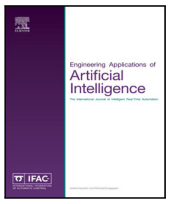
Engineering Applications of Artificial Intelligence 139 (2025) 109540

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Compact representation for memory-efficient storage of images using genetic algorithm-guided key pixel selection

Samir Malakar *, Nirwan Banerjee, Dilip K. Prasad

Department of Computer Science, UiT The Arctic University of Norway, Tromsø, Norway

ARTICLE INFO

Keywords:
Compact representation
Genetic algorithm
Memory efficient
Gaussian kernel
Storage reduction

ABSTRACT

In the past few years, we have observed rapid growth in digital content. Even in the biological domain, the arrival of microscopic and nanoscopic images and videos captured for biological investigations increases the need for space to store them. Hence, storing these data in a storage-efficient manner is a pressing need. In this work, we have introduced a compact image representation technique with an eye on preserving the shape that can shrink the memory requirement to store. The compact image representation is different from image compression since it does not include any encoding mechanism. Rather, the idea is that this mechanism stores the positions of key pixels, and when required, the original image can be regenerated. The genetic algorithm is used to select key pixels, while the Gaussian kernel performs the reconstruction task with the help of the positions of the selected key pixels. The model is tested on four different datasets. The proposed technique shrinks the memory requirement by 87% to 98% while evaluated using the bit reduction rate. However, the reconstructed images' quality is a bit low when evaluated using metrics like structural similarity index (ranges between 0.81 to 0.94), or root means squared error (ranges between 0.06 to 0.08). To investigate the impact of quality reduction in reconstructed images in real-life applications, we performed image classification using reconstructed samples and found 0.13% to 2.30% classification accuracy reduction compared to when classification is done using original samples. The proposed model's performance is comparable to state-of-the-art's similar solutions.

1. Introduction

In the last few decades, the world has seen huge growth in digital content in the form of texts, images, videos, sensor data, and many more. The world today is digitized to an unbelievable extent. Many studies have shown that countries with higher gross domestic product (GDP) index are equipped with digitization power. The study[1] estimates, 70% of the globe's GDP underwent some form of digitization at the end of 2022. This study also estimated that by 2025, people will generate 463 exabytes of data each day. A huge surge in the amount of data shortly is predicted and inevitable. Among these digital contents, vision data is more information-dense than the other avenues of data. A video is essentially a stack of images in the fourth dimension. Recently, we come across a huge amount of image and video data generated and transmitted in our daily lives in some form. Social media is one such place that generates an ample amount of such data.

Apart from this, the proliferation of high-content microscopes, used in biological research, boosts this further. For example, a single high-content nanoscopy microscope image with a resolution of 2048×2048 pixels, acquired in 12-bit depth and with three color channels, can have a file size of around 36 megabytes. Lattice Light-Sheet Microscope (LLSM), to capture 3D images of live fruit fly embryos over several hours, generating image datasets of up to 26 terabytes in size. Howard Hughes Medical Institute's Janelia Research Campus is capable of generating terabyte-sized image datasets of live biological samples with subcellular resolution. The amount of data a nanoscopy spatio-temporal image generates is almost equivalent to the amount of data used by humans in 2022 (Torres-García et al., 2022). It is especially of note that high-quality microscopy contains very crucial information that is especially useful and can be the potential source of future breakthroughs in the domain of medical science. This obscene amount of data naturally leads to an important question "how do we store the huge amount of data generated?". For a few years now, not only the biological scientific community but also the vision research community are going to face an abrupt halt in figuring out this problem.

Even if we are able to store such gigantic data somehow, the issue related to sustainability is not solved. Wu et al. (2022) have mentioned that increased data storage means increased embodied carbon

* Corresponding author.
E-mail addresses: s.malakar@uit.no (S. Malakar), nirwan.banerjee@uit.no (N. Banerjee), dilip.prasad@uit.no (D.K. Prasad).

[1] https://techjury.net/blog/how-much-data-is-created-every-day/.

https://doi.org/10.1016/j.engappai.2024.109540
Received 28 February 2024; Received in revised form 25 September 2024; Accepted 21 October 2024
Available online 4 November 2024

footprint, which is one of the major concerns when thinking about sustainability. In the study[2], Addis measured cloud-based digital data preservation system emits 7800 $kgCO_2eq$, a unit to measure gross carbon emissions, to store 1 petabyte (PB) astronomy research related images used in ARCHIVER project[3] for 1 years. The measurement was made by deploying images in the Google Cloud platform. In another study, Monserrate (2022) mentioned that cloud-based data storage has a larger carbon footprint as compared to the airline industry. In the case of energy, Hu (2015) in his study established that energy needed to run a data center requires electricity consumed by nearly 50,000 homes. The data centers along with edge devices like laptops, smartphones, and tablets are responsible for 2% of global CO_2 emission (Burrington, 2015; Koomey and Masanet, 2021).

We can mitigate the discussed bottlenecks by storing data, specifically vision data that takes larger storage space to store, with lesser space. This vision can be achieved by representing images compactly. An image is a collection of pixels, each having an intensity value. The dimension of an image controls the number of pixels and more pixels means a larger storage requirement. So our idea is to identify a set of pixel positions from which we can retrieve the image by employing a 2D convolution with the help of a pre-defined kernel. Henceforth, we will call such pixels as "key pixels". So by the phrase "compact image representation" we mean representing an image with the key pixels only and storing such pixel positions, even in array form, might shrink the storage requirement compared to its actual storage requirement while storing all the pixels.

Let us define the problem with a better understanding. An image (say, $I = \{f(x,y) : f(x,y) \in \{0,1,\ldots,255\} \wedge (x,y) \in [1,H] \times [1,W]\}$) having dimension $H \times W$ (H, and W represent the height and width of I respectively) contains N number of pixels i.e., $N = H * W$. Now, if the number of key pixels (say, N') is much lower than N (mathematically, $N' \ll N$) with the property $N * b \leq N' * b'$, where b, b' represent the number of bits required to store intensity value of a single pixel in I and position of a pixel value respectively then we can shrink the storage requirement for I. It is noteworthy to mention that b is controlled by the bit depth of I while the dimension of I controls b' i.e., $b' = \lceil log_2(max\{H,W\})\rceil$. Unlike compression techniques, our idea of "compact image representation" does not include any coding scheme to lessen the use of bits to store pixel information (i.e., intensity value). Rather it searches for the key pixels that guide the convolution operator-inspired image regeneration process with minimal loss in shape and pixel intensity information, the two key features of an image.

The pixels on the skeleton, edges, or boundary of objects present in an image are good candidates to be used as key pixels as they possess a certain percentage of data pixels present in an image. In other words, methods like edge detection (Canny, 1986; El-Sayed and Hafeez, 2012; Pramanik et al., 2022; Dey et al., 2022c), skeletonization (Guo and Hall, 1992; Zhang and Suen, 1984; Lee et al., 1994; Ko et al., 2021), boundary detection (Martin et al., 2004; Marmanis et al., 2018), object detection (Ali et al., 2019b,a) followed by boundary detection can be thought of as a key pixels generation process. But generating the image from them causes sufficient loss in shape and pixels' intensity information of the image (see Fig. 1). In this figure, in the first column, we have shown data pixels' positions (termed as "Representer") generated using the Canny edge detector (Canny, 1986), objects' boundary pixels, objects' skeleton pixels generated using the method by Lee et al. (1994), random data pixels selected using Binomial distribution with probability 0.5 (i.e., if a data pixel is not selected then it is converted to background pixel), and key pixels selected using the present method as pixels for compact representation of an image. Next, the representers are convolved with the Gaussian kernel with varying kernel size during the reconstruction process (the reconstruction process is described in detail in Section 2.1) and the reconstructed images are shown in 2nd, 3rd and 4th columns. In the figure legends, R, P, F, M, S, $\#R$, and $\#O$ represent recall, precision, F1-score, root means square error (RMSE) score, structural similarity score (SSIM), number of data pixels in the reconstructed image, and number of data pixels in the original image, respectively. The outputs show that irrespective of the Gaussian kernel size used during the reconstruction process, the randomly selected data pixels as an image compact representer provide the best score in terms of recall, precision, F1-score, RMSE, and SSIM (see 4th row), while the skeleton possesses the least number of pixels in the representer (see $\#R$ value in the 1st column), followed by boundary and edge pixels. It is also clear from Fig. 2, where reconstructed images are shown on a part of the image for better visualization, that the reconstructed image from randomly selected key pixels is better than others. However, it selects almost 50% of the pixels, which is one of its major drawbacks.

The illustrations through Figs. 1 and 2 motivate us to suppress data pixels (i.e., randomly converting some foreground pixels to background pixels) intelligently to obtain a better set of key pixels that will be lesser in count and retain good similarity with the original image after reconstructing the image from the selected key pixels using the Gaussian kernel-based convolution operator. Therefore, we plan to suppress foreground pixels in a controlled way so that we can achieve very close shape and pixel intensity information of the original image after reconstruction from the selected key pixels. In short, the search technique designed here is to select a set of effective foreground pixels as key pixels. An image with n foreground pixels has 2^n-1 possible pixel subsets and one of them may be the best. Searching for the best solution is an NP-complete problem (Sarkar et al., 2019; Malakar et al., 2020b; Mukhopadhyay et al., 2023). This makes the selection process computationally expensive, which is against the objective of sustainability of AI models (Van Wynsberghe, 2021). Therefore, rather than searching for the best solution, we search for the near-optimal solution. Statistical methods or filter-based selection approaches like Chi-square score-based selection (Omar and Abd El-Hafeez, 2024), mutual information (Vergara and Estévez, 2014), and RelifF (Ghosh et al., 2017, 2019) could be used for the same. However, their performances are not at par with the other forms, like wrapper (Acampora et al., 2023; Das et al., 2022; Malakar et al., 2020b), and hybrid (Dey et al., 2022a; Ghosh et al., 2019) selection processes.. However, meta-heuristic-based optimization algorithms are a better choice for such an objective (Dey et al., 2022b,a).

The working procedure of meta-heuristics-based optimization algorithms can be of two types (Pan et al., 2022). One that was initially designed for solving optimization problems in the continuous domain and then transformed to their binary version to meet the optimization requirement in the binary domain. This discretization process employed several transfer functions, quantum approaches, thresholding techniques, etc. (Nadimi-Shahraki et al., 2021). Algorithms like particle swarm optimization (PSO) (Kennedy and Eberhart, 1995; Sarkar et al., 2019; Eman et al., 2023), gravitational search algorithm (GSA) (Rashedi et al., 2009, 2018), grey wolf optimization (GWO) (Mirjalili et al., 2014; Saabia et al., 2019; Eliwa et al., 2023), moth-flame optimization (Mirjalili, 2015; Xu et al., 2019), and Gazelle optimization algorithm (GOA) (Taha et al., 2023) fall under this category. The binary version of these algorithms has been successfully applied to many real-life applications that require binary optimization algorithms (Pan et al., 2022). But the appropriate choice for discretization is the pressing need for a better solution, which is another research problem. For example, Mirjalili and Lewis (2013) and Malakar et al. (2023) showed in their research how the transfer functions, used for transforming this category of optimization algorithms from continuous domain to discrete domain, affect the end result. On the contrary, optimization algorithms like the genetic algorithm (GA) (Holland, 1992; Acampora et al., 2023), ant colony optimization (ACO) (Dorigo et al., 2006; Karimi et al., 2023), memetic algorithm (Ghosh et al., 2017, 2019), and discrete fish migration optimization (Pan et al., 2021,

[2] https://www.dpconline.org/blog/blog-matthew-addis-carbon-footprint.
[3] https://archiver-project.eu/.

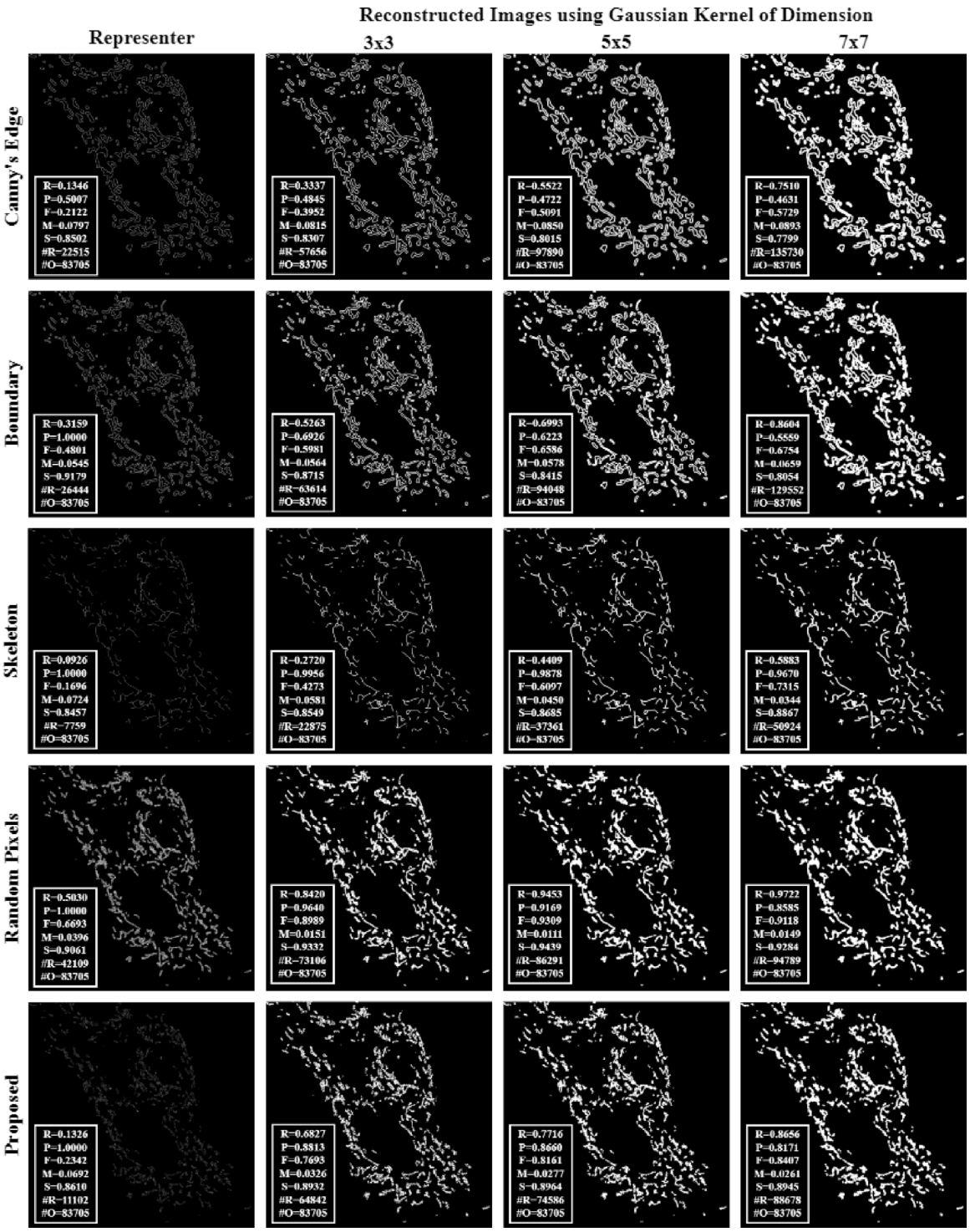


Fig. 1. Illustration of different compact image representation and their effect after reconstruction with the help of a fluorescent microscopy image showing mitochondria in the living cell. The image is taken from UiTMito dataset (Sekh et al., 2021). In the figure legends, R, P, F, M, S, #R, and #O represent recall, precision, F1-score, RMSE score, SSIM, number of data pixels in the reconstructed image, and number of data pixels in the original image respectively.

2022) were designed to work especially on binary domains. Thus, these algorithms are free from transforming the process from continuous space to discrete space, and we feel this category of algorithms is the most suitable for the present optimization problem.

Here, we would also like to mention that by the free-lunch theorem (Wolpert and Macready, 1997) no optimization algorithm is suitable for all applications that inherit the need for an optimization algorithm. Thus, without searching for the most suitable meta-heuristics algorithm for the current problem, we use the GA to decide on a near-optimal set of foreground pixels as key pixels. The computational time of meta-heuristics algorithms like GA, ACO, PSO, and GWO is directly proportional to the dimension of the search space (Mohiuddin et al., 2023). The time complexity of the GA algorithm is $\sim O(D * F * P_s * Max_{iter})$, where D, F, P_s, and Max_{iter} represent the dimension of search space, complexity of fitness function, the population size of GA, and the maximum iteration number of GA respectively, while '*' indicates the arithmetic multiplication operator. Since time complexity

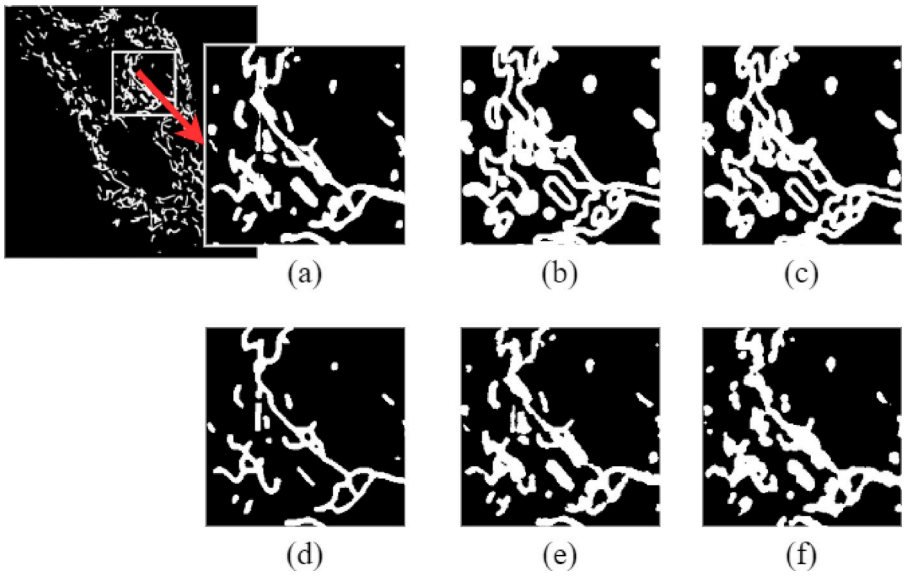


Fig. 2. Illustration of reconstructed images generated from different compact image representations with the help of a cropped part (marked within a rectangle in (a)) of fluorescent microscopy image showing mitochondria image taken from UiTMito dataset. In the reconstruction process, a 7 × 7 Gaussian kernel is used. Here, (a): original image with the magnified cropped region, (b): a reconstructed image generated from Canny edges, (c): a reconstructed image generated from boundary pixels, (d): a reconstructed image generated from skeleton pixels, (e): a reconstructed image generated from randomly selected key pixels, and (f): reconstructed image generated from key pixels selected using GA.

depends on the dimension of search space, selecting key pixels only from foreground (i.e., data) pixels reduces the compute time. The major contributions of the present work are as follows:

- An GA-based technique is proposed for the compact representation of images in a memory-efficient way.
- To the best of our knowledge, image compact representation for memory-efficient storage is in its initial phase.
- A multi-objective function is designed to keep a trade-off between shape and pixel intensity information preservation.
- A meta-heuristic algorithm is used for the first time beyond problems like continuous function optimization, feature selection, and variable selection.
- Three different types of metrics and four datasets are used to evaluate the model performance.
- The proposed method outperforms the existing possible compact image representation techniques.

2. Present work

In this work, we have proposed a method for compact representation of images. This method first applies Otsu's segmentation method (Otsu, 1979) to obtain the foreground (or data) pixels of an input image of dimension $H \times W$ (say, $I_B = \{f'(x,y) : f'(x,y) \in \{0,1\} \wedge (x,y) \in [1,H] \times [1,W]\}$) and then selects some foreground pixels as the key pixels. The aim is to store the positions of the key pixels (say, KP_{pos}) for future use. KP_{pos} generation process is shown in Algorithm 1. The key pixels are selected so that the original image (say, $I = \{f(x,y) : f(x,y) \in \{0,1,\ldots,255\} \wedge (x,y) \in [1,H] \times [1,W]\}$) can be restored by convolving with the Gaussian kernel (say, K) with optimal reconstruction error. The reconstruction process is described in Algorithm 2. In the selection process, we use GA and its objective function comprises three parameters: one to retain shape, one for retaining intensity information, and the last one to control the number of pixels to be selected. In other words, the objective function is so designed that it can control the trade-off between shape distortion and pixel intensity information loss while using a near optimal number of pixels. The overall diagram of the proposed method is shown in Fig. 3 and the major steps are described hereafter. It is to be noted here that the key pixel selection process (i.e., GA) uses the Gaussian kernel based image reconstruction process to estimate pixel intensity loss.

Algorithm 1 Compact Image Representation Generation from Input Image

Require: Original image (i.e., I),
Ensure: Compact image representation i.e., a list of key pixel position (say, KP_{pos})
$H, W \leftarrow$ getImageShape(I) ▷ H is height and W is width
$I_B \leftarrow$ applyOtsuThresholding (I)
$i \leftarrow 1$
$j \leftarrow 1$
$D_{cnt} \leftarrow 0$ ▷ D_{cnt} is total number of data pixels
$DP_{pos} \leftarrow \{\}$ ▷ DP_{pos} is data pixels' positions
*********Encoding I_B to chromosome like structure *********
while $i \leq H$ **do**
  while $i \leq H$ **do**
    if $I_B(i,j)$ ==‘1’ **then** ▷ Binary value ‘1’ represents data pixel
      $DP_{pos} \leftarrow DP_{pos} \cup \{(i,j)\}$
      $D_{cnt} \leftarrow D_{cnt} + 1$
    end if
    $j \leftarrow j + 1$
  end while
  $i \leftarrow i + 1$
end while
$Crom_{len} \leftarrow D_{cnt}$
$KP_{pos} \leftarrow$applyGA(Max_{itr}, P_s, C_p, M_p, $Crom_{len}$) ▷ Parameters are defined in Algorithm 3

2.1. Image reconstruction from a given set of key pixels' position

As already mentioned that the key pixel selection process uses the Gaussian kernel based image reconstruction process and thus we describe the reconstruction process prior to describing other processes. In the reconstruction process, we use the Gaussian kernel (i.e., K) of size ($k \times k$) that is convoluted with the ($I' = \{f'(x,y) : f'(x,y) \in \{0,1\} \wedge (x,y) \in [1,H] \times [1,W]\}$) generated from I_B of an original image (i.e., I) using GA and obtain the reconstructed image (say, $I'' = \{f''(x,y) :$

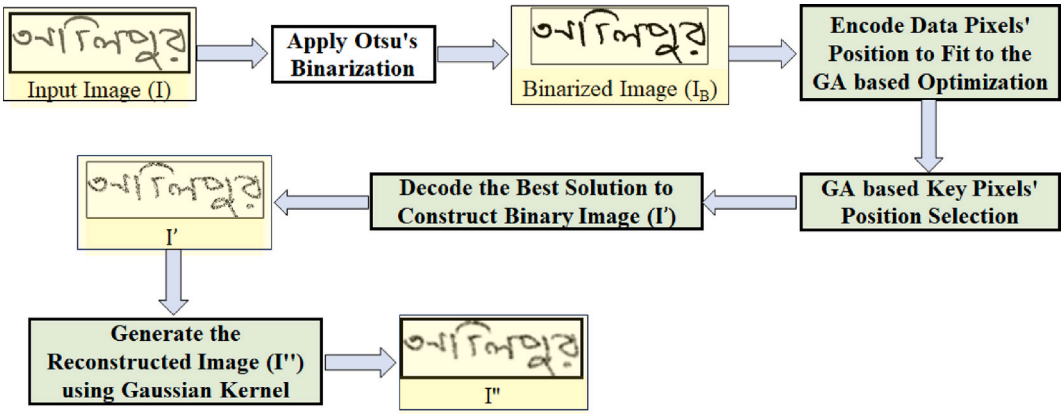


Fig. 3. Flowchart showing the steps used in the present method.

Algorithm 2 Image Reconstruction from Compact Image Representation

Require: List of key pixel position (say, KP_{pos}), List of data pixel positions (say, DP_{pos}), Length of chromosome ($Crom_{len}$), Kernel Size (say, k), Image height (say, H), and Image width (say, W)
Ensure: Reconstructed image (i.e., I'')
$I' \leftarrow$ initializeOneMatrix(H, W) ▷ '1' represents not-data pixel
$i \leftarrow 1$
*********Decoding chromosome to I'*********
while $i \leq Crom_{len}$ **do**
  if $KP_{pos}(i)$ ==='1' **then** ▷ Binary value '1' represents key pixel
    $I'(DP_{pos}(i)) \leftarrow 0$ ▷ '0' represents data pixel
  end if
  $i \leftarrow i + 1$
end while
$K \leftarrow$ getGaussianKernel(k)
$I'' \leftarrow$ applyConvolution(I', K) ▷ GaussianBlur function from pytorch is used
$I'' \leftarrow$ applyContrastStretching(I'')
$I'' \leftarrow$ applyAreaClosing(I'')
$I'' \leftarrow I'' * 255$ ▷ Transform I'' as grayscale image

$f''(x, y) \in [0, 1] \wedge (x, y) \in [1, H] \times [1, W]\}$) i.e., $I'' = I' \circledast K$, where $\circledast$ is the convolutional operator and H, and W represent height and width of the images. Next we have employed contrast stretching on I'' to adjust the contrast of obtained using Gaussian blur process. Finally, we have employed the gray level morphological closing operator with 3×3 structuring element on I'' to generate the final reconstructed image. The target of the present work is to degrade I to generate I' in a controlled way so that $I \sim (255 * I'')$. The entire reconstruction process is pictorially illustrated in Fig. 4 while the process is described in Algorithm 2.

2.2. Encoding binarized image to fit for GA

The GA-based pixel selection process cannot be directly employed on the binarized image (i.e., I_B). Therefore, we encode the I_B to fit it into the GA-based selection process. The basic component on which GA works is chromosomes. Thus, we encode I_B to form a chromosome-like structure (see Fig. 5). At first, we have listed the data pixels' position in a list of 2D points shown in Fig. 5. The length of the list is considered as the length of the chromosome and the indices of elements in the list as the indices of the chromosome. A '1' value at the ith location in the near-optimal chromosome represents that the pixel position associated with this index is a key pixel, while '0' indicates otherwise. This process is explained using a toy example in Fig. 5 and the steps are illustrated in Algorithm 1.

2.3. GA-based key pixels selection

The GA is a popular meta-heuristic algorithm that follows Darwinian's theory of evolution of natural selection and genetics known as "survival of the fittest". The GA is employed on a set of candidate solutions, alternatively known as chromosomes, aiming at obtaining the near-optimal solution by generating better chromosomes with the help of two genetic operators: crossover and mutation. The set of candidate solutions is called "population". A chromosome's fitness score, which is calculated using an objective function (see Section 2.3.2) decides its rank in the population it belongs to. The selection process continues generating candidate solutions till it finds a satisfactory solution or reaches the maximum number of allowed generations. The crossover (used for exploring search space) and mutation (an operation used to exploit search space) operators of GA allow it to explore and exploit any large search spaces efficiently. This section discusses important components of the GA and their setting in our work. The entire process is shown in Fig. 6 and the algorithm is shown in Algorithm 3.

2.3.1. Initial population generation

The population is a collection of candidate solutions. Let, the population consist of P_s ($\in \mathbb{N}$) number of candidate solutions (candidate to qualify as the near-optimal solution). A candidate solution is a binary string, also known as a chromosome, having length n, the number of foreground pixels. In the candidate solution, a "1" indicates the pixel position is selected as key pixels, and "0" means non-selection. In our case, candidate solutions of the initial population are generated randomly, and the value of $P_s = 20$ is set empirically.

2.3.2. Fitness of chromosomes

Chromosomes in a population are ranked by their fitness score. A higher fitness score means a better candidate solution. The fitness score of a chromosome is measured with an objective function specially designed for the problem at hand. The objective function comprised three parameters to keep a good balance among pixel reduction rate (say, r), shape similarity index (say, s), and gray scale similarity index (say, g). The objective function (say, $f(r, s, t)$) used here to measure the fitness score of a candidate solution (say, λ) is defined by Eq. (1).

$$\lambda = f(r, s, t) = \alpha * r + \beta * s + \gamma * g \quad (1)$$

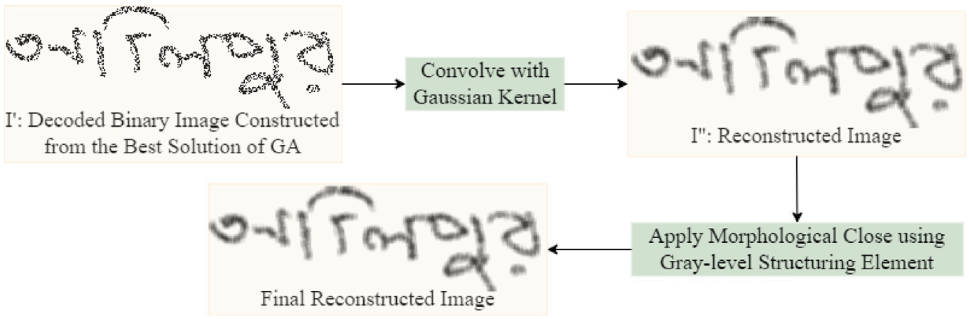


Fig. 4. Flowchart showing the steps used to generate reconstructed image (i.e., I'') from decoded binary image (i.e., I') from selected key pixel positions using GA.

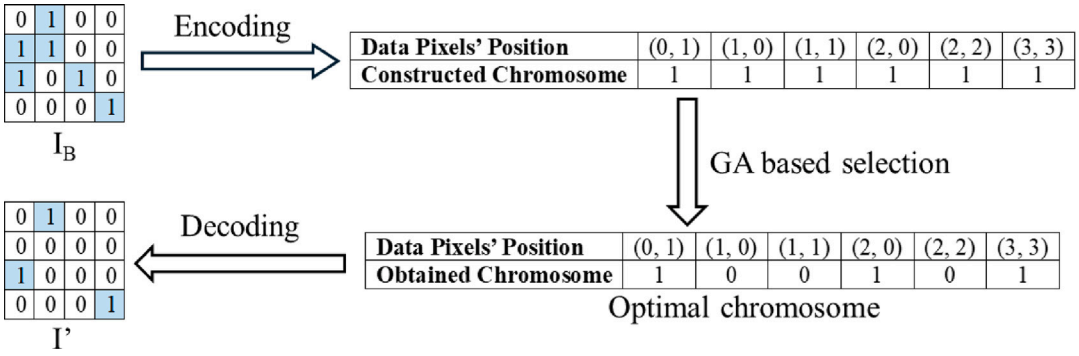


Fig. 5. Encoding of the binarized image (i.e., I_B) to chromosome structure and reconstruction of the binarized image (i.e., I') from selected chromosome after employing GA using a toy example. In this figure, encoding refers to the process of converting I_B into the chromosomal structure fit for GA and decoding refers to the process of constructing I' from the selected near-optimal chromosome.

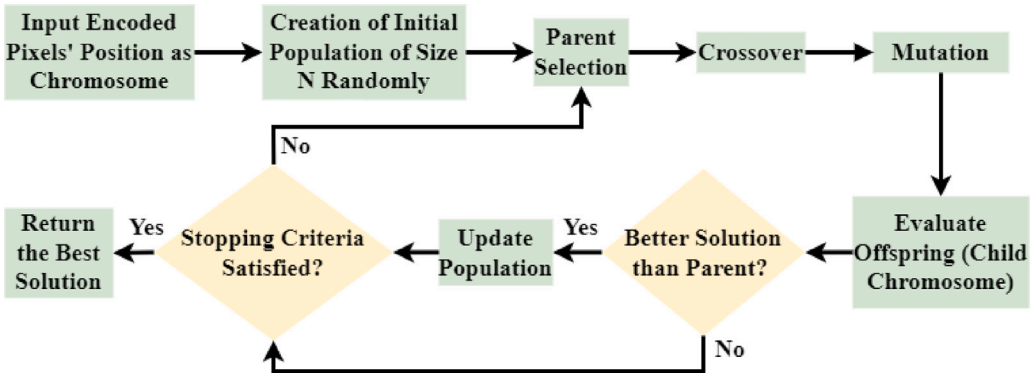


Fig. 6. Flowchart showing the steps used in the present GA-based guided denoiser.

In Eq. (1), α, β, and γ are three constant values. Experimentally, we found that $\alpha = 0.4$, $\beta = 0.3$, and $\gamma = 0.3$ are suitable choices. The parameter r is defined using Eq. (2).

$$r = \frac{|\{(x,y) : f_B(x,y) = 1\}| - |\{(x,y) : f'(x,y) = 1\}|}{|\{(x,y) : f_B(x,y) = 1\}|} \tag{2}$$

To calculate s, we have considered F1-score between I_B and binarized version of I'' (say, $I''_B = \{f''_B(x,y) : f''_B(x,y) \in 0,1 \wedge (x,y) \in [1,H] \times [1,W]\}$). Otsu's thresholding approach is used to generate I''_B from I''. s is calculated using Eq. (3)

$$s = \frac{2 * Re * Pr}{Re + Pr} \tag{3}$$

In Eq. (3), Re and Pr represent recall (see Eq. (4)) and precision (see Eq. (5)) respectively.

$$Re = \frac{|\{(x,y) : f_B(x,y) = 1 \wedge f''_B(x,y) = 1\}|}{|\{(x,y) : f_B(x,y) = 1 \wedge f''_B(x,y) = 1\}| + |\{(x,y) : f_B(x,y) = 1 \wedge f''_B(x,y) = 0\}|} \tag{4}$$

$$Pr = \frac{|\{(x,y) : f_B(x,y) = 1 \wedge f''_B(x,y) = 1\}|}{|\{(x,y) : f_B(x,y) = 1 \wedge f''_B(x,y) = 1\}| + |\{(x,y) : f_B(x,y) = 0 \wedge f''_B(x,y) = 1\}|} \tag{5}$$

Finally, for g we have measured root mean squared error (RMSE) between I, and I'' (say, m) using Eq. (6) first, and then g is calculated by taking the reciprocal of m i.e., $g = \frac{1}{m}$. This is done to comply with the fact that a smaller RMSE score implies a better reconstruction of the gray-scale image.

$$m = \sqrt{\frac{1}{H * W} \sum_{x=1}^{H} \sum_{y=1}^{W} [f''(x,y) - f(x,y)]^2} \tag{6}$$

2.3.3. Crossover

The crossover operator helps the GA explore the search space better. Single-point crossover, two-point crossover, and uniform crossover are

Algorithm 3 GA-based Key Pixels Selection

Require: Stopping condition (say, Max_{itr}), Population size (P_s), Crossover probability (say, C_p), Mutation probability (say, M_p), encoded chromosome length (say $Crom_{len}$)
Ensure: Optimal chromosome (say, Op_{Crom}) of length $Crom_{len}$

```
Pop ← generateRandomPopulation(P_s)
Pop_fit ← evaluateFitnessOfChromosome(Pop)
while ¬C do
    P' ← {} ∪ performElitism(Pop_fit)
    while |P'| < P_s| do
        p_1, p_2 ← performParentSelection(Pop_fit)
        Fit_p1 ← getFitnessOfChromosome(p_1)
        Fit_p2 ← evaluateFitnessOfChromosome(p_2)
        P' ← P' − {p_1, p_2}
        o_1, o_2 ← performCrossover(p_1, p_2, C_p)
        o_1 ← performMutation(o_1, M_p)
        o_2 ← performMutation(o_2, M_p)
        Fit_o1 ← evaluateFitnessOfChromosome(o_1)
        Fit_o2 ← evaluateFitnessOfChromosome(o_2)
        if max(Fit_o1, Fit_o2) > max(Fit_p1, Fit_p2) then
            P' ← P' ∪ {o_1, o_2}
        else
            P' ← P' ∪ {p_1, p_2}
        end if
    end while
    Pop ← P'
end while
Op_Crom ← top-ranked (according to fitness score) in Pop
```

present in the literature. We have used the uniform crossover (Spears and De Jong, 1995) is used due to its better applicability in the larger search spaces (Malakar et al., 2020a). In this crossover scheme, we first select two chromosomes from the entire population following the roulette wheel selection algorithm as parent chromosomes, and then the bits present there are exchanged based on a probability value, known as crossover probability (say, C_p), to obtain two child chromosomes. We have employed crossover operators for $\mu \in [3, 5]$ times and $C_p = 0.85$ is set empirically.

2.3.4. Mutation

The mutation operator guides GA in exploiting the search space efficiently. In this phase, some of the bits in a chromosome are flipped based on a probability score, called mutation probability (say, M_p). Here, we choose $M_p = 0.03$ empirically.

2.3.5. Stopping criteria

GA is an iterative process, requiring a stopping criterion: the maximum number of generations allowed (say, Max_{itr}). We have set the number $Max_{itr} = 15$ empirically.

3. Experimental results

GA has been used in the present work to select key pixels. The positions of the key pixels and the image dimension will be stored for further use. The present method is evaluated on four datasets, and the results are analyzed qualitatively and quantitatively. In the following subsections, we first describe the datasets in use, evaluation metrics, and parameters used in GA, and then describe the results obtained. It is to be mentioned here that we have included the recognition performances along with other quantitative metrics described later for analyzing the results.

3.1. Datasets in use

For experimental purposes, we have used four datasets; two contain images of handwritten digits and words, and the other two have fluorescence microscopy. MNIST (Yan et al., 1998) and CMATERdb2.1.2 (Bhowmik et al., 2019) are considered as handwritten digits and word databases, respectively, while UitMito (Sekh et al., 2021) and 2D HeLa (Boland and Murphy, 2001) datasets contain fluorescent microscopy images. Such datasets are chosen to include diversified data while having gray-scale images. A summary of these datasets is provided in Table 1 as well as described a bit below.

MNIST: This dataset (Yan et al., 1998) is of handwritten digit images that have been widely used to benchmark different image classification and machine learning problems. The dataset was collected by the authors and scanned using a flatbed scanner. It consists of 70,000 sample images of handwritten digits distributed equally over all the classes, i.e., it contains 7000 sample images per class. All the samples are in grayscale with the resolution of 28 × 28 pixels. This dataset is used for a 10-class classification problem.

CMATERdb2.1.2: Bhowmik et al. (2019) took the initiative to prepare this dataset that contains images of handwritten Bangla words. It is suitable for evaluating the performance of character extraction algorithms from handwritten words (Malakar et al., 2021, 2011) are used to measure the performance of handwritten word image recognition using a holistic approach (Malakar et al., 2017, 2020c). It contains handwritten words representing 120 popular city names in West Bengal, India. 150 samples written by writers with varying ages, genders, and educational qualifications per city name are present in this dataset. In short, classically, this database is used for 120 class classification or pattern classification problems.

2D HeLa: Boland and Murphy (2001) prepared the 2D HeLa dataset, in short, we call it hereafter HeLa, and made it public for further research. It consists of images of biological samples that are generated using fluorescence microscopy. HeLa cells were stained with organelle-specific fluorescent dyes while capturing the images. The dataset includes images of 10 organelles and thus it is suitable for 10 class classification problems.

UitMito: Sekh et al. (2021) generated this dataset consisting of fluorescence microscopy images of live cells stained with a mitochondrial-specific fluorescent dye. The dataset contains 1000 2D grayscale images, each with a resolution of 1024 × 1024 pixels, and was captured over 1000 s.

3.2. Metrics in use

Apart from the qualitative evaluation, we have also performed a quantitative assessment of the proposed work. Two different approaches are followed while designing quantitative metrics. One concentrates on how much space is used while storing the data in the memory. Here, no bit encoding mechanism is used rather the number of integer values required to store is considered. For this type of quantitative measurement, we have used three metrics viz., actual pixel reduction rate (see Eq. (7)), foreground pixel reduction rate ($FPRR$) (see Eq. (8)), and byte reduction rate (BRR) (see Eq. (9)). The other one is the retrieval performance i.e., how much information of the original image is retained in the reconstructed image generated from its compact representation. Here we have used the recall (see Eq. (4)), precision (see Eq. (5)), F1-score (see Eq. (3)), structural similarity index (SSIM) score (see Eq. (10)), RMSE (see Eq. (6)). Apart from these, we have also tested the quality of the generated images with the help of classification performance for the datasets for which the classification problem is defined, i.e., MNIST, CMATERdb, and HeLa.

- $APRR$: It is defined by the ratio of the number of pixels used to represent an image (i.e., the number of pixels whose position we

Table 1
Summary of sample counts in the datasets in use. Here resolution is $W \times H$.

Dataset	# samples	Resolution	Capturing device	Accessibility
MNIST	70 000	28 × 28	Scanner	Public
CMATERdb2.1.2	18 000	Varying	Scanner	Public
HeLa	862	512 × 382	Fluorescence microscopy	Public
UiTMito	1000	1024 × 1024	Fluorescence microscopy	In-house[a]

[a] Represents that a portion of the dataset is public.

will store instead of the original image) and the total number of pixels present in the image. It is defined by the Eq. (7).

$$APRR = \frac{|\{(x,y) : f'(x,y) = 1\}|}{H * W} \tag{7}$$

- $FPRR$: It is a similar measure as APRR but here ratio is calculated concerning the number of active or foreground pixels and it is defined by the Eq. (8).

$$FPRR = \frac{|\{(x,y) : f'(x,y) = 1\}|}{|\{(x,y) : f_B(x,y) = 1\}|} \tag{8}$$

- BRR: It is a measure that is similar to the compression ratio. It is calculated as the ratio between the number of bits needed to store the compact representation of the image and the number of bits needed to store the original image in its uncompressed form (see Eq. (8)). In Eq. (8), b and b' depend on the dimension and bit depth of the input image to be represented compactly. In our computation, bit depth is considered as 8 (i.e., $b' = 8$), while $b = \lceil \frac{\log_2(max\{H,W\})}{8} \rceil * 8$ considering bit requirement to store an integer number as multiple of 8. This means if $max\{H,W\} < 256$, we set $b = 8$ while if $max\{H,W\} \in [256, 65535]$ we set $b = 16$

$$BRR = \frac{2 * |\{(x,y) : f'(x,y) = 1\} + 1| * b}{|\{H * W * b'\}|} \tag{9}$$

- $SSIM$: SSIM score finds the similarity between two images. Here, we use this metric along with the RMSE score to have an understanding of the reconstruction similarity between I and I''. The score is defined in Eq. (10).

$$SSIM = \frac{(2 * \mu_I * \mu_{I''} + c_1) * (2 * \sigma_{II''} + c_2)}{(\mu_I^2 + \mu_{I''}^2 + c_1)(\sigma_I^2 + \sigma_{I''}^2 + c_2)} \tag{10}$$

In Eq. (10), μ_I, and $\mu_{I''}$ represent the mean of the pixel intensities in I, and I'' respectively while standard deviations of all pixel intensities present in I, and I'' are represented by σ_I, and $\sigma_{I''}$ respectively. Also, $\sigma_{II''}$ is the covariance between I and I''. $c_1 = (k_1 * L)^2, c_2 = (k_2 * L)^2$ are two variables to stabilize the division with a weak denominator where L is the dynamic range of pixel values, while $k_1 = 0.01, k_2 = 0.03$ are two constant values.

3.3. Parameter selection

Two factors can significantly affect the end performance of the proposed model: (1) the weight values in the objective function i.e., α, β, and γ in Eq. (1), and (2) the size and type of the kernel that convolves with the binary image generated from compact representation (i.e., I') to generate the reconstructed image (i.e., I''). To decide on these two factors, we have experimented with a subset from each dataset by selecting 5% of the samples randomly. The parameters α, β, and γ determine the contribution of r, s, and t respectively (see Eq. (1)) in a multi-criteria/objective function. The higher the value of α, the lower the number of selected key pixels i.e., lesser memory requirement to store an image. On the contrary, higher values of β, and γ ensure a better quality of the reconstructed image. So, we have tried to keep the values close to each other and thus these values are chosen from [0.2, 0.5] experimentally. To decide the values of α, β, and γ, we have experimented on the mentioned subset of the MNIST dataset using a 3 × 3 Gaussian kernel. The results for a few tested combinations are shown in Table 2. From the results, it can be concluded that $\alpha = 0.4$, $\beta = 0.3$, and $\gamma = 0.3$ are good choices and we use these values in our further experiments.

Table 2
Performance of the proposed image compact representation technique with different values of parameters α, β, and γ in Eq. (1). Here GA selects key pixels from the binary version of an image (i.e., I_B), and the 3 × 3 Gaussian kernel is convolved with I' to generate I''. The metrics used here quantify the reconstruction error from I' using the corresponding kernels. ↑, and ↓ represent larger means better results and smaller means better results, respectively.

α	β	γ	Recall (↑)	Precision (↑)	F1-score (↑)	SSIM (↑)	RMSE (↓)
0.2	0.3	0.5	.82 ± .11	.69 ± .09	.75 ± .08	.72 ± .08	.13 ± .02
0.2	0.5	0.3	.81 ± .10	.72 ± .09	.76 ± .09	.74 ± .09	.11 ± .02
0.5	0.2	0.3	.75 ± .13	.77 ± .06	.76 ± .07	.74 ± .08	.14 ± .02
0.5	0.3	0.2	.74 ± .11	.76 ± .08	.76 ± .09	.75 ± .10	.15 ± .02
0.3	0.3	0.4	.82 ± .13	.74 ± .09	.77 ± .08	.73 ± .09	.11 ± .02
0.3	0.4	0.3	.79 ± .14	.75 ± .09	.77 ± .07	.75 ± .08	.12 ± .02
0.4	0.3	0.3	.80 ± .12	.76 ± .07	.78 ± .07	.76 ± .08	.10 ± .02
0.2	0.4	0.4	.81 ± .16	.70 ± .11	.76 ± .09	.73 ± .09	.11 ± .02
0.4	0.2	0.4	.80 ± .13	.74 ± .09	.77 ± .09	.74 ± .10	.12 ± .02
0.4	0.4	0.2	.78 ± .12	.75 ± .10	.77 ± .09	.75 ± .09	.11 ± .02

The shape of the reconstructed images might differ based on the kernel size due to the stroke width of the data parts in the input images, while the weight of different kernels may lead to different pixel intensity information in the reconstructed images. To decide on this factor, we have used three different kernel sizes, viz., 3 × 3, 5 × 5, and 7 × 7, and three different types of kernel functions, namely, the Gaussian kernel, the point spread function (PSF) kernel, and the mean kernel. The obtained results are recorded in Table 3. From this table, it is clear that the Gaussian kernel outperforms the others when considering the same kernel size. For MNIST and CMATERdb 5 × 5 kernel provides the best performance, while on the other two datasets (i.e., HeLa and UiTMito) 7 × 7 kernel provides the best performance. Values of all the parameters used in this work are listed in Table 4.

3.4. Results on entire dataset

We have employed our model on the four datasets described earlier. In these experiments, we have used the selected parameter values of Eq. (1), kernel size, and kernel type as decided in Section 3.3 through experimentation on subsets of the datasets. We have also used the empirically selected parameter values for GA. All the parameters and their corresponding values are listed in Table 4. The results are shown in Table 5. It can be seen from this table that the proposed method achieved .84 ± .14, .84 ± .07, .78 ± .13, and .88 ± .05 F1-score on MNIST, CMATERdb, HeLa, and UiTMito datasets respectively. So, we obtained the best reconstructed performance on the UiTMito dataset while the worst is on the HeLa dataset. Similarly, if we consider the SSIM score, then the proposed method performs poorly on MNIST and CMATERdb datasets while performing well for the other two datasets. If we consider the space requirement reduction, then it can be seen

Table 3
Performance of the proposed image compact representation technique. Here GA selects key pixels from the binary version of an image (i.e., I_B), and the mentioned kernels are convolved with I' to generate I''.

Dataset	Kernel	Kernel size	Recall (↑)	Precision (↑)	F1-score (↑)	SSIM (↑)	RMSE (↓)
MNIST	Mean	3 × 3	.80 ± .12	.76 ± .07	.78 ± .07	.76 ± .08	.10 ± .02
		5 × 5	.81 ± .13	.78 ± .11	.79 ± .09	.77 ± .10	.09 ± .03
		7 × 7	.77 ± .12	.78 ± .15	.77 ± .11	.76 ± .11	.11 ± .04
	PSF	3 × 3	.76 ± .10	.73 ± .08	.74 ± .07	.73 ± .08	.12 ± .02
		5 × 5	.79 ± .12	.74 ± .11	.76 ± .09	.75 ± .10	.10 ± .03
		7 × 7	.75 ± .11	.74 ± .13	.74 ± .12	.74 ± .11	.13 ± .04
	Gaussian	3 × 3	.82 ± .10	.79 ± .06	.80 ± .06	.80 ± .08	.07 ± .02
		5 × 5	.87 ± .13	.82 ± .11	.85 ± .09	.83 ± .10	.06 ± .03
		7 × 7	.83 ± .15	.79 ± .10	.81 ± .10	.81 ± .09	.08 ± .04
CMATERdb	Mean	3 × 3	.80 ± .07	.78 ± .05	.79 ± .05	.75 ± .06	.08 ± .02
		5 × 5	.82 ± .07	.78 ± .05	.80 ± .05	.78 ± .06	.07 ± .02
		7 × 7	.80 ± .06	.78 ± .07	.79 ± .08	.76 ± .06	.09 ± .02
	PSF	3 × 3	.79 ± .07	.76 ± .04	.77 ± .04	.72 ± .07	.11 ± .02
		5 × 5	.80 ± .08	.76 ± .05	.78 ± .06	.75 ± .07	.09 ± .02
		7 × 7	.77 ± .05	.74 ± .07	.75 ± .05	.73 ± .08	.12 ± .02
	Gaussian	3 × 3	.82 ± .07	.85 ± .05	.84 ± .05	.81 ± .06	.06 ± .02
		5 × 5	.86 ± .07	.83 ± .05	.85 ± .05	.82 ± .06	.04 ± .02
		7 × 7	.85 ± .07	.83 ± .05	.84 ± .05	.80 ± .06	.06 ± .02
HeLa	Mean	3 × 3	.75 ± .12	.70 ± .13	.72 ± .12	.81 ± .11	.11 ± .07
		5 × 5	.74 ± .11	.71 ± .12	.72 ± .10	.82 ± .10	.10 ± .07
		7 × 7	.77 ± .11	.74 ± .15	.75 ± .12	.84 ± .10	.10 ± .06
	PSF	3 × 3	.76 ± .09	.69 ± .13	.70 ± .10	.79 ± .10	.12 ± .07
		5 × 5	.72 ± .10	.68 ± .13	.70 ± .10	.80 ± .10	.11 ± .06
		7 × 7	.72 ± .11	.71 ± .15	.71 ± .12	.81 ± .11	.10 ± .06
	Gaussian	3 × 3	.76 ± .09	.74 ± .12	.75 ± .09	.89 ± .10	.11 ± .06
		5 × 5	.76 ± .11	.74 ± .14	.76 ± .13	.91 ± .10	.11 ± .06
		7 × 7	.81 ± .10	.77 ± .15	.79 ± .11	.93 ± .09	.09 ± .05
UiTMito	Mean	3 × 3	.83 ± .10	.79 ± .11	.81 ± .10	.86 ± .09	.05 ± .03
		5 × 5	.83 ± .10	.81 ± .10	.82 ± .09	.88 ± .09	.05 ± .05
		7 × 7	.88 ± .09	.84 ± .11	.86 ± .10	.90 ± .08	.04 ± .03
	PSF	3 × 3	.84 ± .09	.81 ± .11	.82 ± .10	.85 ± .10	.06 ± .05
		5 × 5	.84 ± .09	.82 ± .10	.83 ± .10	.87 ± .09	.05 ± .04
		7 × 7	.87 ± .08	.83 ± .12	.85 ± .11	.89 ± .10	.05 ± .04
	Gaussian	3 × 3	.85 ± .09	.84 ± .10	.84 ± .10	.90 ± .09	.06 ± .03
		5 × 5	.86 ± .09	.83 ± .11	.84 ± .10	.92 ± .09	.04 ± .03
		7 × 7	.91 ± .08	.87 ± .11	.89 ± .09	.95 ± .08	.03 ± .03

Table 4
List of parameters and their values used in this work to cite the results on the entire datasets.

Parameter	Description	Value
α	Weight of pixel reduction rate in Eq. (1)	0.4
β	Weight of shape similarity index in Eq. (1)	0.3
γ	Weight of gray-scale similarity index in Eq. (1)	0.3
k	Kernel size in reconstruction process (MNIST and CMATERdb)	5
	Kernel size in reconstruction process (HeLa and UiTMito)	7
P_s	Population size	20
C_p	Crossover probability	0.85
M_p	Mutation probability	0.03
Max_{itr}	Maximum generation for stopping criterion	15

that the proposed technique reduces the memory requirement (see BRR score) by around 95.00%, 87.00%, 98.00%, and 96.00% for MNIST, CMATERdb, HeLa, and UiTMito datasets respectively.

3.5. Execution time

The key pixels selection process for compact image representation uses GA. Theoretically, the time complexity of the GA algorithm is $\sim O(D * F * P_s * Max_{iter})$, where D, F, P_s, and Max_{iter} represent the dimension of search space, complexity of fitness function, the population size of GA, and the maximum iteration number of GA respectively, while '*' indicates the arithmetic multiplication operator. Practically, out of these factors, the execution time mostly depends a lot on the nature of the objective function i.e., fitness evaluation

Table 5
Performance of the proposed memory efficient compact representation of images technique on the entire dataset.

Performance metrics	Dataset in use			
	MNIST	CMATERdb	HeLa	UiTMito
Recall	.86 ± .13	.85 ± .06	.80 ± .11	.90 ± .04
Precision	.82 ± .15	.83 ± .07	.76 ± .14	.87 ± .06
F1-score	.84 ± .14	.84 ± .07	.78 ± .13	.88 ± .05
SSIM	.82 ± .11	.81 ± .08	.92 ± .04	.94 ± .04
RMSE	.08 ± .05	.07 ± .06	.11 ± .07	.06 ± .03
APRR	.02 ± .01	.04 ± .02	.01 ± .01	.01 ± .01
FPRR	.17 ± .05	.21 ± .05	.15 ± .03	.12 ± .01
BRR	.04 ± .02	.13 ± .07	.02 ± .03	.04 ± .01

Table 6
Single-time execution time of important operations on the proposed method. Time is recorded in seconds (approximated at 4-decimal place). '–' indicate time is negligible to 4th-decimal place. Here, resolution is provided as $W \times H$. OP # is used to provide a unique ID to each operation.

Op #	Operation	Execution time (in second) on image of resolution			
		28 × 28	442 × 221	512 × 382	1024 × 1024
01	Image acquisition[a]	0.0050	0.4350	0.3240	0.0200
02	Image binarization[a]	0.0002	0.0003	0.0004	0.0009
03	Encoding I_B to form chromosome	0.0013	0.0433	0.3542	1.8403
04	Reconstructing I' from KP_{pos}	0.0001	0.0011	0.0048	0.0156
05	I'' generation (i.e., $I'' = I' \circledast K$)[a]	0.0005	0.0006	0.0008	0.0011
06	Contrast stretching of I''	0.0029	0.0927	0.7432	3.9837
07	Area closing on I''[a]	0.0046	0.0428	0.2734	2.1026
08	RMSE loss calculation[a]	0.0004	0.0008	0.0045	0.0204
09	Shape similarity index estimation	0.0051	0.1674	1.3155	6.9860
10	Sorting candidate solutions	0.0004	0.0018	0.0071	0.0241
11	Crossover	–	0.0020	0.0090	0.0278
12	Mutation	–	0.0007	0.0036	0.0125

[a] Indicates that this operation uses a built-in Python library function while a user-defined function is used in other cases.

Table 7
Comparison with the other image compact representation techniques considered here for comparison. Images are reconstructed using the Gaussian kernel with specified kernel size (5 × 5 for MNIST and CMATERdb and 7 × 7 for HeLa and UiTMito).

Dataset	Representer	F1-score (↑)	SSIM (↑)	RMSE (↓)	APRR (↓)	FPRR (↓)	BRR (↓)
MNIST	Canny edge	.39 ± .07	.49 ± .07	.08 ± .04	.09 ± .02	.76 ± .21	.20 ± .08
	Boundary	.60 ± .12	.68 ± .14	.09 ± .04	.09 ± .02	.74 ± .20	.19 ± .06
	Skeleton	.60 ± .09	.69 ± .10	.06 ± .05	.04 ± .01	.35 ± .09	.08 ± .05
	Random pixels	.83 ± .10	.83 ± .09	.09 ± .04	.07 ± .02	.50 ± .05	.15 ± .09
	Proposed	.84 ± .14	.82 ± .11	.08 ± .05	.02 ± .01	.17 ± .05	.04 ± .02
CMATERdb	Canny edge	.56 ± .04	.61 ± .12	.25 ± .05	.06 ± .01	.36 ± .06	.26 ± .06
	Boundary	.51 ± .07	.57 ± .10	.19 ± .06	.06 ± .01	.33 ± .07	.24 ± .04
	Skeleton	.57 ± .08	.65 ± .11	.25 ± .07	.08 ± .02	.46 ± .20	.32 ± .09
	Random pixels	.84 ± .05	.82 ± .05	.07 ± .02	.09 ± .03	.50 ± .01	.38 ± .11
	Proposed	.84 ± .07	.81 ± .08	.07 ± .06	.04 ± .02	.21 ± .05	.13 ± .07
HeLa	Canny edge	.50 ± .17	.59 ± .10	.04 ± .08	.04 ± .04	.13 ± .11	.01 ± .01
	Boundary	.42 ± .18	.53 ± .10	.04 ± .09	.01 ± .01	.27 ± .18	.03 ± .02
	Skeleton	.33 ± .17	.51 ± .11	.20 ± .11	.10 ± .05	.21 ± .06	.39 ± .19
	Random pixels	.76 ± .12	.91 ± .10	.02 ± .03	.02 ± .02	.52 ± .06	.08 ± .08
	Proposed	.78 ± .13	.92 ± .04	.11 ± .07	.01 ± .01	.15 ± .03	.02 ± .03
UiTMito	Canny edge	.79 ± .01	.80 ± .06	.08 ± .03	.02 ± .01	.29 ± .02	.08 ± .03
	Boundary	.49 ± .02	.90 ± .03	.06 ± .02	.02 ± .01	.28 ± .02	.08 ± .02
	Skeleton	.80 ± .03	.91 ± .03	.03 ± .01	.01 ± .01	.11 ± .01	.03 ± .01
	Random pixels	.89 ± .01	.93 ± .02	.02 ± .01	.04 ± .01	.50 ± .01	.14 ± .04
	Proposed	.88 ± .05	.94 ± .04	.06 ± .03	.01 ± .01	.12 ± .01	.04 ± .01

strategy. The fitness evaluation depends a lot on data volume (here, image resolution). This is because the same is called several times in the process. Therefore, we have measured the execution time of all the import functions used in the current process with varying image resolutions for a single-time execution and recorded in Table 6. All the times are measured in the Google Colab free version without using GPUs. The operations having identification (ID) code (i.e., OP # in Table 6) 01 is executed only once in the proposed work. The time taken in the image reconstruction process from key pixel positions (i.e., KP_{pos}) is the time taken to execute OP # 04, 05, 06, and 07. So the reconstruction process takes 6.103 (i.e., 0.0156+0.0011+3.9837+2.1026 = 6.103) seconds for an image of resolution 1024 × 1024 while it takes ∼0.0081 s for an image of dimension 28 × 28. The operations related to calculating fitness score of a candidate solution of GA are OP # 02 to OP # 09 excluding OP #03. From, this table, it is clear that these operations consume a lot of time and these times increase heavily with the increase in image resolution. It is also noteworthy to observe that crossover and mutation operations' execution time increases as image resolution increases. This happens because, as the image resolution increases, the length of the chromosome (or dimension of search space i.e., F) also increases.

3.6. Comparison with other compact representation methods

It has already been mentioned that key pixels generated using the Canny edge detection algorithm (Canny, 1986), skeleton extraction techniques (Lee et al., 1994), boundary detection algorithm (Martin et al., 2004), and randomly selected key pixels can also be considered compact representations of the images. Therefore, we have reconstructed the images from the compact representations where Canny edge pixels, skeleton pixels, boundary pixels, and randomly selected key pixels are to be stored using the same Gaussian kernel used for the current work, i.e., 5 × 5 kernel for MNIST and CMATERdb, and 7 × 7 kernel for HeLa and UiTMito datasets. The comparative results are shown in Table 7. From these results, it can be noticed that the F1-score, SSIM, and RMSE scores of reconstructed images from randomly selected key pixels are very close (and sometimes even better) to the scores obtained from the proposed method. However, the proposed method selects much fewer key pixels than the randomly selected pixels and is thus more memory-efficient for storing (see BRR score). If we concentrate on the memory requirement for storing a compact version of an image, then we can see that except for the UiTMito dataset, the proposed performs the best. Even, it uses fewer key pixels than that of

Table 8
Comparison of performance of the proposed method with deep learning-based techniques: GU-Net (Banerjee et al., 2023), and GU-Net++ (Banerjee et al., 2024).

Dataset	Method	SSIM (↑)	RMSE (↓)	APRR (↓)	BRR (↓)
MNIST	GU-Net	.83 ± .02	.06 ± .02	.04 ± .01	.08 ± .02
	GU-Net++	.92 ± .02	.03 ± .01	.04 ± .01	.08 ± .02
	Proposed	.82 ± .11	.08 ± .05	.02 ± .01	.04 ± .02
CMATERdb	GU-Net	.79 ± .03	.11 ± .02	.03 ± .01	.11 ± .02
	GU-Net++	.77 ± .03	.14 ± .04	.03 ± .01	.11 ± .02
	Proposed	.81 ± .08	.07 ± .06	.04 ± .02	.13 ± .07
HeLA	GU-Net	.89 ± .03	.24 ± .04	.21 ± .02	.44 ± .09
	GU-Net++	.74 ± .05	.22 ± .07	.21 ± .02	.44 ± .09
	Proposed	.92 ± .04	.11 ± .07	.01 ± .01	.02 ± .03
UiTMito	GU-Net	.90 ± .09	.20 ± .04	.03 ± .01	.11 ± .02
	GU-Net++	.92 ± .05	.16 ± .05	.03 ± .02	.11 ± .02
	Proposed	.94 ± .04	.06 ± .03	.01 ± .01	.04 ± .01

the skeleton in three of the four datasets. In summary, if we consider the reconstruction similarity and memory requirement for storing the compact representation of an image, then it can be safely commented that the proposed method performs far better than the alternative ways of storing an image in a memory-efficient way.

Apart from comparison with classical methods, we have compared the performances of the present method with two recent deep learning-based methods: guided U-Net (Banerjee et al., 2023) (in short, GU-Net) and its improved version, GU-Net++ (Banerjee et al., 2024). In GU-Net, the authors have trained a U-Net with three loss functions: shape, budget, and skeleton and PSF-based image reconstruction process. In Banerjee et al. (2024), the authors have tried to mitigate the issues of the reconstruction process through the use of the conditional generative adversarial network (cGAN). It is to be noted here that GU-Net and GU-Net++ need training data to build the learned module while the current method does not have such a requirement. Moreover, the volume of the training dataset decides their performance which is also clear from their performances on four variety of datasets. To have a fair comparison with the present technique, these two methods are trained and evaluated on all the four datasets mentioned earlier and the comparative performances are shown in Table 8. The GU-Net and GU-Net++ are evaluated on the test samples shown in Table 9 for MNIST, CMATERdb, and HeLa while for UiTMito 20% of all the samples (see Table 1 i.e., on 200 samples. However, the proposed method is evaluated on all the samples as mentioned in Table 1. From these results, it can be noticed that the performance of the proposed method is better than that of GU-Net, and GU-Net++ methods except the MNIST dataset. The reason behind that could be these deep-learning methods get enough train data to generalize the same. However, if the train samples are less then the current approach is better. Moreover, to work on the UiTMito dataset, the images are partitioned into patches of resolution 256 × 256 pixels following the approach taken by Sekh et al. (2021). But no such special care is not required for the present technique which is advantageous over deep learning-based techniques.

3.7. Discussion

The results described are related to the reconstruction metrics and storage requirements. We found reconstruction errors when the Gaussian kernel is convolved with the decoded binary image from the selected near optimal key pixel positions (i.e., I') and generates. That means there is a clear difference between the original image and the reconstructed image. So the stated metrics do not describe how the reconstruction error will affect real-time applications. To have the answer to this query, we have considered image classification as a real-life application. The datasets: MNIST, CMATERdb, and HeLa have been previously used as classification problems. It is noteworthy to mention here that our objective does not include generating the best classification accuracy for any particular dataset. Rather, our goal is to test how the classification performance could be affected by the loss obtained during the reconstruction process from the key pixels.

We have used EfficientNet-B0 (He et al., 2016) here for performing the classification tasks. The pre-trained EfficientNet-B0 model trained on ImageNet is fine-tuned on respective datasets to perform image classification. Two different experimental setups are followed to perform the classification. In the first setup, we have used the original train and test samples from the datasets to evaluate classification performance. In the other setup, we have used the corresponding reconstructed images and tested on the reconstructed images, considering that the original images will not be available once they are stored using their compact version. The train, validation, and test sample sizes are recorded in Table 9 for the said datasets used for the classification task. The results are presented in Fig. 7. From these results, it can be seen that the classification accuracy obtained on reconstructed MNIST, CMATERdb, and UiTMito images has dropped by 0.17%, 0.14%, and 2.30% respectively. Here, the classification accuracy obtained on the reconstructed images from the randomly selected key pixels is also very close to the proposed one. It is to be noted here that this inspires the present method of randomly selected key pixels approach. In other cases, including the deep learning approaches: GU-Net and GU-Net++, the classification performances are not that promising compared to the original one except for the MNIST dataset. With these results, it can be safely commented that the proposed GA-based key pixels' selection method not only returns less number of key pixels to store the image in a memory-efficient way but also the reconstructed images could be directly useful in real-life applications like image classification.

The proposed approach is planned to lessen the carbon footprint generated while storing vision data in cloud or edge devices for a longer period. So it is required to analyze whether the proposed approach is beneficial for obtaining such a goal. To evaluate the same, we have stored the high-quality microscopy images from the UiTMito (Sekh et al., 2021) dataset using lossless as well as lossy compression techniques. The reason behind such a choice is the higher resolution of the images in the dataset and thus, a better explanation may be cited. When we store an image of resolution 1024 × 1024 from the UiTMito dataset using 24, and 8-bit bit map representation, the image sizes are 1 and 3 Megabytes (MB), respectively. When we store the image using portable network graphics (PNG) format, the storage requirement becomes 747 kilobytes (KB). However, when the same image is stored using an 8-bit graphics interchange format (gif) and joint photographic experts group (JPEG), the memory requirement becomes 239 KB and 95.3 KB, respectively. However, in our storing system, the memory requirement is ∼40.96 KB (i.e., below 50% of storage requirement for JPEG), considering 0.04 as BRR value for the UiTMito dataset. This study illustrates the need for the proposed approach to store vision data for long-term storage.

3.8. Advantages, limitations and potential future research

In this section, we have first summarized the advantages of the proposed method and then discussed its limitations, including potential future research. The advantages of the proposed method are as follows:

- This method picks less than 5% pixels of the original image to store. Moreover, since it selects key pixels only from foreground pixels, it will always shrink the storage requirement, which may not be true for image compression techniques.
- Since the method does not use any deep learning method, this method could be applied to images of any resolution, which is not true for methods like GU-Net and GU-Net++.
- The current approach does not require any training samples. It is designed to work on a dataset comprised of only one sample. Thus, it could be directly integrated with image-capturing techniques to store the images, like image compression techniques.

Table 9
Summary of sample counts in the datasets in use.

Dataset	# samples	# train samples	# validation samples	# test samples
MNIST	70,000	48,000	12,000	10,000
CMATERdb2.1.2	18,000	11,520	2880	3600
HeLa	862	552	138	172

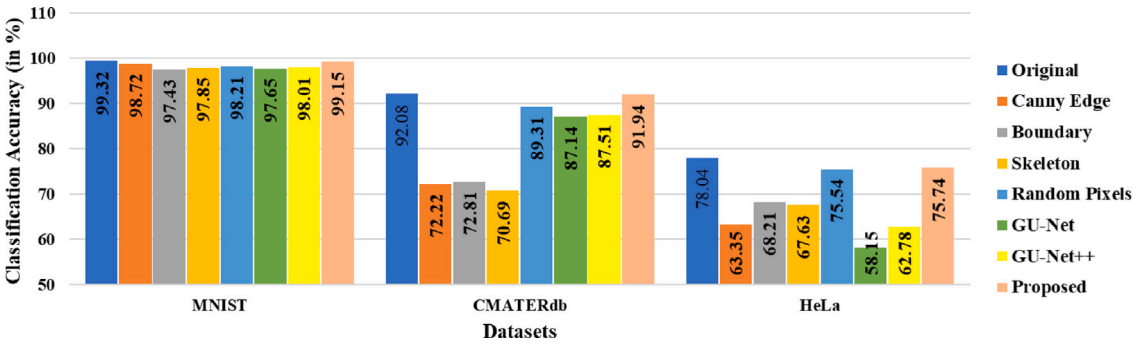


Fig. 7. Performance comparison of the proposed method in terms of classification accuracy with other image compact representation techniques considered here for comparison. Here, the reconstructed images are classified and images are reconstructed using the Gaussian kernel with specified kernel size (5×5 for MNIST and CMATERdb datasets and 7×7 for HeLa and UiTMito datasets).

- The use of kernel-based image reconstruction is compute-efficient reconstruction compared to image compression technique.

Despite its different advantages, this approach requires further improvement by considering its eye-catching limitations. The limitations along with the possible solutions are discussed below.

- The key pixels are selected from the foreground pixels. The majority of the failure that occurs during this is due to the segmentation technique, which is here Otsu's algorithm (Otsu, 1979) as it fails to generate good foreground pixels when the input image is complex. Thus, a more sophisticated image segmentation technique is required to obtain better forground pixels. The unsupervised segmentation techniques like watershed algorithm-based image segmentation (Ng et al., 2006; Guo et al., 2022), pixel-level clustering (Hoang and Kang, 2024), and U2Seg (Niu et al., 2024) could be used in the future considering their impacts on image segmentation problems. Techniques like (Ali et al., 2019a,b) can also be used to extract object regions. Nevertheless, the recent deep learning-aided segmentation techniques like segment anything (Kirillov et al., 2023) could be used.
- The execution time largely depends on the nature of the objective function (see Section 3.5). As the image resolution increases, the execution time for the shape similarity index (measured using F1-score) and RMSE loss calculations increases heavily (see Table 6). Also, the image reconstruction time increases. This issue may be resolved by excluding such a loss calculation and introducing a new measure that can work in compact representation. Such an approach will make the process more energy-efficient.
- The execution time is more for images of larger dimensions (see Table 6), as the more the number of data pixels, the selection process takes more time. So we have a plan to devise some mechanism that can pre-select some important data pixels. So, one can first employ a filter-based approach (Omar and Abd El-Hafeez, 2024; Vergara and Estévez, 2014) and then employ a meta-heuristic algorithm like GA on the reduced number of data pixels. A similar approach has been used by Malakar et al. (2023) to select a feature subset before using PSO for diagnostic attribute selection from the datasets having a large number of attributes to select from.
- The reconstruction performance depends on kernel size as well as the kernel to be used. So in the future, generative artificial intelligence (GenAI) aided reconstruction process could be designed to have a better reconstruction from I'. Banerjee et al. (2024) recently used cGAN to tackle such an issue. However, its performance drops when training data is less. Therefore, one can plan to use the noise-like image (i.e., I') can be fed to GenAI techniques as a noise model instead of generating them from the Gaussian noise.

4. Conclusion

The present work is an initiative to store vision data in a more memory-efficient and shape-preserving way. We call this initiative a "compact image representation" of an image that differs from well-known image compression since no encoding mechanism is used here to store an image. Rather it tries to locate a near-optimal set of key pixels from where the original image could be reconstructed using some reconstruction technique. Here, we have used GA to select such a near-optimal set of key pixels while the Gaussian kernel-based convolution operator is used in the reconstruction process. The proposed technique is evaluated on four datasets: MNIST, CMATERdb, HeLa, and UiTMito. Performance is evaluated using metrics like F1-score, SSIM score, RMSE, APRR, FPRR, BRR, and classification accuracy. The average F1-score, SSIM score, RMSE, APRR, FPRR, and BRR lies in [.78, .88], [.81, .94], [.06, .08], [.01, .04], [.12, .21], [.02, .13] respectively for these datasets. These results infer that the proposed model not only reduces the memory requirements measured using APRR, FPRR, and BRR to store the images but also obtains good reconstruction similarity measured using F1-score, SSIM score, and RMSE. However, the reconstruction metrics show that the proposed image reconstruction process generates images that have good reconstruction loss. To investigate the impacts of the reconstructed images generated with the stated reconstruction loss on real-life applications, a classification task is performed using reconstructed images. These experiments show that the classification accuracy drops by 0.17%, .14%, and 2.30% on MNIST, CMATERdb, and HeLa datasets respectively. The performances of the proposed method are also comparable with deep learning-based methods: GU-Net, and GU-Net++. In short, being the work as first of its kind, the results are promising.

Though the proposed method performs satisfactorily there is some scope for improvement. First and foremost, the execution time for generating key pixels is needed to improve. Secondly, the F1-score (or shape similarity) for the datasets is close to 80% and the SSIM

score ranges between 82 to 94) on these datasets. So in the future, a better reconstruction algorithm is required along with a better selection process for key pixels. The proposed technique is designed for binary or gray-scale images. So the work can be extended to multi-channel images in the future. It is also noteworthy to mention that the datasets used here are not diverse enough to comment on the generalizability of the process. Therefore, in the future, the method could be applied to diverse data (like images from medical, satellite, and complex images) to investigate its generalizability. For real-life applications (here classification), we have reconstructed images from their compact representation. Therefore, in the future, the same may be tried from its compact representation.

CRediT authorship contribution statement

Samir Malakar: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Nirwan Banerjee:** Writing – review & editing, Writing – original draft, Visualization, Validation, Investigation, Data curation. **Dilip K. Prasad:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Conceptualization.

Declaration of competing interest

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

Acknowledgments

This work was supported by the Research Council of Norway Project (nanoAI, Project ID: 325741), H2020 Project (OrganVision, Project ID: 964800), and VirtualStain (UiT, Cristin Project ID: 2061348).

Data availability

The authors do not have permission to share data.

References

Acampora, G., Chiatto, A., Vitiello, A., 2023. Genetic algorithms as classical optimizer for the Quantum Approximate Optimization Algorithm. Appl. Soft Comput. 142, 110296.

Ali, A.A., El-Hafeez, T.A., Mohany, Y.K., 2019a. An accurate system for face detection and recognition. J. Adv. Math. Comput. Sci. 33 (3), 1–19.

Ali, A.A., El-Hafeez, T.A., Mohany, Y.K., 2019b. A robust and efficient system to detect human faces based on facial features. Asian J. Res. Comput. Sci. 2 (4), 1–12.

Banerjee, N., Malakar, S., Gupta, D.K., Horsch, A., Prasad, D.K., 2023. Guided U-Net aided efficient image data storing with shape preservation. In: Asian Conference on Pattern Recognition. Springer, pp. 317–330.

Banerjee, N., Malakar, S., Horsch, A., Prasad, D., 2024. GUNet++: Guided U-Net based compact image representation with improved. J. Opt. Soc. Amer. A 41 (10), 1–8.

Bhowmik, S., Malakar, S., Sarkar, R., Basu, S., Kundu, M., Nasipuri, M., 2019. Off-line Bangla handwritten word recognition: a holistic approach. Neural Comput. Appl. 31, 5783–5798.

Boland, M.V., Murphy, R.F., 2001. A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of hela cells. Bioinformatics 17 (12), 1213–1223.

Burrington, I., 2015. The environmental toll of a Netflix binge. Atlantic 16.

Canny, J., 1986. A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. (6), 679–698.

Das, S., Chatterjee, A., Dey, S., Saha, S., Malakar, S., 2022. Breast cancer detection from histology images using deep feature selection. In: Proceedings of International Conference on Frontiers in Computing and Systems: COMSYS 2021. Springer, pp. 323–330.

Dey, C., Bose, R., Ghosh, K.K., Malakar, S., Sarkar, R., 2022a. LAGOA: Learning automata based grasshopper optimization algorithm for feature selection in disease datasets. J. Ambient Intell. Humaniz. Comput. 1–20.

Dey, S., Roychoudhury, R., Malakar, S., Sarkar, R., 2022b. An optimized fuzzy ensemble of convolutional neural networks for detecting tuberculosis from Chest X-ray images. Appl. Soft Comput. 114, 108094.

Dey, S., Roychoudhury, R., Malakar, S., Sarkar, R., 2022c. Screening of breast cancer from thermogram images by edge detection aided deep transfer learning model. Multimedia Tools Appl. 81 (7), 9331–9349.

Dorigo, M., Birattari, M., Stutzle, T., 2006. Ant colony optimization. IEEE Comput. Intell. Mag. 1 (4), 28–39.

El-Sayed, M.A., Hafeez, T.A.-E., 2012. New edge detection technique based on the Shannon entropy in gray level images. arXiv preprint arXiv:1211.2502.

Eliwa, E.H.I., El Koshiry, A.M., Abd El-Hafeez, T., Farghaly, H.M., 2023. Utilizing convolutional neural networks to classify monkeypox skin lesions. Sci. Rep. 13 (1), 14495.

Eman, M., Mahmoud, T.M., Ibrahim, M.M., Abd El-Hafeez, T., 2023. Innovative hybrid approach for masked face recognition using pretrained mask detection and segmentation, robust PCA, and KNN classifier. Sensors 23 (15), 6727.

Ghosh, M., Malakar, S., Bhowmik, S., Sarkar, R., Nasipuri, M., 2017. Memetic algorithm based feature selection for handwritten city name recognition. In: Computational Intelligence, Communications, and Business Analytics: First International Conference, CICBA 2017, Kolkata, India, March 24–25, 2017, Revised Selected Papers, Part II. Springer, pp. 599–613.

Ghosh, M., Malakar, S., Bhowmik, S., Sarkar, R., Nasipuri, M., 2019. Feature selection for handwritten word recognition using memetic algorithm. Adv. Intell. Comput. 103–124.

Guo, Z., Hall, R.W., 1992. Fast fully parallel thinning algorithms. CVGIP: Image Underst. 55 (3), 317–328.

Guo, Q., Wang, Y., Yang, S., Xiang, Z., 2022. A method of blasted rock image segmentation based on improved watershed algorithm. Sci. Rep. 12 (1), 7143.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

Hoang, C.M., Kang, B., 2024. Pixel-level clustering network for unsupervised image segmentation. Eng. Appl. Artif. Intell. 127, 107327.

Holland, J.H., 1992. Genetic algorithms. Sci. Am. 267 (1), 66–73.

Hu, T.-H., 2015. A Prehistory of the Cloud. MIT Press.

Karimi, F., Dowlatshahi, M.B., Hashemi, A., 2023. SemiACO: A semi-supervised feature selection based on ant colony optimization. Expert Syst. Appl. 214, 119130.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks. Vol. 4, IEEE, pp. 1942–1948.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.-Y., et al., 2023. Segment anything. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4015–4026.

Ko, D.H., Hassan, A.U., Majeed, S., Choi, J., 2021. SkelGAN: A font image skeletonization method. J. Inf. Process. Syst. 17 (1), 1–13.

Koomey, J., Masanet, E., 2021. Does not compute: Avoiding pitfalls assessing the Internet's energy and carbon impacts. Joule 5 (7), 1625–1628.

Lee, T.-C., Kashyap, R.L., Chu, C.-N., 1994. Building skeleton models via 3-D medial surface axis thinning algorithms. CVGIP, Graph. Models Image Process. 56 (6), 462–478.

Malakar, S., Ghosh, M., Bhowmik, S., Sarkar, R., Nasipuri, M., 2020a. A GA based hierarchical feature selection approach for handwritten word recognition. Neural Comput. Appl. 32, 2533–2552.

Malakar, S., Ghosh, M., Chaterjee, A., Bhowmik, S., Sarkar, R., 2020b. Offline music symbol recognition using Daisy feature and quantum Grey wolf optimization based feature selection. Multimedia Tools Appl. 79, 32011–32036.

Malakar, S., Ghosh, P., Sarkar, R., Das, N., Basu, S., Nasipuri, M., 2011. An improved offline handwritten character segmentation algorithm for Bangla script. In: IICAI. pp. 71–90.

Malakar, S., Paul, S., Kundu, S., Bhowmik, S., Sarkar, R., Nasipuri, M., 2020c. Handwritten word recognition using lottery ticket hypothesis based pruned CNN model: a new benchmark on CMATERdb2. 1.2. Neural Comput. Appl. 32, 15209–15220.

Malakar, S., Sarkar, R., Basu, S., Kundu, M., Nasipuri, M., 2021. An image database of handwritten bangla words with automatic benchmarking facilities for character segmentation algorithms. Neural Comput. Appl. 33, 449–468.

Malakar, S., Sen, S., Romanov, S., Kaplun, D., Sarkar, R., 2023. Role of transfer functions in PSO to select diagnostic attributes for chronic disease prediction: An experimental study. J. King Saud Univ.-Comput. Inf. Sci. 35 (9), 101757.

Malakar, S., Sharma, P., Singh, P.K., Das, M., Sarkar, R., Nasipuri, M., 2017. A holistic approach for handwritten Hindi word recognition. Int. J. Comput. Vis. Image Process. (IJCVIP) 7 (1), 59–78.

Marmanis, D., Schindler, K., Wegner, J.D., Galliani, S., Datcu, M., Stilla, U., 2018. Classification with an edge: Improving semantic image segmentation with boundary detection. ISPRS J. Photogramm. Remote Sens. 135, 158–172.

Martin, D.R., Fowlkes, C.C., Malik, J., 2004. Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Trans. Pattern Anal. Mach. Intell. 26 (5), 530–549.

Mirjalili, S., 2015. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl.-based Syst. 89, 228–249.

Mirjalili, S., Lewis, A., 2013. S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evol. Comput. 9, 1–14.
Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61.
Mohiuddin, S., Sheikh, K.H., Malakar, S., Velásquez, J.D., Sarkar, R., 2023. A hierarchical feature selection strategy for deepfake video detection. Neural Comput. Appl. 35 (13), 9363–9380.
Monserrate, S.G., 2022. The cloud is material: on the environmental impacts of computation and data storage. MIT Case Stud. Soc. Ethical Responsib. Comput. http://dx.doi.org/10.21428/2c646de5.031d4553.
Mukhopadhyay, S., Hossain, S., Malakar, S., Cuevas, E., Sarkar, R., 2023. Image contrast improvement through a metaheuristic scheme. Soft Comput. 27 (18), 13657–13676.
Nadimi-Shahraki, M.H., Taghian, S., Mirjalili, S., Ewees, A.A., Abualigah, L., Abd Elaziz, M., 2021. MTV-MFO: Multi-trial vector-based Moth-flame Optimization Algorithm. Symmetry 13 (12), 2388.
Ng, H., Ong, S., Foong, K., Goh, P.-S., Nowinski, W., 2006. Medical image segmentation using k-means clustering and improved watershed algorithm. In: 2006 IEEE Southwest Symposium on Image Analysis and Interpretation. IEEE, pp. 61–65.
Niu, D., Wang, X., Han, X., Lian, L., Herzig, R., Darrell, T., 2024. Unsupervised universal image segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22744–22754.
Omar, A., Abd El-Hafeez, T., 2024. Optimizing epileptic seizure recognition performance with feature scaling and dropout layers. Neural Comput. Appl. 36 (6), 2835–2852.
Otsu, N., 1979. A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. 9 (1), 62–66.
Pan, J.-S., Hu, P., Chu, S.-C., 2021. Binary fish migration optimization for solving unit commitment. Energy 226, 120329.
Pan, J.-S., Hu, P., Snášel, V., Chu, S.-C., 2022. A survey on binary metaheuristic algorithms and their engineering applications. Artif. Intell. Rev. 1–67.
Pramanik, R., Dey, S., Malakar, S., Mirjalili, S., Sarkar, R., 2022. TOPSIS aided ensemble of CNN models for screening COVID-19 in chest X-ray images. Sci. Rep. 12 (1), 15409.
Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S., 2009. GSA: a gravitational search algorithm. Inf. Sci. 179 (13), 2232–2248.
Rashedi, E., Rashedi, E., Nezamabadi-Pour, H., 2018. A comprehensive survey on gravitational search algorithm. Swarm Evol. Comput. 41, 141–158.
Saabia, A.A.-B., El-Hafeez, T., Zaki, A.M., 2019. Face recognition based on grey wolf optimization for feature selection. In: Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2018 4. Springer, pp. 273–283.
Sarkar, S., Ghosh, M., Chatterjee, A., Malakar, S., Sarkar, R., 2019. An advanced particle swarm optimization based feature selection method for tri-script handwritten digit recognition. In: Computational Intelligence, Communications, and Business Analytics: Second International Conference, CICBA 2018, Kalyani, India, July 27–28, 2018, Revised Selected Papers, Part I 2. Springer, pp. 82–94.
Sekh, A.A., Opstad, I.S., Godtliebsen, G., Birgisdottir, Å.B., Ahluwalia, B.S., Agarwal, K., Prasad, D.K., 2021. Physics-based machine learning for subcellular segmentation in living cells. Nat. Mach. Intell. 3 (12), 1071–1080.
Spears, W.M., De Jong, K.D., 1995. On the Virtues of Parameterized Uniform Crossover. Technical Report, Naval Research Lab Washington DC.
Taha, M.E., Mostafa, T., El-Rahman, A., Abd El-Hafeez, T., 2023. A novel hybrid approach to masked face recognition using robust PCA and GOA optimizer. Sci. J. Damietta Fac. Sci. 13 (3), 25–35.
Torres-García, E., Pinto-Cámara, R., Linares, A., Martínez, D., Abonza, V., Brito-Alarcón, E., Calcines-Cruz, C., Valdés-Galindo, G., Torres, D., Jabloñski, M., et al., 2022. Extending resolution within a single imaging frame. Nature Commun. 13 (1), 7452.
Van Wynsberghe, A., 2021. Sustainable AI: AI for sustainability and the sustainability of AI. AI Ethics 1 (3), 213–218.
Vergara, J.R., Estévez, P.A., 2014. A review of feature selection methods based on mutual information. Neural Comput. Appl. 24, 175–186.
Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1 (1), 67–82.
Wu, C.-J., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Aga, F., Huang, J., Bai, C., et al., 2022. Sustainable ai: Environmental implications, challenges and opportunities. Proc. Mach. Learn. Syst. 4, 795–813.
Xu, Y., Chen, H., Luo, J., Zhang, Q., Jiao, S., Zhang, X., 2019. Enhanced Moth-flame optimizer with mutation strategy for global optimization. Inform. Sci. 492, 181–203.
Yan, L., Corinna, C., Burges, C., 1998. The MNIST dataset of handwritten digits.
Zhang, T.Y., Suen, C.Y., 1984. A fast parallel algorithm for thinning digital patterns. Commun. ACM 27 (3), 236–239.