classification performance (Adhikari et al., 2019; Zheng and Yang, 2019; Zeberga et al., 2022; Onan, 2023; Eang and Lee, 2024; Yu et al., 2024; Liao et al., 2024; Li and Jia, 2025).

Based on the analysis of previous deep learning-based text classification methods, we choose BERT-RCNN as the model for data classification. BERT-RCNN combines the advantages of RNN and CNN to deeply understand semantic information in text and capture both global and local contextual relationships. Thus, this model can obtain in-depth and comprehensive text features, achieving prominent classification performance.

7. Deployment considerations and future improvements

7.1. Practical considerations and future directions

While our model demonstrates strong performance under typical seasonal patterns, we acknowledge certain limitations in handling complex or compound seasonality and abrupt shifts in user behavior. For instance, emerging trends (e.g., social media-driven fads or atypical weather events) may cause access patterns that deviate from historical norms, thereby affecting classification accuracy. Future work will involve validating and refining our approach across larger, more diverse real-world datasets and workload types to ensure broader applicability.

In future work, we plan to incorporate adaptive learning mechanisms such as online fine-tuning and real-time semantic drift detection to improve robustness. Moreover, integrating external real-time signals (e.g., trending queries or news feeds) may help our system distinguish between seasonal and non-seasonal anomalies more effectively.

Another practical consideration is the computational complexity of the BERT-RCNN model. Compared to traditional models such as K-means or standard CNNs, BERT-RCNN requires significantly more resources during training and inference. While this overhead enables superior semantic feature extraction and classification accuracy, it may present challenges in industrial environments with limited processing capacity or real-time constraints. To address this, lightweight deployment strategies such as model pruning, quantization, or knowledge distillation can be adopted to reduce latency and resource consumption. In addition, inference tasks can be offloaded to dedicated edge devices or executed in batch mode to balance efficiency and performance. Overall, the increased computation cost is justifiable given the substantial gains in storage optimization, especially when applied to high-volume workloads.

7.2. Privacy and data protection considerations

Our semantic-driven storage optimization approach operates on textual metadata such as aggregated access keywords and trends, rather than raw personal or user-identifiable data. The dataset used in our experiments is derived from publicly accessible sources (e.g., Baidu Index) and does not include direct personal identifiers. Therefore, the analysis does not target or profile individual users.

In real-world scenarios, we acknowledge that privacy and regulatory considerations must be addressed. Our model is compatible with deployment strategies that keep data local (e.g., on-premise or edge computing) to reduce potential exposure. Additionally, preprocessing modules can be configured to exclude sensitive fields where necessary. For more stringent privacy requirements, advanced mechanisms such as differential privacy or federated learning could be integrated to enhance protection. These measures allow storage system operators to benefit from semantic analysis while maintaining alignment with general data protection principles.

8. Conclusion and future work

In this study, we propose a method for extracting seasonal semantic features from access-related data using a BERT-RCNN framework to support cost-effective data classification and storage. The core idea is to leverage BERT's contextual embeddings and RCNN's sequential modeling to identify long-period seasonal patterns embedded in textual metadata, which are then used to guide data placement decisions. The benefits of our solution are two-fold. First, it compensates for the limitations of traditional frequency-based classification by incorporating deeper semantic understanding of access logs. Second, it introduces a new seasonal-aware classification dimension by detecting recurring topic patterns across time. These contributions collectively enhance the precision of storage-level decision making. The observed improvements in classification accuracy and system efficiency stem from the hybrid architecture of BERT-RCNN, which integrates semantic and sequential feature extraction. This structural design enables more accurate modeling of complex access behaviors, leading to reduced energy consumption, lower operational cost, and improved response time in storage systems. Nonetheless, the effectiveness of our method depends on the presence of identifiable seasonal access patterns and the availability of meaningful textual metadata. In scenarios with abrupt user behavior changes or weak semantic signals, classification performance may degrade. We consider these factors important limitations to be addressed in future work.

As a future research direction, we will continue to mine data access features from multiple perspectives to enhance classification accuracy and support the construction of large-scale, cost-effective storage systems. In the next step, we will validate and refine our approach across diverse real-world datasets and hybrid storage environments composed of NVM, SSD, and HDD. To improve adaptability in cases where seasonal signals are weak or user behavior changes abruptly, we plan to explore incremental and online learning strategies that can dynamically respond to evolving access patterns. Additionally, to address the dependence on high-quality textual metadata, we will investigate adapting the BERT-RCNN model for multilingual and domain-specific data, and explore methods to integrate auxiliary signals such as query logs or event data. We also plan to incorporate privacy-preserving technologies such as federated learning and differential privacy to ensure secure and distributed deployment in regulated environments.

CRedit authorship contribution statement

Zhu Yuan: Writing – original draft, Validation, Software, Methodology, Conceptualization. **Xueqiang Lv:** Supervision, Project administration, Funding acquisition. **Yunchao Gong:** Writing – review & editing, Validation. **Ping Xie:** Writing – review & editing, Funding acquisition. **Xiao Qin:** Writing – review & editing, Funding acquisition. **Xindong You:** Writing – review & editing, Software, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by Science Research Project of Hebei Education Department under Grant No. QN2025011, Doctoral Research Start-up Fund Program of The National Police University for Criminal Justice under Grant No. BSQ-DW202150, National Natural Science Foundation of China under Grants No. 62171043, 62202061 and 62362057, the Natural Science Foundation of Beijing under Grant No. 4232-025, Beijing Municipal Education Commission, China under Grant No.

KM202311232002 and KM202311232003, the U.S. National Science Foundation under Grant DUE-2424934, the National Highway Traffic Safety Administration, United States under Grant 451861-19158, the Alabama Research and Development Enhancement Fund under Grant 1ARDEF25 02, and Wright Media, LLC under Grants 240250 and 240311.

Data availability

Data will be made available on request.

References

- Abdelwhab, O., 2022. Feature selection based on term frequency for Arabic text classification using multilayer perceptron. In: *International Conference on Managing Business Through Web Analytics*. Springer, pp. 101–109.
- Adhikari, A., Ram, A., Tang, R., Lin, J., 2019. Docbert: Bert for document classification. arXiv preprint arXiv:1904.08398.
- Ahmed, M., Seraj, R., Islam, S.M.S., 2020. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics* 9 (8), 1295.
- Anon, 2018. Storage review enterprise lab - enterprise reviews. [Online], <https://www.storagereview.com/enterprise>.
- Arora, S., Bala, A., 2021. An intelligent energy efficient storage system for cloud based big data applications. *Simul. Model. Pr. Theory* 108, 102260.
- Bahdanau, D., Cho, K., Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
- Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S., 1999. Web caching and zipf-like distributions: Evidence and implications. In: *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. the Future Is now (Cat. No. 99CH36320)*, vol. 1, IEEE, pp. 126–134.
- Cai, J., Li, J., Li, W., Wang, J., 2018. Deep learning model used in text classification. In: *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing. ICCWAMTIP, IEEE*, pp. 123–126.
- Calheiros, R.N., Ranjan, R., Beloglazov, A., Rose, C.A.F.D., Buyya, R., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pr. Exp.* 41 (1), 23–50.
- Chen, Y., 2015. *Convolutional Neural Network for Sentence Classification*. University of Waterloo.
- Conneau, A., Schwenk, H., Barrault, L., Lecun, Y., 2016. Very deep convolutional networks for text classification. arXiv preprint arXiv:1606.01781.
- DelValle, R., Kaushik, P., Jain, A., Hartog, J., Govindaraju, M., 2017. Exploiting efficiency opportunities based on workloads with electron on heterogeneous clusters. In: *Proceedings of The10th International Conference on Utility and Cloud Computing*, pp. 67–77.
- Deng, L., Ge, Q., Zhang, J., Li, Z., Yu, Z., Yin, T., Zhu, H., 2022. News text classification method based on the GRU_CNN model. *Int. Trans. Electr. Energy Syst.* 2022.
- Deshpande, A., Kacham, P., Pratap, R., 2020. Robust k -means++. In: *Conference on Uncertainty in Artificial Intelligence. PMLR*, pp. 799–808.
- Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- Eang, C., Lee, S., 2024. Improving the accuracy and effectiveness of text classification based on the integration of the bert model and a recurrent neural network (RNN_Bert-Based). *Appl. Sci.* 14 (18), 8388.
- Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A.Y., Foufou, S., Bouras, A., 2014. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Trans. Emerg. Top. Comput.* 2 (3), 267–279.
- Field, A., 2013. *Discovering Statistics Using IBM SPSS Statistics*. sage.
- Fontugne, R., Abry, P., Fukuda, K., Veitch, D., Cho, K., Borgnat, P., Wendt, H., 2017. Scaling in internet traffic: a 14 year and 3 day longitudinal study, with multiscale analyses and random projections. *IEEE/ACM Trans. Netw.* 25 (4), 2152–2165.
- Gao, B., Liu, W., Liu, G., Nie, F., 2024a. Resource knowledge-driven heterogeneous graph learning for website fingerprinting. *IEEE Trans. Cogn. Commun. Netw.* 10 (3), 968–981.
- Gao, B., Liu, W., Liu, G., Nie, F., Huang, J., 2024b. Multi-level resource-coherent graph learning for website fingerprinting attacks. *IEEE Trans. Inf. Forensics Secur.*
- Gao, B., Liu, W., Liu, G., Nie, F., Huang, J., Zhang, H., 2025. Lightweight graph mining for website fingerprinting guided by structure knowledge. *IEEE Trans. Comput. Soc. Syst.*
- Iodice, M., Candela, M., Di Battista, G., 2017. The internet pendulum: On the periodicity of internet topology measurements. arXiv preprint arXiv:1709.05969.
- Jennings, S., Magrath, J., et al., 2009. What happened to the seasons. In: *Conference on Seasonality, Future Agriculture Consortium International*. IDS Sussex, UK.
- Kaushik, R.T., Cherkasova, L., Campbell, R., Nahrstedt, K., 2010. Lightning: self-adaptive, energy-conserving, multi-zoned, commodity green cloud storage system. In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pp. 332–335.

- Kavalanekar, S., Worthington, B., Zhang, Q., Sharda, V., 2008. Characterization of storage workload traces from production windows servers. In: 2008 IEEE International Symposium on Workload Characterization. IEEE, pp. 119–128.
- Kotera, I., Egawa, R., Takizawa, H., Kobayashi, H., 2008. Modeling of cache access behavior based on zipf's law. In: Proceedings of the 9th Workshop on MEMory Performance: DEaling with Applications, Systems and Architecture. pp. 9–15.
- Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., Brown, D., 2019. Text classification algorithms: A survey. *Information* 10 (4), 150.
- Lai, S., Xu, L., Liu, K., Zhao, J., 2015. Recurrent convolutional neural networks for text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29, (no. 1).
- Lannelongue, L., Grealey, J., Inouye, M., 2021. Green algorithms: quantifying the carbon footprint of computation. *Adv. Sci.* 8 (12), 2100707.
- Li, X., Jia, L., 2025. English text topic classification using BERT-based model. *J. Comput. Methods Sci. Eng.* 25 (1), 669–684.
- Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P.S., He, L., 2022. A survey on text classification: From traditional to deep learning. *ACM Trans. Intell. Syst. Technol.* 13 (2), <http://dx.doi.org/10.1145/3495162>.
- Li, Y.J., Zhang, H.J., Pan, W.M., Feng, R.-J., Zhou, Z.-Y., 2021. Microblog rumor detection based on bert-DPCNN. In: Artificial Intelligence in China: Proceedings of the 2nd International Conference on Artificial Intelligence in China. Springer, pp. 524–530.
- Liao, W., Liu, Z., Dai, H., Wu, Z., Zhang, Y., Huang, X., Chen, Y., Jiang, X., Liu, D., Zhu, D., et al., 2024. Mask-guided BERT for few-shot text classification. *Neurocomputing* 610, 128576.
- Liao, B., Yu, J., Zhang, T., Binglei, G., Hua, S., Ying, C., 2015. Energy-efficient algorithms for distributed storage system based on block storage structure reconfiguration. *J. Netw. Comput. Appl.* 48, 71–86.
- Liu, G., Guo, J., 2019. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing* 337, 325–338.
- Liu, P., Qiu, X., Huang, X., 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.
- Louis, B., Mitra, K., Saguna, S., Åhlund, C., 2015. Cloudsimdisk: Energy-aware storage simulation in clouds. In: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing. UCC, IEEE, pp. 11–15.
- Ma, J., 2021. Overview and empirical research on file correlation in cloud storage. *Procedia Comput. Sci.* 188, 33–39.
- Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., Gao, J., 2021. Deep learning-based text classification: a comprehensive review. *ACM Comput. Surv.* 54 (3), 1–40.
- Ming, L., Hu, X., 2015. Recurrent convolutional neural network for object recognition. In: IEEE Conference on Computer Vision & Pattern Recognition. pp. 3367–3375.
- Onan, A., 2023. Hierarchical graph-based text classification framework with contextual node embedding and BERT-based dynamic fusion. *J. King Saud University-Computer Inf. Sci.* 35 (7), 101610.
- Orlove, B., 2021. How people name seasons. In: *Weather, Climate, Culture*. Routledge, pp. 121–140.
- Park, H.-S., Jun, C.-H., 2009. A simple and fast algorithm for K-medoids clustering. *Expert Syst. Appl.* 36 (2), 3336–3341.
- Potika, K., Eirinaki, M., Vitali, M., Bernasconi, A., Fujioka, H., 2025. Special issue on big data computing service and machine learning applications. *Future Gener. Comput. Syst.* 107836.
- Razno, M., 2019. Machine learning text classification model with NLP approach. *Comput. Linguist. Intell. Syst.* 2, 71–73.
- Riska, A., Riedel, E., 2006. Disk drive level workload characterization. In: *USENIX Annual Technical Conference, General Track*, vol. 2006, pp. 97–102.
- She, X., Zhang, D., 2018. Text classification based on hybrid CNN-LSTM hybrid model. In: 2018 11th International Symposium on Computational Intelligence and Design, Vol. 2. ISCID, IEEE, pp. 185–189.
- Smith, S., 2014. *Seasons*. Capstone.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. *Adv. Neural Inf. Process. Syst.* 30.
- Wan, C.X., Li, B., 2022. Financial causal sentence recognition based on BERT-CNN text classification. *J. Supercomput.* 1–25.
- Xie, T., 2008. Sea: A striping-based energy-aware strategy for data placement in raid-structured storage systems. *IEEE Trans. Comput.* 57 (6), 748–761.
- Xu, D., Tian, Y., 2015. A comprehensive survey of clustering algorithms. *Ann. Data Sci.* 2 (2), 165–193.
- Xu, R., Wunsch, D., 2005. Survey of clustering algorithms. *IEEE Trans. Neural Netw.* 16 (3), 645–678.
- Xue-mei, L., Hai-rui, L., et al., 2023. Text classification of water diversion project inspection text based on BERT-BiLSTM model. *China Rural. Water & Hydropower* (10).
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V., 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Adv. Neural Inf. Process. Syst.* 32.
- Yang, Y., Zhu, J., 2016. Write skew and zipf distribution: Evidence and implications. *ACM Trans. Storage (TOS)* 12 (4), 1–19.
- Yao, L., Mao, C., Luo, Y., 2019. Graph convolutional networks for text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, (no. 01), pp. 7370–7377.
- You, X., Lv, X., Zhao, Z., Han, J., Ren, X., 2020. A survey and taxonomy on energy-aware data management strategies in cloud environment. *IEEE Access* 8, 94279–94293.
- You, X., Sun, T., Sun, D., Liu, X., Lv, X., Buyya, R., 2021. K-ear: Extracting data access periodic characteristics for energy-aware data clustering and storing in cloud storage systems. *Concurr. Comput.: Pr. Exp.* 33 (9), e6096.
- Yu, Z., Li, H., Feng, J., 2024. Enhancing text classification with attention matrices based on BERT. *Expert Syst.* 41 (3), e13512.
- Yu, S., Liu, D., Zhu, W., Zhang, Y., Zhao, S., 2020. Attention-based LSTM, GRU and CNN for short text classification. *J. Intell. Fuzzy Systems* 39 (1), 333–340.
- Yuan, Z., Lv, X., Gong, Y., Liu, B., Yang, H., You, X., 2024a. Text semantics-driven data classification storage optimization. *Appl. Sci.* 14 (3), 1159.
- Yuan, Z., Lv, X., Gong, Y., Xie, P., Yuan, T., You, X., 2024b. Cost-effective data classification storage through text seasonal features. *Future Gener. Comput. Syst.* 158, 472–487.
- Yuan, Z., Lv, X., Xie, P., Ge, H., You, X., 2022. CSEA: A fine-grained framework of climate-season-based energy-aware in cloud storage systems. *Comput. J.* bxac184.
- Zeberga, K., Attique, M., Shah, B., Ali, F., Jembre, Y.Z., Chung, T.-S., 2022. A novel text mining approach for mental health prediction using Bi-LSTM and BERT model. *Comput. Intell. Neurosci.* 2022.
- Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., Liu, Q., 2019. ERNIE: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*.
- Zhang, T., Li, F., 2014. Energy-efficient algorithm based on data classification for cloud storage system. *J. Comput. Appl.* 34 (8), 2267.
- Zhang, J., Li, Y., Tian, J., Li, T., 2018. LSTM-CNN hybrid model for text classification. In: 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference. IAEAC, IEEE, pp. 1675–1680.
- Zhang, Y., Wallace, B., 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.
- Zheng, S., Yang, M., 2019. A new method of improving BERT for text classification. In: *International Conference on Intelligent Science and Big Data Engineering*. Springer, pp. 442–452.
- Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., Xu, B., 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.
- Zulqarnain, M., Ghazali, R., Ghouse, M.G., Mushtaq, M.F., 2019. Efficient processing of GRU based on word embedding for text classification. *JOIV: Int. J. Informatics Vis.* 3 (4), 377–383.