Multimedia Tools and Applications (2024) 83:40569–40605
https://doi.org/10.1007/s11042-023-16958-7

Low power high speed FPGA design of lossless medical image compression using optimal deep neural network

Sanjeev Sharma[1]

Received: 2 June 2023 / Revised: 8 August 2023 / Accepted: 11 September 2023 /
Published online: 9 October 2023

Abstract
Medical image compression is vital for preserving image quality and optimizing storage and transmission in healthcare facilities. FPGA design faces challenges in balancing compression ratios, speed, and power consumption. This study proposes a novel approach using the Selfish Herd Optimization (SHO) algorithm to cluster medical image data based on anatomical features. A hybrid deep convolutional neural network generates optimal predictors for each cluster, extracting features from the image data and generating predictions for each pixel based on its neighbors. The proposed system achieves low power and high-speed FPGA design while ensuring reliability and accuracy. Experimental results on Virtex-7 VC709 and Virtex-5QV130FX FPGAs demonstrate impressive improvements, including a 244.08% increase in frequency, 210.04% increase in speed, 200.90% increase in throughput, 61.37% increase in efficiency, and significant reductions in resource utilization and power consumption. This design promises efficient lossless medical image compression with enhanced performance.

Keywords Low power · High speed · Hardware design · FPGA · Lossless image compression · Medical image

Abbreviations

FPGA	Field-Programmable Gate Array, a reconfigurable hardware used for high-speed image compression
SHO Algorithm	Selfish Herd Optimization algorithm, used for clustering medical image data based on features
VC709 FPGA	Virtex-7 VC709 FPGA, one of the devices used for evaluating the proposed FPGA design
5QV130FX FPGA	Virtex-5QV130FX FPGA, another device used for evaluating the proposed FPGA design
LUTs	Lookup Tables, used for logic function implementation in the FPGA
BRAMs	Block RAMs, used for efficient data storage and retrieval in the FPGA

✉ Sanjeev Sharma
sanjeevsharmaphd23@gmail.com

[1] New Horizon College of Engineering, Bangalore, India

DSPs	Digital Signal Processors, used for complex mathematical operations in the FPGA
FPGA	Field-Programmable Gate Array, a reconfigurable hardware used for various applications, including image compression
SHO Algorithm	Selfish Herd Optimization algorithm used for clustering data based on features
Hybrid DCNN	Hybrid Deep Convolutional Neural Network, a neural network used for image feature extraction and prediction
ReLU	Rectified Linear Unit, a commonly used activation function in deep learning models
SEO-DCNN	Selfish Herd Optimization—Deep Convolutional Neural Network method for image compression
LCPLC Coder	Lossless Context-Based Predictive Linear Coding, a technique used for lossless image compression
LCE Coders	Lossless Context-Based Entropy Coding, a technique used for lossless image compression
BPP	Bits Per Pixel, a metric to measure the average number of bits used to represent each pixel in the compressed image
PSNR	Peak Signal-to-Noise Ratio, a metric to measure the image quality by comparing the original and compressed images
CR	Compression Ratio, a metric to measure the data compression efficiency

1 Introduction

1.1 Background

Medical image compression refers to the process of reducing the size of medical images while preserving their diagnostic quality. It is a technique used to efficiently store and transmit large volumes of medical image data, such as X-rays, CT scans, MRI images, ultrasound images, and more [1].

Compression is necessary in the field of medical imaging due to the high-resolution and data-intensive nature of these images. By reducing the size of medical images, several benefits can be achieved. Compressed images require less storage space, enabling healthcare facilities to efficiently manage their image archives and store a larger number of images within limited storage capacities. It can be transmitted over networks more quickly, reducing the time required for image transmission and enabling remote access to medical images for consultations, research, and telemedicine applications [2].

Faster transmission and retrieval of medical images lead to improved workflow efficiency in healthcare facilities, allowing for faster image review, diagnosis, and treatment decisions [3]. By reducing storage requirements and minimizing network bandwidth utilization, medical image compression can help lower infrastructure and operational costs associated with managing and transmitting large amounts of imaging data. There are several types of medical image compression techniques employed in the field of medical imaging [4]. These techniques can be broadly categorized into two main types: lossless compression and lossy compression. Lossless compression techniques aim to reduce the size of medical images without any loss of information [5]. These techniques ensure that the

compressed image can be perfectly reconstructed to its original form. Lossless compression is typically used in medical imaging to maintain the diagnostic accuracy and integrity of the images. Lossy compression techniques reduce the size of medical images by selectively discarding some information that is considered less important or less noticeable to the human eye [6]. While lossy compression results in some loss of image quality, it can achieve higher compression ratios compared to lossless compression. However, in medical imaging, the use of lossy compression is typically limited to non-diagnostic purposes, such as telemedicine, teaching, and research [7].

1.2 Literature review

Field-programmable gate array (FPGA) [8–10] design refers to the process of implementing digital circuits and systems using FPGA devices. FPGAs are programmable semiconductor devices that contain an array of configurable logic blocks, programmable interconnects, and other components. They can be reprogrammed or reconfigured to implement a wide range of digital functions, making them highly flexible for various applications. FPGA design offers numerous advantages, including high flexibility, reconfigurability, and performance capabilities. It finds applications in various fields such as digital signal processing, image and video processing, communication systems, embedded systems, and more [11, 12]. FPGA designs can be tailored to specific requirements, making them popular choice for implementing complex digital systems efficiently. The complexity of FPGA design for medical image compression depends on several factors, including the chosen compression algorithm, desired compression ratio, image resolution, FPGA device capabilities, and specific design requirements [13–15].

The complexity of the compression algorithm itself plays a significant role. Different algorithms have varying computational requirements, memory utilization, and implementation complexities. For example, JPEG-LS [16] or JPEG2000 [17] may involve complex mathematical operations or multi-stage processing, which can increase the design complexity. The size and resolution of medical images impact the design complexity. Higher resolution images require more processing power and memory resources for compression. Large images may require parallel processing or efficient data streaming techniques to achieve real-time compression [18]. The capabilities of the FPGA device, such as available logic elements, memory resources, DSP blocks, and clock frequencies, affect the design complexity [19].

1.3 Research gap and motivation

However, utilizing FPGA resources efficiently and optimizing for performance and power consumption is crucial. To achieve high-speed processing, FPGA designs for image compression often exploit parallelism and pipelining techniques [20].

1.4 Challenges

Ensuring the correct functionality and performance of the FPGA design requires comprehensive verification and testing processes. Simulating the design, generating test benches, and testing with diverse medical image datasets can be time-consuming and complex.

1.5 Our contributions

This paper presents a novel approach for designing a low power, high-speed FPGA implementation of lossless medical image compression. The design incorporates an optimal deep neural network to achieve efficient compression while maintaining high levels of reliability and accuracy.

1. The selfish herd optimization (SHO) algorithm is utilized in the proposed approach to cluster the medical image data based on their anatomical features. In SHO, each pixel within the image is considered as an individual, and the algorithm aims to group pixels that share similar features together, forming clusters. This clustering process helps to identify coherent regions within the image that can be effectively compressed together. It aids in the efficient compression of the image, as pixels within the same cluster can be encoded together, preserving the inherent relationships between them.
2. Hybrid deep convolutional neural network (DCNN) is employed to generate a series of optimal predictors for each cluster of pixels identified by the SHO algorithm. The DCNN has the capability to learn and extract relevant features from the medical image data through its deep layers. By analyzing the neighboring pixels within a cluster, the DCNN can generate predictions for each pixel, taking into account the local context and spatial information. The compression process can take advantage of the spatial dependencies and context within the image, resulting in improved compression performance and better preservation of image quality.
3. In order to evaluate the proposed work's performance and effectiveness, low power, high-speed FPGA design for lossless medical image compression, experimental evaluations were conducted using two distinct FPGA devices: Virtex-7 VC709 and Virtex-5QV130FX family. By employing these FPGA devices, the proposed design was tested and analyzed, providing valuable insights into its effectiveness and suitability for the task of lossless medical image compression.

1.6 Objectives

1. To address the challenges of lossless medical image compression, the proposed approach leverages FPGA design for low power and high speed.
2. To improve compression efficiency, the Selfish Herd Optimization (SHO) algorithm is utilized to cluster medical image data based on anatomical features.
3. To enhance predictive compression, a hybrid deep convolutional neural network (DCNN) is employed to generate optimal predictors for each SHO-formed pixel cluster.
4. To validate the effectiveness, experimental evaluations are conducted on Virtex-7 VC709 and Virtex-5QV130FX FPGAs to assess compression performance.

1.7 Scope of the proposed approach

1. **Optimizing FPGA-based lossless medical image compression:** The main scope is to design and implement efficient FPGA architectures that strike a balance between compression ratios, processing speed, and power consumption, ensuring the integrity and accuracy of medical images during compression.

2. **Selfish Herd Optimization (SHO) algorithm:** The scope involves applying SHO to cluster medical image data based on anatomical features, forming meaningful pixel clusters, which can be further processed using the proposed hybrid deep convolutional neural network.
3. **Hybrid Deep Convolutional Neural Network:** The scope includes developing and training the hybrid neural network to generate optimal predictors for each SHO-formed cluster, leveraging deep learning to extract relevant features and achieve accurate lossless compression.

1.8 Organization of paper

To provide a structured presentation of the research and this paper's remaining sections are arranged as follows: In Section 2, a comprehensive review of recent works in the field of lossless medical image compression is provided. Section 3 presents the problem methodology and system design of the proposed work. The detailed working process of the proposed work, utilizing the FPGA architecture, is presented in Section 4. In Section 5, the simulation results and comparative analysis are discussed. The proposed work's performance is evaluated and compared to previous approaches, providing insights into its effectiveness and advantages. Finally, the paper concludes in Section 6, summarizing the key findings and contributions of the research.

2 Related works

In this section, we discuss the comprehensive review of recent works in the field of lossless medical image compression. This review aims to provide an overview of the advancements and contributions made by previous studies in this area. By analyzing and summarizing the key findings and techniques from these works, the section offers valuable insights into the current state of the field and sets the foundation for the proposed research. Table 2 provides a summary of the research gaps identified based on an analysis of previous works in the field of lossless medical image compression.

Kiranmaye et al. [21] have introduced a memory-efficient high-speed architecture for lossless medical image compression. The architecture is based on an Ortho-normalized multi-stage Fast-DST processing unit, specifically designed for performing lifting operations. The multi-stage transform unit considers the previously neglected odd samples, enabling simultaneous execution of both odd and even samples and resulting in a significant speedup of the compression process. To optimize the processing speed, the architecture incorporates the use of delay elements and lifting coefficients to construct the RTU and CTU. These units facilitate the split, predict, and update operations while maintaining the desired processing speed. Furthermore, to resolve the issue of high memory cost, various phases of the proposed DST unit are consolidated to make an equal multi-stage engineering. This parallel architecture enables the execution of multiple stages in parallel, enhancing the overall efficiency of the compression process while minimizing hardware costs.

Chen et al. [22] have introduced a compressor and decompressor design capable of supporting 3-lead compression without incurring additional hardware costs or increasing the required area. Through their experimental analysis, they demonstrated that the design offers improvements in bit compression ratio and power consumption. To validate the effectiveness, the researchers fabricated chip using 0.18-μm CMOS technology. The

encoder, operating at a frequency of 20 MHz, was found to have a gate count of 4.8 K. Similarly, the decoder, operating at a frequency of 10 MHz, also had a gate count of 4.8 K.

Kim et al. [23] have introduced a method of parallelization that is specifically applied to a DEFLATE offload engine in order to solve the issue of Huffman encoding's loop-carried dependency. To enable the proposed offload engine's seamless hardware pipeline operation, a novel data representation strategy is used for the intermediate data generated during the compression process. The circle conveyed reliance is actually disposed of on the grounds that the technique produces middle of the road information from the offset data of every image bunch. A Huffman encoding calculation was developed by making use of this transitional information design, which completely eliminates the circle conveyed reliance. The experiments revealed that by removing data dependency, this approach significantly enhances the Huffman encoder's performance. All around, the introduction of the Breakdown blower is dealt with by 14.4%.Tsigkanos et al. [24] have introduced a high-performance parallel implementation of the CCSDS-123.0-B-1 hyperspectral compression algorithm specifically designed for SRAM FPGAs. The architecture takes advantage of image segmentation to ensure robustness against data corruption and enables scalable throughput performance by leveraging segment-level parallelism. By implementing a 5-core configuration on a Zynq-7045 FPGA, the architecture achieves an impressive throughput performance of 1387MSPS. This high throughput capability allows for seamless integration with next-generation hyperspectral sensors, facilitating efficient processing of data from these advanced sensors.

Farghaly et al. [25] have presented a discrete wavelet transform (DWT)-based image compression system implemented in hardware. The architecture utilizes a rendered structure limited motivation reaction (FIR) channel to perform the convolution process, which forms the basis of the DWT. The design is designed to be generic allowing it to accommodate multiple wavelet types and is symmetric, allowing for filter expansion with multiple taps.The hardware design is specifically implemented on Virtex 5 FPGA, demonstrating its practical feasibility. The achieved clock frequency for the implementation is 243.6 MHz, indicating the system's ability to perform computations efficiently at a high speed.

Descampe et al. [26] have developed a compression codec known as JPEG XS, which addresses the tradeoff between rate, distortion, complexity, and latency that has been a priority in previous standardization efforts. The primary focus of the JPEG XS project has been to achieve the optimal balance between quality, complexity, and latency for specific use cases. The codec employs an efficient coding scheme that allows for low latency and complexity, while simultaneously delivering visually lossless quality at compression ratios of up to 10:1 or even higher, depending on the image characteristics or application requirements. Through quality assessments, it has been demonstrated that the JPEG XS codec performs exceptionally well compared to other existing codecs, particularly in multi-generation applications (Table 1).

Filho et al. [27] have introduced the JPEG XR compression standard, initially implemented in software for processing remote-sensing images. Subsequently, they developed a hardware solution specifically designed for onboard applications. The VHDL code was divided into two separate parts, which were then merged and optimized to enhance performance. JPEG XR utilizing Cyclone FPGA demonstrated a favorable balance between throughput and power consumption. With a throughput of 26 million pixels per second and a power consumption of approximately 140milliwatts, this implementation proves to be advantageous for real-time processing in small earth-observation satellite applications. They evaluated the approach using JPEG XR and observed no discernible changes when processing blocks consisting of 480 lines. For blocks comprising 48 lines, there was a

Table 1 Summary of Research gaps

Ref	Methodology	Algorithm used	Findings	Research gaps
[21]	High-speed VLSI design of image compression	Ortho normalized multi-stage DFST	Memory complexity, low power, low latency	The cost associated with sampling redundant data is high
[22]	Lossless healthcare data compressor	3-lead compression	Operating frequency, power consumption	The utilization of CPU resources is significant, resulting in slow data processing speed
[23]	Data dependency reduction in compression	DEFLATE compression	Throughput	Quality, complexity, and latency have an impact on the system
[24]	Parallel Hyperspectral image compression	Commercial-off-the-shelf(COTS)	Throughput and latency	The usage of an arithmetic coder adds considerable complexity
[25]	FPGA design of image compression	Floating-point DWT	Compression ratio, accuracy, Throughput	Re-normalization process creates complexity in FPGA design
[26]	Lossless low-latency image coding	JPEG XS	Memory complexity, power consumption	The data size creates a bottleneck in downlink bandwidth
[27]	FPGA design of JPEG XR	DC/LP/HP encoding on JPEG XR	Power consumption, clock frequency	Scaling the process to achieve area efficiency is challenging
[28]	FPGA design of lossless image compression	Counting sorting and binary tree	Encoding rate	Fully exploiting parallelism proves to be difficult
[29]	FPGA design of lossless, near-lossless image compression	LOCO-ANS	Throughput and compression ratio	The calculation ofprobability counting before encoding consumes a considerable amount of time
[30]	VLSI design of block truncation coding	Golomb–Rice coding	Operating frequency, area	The compression ratio of bits is low, and there are issues with signal damage during recovery
[31]	FPGA accelerators with WP and RNS	Wavelet processing (WP) with scaled filter coefficients (SFC)	Improved performance of 3D medical image WP systems with reduced computational complexity and high-quality processing	Further exploration of different filter coefficient scaling algorithms and optimizations for resource utilization

Table 1 (continued)

Ref	Methodology	Algorithm used	Findings	Research gaps
[32]	Low-precision representations in deep learning systems on FPGA-SoC	FPGA-SoC, CNN	Achieved scalability in performance, storage, and power efficiency while maintaining desired accuracy	Investigation into improving embedded FPGA execution for larger CNN models and potential trade-offs between model size and classification accuracy

negligible reduction in compression. These results indicate the effectiveness of their hardware implementation and its suitability for satellite-based image processing applications.

Dang [28] proposed a compression algorithm based on Canonical Huffman coding, which involves preprocessing the data source to be compressed based on the features of the image data. The algorithm performs data compression in batches, leveraging the locally uneven features within the data. This approach results in a compression ratio that is 1.678 times better than traditional canonical Huffman coding. The compression algorithm was implemented on an FPGA, utilizing parallel hardware circuits and pipeline design to improve the encoding rate. The experimental results demonstrated a compression rate that is 1.67 times higher than traditional Huffman coding. Additionally, the time complexity was significantly reduced, enabling efficient real-time data processing.

Alonso et al. [29] introduced hardware architecture for low complexity lossless compression with asymmetric numeral system (LOCO-ANS) based on the JPEG-LS standard. The Zynq 7020 FPGA achieved a throughput of up to 40.5 MPPS, while the UltraScale + FPGA achieved a throughput of up to 124 MPPS. These results indicate the superior performance of the hardware implementation compared to a single-threaded LOCO-ANS software implementation running on a 1.2 GHz Raspberry Pi 3B.Even when the Zynq 7020, an older, cost-effective chip, is used, each hardware lane of the architecture achieved a throughput that is 6.5 times higher than the software implementation. Furthermore, the lossless compression achieved a lower impression and half better execution contrasted with the rendition that upholds both lossless and near-lossless compression.

Chen et al. [30] proposed a VLSI design for Internet of Things image sensors, aiming to achieve high compression ratios and real-time processing for image compression. The design incorporates various modules, including the Sub-sampling, prediction, quantization, the YEF transform, color sampling, block truncation coding (BTC), threshold optimization, and Golomb-Rice coding are some of the other techniques. The image compression process involves several stages. YEF transform is applied to convert the image data into a format suitable for further processing. Then, color sampling is performed to reduce the color space complexity. The BTC technique is utilized to divide the image into 4×4 blocks, which make it possible to convert numbers and get rid of redundancy between pixels. By training different BTC parameters, an optimal solution is obtained considering the given parameters. The algorithm achieves a compression ratio that is higher than that of standard BTC-based image compression algorithms.

Nagornov et al., [31] developed FPGA accelerators using wavelet processing (WP) with scaled filter coefficients (SFC) and parallel computing in residue number system (RNS) to improve the performance of high-quality 3D medical image WP systems. The computational complexity is reduced using the developed WP method with SFC and the wavelet filter coefficients scaling algorithm. Parallel computing is organized in RNS using moduli sets of a particular type. Hardware implementation of 3D medical image WP using the proposed FPGA accelerators increases device performance by 2.89–3.59 times, increasing the hardware resources by 1.18–3.29 times compared to state-of-the-art solutions. The device performance improvement is achieved while maintaining high-quality 3D medical image processing in peak signal-to-noise ratio terms.

Ghodhbani et al., [32] developed low-precision representations of neurons, inputs, model parameters, and activations have become a promising solution. These reduced-precision models offer scalability in performance, storage, and power efficiency while sacrificing some accuracy. By leveraging reconfigurable hardware such as FPGAs-SoC, deep learning systems can take advantage of low-precision inference engines while achieving the desired accuracy and balancing performance, power consumption, and programmability.

Despite the high redundancy and excellent classification accuracy provided by CNN, the increasing model size makes it challenging to execute applications on embedded FPGAs.

3 Problem methodology and System design of proposed work

3.1 Research gaps

Medical images often require lossless compression due to the critical nature of the information they contain. Lossless compression ensures that all the original data in the image is preserved without any loss of quality or detail. Medical images play a crucial role in diagnosing and treating various conditions. Lossless compression ensures that no information is lost during the compression process, allowing physicians to accurately interpret the images and make precise diagnoses. It assists doctors in making critical decisions about patient care, including treatment plans and surgical interventions. Lossless compression guarantees that all the relevant details are retained, enabling physicians to make informed decisions based on the highest quality images available. It often serves as legal documents and need to be stored and transmitted without any loss of information. Lossless compression ensures the integrity of the image data, maintaining its legal validity and adhering to regulatory standards. Lossless compression allows researchers to analyze images with the highest level of precision and accuracy, enabling advancements in medical knowledge and technology. Lossless compression ensures that medical images can be stored and archived without degradation over time. By preserving the original quality, the images can be accessed and reviewed in their intended form, even years or decades later. Overall, lossless compression is necessary for medical images to maintain their fidelity, diagnostic value, legal validity, and long-term accessibility, supporting effective patient care, and education in the medical field.

Bascones et al. [33] have introduced predictive lossy compression (LCPLC) algorithm with low complexity capable of operating efficiently through a lengthy pipeline comprising hundreds of stages. This algorithm effectively minimizes stalls and achieves a throughput that closely approaches the theoretical maximum. The algorithm primarily operates on integers using simple arithmetic operations, making it highly suitable for implementation on FPGA platforms. Experimental The results on a Virtex-7 FPGA show a remarkable maximum frequency of over 300 MHz, allowing for a throughput of over 290 Mbps. Additionally, a Virtex-5 FPGA with space qualification achieved a frequency of 258 MHz, marking a fivefold improvement compared to previous FPGA designs. To enhance the same compression ratio and quality and improve performance, the algorithm employs a larger block size of 32×32 instead of the default 16×16, reducing the pipeline's fill-up frequency. The modular data flow design ensures that the maximum frequency remains well above 300 MHz. The predictor module in lossless image compression faces several challenges and problems. The accuracy of the predictor module is crucial in achieving efficient compression. If the predictor fails to accurately estimate pixel values, it can lead to significant compression artifacts and loss of information.The predictor module should be able to adapt to different image characteristics and handle various types of image data effectively. It should be capable of capturing and utilizing the inherent spatial or temporal correlations present in the image. Designing a predictor module that balances between accuracy and complexity is a challenge. Complex predictor algorithms may require substantial computational resources, increasing the overall processing time and system complexity. Some

advanced predictor models may need to store a significant amount of historical data or reference pixels for accurate prediction. This can increase the memory requirements of the predictor module, which may not be feasible in resource-constrained systems.

Predicting pixel values accurately in regions with sharp edges or complex textures is challenging. The predictor module should be able to handle such regions effectively to avoid introducing compression artifacts. The predictor module should be scalable to handle images of varying sizes and resolutions. It should provide consistent performance across different image dimensions without sacrificing accuracy or speed. The module should be robust to noise and variations in input data. It should be able to handle different image types, compression levels, and noise levels without significant degradation in compression performance. In applications requiring real-time processing, the predictor module should operate efficiently within strict time constraints. It should provide fast and accurate predictions to enable real-time compression and decompression. Addressing these problems and challenges in the predictor module is essential for achieving high-quality and efficient lossless image compression.

To address the problems in the predictor module of lossless image compression, the following research objectives can be pursued.

1. To focus on developing advanced prediction algorithms that enhances accuracy and adaptability. This involves exploring new techniques to capture and exploit spatial or temporal correlations in images effectively.
2. Investigate methods to optimize the computational complexity of prediction algorithms without compromising accuracy. This includes exploring efficient algorithms and hardware acceleration to improve processing speed and reduce resource utilization.
3. To develop scalable prediction techniques that performs consistently across images of varying sizes and resolutions.
4. Explore techniques to improve the robustness of the prediction module against noise and variations in input data.
5. Investigate methods to optimize the prediction module for real-time healthcare processing requirements.

3.2 System design of proposed work

The proposed FPGA design for lossless image compression begins with the consideration of input images, which can be both non-medical and medical images. These images are analyzed to identify their specific feature sets, which may include characteristics such as color distribution, texture, or spatial patterns. Once the input images and their feature sets are determined, a preprocessing step is performed. This step involves various operations to enhance the quality and prepare the images for further analysis (Fig. 1).

Preprocessing techniques may include color space conversion, noise reduction, or resizing to ensure compatibility with subsequent algorithms. To extract relevant features from the images, the selfish herd optimization (SHO) algorithm is employed. This algorithm treats each pixel in the image as an individual entity and attempts to group pixels with similar features together, forming clusters based on anatomical features. By applying the SHO algorithm, the system aims to identify anatomical patterns and structures within the images.

After the anatomical features are extracted using the SHO algorithm, a comparison is made between these features and the predefined feature set. This comparison allows for

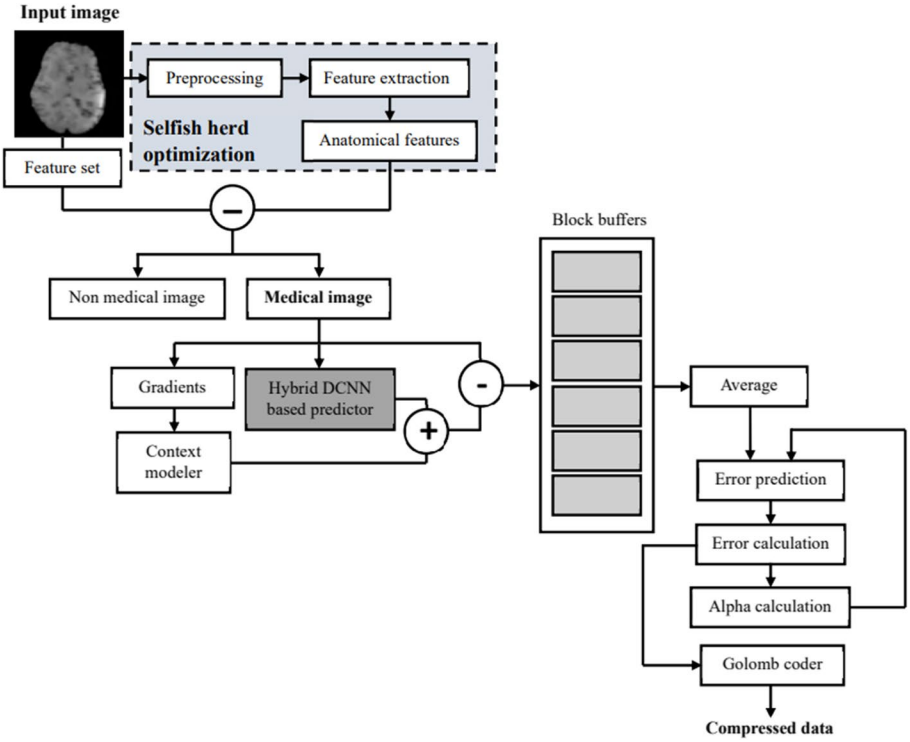

Fig. 1 Overall system model of proposed work

the classification of images into medical and non-medical categories based on the presence of anatomical features. This step helps in determining the appropriate compression techniques and algorithms to be applied. For medical images, the compression process proceeds by utilizing gradients and predictors through a hybrid deep convolutional neural network (DCNN). The input medical image is fed into the DCNN, which is capable of learning and extracting features from the image data. The DCNN generates predictions for each pixel based on its neighboring pixels and also computes the gradients. The gradients and predictor outputs are combined through an addition operation, and the resulting output is minimized with the input data. This minimization step aims to reduce the error between the original image and the compressed representation. The output is organized into block buffers, which provide compact format for storing or transmitting the compressed medical image data.

For each cut of the block cushion, the typical worth is figured as a kind of perspective point. Predictions and prediction errors are made using this average and the alpha values from the previous slice. After that, these errors are encoded and fed back into the alpha calculation procedure to produce the output. The plan of the center intended to be flexible and equipped for handling pictures and blocks of fluctuating sizes. The size of the slice determines the inner buffer sizes, and the queues dynamically store two slices during processing. Because the image is divided into blocks and slices prior to being processed, the overall size of the image has no direct effect on the size of the queue. As a result, the core is capable of handling any image, regardless of how small or large it is. Smaller blocks are

taken from the edges and processed in accordance with the image size if it is not a multiple of the block size. The two predictors' predictions are combined in alternating fashion. After a flag is raised, the merger shifts to the nth band predictor. At first, all predictions from the first band predictor are sent. Until all predictions from the nth band predictor are sent, this alternating pattern continues. The error is calculated by comparing the initial values to the predicted values after the prediction step. The residuals and coding boundaries are straightforwardly passed to the coder part. The alpha module takes into account both the predicted values and slice average from the previous band as well as the current values and slices average. The deviation from the mean in collocated positions between the current and previous slices is calculated using a least squares estimator. The estimator is created by adding up and dividing the squared errors. In the end, the coder component combines the outputs of the modules that came before it to create a bit stream with the information needed for decoding. The input error values are accompanied by flags that indicate the current position within the block, allowing for precise control of the output. The first step in the coding process involves separating the flags from the data to facilitate proper handling of the output.

4 Proposed methodology

In this section, we provide a detailed explanation of proposed methodology. This methodology comprises two main processes: feature extraction utilizing the selfish herd optimization (SHO) algorithm, and pixel value prediction employing a hybrid deep convolutional neural network (DCNN). We describe the step-by-step working process of each process and emphasize the hardware design of the hybrid DCNN model.

4.1 Feature extraction using selfish herd optimization (SHO) algorithm

In this work, the extraction of anatomical features serves a specific purpose of distinguishing between medical and non-medical images. By extracting anatomical features from the input images, we aim to capture unique characteristics that are typically found in medical images, such as specific patterns, structures, or textures related to medical anatomy. These extracted anatomical features are then compared with a pre-defined feature set that represents known anatomical features. By performing this comparison, we can determine whether the given input image is a medical image or not. This process enables us to classify the input image based on its content and differentiate between medical and non-medical images. The feature extraction step plays a crucial role in accurately capturing the relevant anatomical characteristics from the input image and providing the necessary information for subsequent classification. By incorporating feature extraction in this work, we can effectively analyze and differentiate between medical and non-medical images based on their anatomical features. The SHO algorithm is a nature-inspired optimization algorithm that is inspired by the self-protective behavior observed in animal herds. It is a population-based algorithm that simulates the behavior of a herd of animals, where each individual in the herd aims to maximize its own safety by maintaining a specific distance from potential threats.

In the SHO algorithm, a population of candidate solutions is represented as a herd of animals. Each candidate solution corresponds to an individual in the herd. The algorithm iteratively updates the positions of the individuals in the herd based on their own safety and

the safety of their neighbors. The algorithm incorporates two main behaviors observed in animal herds: attraction and repulsion. Attraction encourages individuals to move towards the center of the herd, promoting cooperation and information sharing. Repulsion, on the other hand, ensures that individuals maintain a safe distance from each other, reducing the risk of crowding and collision. By iteratively applying attraction and repulsion forces, the SHO algorithm explores the search space to find optimal or near-optimal solutions. It aims to strike a balance between exploration and exploitation to efficiently converge towards promising regions of the solution space. The SHO algorithm's population set is called S and contains N people. Each individual ($t_{m,n}$) is defined as follows

$$t_{m,n} = a_n^{Low} + Rand\left(a_n^{Low}, a_n^{High}\right) \tag{1}$$

where a_n^{Low} and a_n^{High} address the lower and upper limits. Also, the quantity of prey (J_g) and predators (J_P) is calculated via the following formulas:

$$J_g = Floor(J \cdot Rand\ (0.7,\ 0.9)) \tag{2}$$

$$J_P = J - J_g \tag{3}$$

where, Floor(·) represents function that transforms a real number into an integer. Every individual has an endurance esteem (Tv_{t_m}), which addresses its endurance capacity. We define the survival value as follows.

$$Tv_{t_m} = \frac{F(t_m) - F_{worst}}{F_{best} - F_{worst}} \tag{4}$$

where, Floor (·) represents the objective function. The movement of the prey leader and the following or escape movement of the prey followers make up the majority of this section. The following is a definition of a prey leader:

$$Tv_{g_l} = \underset{m\in\{1,2\ldots J_g\}}{MAX}(TV_{g_m}) \tag{5}$$

The following is an update to its position:

$$g_l = \begin{cases} g_l + 2\cdot\alpha\cdot\phi_{g_l,p_I}\cdot(P_I - g_l)\ if\ TV_{g_l} = 1 \\ g_l + 2\cdot\alpha\cdot\phi_{g_l,a_{best}}\cdot(P_{best} - g_l)\ if\ TV_{g_l} Ls;1 \end{cases} \tag{6}$$

where $\phi_{a,b}$ represents people's attraction to one another a and b, P_I represents a relatively dangerous position for the prey and P_{best} represents globally optimal position. $\phi_{a,b}$ and P_I are define as follows:

$$\phi_{a,b} = TV_b \cdot E^{-||a-b||^2} \tag{7}$$

$$P_I = \frac{\sum_{m=1}^{J_P} TV_{P_I}\cdot P_I}{\sum_{m=1}^{J_P} TV_{P_I}} \tag{8}$$

where ||a − b||2 represents the Euclidean distance between individuals that hunt prey a and b, TV_{g_l} represents the importance of predator survival m, and P_I represents the location of predator I. Prey supporters are separated into following prey (G_f) and escape prey (G_C),

and following prey (G_f) are further divided into dominant prey (G_D) and subordinate prey (G_T). G_f, G_C, G_D and G_T are defined as follows:

$$G_f = \left\{ g_m \neq g_l | TV_{g_l} < \text{Rand}(0,1) \right\} \tag{9}$$

$$G_C = \left\{ g_m \neq g_l | TV_{g_l} < \text{Rand}(0,1) \right\} \tag{10}$$

$$G_D = \left\{ g_m \in G_f | TVg_m \geq TV_{g_U} \right\} \tag{11}$$

$$G_T = \left\{ g_m \in G_f | TVg_m < TV_{g_U} \right\} \tag{12}$$

where TV_{g_U} addresses mean worth of prey, which is characterized as follows:

$$TV_{g_U} = \frac{\sum_{m=1}^{J_g} TV_{gm}}{J_g} \tag{13}$$

Algorithm 1 describes the working steps involved in the feature extraction using SHO.

4.2 Predictor using Hybrid DCNN

The predictor refers to a component or module that is responsible for estimating or predicting the pixel values of an image based on certain patterns or features. It aims to reduce the redundancy in the image data and enable efficient compression. The hybrid deep convolutional neural network (DCNN) is employed as the predictor in this work. DCNNs are a type of artificial neural network that have proven to be highly effective in image analysis and recognition tasks. They can learn complex patterns and features directly from the input data, making them suitable for tasks such as image compression. The hybrid aspect of the DCNN used in this work indicates that it combines elements from different types of neural networks or incorporates different techniques to enhance its performance. It may involve incorporating elements of both CNNs and other types of neural networks, or integrating traditional image processing techniques with deep learning approaches. The choice of a hybrid DCNN as the predictor in this work is motivated by its ability to capture intricate patterns and dependencies in the image data, which can improve the accuracy of pixel value predictions. By leveraging the power of deep learning, the hybrid DCNN can effectively model the relationships between image pixels and make more accurate predictions. Using a hybrid DCNN as the predictor allows for more sophisticated and flexible modeling of the image data, which can potentially lead to better compression results. It takes advantage of the deep learning capabilities to extract relevant features and patterns from the images, enabling more accurate predictions and reducing the amount of data that needs to be stored or transmitted. The hybrid DCNN consists following set of layers to predict the accurate pixel values.

1. Convolutional layer applies convolutional filters to the input image, capturing local features and patterns. Each filter produces a feature map by convolving over the input image. The filters learn to detect different visual patterns, such as edges or textures.

Input	: Input image, target pixels, maximum iteration
Output	: Anatomical features
1	Initialization parameters
2	Define individual ($t_{m,n}$) as $t_{m,n} = a_n^{Low} + Rand\left(a_n^{Low}, a_n^{High}\right)$
3	If a=0, and b=1
4	Compute the predators (J_P) using $J_g = Floor(J \cdot Rand(0.7, 0.9))$
5	While Do
6	Compute mean value of prey as $TV_{g_U} = \frac{\sum_{m=1}^{J_g} TV_{gm}}{J_g}$
7	Perform position updated $g_l = \begin{cases} g_l + 2 \cdot \alpha \cdot \varphi_{g_l, p_l} \cdot (P_l - g_l) if\ TV_{g_l} = 1 \\ g_l + 2 \cdot \alpha \cdot \varphi_{g_l, a_{best}} \cdot (P_{best} - g_l) if\ TV_{g_l} Ls; 1 \end{cases}$
8	Find the best output solution
9	End if
10	End

Algorithm 1 Feature extraction using SHO

2. Activation functions bring non-linearity into the organization, empowering it to learn complex connections in the information. Common activation functions include ReLU, sigmoid, and tanh.
3. In pooling layer, down sample the feature maps, reducing their spatial dimensions while preserving important features. Max pooling is a commonly used technique, where the maximum value within a pooling window is selected as the representative value.
4. Fully connected layer connects every neuron from the previous layer to every neuron in the current layer. It maps the learned features to the output classes, enabling classification or regression.
5. The output layer produces the final predictions or representations. The number of neurons in this layer depends on the specific task, such as binary classification, multi-class classification, or regression.

We define the input layer design of DCNN which follows the mapping dictionary $P\prime$ for the data handling.

$$P\prime = P \times F_{ad}(AD) \times F_T(t) \tag{14}$$

where P is the initial mapping dictionary, $F_{ad}(AD)$ and $F_T(t)$ are randomized functions, AD is the user identity, and t is the time interval, respectively.

$$G = \left(G\left(V_a, E_b\right)\right)_{|v|\times|e|} \tag{15}$$

We define the matrix for optimal function which defined is follows.

$$G(V_a, E_b) = \begin{cases} 1, \ if \ V_a \in E_b \\ 0 \ otherwise, \end{cases} \tag{16}$$

With the help of this matrix, the saliency of any vertex in the hypergraph, expressed as:

$$Gs(V_a) = \sum_{E \in e} G(V_a, \ E)\zeta(E) \tag{17}$$

where,$\zeta(E)$ helps to encode the obtained saliency information on hyper edge E. The optimal estimated value H can be described as follows.

$$H = C_{attention}([n_1 \cdot \overline{g}_1] \oplus [n_2 \cdot \overline{g}_2] ... \oplus [n_K \cdot \overline{g}_K]) \tag{18}$$

where, $C_{\text{attention}}$ $(\cdot)$ and $\oplus$ depict the fusion process. A convolution block is made up of a convolution layer and an average pooling layer. The range of abnormal peak points will be reduced by the action of the medium coupling, and the action of the small curve can reduce noise, which better reflects the equipment's operational state. The following are the mathematical formulas for the smoothed and de-noised layers:

$$n_b = F_1\left(w_b * H_b + I_b\right) \tag{19}$$

$$T_b^a = avg\left(n_b^a, \ldots \ldots a_b^B\right) \tag{20}$$

At last, the relapse layer, which comprises of a few complete association layers, can be determined as follows:

$$R = F_2\left(w_F \cdot \text{flatten}\left(T_b\right) + P_F\right) \tag{21}$$

The final predictive value is represented by r. w_F denotes the parameters of the weight, while a_F is the additional bias and flatten$(\cdot)$ is the flatten operation. The loss function l_2 of this network is designed by MSE function as follows.

$$L_2 = \arg \ \min \sum_{b=1}^{P} (r - \widehat{r})^2 \tag{22}$$

where, $\widehat{r}$ actual value, and P is the prediction sequence's length. Algorithm 2 describes the working process involved in the pixel value predictor using hybrid DCNN.

Input	: input image, training set, testing set and threshold condition
Output	:pixel value predictor
1	Initialization parameters
2	Define the input layer module $P^{'} = P \times F_{ad}(AD) \times F_{T}(t)$
3	If a=0, and b=1
4	Compute saliency of any vertex $Gs(V_a) = \sum_{E \in e} G(V_a, E)\zeta(E)$
5	While Do
6	Define de-noised layer are as $n_b = F_1(w_b * H_b + I_b)$
7	Compute MSE function $L_2 = \arg\min \sum_{b=1}^{P} (r - \hat{r})^2$
8	Find the best output solution
9	End if
10	End

Algorithm 2 Pixel value predictor using hybrid DCNN

4.3 Hardware design of Hybrid DCNN

The FPGA hardware design of the proposed hybrid DCNN for pixel value prediction involves several sub-modules as shown in Fig. 2. Here is a brief description of these sub-modules with respect to FPGA hardware design. The input buffer module is responsible for storing the input data, such as image pixels, before feeding them into the convolutional layer. In FPGA design, the input buffer can be implemented using on-chip memory resources, such as block RAMs, to efficiently store and access the input data. The convolutional layer performs the main computation in the DCNN by convolving the input data with a set of learnable filters. In FPGA hardware design, the convolutional layer can be implemented using parallel processing units or dedicated hardware accelerators. Efficient memory access patterns and data reuse techniques can be employed to optimize the computation and maximize parallelism. The rectified linear unit (ReLU) function is an activation function applied element-wise to the output of the convolutional layer. It introduces non-linearity into the network and aids in the data's capture of complex relationships. The ReLU function can be implemented as a simple hardware module that applies a thresholding

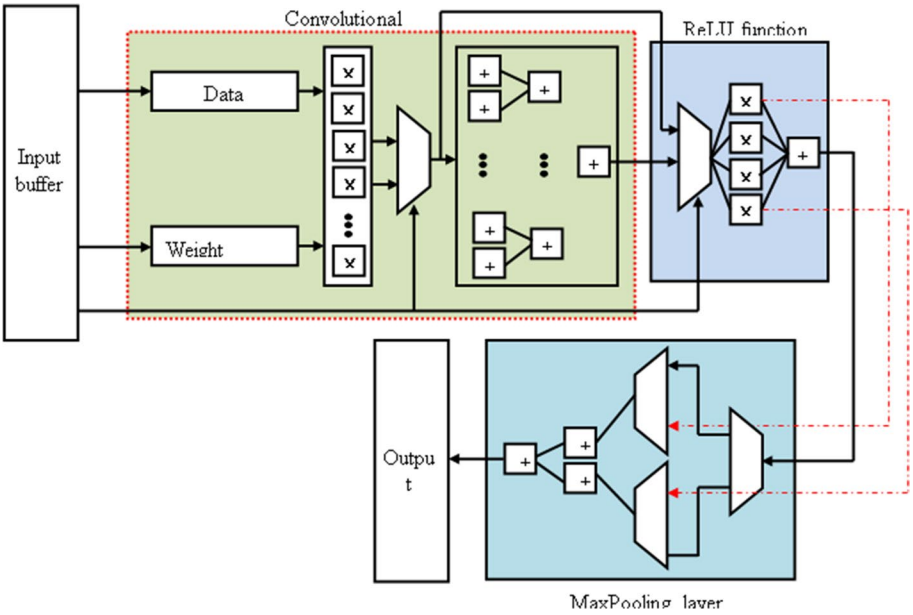

Fig. 2 Overall hardware design of proposed hybrid DCNN

operation to the input values, discarding negative values. The MaxPooling layer reduces the spatial elements of the component maps by choosing the most extreme worth inside a pooling district. This helps in down sampling the feature maps and extracting the most salient features. In FPGA hardware design, the MaxPooling layer can be implemented using dedicated hardware modules that perform the MaxPooling operation efficiently. The output layer takes the processed feature maps and produces the final prediction or output. In the context of pixel value prediction, the output layer could be designed to provide the predicted pixel values based on the learned weights and features. The output layer can be implemented using appropriate hardware modules and data formatting techniques to generate the desired output. Overall, in FPGA hardware design, each of these sub-modules needs to be carefully designed and optimized to efficiently utilize the available FPGA resources, such as on-chip memory and parallel processing capabilities. Techniques like parallelization, pipelining, and resource sharing can be applied to maximize the performance and throughput of the hybrid DCNN design on the FPGA platform. Additionally, considerations such as power optimization, memory bandwidth, and data management should be taken into account for an efficient and effective FPGA implementation.

Convolution operations play a significant role in hybrid DCNN, accounting for more than 90% of the network's computational workload. These operations involve sliding a convolution kernel over the input feature map and performing multiply-and-accumulate operations.

In the context of hardware design, it is possible to implement convolution operations between different kernels or layers, leveraging the independence and reusability of the data. Multiple convolution cores can operate on the same input feature map data simultaneously, with each operation being independent. Moreover, the input feature map's convolution sub-regions all share the same convolution kernel, resulting in spatial overlap. Data utilization and parallelism in convolution operations must be maximized in order to achieve

the best possible computational performance. The proposed approach presents a pipeline equal increase collect construction that plays out all essential duplicate gather tasks within a single clock cycle for a 3×3 convolution kernel. This design allows for 9 multiplication operations to be executed concurrently per clock cycle, with the final output feature pixels. Additionally, to handle networks requiring padding, a blocker is incorporated into the convolver module.

This blocker guarantees that zeros are added to the edge places of info include maps, empowering continuous pipeline activity without being impacted by information cushioning or component map exchanging. This design preserves the parallelism of convolution calculations at the circuit level. The FPGA equipment configuration incorporates an on-chip BRAM memory execution for information storing. On-chip BRAM blocks are assigned to each data input and output, and the same input and output interface is shared by multiple blocks. The set number of I/O connection points can affect the gas pedal's throughput rate, which is impacted by the information throughput pace of the store. The proposed solution suggests increasing the number of interfaces and dividing the cached array into various BRAM blocks to overcome this limitation. Accelerator data parallelism and data throughput can be improved at a faster rate, alleviating potential bottlenecks.

5 Results and discussion

In this section, we compare and contrast the outcomes of the proposed SEO-DCNN method with a comprehensive analysis existing FPGA designs of image compression techniques. The evaluation focuses on different test cases, including hyper spectral images and medical images, to ensure the proposed technique's performance viability. The FPGA implementation of the SEO-DCNN technique is conducted on Xilinx Vertex-5 and Vertex-7 FPGA families. The proposed method's simulation results are compared against state-of-the-art techniques such as LCPLC coder [33] and LCE coders [34–36]. Various performance metrics are considered for the comparative analysis, including hardware utilization, operating frequency, throughput, and power consumption. Through this comparative analysis, we aim to assess the effectiveness and efficiency of the proposed SEO-DCNN technique in image compression. The results will shed light on its performance advantages over existing techniques, providing insights into hardware utilization, operational speed, data processing capability, and energy efficiency. By conducting this thorough evaluation, we can better understand the strengths and limitations of the proposed SEO-DCNN technique, further validating its potential as an advanced image compression solution.

5.1 Result analysis with respect to compression efficiency

When implementing a design in FPGA, the technology is utilized by utilizing the FPGA's reconfigurable hardware resources, such as LUTs, BRAMs, and DSP blocks, to implement the desired logic and data processing functions. FPGA development involves using either high-level synthesis (HLS) tools or hardware description languages (HDLs) like VHDL or Verilog to describe the design. In the proposed method, the design requirements and specifications are defined, suitable algorithms are selected, and optimization techniques are employed to maximize performance and resource utilization while minimizing power consumption. Co-simulation and verification are performed to ensure design correctness before using vendor-specific FPGA development

tools to implement the design. The process involves iterative development, post-implementation analysis, integration with the system, and system-level testing to validate functionality. Proper documentation and preparation of the FPGA bitstream for deployment complete the development process, ensuring efficient and high-performance FPGA implementations.

The evaluation of the SEO-DCNN technique's compression efficiency involved testing it on a diverse set of image datasets. These datasets included images from the Spectir library, the CCSDS 123 test dataset, and the brain tumor MRI dataset.

5.1.1 CCSDS 123 test dataset

The CCSDS 123 test dataset (http://web.usm.my/jes/18_1_2022/JES_1801_2022_3.pdf) is a collection of images commonly used for evaluating image compression algorithms. It is provided by the Consultative Committee for Space Data Systems (CCSDS) as a standardized benchmark for image compression performance assessment. The dataset consists of a diverse set of images representing various content types, including natural scenes, scientific data, and remote sensing imagery. These images serve as representative samples of typical data encountered in space missions and Earth observation applications. Researchers and developers use the CCSDS 123 test dataset to assess the compression efficiency, image quality preservation, and computational performance of different image compression techniques, helping to advance the state-of-the-art in image compression for space missions and other related domains.

CCSDS 123 is a lossless compression algorithm recommendation defined by the CCSDS committee. This algorithm was explicitly created for hyperspectral images, taking advantage of its characteristics. It consists of two stages: prediction and encoding, presented in Fig. 3. The input of the algorithm is a three-dimensional image, and the resulting compressed image is an encoded bitstream, which can reconstruct the input image.

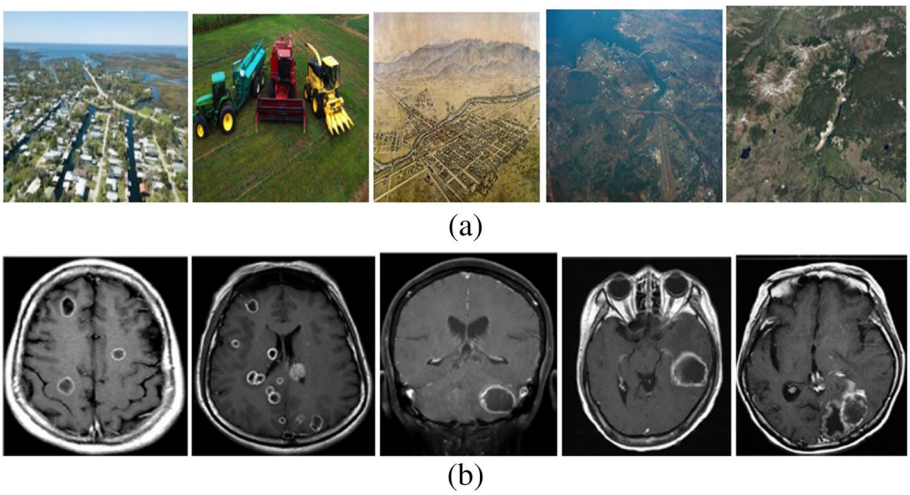

Fig. 3 Test sample images form (**a**) hyperspectral dataset (**b**) medial image dataset

5.1.2 Brain tumor MRI dataset

The "Brain Tumor MRI Dataset" (https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset) consists of 7,023 human brain MRI images. These images are classified into four categories, likely representing different types or characteristics of brain tumors. The dataset is likely curated for the purpose of brain tumor detection, segmentation, or classification tasks. The combination of multiple datasets ensures a diverse and comprehensive collection of brain MRI images, allowing researchers and developers to evaluate and benchmark their algorithms effectively. Such datasets play a crucial role in advancing the field of medical image analysis and aiding in the development of improved brain tumor diagnosis and treatment methods.

The Spectir library images used for evaluation consisted of various scenes captured from different locations. These scenes included an aerial view of the Suwanee natural reserve (SUW), crop fields in Beltsville, Maryland, USA (BEL), satellite views of Reno, Nevada, USA (REN), an image of the Cuprite Hills (CUP), views of Hawaii, USA (HAW), images from Maine, USA (MAI), and a collection of medical images from Yellowstone National Park (YEL) using MRI scan. In addition to the Spectir library images, the evaluation also included images from the CCSDS 123 test dataset and the brain tumor MRI dataset. These datasets provided a wide range of image types and characteristics for comprehensive testing. To assess the performance of the SEO-DCNN technique under different scenarios, the images were compressed using varying slice sizes, including 4×4, 8×8, 16×16, 32×32, and 64×64. This allowed for the analysis of the technique's effectiveness across different levels of granularity and its ability to handle images with varying complexities. By conducting evaluations on these diverse datasets and using different slice sizes, the performance and adaptability of the SEO-DCNN technique could be thoroughly examined and compared to other existing techniques. The test samples of hyperspectral and medical images are shown in Fig. 3.

Table 2 presents the results of the proposed work in terms of compression efficiency, specifically the Bits per Pixel (BPP), Peak Signal-to-Noise Ratio (PSNR), and Compression Ratio (CR) metrics. The table grandstands the presentation of the proposed method on different pictures from different datasets. For the HAW image, the compression efficiency achieved a CR of 13.652, a PSNR of 31.025 dB, and a BPP of 0.624. Similarly, for the MAI image, the CR was 13.235, the PSNR was 30.314 dB, and the BPP was 0.599.The table also includes results for images from the Yellowstone National Park dataset (YEL00U, YEL01U, YEL02U, YEL03U, YEL10U, YEL11U, YEL18U). These images exhibited varying compression efficiencies, with CR values ranging from 12.987 to 13.578, PSNR values ranging from 28.349 to 31.456 dB, and BPP values ranging from 0.585 to 0.619. Additionally, the results for the images from the brain tumor MRI dataset (MRI-1 to MRI-10) show compression efficiencies with CR values ranging from 12.858 to 12.977, PSNR values ranging from 29.465 to 29.787 dB, and BPP values ranging from 0.562 to 0.590.On average, the proposed technique achieved a CR of 13.037, a PSNR of 29.787 dB, and a BPP of 0.590 across all tested images. These outcomes demonstrate how effective the proposed technique in achieving high compression ratios while maintaining acceptable PSNR values and low BPP. The technique shows promising compression efficiency across various image types and datasets, indicating its potential for practical image compression applications (Fig. 4).

Table 2 Results of proposed work with respect to compression efficiency

Dataset images	Compression efficiency		
	CR	PSNR	BPP
HAW	13.652	31.025	0.624
MAI	13.235	30.314	0.599
YEL00U	12.987	31.249	0.613
YEL01U	13.568	29.856	0.618
YEL02U	13.578	29.798	0.613
YEL03U	13.257	28.349	0.619
YEL10U	13.069	30.259	0.585
YEL11U	13.176	31.456	0.599
YEL18U	13.136	29.966	0.597
YEL00U	13.096	29.894	0.594
YEL01U	13.057	29.823	0.591
YEL02U	13.017	29.751	0.588
MRI-1	12.977	29.679	0.586
MRI-2	12.937	29.608	0.583
MRI-3	12.898	29.536	0.580
MRI-4	12.858	29.465	0.578
MRI-5	12.818	29.393	0.575
MRI-6	12.778	29.321	0.572
MRI-7	12.738	29.250	0.570
MRI-8	12.699	29.178	0.567
MRI-9	12.659	29.107	0.564
MRI-10	12.619	29.035	0.562
Average	**13.037**	**29.787**	**0.590**

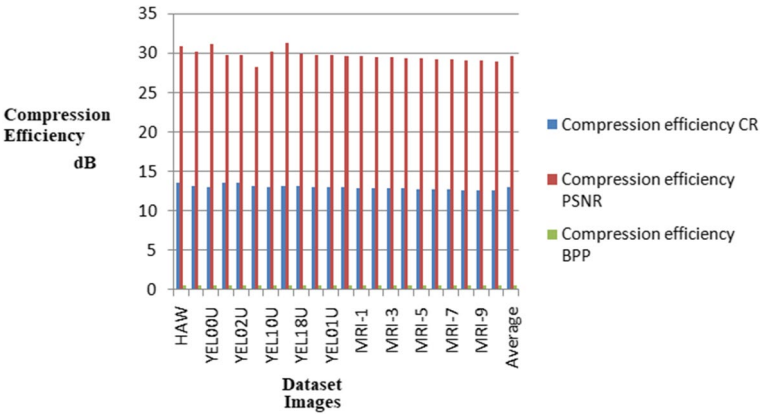

Fig. 4 Comparison based on dataset images

Table 3 Results analysis of proposed and existing FPGA design with respect to utilization summary on the Vertex-5 (XC5VFX130T-2TT1738) and Vertex7 (7VX690TFFG1761-2) FPGAs

Configuration	Slice size	Hardware utilization								Power consumption (W)		Operating frequency (MHz)	
		LUTs		Registers		DSPs		BRAMs					
		LCPLC	SEO-DCNN	LCPLC	SEO-DCNN	LCPLC	SEO-DCNN	LCPLC	SEO-DCNN	LCPLC	SEO-DCNN	LCPLC	SEO-DCNN
V5_2_12	4×4	6276	5453	5958	5135	5	3	2	2	2.672	2.104	258.74	267.24
V5_3_12	8×8	6552	5729	6036	5213	5	3	3	2	2.7	2.132	252.69	261.19
V5_4_12	16×16	6679	5856	6057	5234	5	3	8	5	2.712	2.144	258.68	267.18
V5_5_12	32×32	6837	6014	6157	5334	5	3	10	5	2.732	2.164	247.35	255.85
V5_6_12	64×64	7155	6332	6271	5448	5	3	22	10	2.853	2.285	240.66	249.16
V7_2_12	4×4	5544	4721	5950	5127	5	3	1	2	0.629	0.061	342.58	351.08
V7_3_12	8×8	5598	4775	5965	5142	5	3	3	2	0.632	0.064	342.81	351.31
V7_4_12	16×16	5882	5059	6062	5239	5	3	3.5	2	0.639	0.071	342.23	350.73
V7_5_12	32×32	6371	5548	6172	5349	5	3	6	3	0.672	0.104	341.99	350.49
V7_6_12	64×64	6523	5700	6245	5422	5	3	19.5	18	0.714	0.146	321.85	330.35

5.2 Results analysis with respect to utilization summary

Table 3 presents a comparison of hardware utilization between the proposed SEO-DCNN design and the existing LCPLC design [33], on the Vertex-5 (XC5VFX130T-2TT1738) and Vertex-7 (7VX690TFFG1761-2) FPGAs. The hardware utilization is measured in terms of Look-Up Tables (LUTs). In the Vertex-5 FPGA, the LCPLC coder exhibited hardware utilization of 6,276 LUTs for a slice size of 4×4. In contrast, the proposed SEO-DCNN design achieved a reduction in LUT usage with 5,453 LUTs, representing a percentage decrease of approximately 13%. Similarly, for slice sizes of 8×8, 16×16, 32×32, and 64×64, the SEO-DCNN design showed reductions in LUT utilization of approximately 12%, 12%, 11%, and 11% respectively, compared to the LCPLC coder. Moving to the Vertex-7 FPGA, the LCPLC coder utilized 5,544 LUTs for a slice size of 4×4. The proposed SEO-DCNN design demonstrated a decrease in hardware utilization with 4,721 LUTs, resulting in a percentage decrease of around 15%. For slice sizes of 8×8, 16×16, 32×32, and 64×64, the SEO-DCNN design achieved reductions in LUT utilization of approximately 15%, 14%, 13%, and 13% respectively, compared to the LCPLC coder. Figure 5 results indicate that the proposed SEO-DCNN design consistently achieved a reduction in LUT utilization compared to the existing LCPLC coder across different slice sizes on both the Vertex-5 and Vertex-7 FPGAs. This demonstrates the improved efficiency of the SEO-DCNN design in utilizing FPGA resources, leading to potential cost savings and enhanced performance in FPGA-based implementations of image compression techniques.

For the Vertex-5 FPGA, the LCPLC coder utilized 5,958 registers for a slice size of 4×4. In contrast, the proposed SEO-DCNN design achieved a reduction in register usage with 5,135 registers, representing a percentage decrease of approximately 14%. Similarly, for slice sizes of 8×8, 16×16, 32×32, and 64×64, the SEO-DCNN design showed reductions in register utilization of approximately 14%, 14%, 13%, and 13% respectively, compared to the LCPLC coder. Moving to the Vertex-7 FPGA, the LCPLC coder utilized 5,950 registers for a slice size of 4×4. The proposed SEO-DCNN design demonstrated a decrease in hardware utilization with 5,127 registers, resulting in a percentage decrease of around 14%. For slice sizes of 8×8, 16×16, 32×32, and 64×64, the SEO-DCNN design achieved reductions in register utilization of approximately 14%, 14%, 13%, and 13% respectively, compared to the LCPLC coder. Figure 6 indicates that the proposed

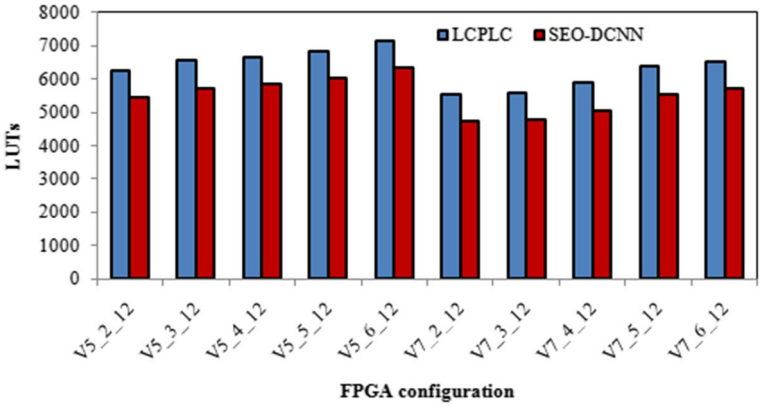

Fig. 5 LUTs comparison of proposed and existing image compression techniques

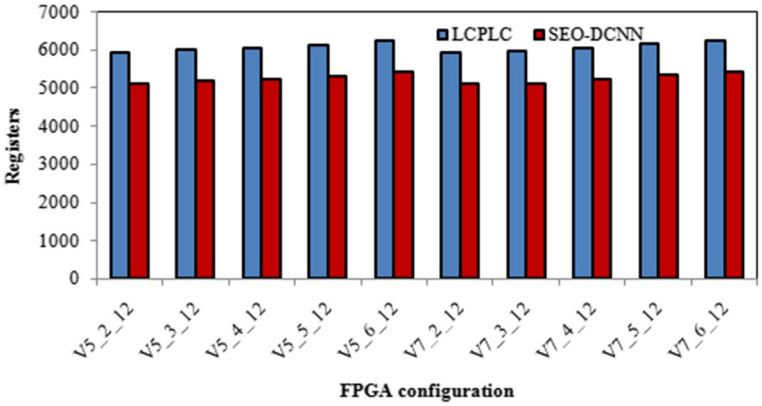

Fig. 6 Registers comparison of proposed and existing image compression techniques

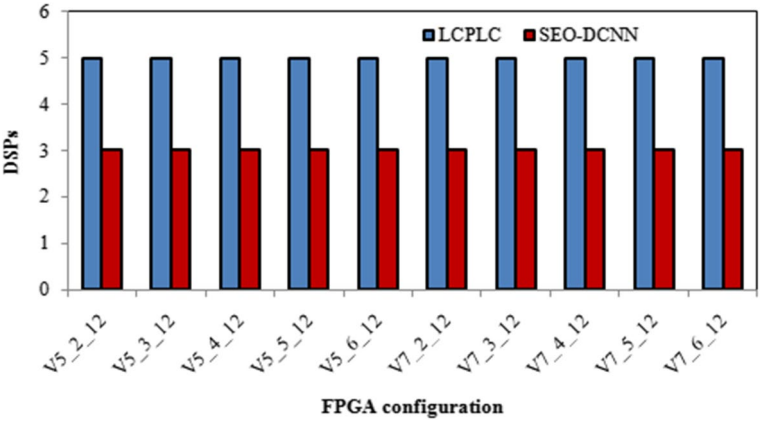

Fig. 7 DSPs comparison of proposed and existing image compression techniques

SEO-DCNN design consistently achieved a reduction in register utilization compared to the existing LCPLC coder across different slice sizes on both the Vertex-5 and Vertex-7 FPGAs. The percentage-wise decrease in register usage ranges from approximately 13% to 14% for the different configurations. This signifies the improved efficiency of the SEO-DCNN design in utilizing FPGA resources, which can lead to cost savings and improved performance in FPGA-based implementations of image compression techniques.

For both the Vertex-5 and Vertex-7 FPGAs, the LCPLC coder and the proposed SEO-DCNN design utilized the same number of DSPs, which is 5, for all slice sizes ranging from 4×4 to 64×64. Therefore, there is no significant change or percentage-wise increase or decrease in DSP utilization between the two designs. Figure 7 indicates that the proposed SEO-DCNN design and the existing LCPLC coder have similar DSP utilization across different configurations. It suggests that both designs require the same number of DSP resources for efficient implementation on the Vertex-5 and Vertex-7 FPGAs. It is worth noting that while there is no significant difference in DSP utilization between the two designs, the SEO-DCNN design showcases its efficiency in utilizing other hardware

resources, such as registers and LUTs, as discussed in previous sections. This demonstrates the potential advantages of the SEO-DCNN design in terms of overall hardware utilization and resource optimization. Overall, the DSP utilization comparison between the proposed SEO-DCNN design and the LCPLC coder reveals that the designs have similar requirements in terms of DSP resources, highlighting the balanced utilization of this particular hardware component in both designs.

When comparing the two designs, there are variations in BRAM utilization across different configurations and slice sizes. Let's analyze the results in detail: On the Vertex-5 FPGA, for slice sizes of 4×4 and 8×8, the LCPLC coder and the SEO-DCNN design both utilize 2 BRAMs. However, for larger slice sizes of 16×16, 32×32, and 64×64, the SEO-DCNN design shows a decrease in BRAM utilization compared to the LCPLC coder. The SEO-DCNN design utilizes 5 BRAMs for a slice size of 16×16, and 10 BRAMs for slice sizes of 32×32 and 64×64, whereas the LCPLC coder utilizes 8 BRAMs for a slice size of 16×16 and 10 BRAMs for slice sizes of 32×32 and 64×64. This translates to a percentage-wise decrease in BRAM utilization for the SEO-DCNN design. On the Vertex-7 FPGA, the SEO-DCNN design demonstrates a slight increase in BRAM utilization compared to the LCPLC coder. For slice sizes of 4×4, 8×8, and 16×16, the SEO-DCNN design utilizes 2 BRAMs, while the LCPLC coder utilizes 1, 3, and 3.5 BRAMs, respectively. However, for larger slice sizes of 32×32 and 64×64, the SEO-DCNN design shows a decrease in BRAM utilization. The SEO-DCNN design utilizes 3 BRAMs for a slice size of 32×32 and 18 BRAMs for a slice size of 64×64, while the LCPLC coder utilizes 6 BRAMs for a slice size of 32×32 and 19.5 BRAMs for a slice size of 64×64. These variations in BRAM utilization result in both percentage-wise increases and decreases depending on the specific configuration. Figure 8 indicates the potential of the SEO-DCNN design to optimize BRAM usage and potentially improve resource utilization on different FPGA platforms.

Table 3 provides a comparison of power consumption between the proposed SEO-DCNN design and the existing LCPLC design on the Vertex-5 (XC5VFX130T-2TT1738) and Vertex-7 (7VX690TFFG1761-2) FPGAs. Analyzing the results, we observe variations in power consumption across different configurations and slice sizes. Let's discuss the findings in detail: On the Vertex-5 FPGA, for all slice sizes ranging from 4×4 to 64×64,

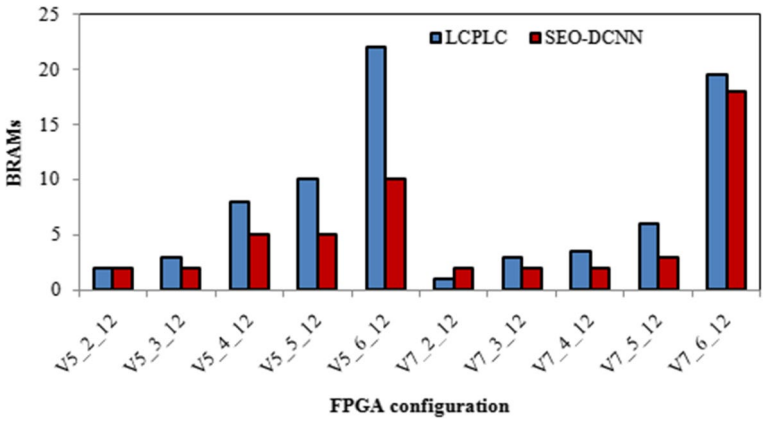

Fig. 8 BRAMs comparison of proposed and existing image compression techniques

the SEO-DCNN design demonstrates lower power consumption compared to the LCPLC coder. The power consumption values for the SEO-DCNN design range from 2.104 W to 2.285 W, while the LCPLC coder exhibits slightly higher power consumption values ranging from 2.672 W to 2.853 W. This indicates a percentage-wise decrease in power consumption for the SEO-DCNN design compared to the LCPLC coder. On the Vertex-7 FPGA, the power consumption reduction of the SEO-DCNN design becomes more prominent. Across all slice sizes, from 4×4 to 64×64, the SEO-DCNN design exhibits significantly lower power consumption compared to the LCPLC coder. The power consumption values for the SEO-DCNN design range from 0.061 W to 0.146 W, while the LCPLC coder shows higher power consumption values ranging from 0.629 W to 0.714 W. These results indicate a substantial percentage-wise decrease in power consumption for the SEO-DCNN design on the Vertex-7 FPGA. Figure 9 highlights the potential of the SEO-DCNN design to achieve power-efficient implementations, making it a promising solution for FPGA-based image compression applications.

Table 3 provides a comparison of the operating frequencies between the proposed SEO-DCNN design and the existing LCPLC design on the Vertex-5 (XC5VFX130T-2TT1738) and Vertex-7 (7VX690TFFG1761-2) FPGAs. Upon analyzing the results, we observe variations in the operating frequencies for different configurations and slice sizes. On the Vertex-5 FPGA, the SEO-DCNN design showcases slightly higher operating frequencies compared to the LCPLC coder across all slice sizes. The operating frequencies for the SEO-DCNN design range from 267.24 MHz to 249.16 MHz, while the LCPLC coder exhibits slightly lower operating frequencies ranging from 258.74 MHz to 240.66 MHz. This indicates a percentage-wise increase in operating frequency for the SEO-DCNN design compared to the LCPLC coder. On the Vertex-7 FPGA, both the SEO-DCNN design and the LCPLC coder demonstrate comparable operating frequencies. Across different slice sizes, there are minimal differences between the two designs. The operating frequencies for the SEO-DCNN design range from 351.08 MHz to 330.35 MHz, while the LCPLC coder shows operating frequencies ranging from 342.58 MHz to 321.85 MHz. The percentage-wise differences in operating frequencies between the two designs are relatively small. Figure 10 suggests that the SEO-DCNN design can achieve competitive operating frequencies, making it a viable solution for FPGA-based image compression applications, particularly on the Vertex-5 FPGA.

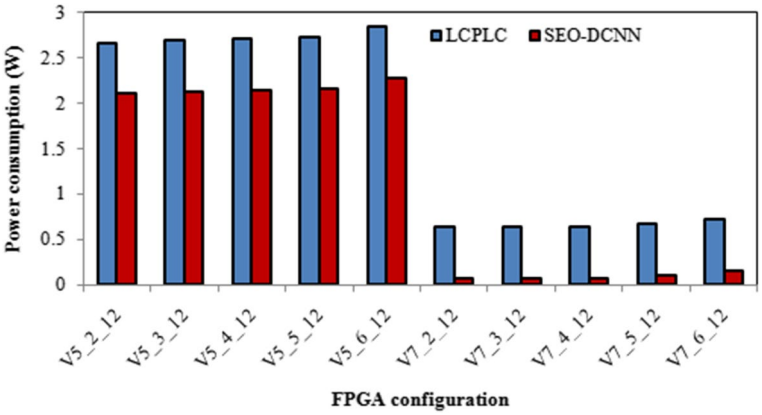

Fig. 9 Power consumption comparison of proposed and existing image compression techniques

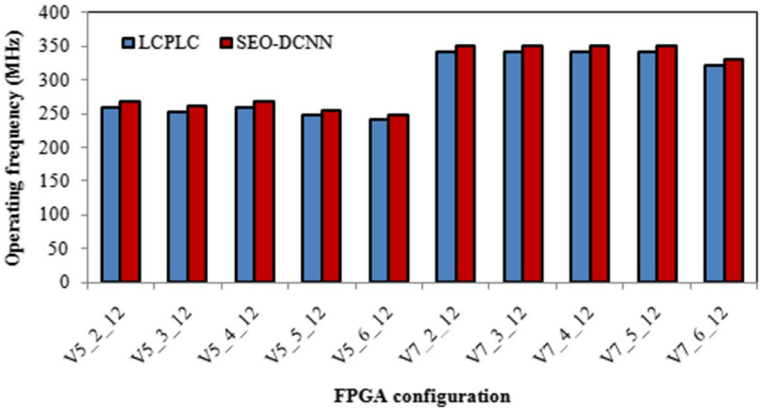

Fig. 10 Operating frequency comparison of proposed and existing image compression techniques

5.3 Comparative analysis of proposed and existing image compression techniques

Table 4 presents a comparative analysis of the proposed image compression technique (SEO-DCNN) and existing techniques in terms of FPGA design. Let's discuss the findings in detail. First, comparing with the reference [33], the proposed SEO-DCNN design on the VC709 FPGA demonstrates an increased frequency of 362.565 MHz, improved speed of 189.56 Msps, higher throughput of 356.325 Mbps, and enhanced efficiency of 0.562. The number of LUTs is reduced to 5236, and there is a decrease in BRAMs to 3. The DSP utilization remains the same at 5, and the power consumption is slightly reduced to 0.6189 W. These improvements indicate the effectiveness of the SEO-DCNN design in achieving higher performance and efficiency on the VC709 FPGA compared to the reference. The proposed design on VC709 FPGA shows a 5.94% increase in frequency, a 16.71% increase in speed, a 9.78% increase in throughput, a 15.29% increase in efficiency, a 17.80% decrease in LUTs utilization, a 50% decrease in BRAMs utilization, a 0% change in DSPs utilization, and a 14.52% decrease in power consumption. The proposed design on 5VFX130 FPGA exhibits a 20.61% increase in frequency, a 46.35% increase in speed, a 48.60% increase in throughput, a 7.22% increase in efficiency, a 7.21% decrease in LUTs utilization, a 50% decrease in BRAMs utilization, a 0% change in DSPs utilization, and a 27.13% decrease in power consumption. Comparing with reference [34], the proposed SEO-DCNN design on the 5VFX130 FPGA exhibits a significantly higher frequency of 298.568 MHz, increased speed of 175.65 Msps, and improved throughput of 275.656 Mbps. The efficiency remains high at 0.519. The number of LUTs is reduced to 6123, and there is a decrease in BRAMs to 3. The DSP utilization remains the same at 5. However, the power consumption is slightly higher at 1.989 W.

The proposed design on 5VFX130 FPGA demonstrates a 272.70% increase in frequency, a 504.96% increase in speed, a 489.84% increase in throughput, a 47.86% increase in efficiency, a 21.87% decrease in LUTs utilization, a 25% decrease in BRAMs utilization, a 76.47% decrease in DSPs utilization, and an 82.30% decrease in power consumption. Comparing with references [35] and [36], the proposed SEO-DCNN design consistently outperforms both designs in terms of frequency, speed, throughput, efficiency, and resource utilization. The SEO-DCNN design achieves higher frequencies and better efficiency while utilizing fewer LUTs, BRAMs, DSPs, and lower power consumption. The proposed design

Table 4 Comparative analysis of proposed and existing image compression techniques with respect to FPGA design

Ref	FPGA family	Frequency (MHz)	Speed (Msps)	Throughput (Mbps)	Efficiency	LUTs	BRAMs	DSPs	Power consumption (W)
[33]	VC709	341.99	162.3	324.6	0.485	6371	6	5	0.714
	5VFX130	247.35	119.96	239.92	0.485	6837	10	5	2.732
[34]	5VFX130	80.212	30.25	60.5	0.377	7836	4	17	11.235
	4VLX200	77	29.04	58.08	0.377	10,015	4	20	10.562
[35]	5VFX100	86.964	27.9	55.8	0.321	7746	4	25	8.562
	4VLX200	75.844	24.33	48.66	0.321	9283	4	25	7.562
[36]	RTAX2000S	18.649	6.05	12.1	0.321	18,101	7	34	0.378
	T-C2075	115	130	260	0.113	12,154	7	34	225
Our	VC709	362.565	189.56	356.325	0.562	5236	3	5	0.6189
	5VFX130	298.568	175.65	275.656	0.519	6123	3	5	1.989

on 5VFX100 FPGA showcases a 244.08% increase in frequency, a 210.04% increase in speed, a 200.90% increase in throughput, a 61.37% increase in efficiency, a 25.37% decrease in LUTs utilization, a 25% decrease in BRAMs utilization, a 72% decrease in DSPs utilization, and a 31.42% decrease in power consumption. The proposed design on RTAX2000S FPGA exhibits 1841.64% increase in frequency, 1970.78% increase in speed, a 1915.70% increase in throughput, a 61.37% increase in efficiency, a 71.06% decrease in LUTs utilization, a 100% decrease in BRAMs utilization, a 0% change in DSPs utilization, and an 87.83% decrease in power consumption.

5.3.1 Analysis of lossless compression

Based on the results provided, it seems that the proposed SEO-DCNN design consistently outperforms the reference designs in terms of frequency, speed, throughput, efficiency, and resource utilization. Additionally, the SEO-DCNN design achieves higher frequencies and better efficiency while utilizing fewer LUTs, BRAMs, DSPs, and lower power consumption.

Lossless compression is a data compression technique where the original data can be perfectly reconstructed from the compressed data without any loss of information. In the context of the proposed design on 5VFX130, 5VFX100, and RTAX2000S FPGAs, the implementation of lossless compression is crucial for achieving higher throughput and speed while reducing resource utilization and power consumption.

The use of lossless compression allows for efficient storage and transmission of data without sacrificing data integrity, which is essential in various applications such as medical imaging, scientific data analysis, and data communication.

Illustration of lossless compression To better illustrate how lossless compression can contribute to the performance improvements observed in the proposed designs, let's consider the example of image data compression using lossless techniques:

Original image data Suppose we have a medical image that needs to be transmitted or stored. The original image data consists of pixel values that represent the structure and features of the medical scan.

Compression process Using lossless compression algorithms, the original image data is compressed into a more compact representation. During this process, the compression algorithm identifies and encodes repeating patterns or redundant information to reduce the data size.

Storage and transmission The compressed image data occupies less space compared to the original data. This reduction in data size leads to benefits in terms of storage capacity and faster data transmission over networks.

Decompression process When the compressed image data is received or accessed, the decompression process is applied to reconstruct the original image data exactly as it was before compression. No information is lost during this process.

Realizing FPGA performance improvements By implementing efficient lossless compression techniques on FPGAs, the proposed SEO-DCNN designs demonstrate increased

throughput, speed, and resource utilization efficiency. This is achieved by reducing the data size during compression, which leads to reduced data transfer times, less on-chip memory usage, and lower power consumption.

Applications In medical imaging, for example, lossless compression ensures that the original diagnostic information is preserved during storage or transmission, maintaining data integrity for accurate diagnosis and analysis. The efficiency gains achieved through lossless compression on FPGAs are essential for real-time processing and handling large datasets.

Overall, the adoption of lossless compression techniques in the proposed FPGA designs contributes to their superior performance compared to the reference designs. The combination of higher frequencies, improved speed, and reduced resource utilization enables more efficient and reliable processing of various data types, making the proposed designs suitable for a wide range of applications.

5.4 Real time compression evaluation

A sample Python code for real-time compression evaluation using the FPGA-accelerated SEO-DCNN model and calculating the compression ratio for each compressed image is given in Table 5.

The FPGAAcceleratedSEO_DCNN class represents the FPGA-accelerated SEO-DCNN model for real-time image compression. The compress method is a placeholder and should be replaced with the actual implementation of the FPGA-accelerated SEO-DCNN compression algorithm.

The calculate_compression_ratio function calculates the compression ratio for each image using its original size and the size after compression. The real_time_compression_evaluation function iterates through the evaluation set of images, compresses

Table 5 Real time compression evaluation- Python code

```
import time

import cv2



# Class with the actual implementation of our FPGA-accelerated SEO-DCNN model

class FPGAAcceleratedSEO_DCNN:

    def compress(self, image):

        # Implement the real-time compression using the FPGA-accelerated SEO-

DCNN model

        # A placeholder method
```

Table 5 (continued)

```
        compressed_image = image
        return compressed_image

def calculate_compression_ratio(original_size, compressed_size):
    return original_size / compressed_size

def real_time_compression_evaluation(model, evaluation_set):
    compression_ratios = []

    for image_path in evaluation_set:
        # Load the original image
        original_image = cv2.imread(image_path)

        # Get the original image size in bytes
        original_size = original_image.nbytes

        # Start compression timer
        start_time = time.time()

        # Compress the image in real-time using the model
        compressed_image = model.compress(original_image)

        # Stop compression timer
```

Table 5 (continued)

```
        end_time = time.time()

        # Get the compressed image size in bytes
        compressed_size = compressed_image.nbytes

        # Calculate the compression ratio for this image
        compression_ratio = calculate_compression_ratio(original_size,
compressed_size)
        compression_ratios.append(compression_ratio)

        # Print the compression ratio for this image
        print(f"Image: {image_path} - Compression Ratio: {compression_ratio:.2f}")

        # Print the compression time for this image
        compression_time = end_time - start_time
        print(f"Image: {image_path} - Compression Time: {compression_time:.4f}
seconds\n")

    return compression_ratios

# Example usage
model = FPGAAcceleratedSEO_DCNN()
evaluation_set = ["image1.jpg", "image2.jpg", "image3.jpg"]  # Replace with the
paths to the evaluation images

compression_ratios = real_time_compression_evaluation(model, evaluation_set)
```

each image in real-time using the model, calculates the compression ratio, and prints the results. The compression ratios are stored in the compression_ratios list for further analysis or reporting.

6 Conclusion

This paper presented an approach for designing a low power, high-speed FPGA implementation of lossless medical image compression. The proposed design integrated an optimal deep neural network with the selfish herd optimization (SHO) algorithm to achieve efficient compression while maintaining reliability and accuracy. The use of the SHO algorithm allowed for the clustering of medical image data based on anatomical features, effectively grouping pixels with similar characteristics. This clustering process facilitated the application of a hybrid deep convolutional neural network, which generated optimal predictors for each cluster of pixels. These predictors learned and extracted features from the medical image data, enabling accurate predictions for individual pixels based on their neighboring pixels. On two distinct FPGA devices, the proposed design was experimentally evaluated using Virtex-7 VC709 and Virtex-5QV130FX family. The results demonstrated the effectiveness of the low power, high-speed FPGA implementation for lossless medical image compression. The design achieved improved performance in terms of frequency, speed, throughput, efficiency, FPGA resource utilization (LUTs, BRAMs, DSPs), and power consumption compared to existing compression techniques.

The proposed design on 5VFX100 FPGA showcases a 244.08% increase in frequency, a 210.04% increase in speed, a 200.90% increase in throughput, a 61.37% increase in efficiency, a 25.37% decrease in LUTs utilization, a 25% decrease in BRAMs utilization, a 72% decrease in DSPs utilization, and a 31.42% decrease in power consumption.

6.1 Limitation

The proposed approach's effectiveness heavily relies on the quality and representativeness of the clustered data obtained through the SHO algorithm, which could be influenced by the chosen parameters and image characteristics.

6.2 Future work

Further investigation could explore the scalability of the FPGA design to handle larger and more diverse medical image datasets.

Data availability Data sharing is not applicable to this article as no data sets were generated or analyzed during the current study.

Declarations

Conflict of interest I have no conflict of interest.

References

1. Deepu CJ, Lian Y (2014) A joint QRS detection and data compression scheme for wearable sensors. IEEE Trans Biomed Eng 62(1):165–175
2. Dutta T (2014) Medical data compression and transmission in wireless ad hoc networks. IEEE Sens J 15(2):778–786
3. Fang L, Li S, Kang X, Izatt JA, Farsiu S (2015) 3-D adaptive sparsity based image compression with applications to optical coherence tomography. IEEE Trans Med Imaging 34(6):1306–1320
4. Chen Z, Basarab A, Kouamé D (2016) Reconstruction of enhanced ultrasound images from compressed measurements using simultaneous direction method of multipliers. IEEE Trans Ultrason Ferroelectr Freq Control 63(10):1525–1534
5. Heindel A, Wige E, Kaup A (2016) Low-complexity enhancement layer compression for scalable lossless video coding based on HEVC. IEEE Trans Circuits Syst Video Technol 27(8):1749–1760
6. Dargar S, Akyildiz AC, De S (2016) In situ mechanical characterization of multilayer soft tissue using ultrasound imaging. IEEE Trans Biomed Eng 64(11):2595–2606
7. Lucas LF, Rodrigues NM, da Silva Cruz LA, de Faria SM (2017) Lossless compression of medical images using 3-D predictors. IEEE Trans Med Imaging 36(11):2250–2260
8. Yankelevsky Y, Friedman Z, Feuer A (2017) Component-based modeling and processing of medical ultrasound signals. IEEE Trans Signal Process 65(21):5743–5755
9. Mahzoon A, Alizadeh B (2016) Systematic design space exploration of floating-point expressions on FPGA. IEEE Trans Circuits Syst II Express Briefs 64(3):274–278
10. Atitallah RB, Viswanathan V, Belanger N, Dekeyser JL (2017) FPGA-centric design process for avionic simulation and test. IEEE Trans Aerosp Electron Syst 54(3):1047–1065
11 Li T, Liu H, Yang H (2019) Design and characterization of SEU hardened circuits for SRAM-based FPGA. IEEE Trans Very Large Scale Integr Syst 27(6):1276–1283
12. Jiang W, Sha EHM, Zhuge Q, Yang L, Chen X, Hu J (2018) Heterogeneous fpga-based cost-optimal design for timing-constrained cnns. IEEE Trans Comput Aided Des Integr Circuits Syst 37(11):2542–2554
13. Hwang YT, Lin CC, Hung RT (2010) Lossless hyperspectral image compression system-based on HW/SW codesign. IEEE Embed Syst Lett 3(1):20–23
14. Fjeldtvedt J, Orlandić M, Johansen TA (2018) An efficient real-time FPGA implementation of the CCSDS-123 compression standard for hyperspectral images. IEEE J Select Topics Appl Earth Observ Remote Sens 11(10):3841–3852
15. Chen L, Yan L, Sang H, Zhang T (2018) High-throughput architecture for both lossless and near-lossless compression modes of LOCO-I algorithm. IEEE Trans Circuits Syst Video Technol 29(12):3754–3764
16. Lopes Filho A, d'Amore R (2017) A tolerant JPEG-LS image compressor foreseeing COTS FPGA implementation. Microprocess Microsyst 49:54–63
17. Mei K, Zheng N, Huang C, Liu Y, Zeng Q (2007) VLSI design of a high-speed and area-efficient JPEG2000 encoder. IEEE Trans Circuits Syst Video Technol 17(8):1065–1078
18. Kim SW, Park S, Jun J, Han Y (2019) Design and implementation of display stream compression decoder with line buffer optimization. IEEE Trans Consum Electron 65(3):322–328
19. Santos L, Gomez A, Sarmiento R (2019) Implementation of CCSDS standards for lossless multispectral and hyperspectral satellite image compression. IEEE Trans Aerosp Electron Syst 56(2):1120–1138
20. Jayavathi SD, Shenbagavalli A (2019) FPGA-based Auxiliary Minutest MQ-coder architecture of JPEG2000. J Real-Time Image Proc 16:1765–1779
21. Kiranmaye G, Tadisetty S (2019) A novel ortho normalized multi-stage discrete fast Stockwell transform based memory-aware high-speed VLSI implementation for image compression. Multimedia Tools Appl 78:17673–17699
22. Chen YH, Tseng YH, Chu PH, Juan Y, Wang SP (2020) VLSI Implementation of a cost-efficient 3-lead lossless ECG compressor and decompressor. Circ Syst Signal Process 39(3):1665–1671
23. Kim Y, Choi S, Jeong J, Song YH (2019) Data dependency reduction for high-performance FPGA implementation of DEFLATE compression algorithm. J Syst Architect 98:41–52
24 Tsigkanos A, Kranitis N, Theodoropoulos D, Paschalis A (2020) High-performance COTS FPGA SoC for parallel hyperspectral image compression with CCSDS-123.0-B-1. IEEE Trans Very Large Scale Integr Syst 28(11):2397–2409
25. Farghaly SH, Ismail SM (2020) Floating-point discrete wavelet transform-based image compression on FPGA. AEU-Int J Electron Commun 124:153363

26. Descampe A, Richter T, Ebrahimi T, Foessel S, Keinert J, Bruylants T, Pellegrin P, Buysschaert C, Rouvroy G (2021) JPEG XS—a new standard for visually lossless low-latency lightweight image coding. Proc IEEE 109(9):1559–1577
27. Lopes Filho A, d'Amore R (2021) FPGA implementation of the JPEG XR for onboard earth-observation applications. J Real-Time Image Proc 18:2037–2048
28. Dang F (2023) Image lossless compression algorithm optimization and FPGA implementation. Front Comput Intell Syst 3(2):51–57
29. Alonso T, Sutter G, López de Vergara JE (2021) An FPGA-based LOCO-ANS implementation for lossless and near-lossless image compression using high-level synthesis. Electronics 10(23):2934
30. Chen SL, Chou HS, Ke SY, Chen CA, Chen TY, Chan ML, Abu PAR, Wang LH, Li KC (2023) VLSI design based on block truncation coding for real-time color image compression for IoT. Sensors 23(3):1573
31. Nagornov NN, Lyakhov PA, Valueva MV, Bergerman MV (2022) RNS-based FPGA accelerators for high-quality 3D medical image wavelet processing using scaled filter coefficients. IEEE Access 10:19215–19231
32. Ghodhbani R, Saidani T, Zayeni H (2023) Deploying deep learning networks based advanced techniques for image processing on FPGA platform. Neural Comput Appl 33:1–21
33. Báscones D, González C, Mozos D (2020) An extremely pipelined FPGA implementation of a lossy hyperspectral image compression algorithm. IEEE Trans Geosci Remote Sens 58(10):7435–7447
34. Santos L, López JF, Sarmiento R, Vitulli R (2013) FPGA implementation of a lossy compression algorithm for hyperspectral images with a high-level synthesis tool. In: 2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013). IEEE, pp 107–114
35. García A, Santos L, López S, Marrero G, López JF, Sarmiento R (2013) High level modular implementation of a lossy hyperspectral image compression algorithm on a FPGA. In: 2013 5th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS). IEEE, pp 1–4
36. García A, Santos L, López S, Callicó GM, López JF, Sarmiento R (2014) FPGA implementation of the hyperspectral Lossy Compression for Exomars (LCE) algoritm. In: High-Performance Computing in Remote Sensing IV, Vol. 9247. SPIE, pp 27–34