Guided U-Net aided Efficient Image Data Storing with Shape Preservation

Nirwan Banerjee[1,*], Samir Malakar[1], Deepak Kumar Gupta[1], Alexander Horsch[1], and Dilip K. Prasad[1]

Bio-AI Lab, Department of Computer Science, UiT The Arctic University of Norway, Tromsø, 9037, Norway
{nirwan.banerjee, s.malakar, deepak.k.gupta, alexander.horsch, dilip.prasad}@uit.no
[*]Corresponding author nirwan.banerjee@uit.no

Abstract. The proliferation of high-content microscopes ($\sim$ 32 GB for a single image) and the increasing amount of image data generated daily have created a pressing need for compact storage solutions. Not only is the storage of such massive image data cumbersome, but it also requires a significant amount of storage and data bandwidth for transmission. To address this issue, we present a novel deep learning technique called Guided U-Net (GU-Net) that compresses images by training a U-Net architecture with a loss function that incorporates shape, budget, and skeleton losses. The trained model learns to selects key points in the image that need to be stored, rather than the entire image. Compact image representation is different from image compression because the former focuses on assigning importance to each pixel in an image and selecting the most important ones for storage whereas the latter encodes information of the entire image for more efficient storage. Experimental results on four datasets (CMATER, UiTMito, MNIST, and HeLA) show that GU-Net selects only a small percentage of pixels as key points (3%, 3%, 5%, and 22% on average, respectively), significantly reducing storage requirements while preserving essential image features. Thus, this approach offers a more efficient method of storing image data, with potential applications in a range of fields where large-scale imaging is a vital component of research and development.

Keywords: Compact Image Representation · Guided U-Net · Budget Loss · Shape Loss · Skeleton Loss · Storage Efficient

1 Introduction

Rapid advancement in digital technology has led to an exponential increase in the data generated daily. With the availability of the internet, social media, and smartphones, people are generating a considerable amount of digital content like texts, images, audio, and video. The sheer volume of data generated is quite large, with estimates suggesting that humans create and consume around 328.77 million terabytes of data daily and 120 zettabytes of data every year, with videos accounting for over half of internet traffic [1]. Even a single image from the domains like microscopy, nanoscopy, telescope, and satellite can generate a very large amount of data, similar in the orders of magnitude to the amount of data used by humans in a year. Specifically, the latest advancements in microscopy, like the CNI v2.0 microscope[15], can generate a single image of up to 32 gigabytes (Table 1). In contrast, multiple images in a z-stack or a time-series video can exceed a terabyte. Table 1 cites more such cases.

Table 1: Illustration of storage requirements for different microscopic data

Type of cell organelle	**Microscope**	**Image dimension**	**Size of image**
Mitochondria	Deconvolutional	2048×2048	8.38 MB
	OMX	1024×1024	2.09 MB
	Confocal	512×512	0.52 MB
	RCM scan	2048×2048	8.38 MB
Vesicles, Lysosomes	Deconvolutional	2048×2048	8.38 MB
	Confocal	100×100	0.02 MB
Membrane / Cytoskeleton for Actin or Microtubule	Deconvolutional	2048×2048	8.38 MB
	Epiflourescent	2048×2048	8.38 MB
Liver Tissue	CNI v2.0	10240×10240	$\sim$ 32000 MB

Storing and analyzing such large-sized data pose significant challenges to the scientific community, particularly in the field of biological sciences, where high-quality microscopy provides crucial information for potential breakthroughs in medical science. As we continue generating such voluminous data, a few of the concerns, but not limited to, that may arise in the future are as follows.

- **Storage issue:** One of the primary reasons for the inefficient storing of increased amounts of data is the requirement of expensive storage space. Traditional data storage methods, such as on-premise devices, can quickly become cost-prohibitive as data volumes increase. Thus, organizations may require to invest in buying storage from options like cloud storage, which can pose an extra cost to AI-driven products. Moreover, these increased numbers of data centers, in turn, convert these solutions into less sustainable and environmentally unfriendly ones.
- **Energy consumption:** Another issue with storing large amounts of data is the energy required. Data centers, which store massive amounts of data, are some of the largest energy consumers in the world. More data means more use of energy to store.
- **Data transmission cost:** The increased volume of data means an increased number of bits to transmit while sending data from one device to another. This scenario will consume more time and hence cost.

[1] https://explodingtopics.com/blog/data-generated-per-day

Overall, storing data in its original form is resource-intensive and ill impacts the environment. Thus, developing strategies for storing data using their compressed representation is vital to retrieve essential information only when required. By the term compressed representation of a datum (say, $\mathbb{D}$), we mean here storing it with less memory (say, $\mathbb{D}'$), i.e., $memoryUsage(\mathbb{D}') \ll memoryUsage(\mathbb{D})$. However, in the present work, we try to tackle this imminent problem in the premise of images. The work aims to represent images with fewer pixels while preserving their shape after kernel-based image reconstruction. Intuitively, to achieve the goal, we have tried to represent an input image (say, I with dimension $H \times W$) by selecting its key points. Let the method select m ($\ll H * W$) number of key points. The key points can be stored as x and y coordinates, costing $2*m$. We can construct a new image (say, O with dimension $H \times W$) by setting the key points, and then by employing kernel-based convolution, the I can be reconstructed. Lowering the value of m, we can achieve more storage efficiency.

In this paper, we propose a deep learning-based method for the compressed representation of an image. Please note that we have restricted ourselves to gray-scale images. Our method involves training a deep neural network (U-Net architecture[11]) to learn a compressed representation of images aiming to preserve the overall shape of the image. During model training, we apply point spread function (PSF) kernel-based image reconstruction from the set of selected points that are required to remember. The model is guided by three different losses, viz., budget, shape, and skeleton, and thus we call this model Guided U-Net (GU-Net). GU-Net can effectively reduce the storage requirement for an image to store.

2 Related Studies

The current work deals with representing image data in a compressed way while aiming to preserve their shapes. Some similar approaches are boundary-based representation, skeletonization, edge-based representation, and the like. Skeletonization is a popular approach for compact representation since it uses the least number of pixels. Hence, we discuss some state-of-the-art skeletonization approaches ranging from classical to deep learning.

2.1 Classical Approaches

Several conventional approaches exist for generating the skeleton of an image. Here, for simplicity, we discuss four well-known classical approaches. The thinning algorithms are one of the most popular approaches for skeletonization. Thinning refers to removing pixels from a component's boundary in an image until obtaining single pixel width. Some popular thinning methods proposed by Zhang et al. [17], Guo et al. [7], and Rosenfeld [12] relied on a set of rules to remove pixels to form the skeleton iteratively. However, thinning algorithms are sensitive to noise, and they may produce multiple skeletons or break the connectivity of the object in the image. The Medial Axis Transform (MAT) [6] is another popular approach for skeletonization. The MAT is a mathematical representation of the central axis of an object. It is obtained by estimating the Voronoi diagram of the object boundary. The MAT has the advantage of preserving the connectivity of the object in the image and can handle noise well. However, generating the Voronoi diagram can be computationally expensive, especially for large-sized microscopy and nanoscopy images/videos. Morphological operations, such as erosion and dilation, are primarily used in morphological skeletonization. The morphological skeletonization algorithm involves iterating a set of morphological operations until a skeleton is obtained. Methods following this approach have the advantage of being implementation friendly and can handle noise to some extent. However, the resulting skeleton may be thicker than the skeleton obtained by other methods. Distance transform that computes the distance of each pixel in an object from the object boundary is also used to obtain the skeleton of an object by finding the points where the distance function is maximized. Distance transform-based skeletonization has the advantage of being fast and efficient. However, it may not preserve the connectivity of the object in the image and is susceptible to noise.

2.2 Deep Learning Approaches

Recently researchers have been focusing on designing deep-learning methods to extract the skeleton from an image. It treats the problem as either a pixel-to-pixel classification or an image-to-image translation. In both cases, a deep learning-aided segmentation protocol is used. A few deep learning-based methods dealing mentioned problem are discussed here. DeepSkeleton [14] proposes a multi-task learning framework that simultaneously learns the object skeleton and the object's scale at each pixel. The method is based on holistic edge detection that produces a set of side outputs used to refine the object skeleton at different scales. This framework consists of two key components: a Scale-associated Deep Side Output (SDSO) and a multi-task loss function. The SDSO module uses a series of convolutional layers to extract features at different scales from the input image. The multi-task loss function combines a skeletonization loss and a scale estimation loss, which jointly optimize the network to produce high-quality skeletons at multiple scales. It can be used on natural images. PSPU-SkelNet [2] uses three U-Net architectures for extracting point clouds from a given shape point cloud. The authors also introduce a novel loss function called the Symmetric Chamfer Distance (SCD) loss, which considers the extracted skeleton's accuracy and completeness. The SCD loss is defined as the average distance between each point on the Ground Truth (GT) skeleton and its nearest point on the predicted skeleton, and vice versa. SkelGAN [9] tries font skeletonization using a modified U-Net structure and a PatchGAN discriminator. The authors also proposed a novel loss function called the skeleton consistency loss, which encourages the generator to produce skeletons that match the structure of the input font image and have consistent topology and connectivity. In work [1], the authors used MAT to generate the skeleton from a binary image. This algorithm first computes the MAT of the object, and then the skeleton is obtained by pruning the MAT based on a set of criteria, such as the degree of curvature and the distance of points from the boundary of the object.

The aforementioned approaches focus primarily on skeleton generation from natural or binary images. It is noteworthy to mention that our focus is not on generating the skeleton of an image but on obtaining a compact representation of an image from where we can regenerate the image using some simple but effective convolutional operator. Thus, our representation might differ visually from the actual skeleton but the regenerated images look similar to the actual image. This discussion is clear from the images shown in Fig. 1. In this figure, we showcase the problem associated with skeleton-based (see Fig. 1) or edge-based (see Fig. 1) reconstruction when employed on an epifluorescent mitochondria images. This figure also contains output from the current method (see Fig. 1).

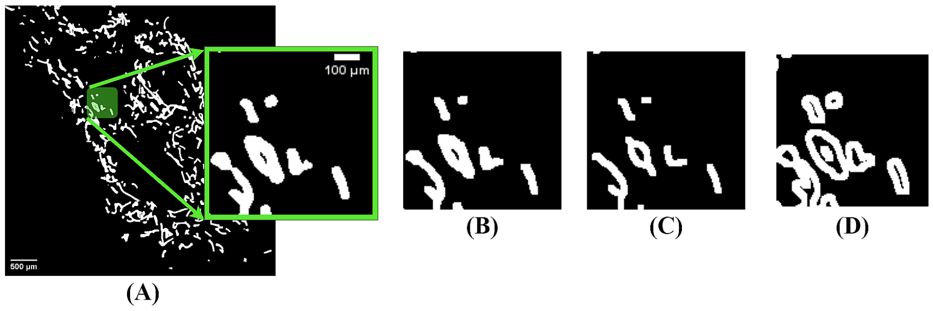
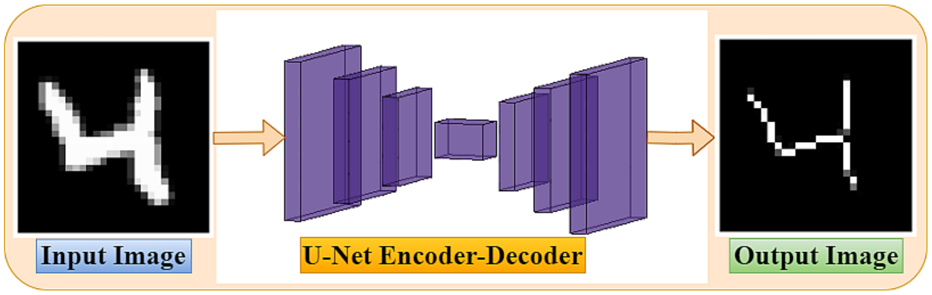


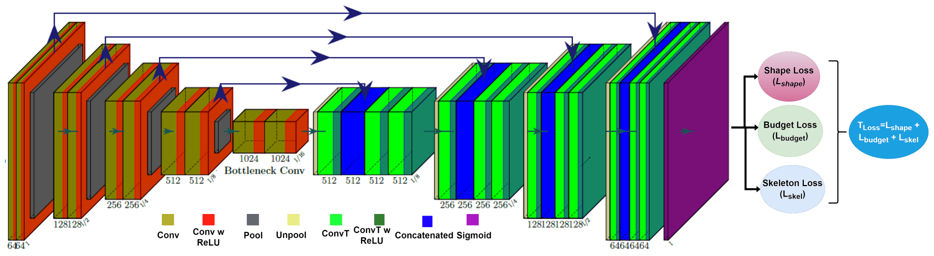
Fig. 1: Visual comparison of (A) original image taken from UiTMito [13] against reconstructed images obtained by employing PSF-based reconstruction from the compact representations generated using (B) our method (SSIM score= 0.9766) (C) skeletonization [7] (SSIM score = 0.9516) (D) edge detection [5] (SSIM score = 0.9040)

3 GU-Net: Detailed Description

In this section, we introduce GU-Net, a semi-supervised deep learning model used for shape-preserving compact representation of 2D data. We consider 2D data as a set of points in Euclidean space, and GU-Net suppresses data points while trying to maintain the original shape. A crucial aspect of our method is the ability to reconstruct the original data from its compact version. We employ our reconstruction technique (see section 3.3). We use the U-Net architecture as a base segmentation network to segment an image into a set of points. However, we incorporate three loss functions to constrain the segmentation process: the skeleton loss encourages the model to select points near the skeleton of an input object; the shape loss guides the model to generate images with persistent shapes; and the budget loss controls the number of selected points, which the user specifies. Fig. 2b shows the overall architecture of the proposed method.



(a) Overall working procedure of the proposed method



(b) Used U-Net architecture with associated loss functions

Fig. 2: Proposed GU-Net architecture that accepts an image and generates its compact representation

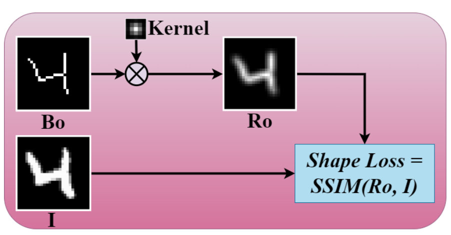
3.1 Base Segmentation Network

The segmentation network used in GU-Net shares many similarities with the U-Net architecture [11]. Specifically, the up and down sampling components remain unchanged, while modifications have been made to the input feeding mechanism and the loss function. The selection of the U-Net model for this problem is based on its resilience in diverse applications that require accurate segmentation of images, such as biomedical image analysis. The strength of this architecture lies in its ability to generate high-resolution output images while maintaining the spatial information of the input image. This feature makes it particularly useful in object detection and recognition tasks, where preserving object boundaries is crucial. Moreover, the U-Net architecture can handle various input sizes and is computationally efficient, making it suitable for real-time applications.

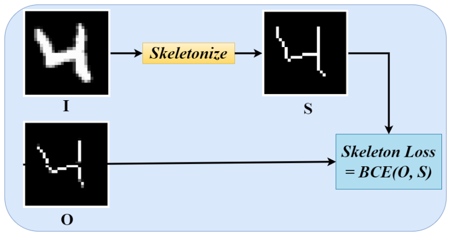
During model design, instead of providing binary masks as labels to the U-Net architecture, we provide the skeleton of the input images generated by the method proposed by Zang et al. [17] as segmentation GT for the network. Such a setting encourages the network to select key points near the skeleton of the input image. One can feed the edges to the network as GT images, but such a setting will lead to selecting more key points. In addition to this, three loss functions (discussed in subsection 3.2) have been designed here to guide the network to select better key points.

3.2 Loss Functions

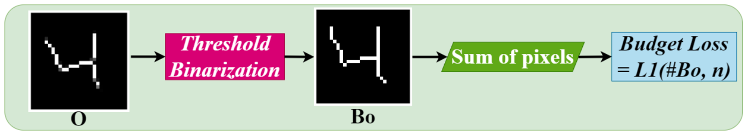
It has already been mentioned that GU-Net is guided by three different loss functions: skeleton loss, budget loss, and shape loss. Here, we discuss these loss functions.



(a) Shape loss



(b) Skeleton loss



(c) Budget loss

Fig. 3: Computation of different loss functions used in this work

Given an image I, output image O is generated by a segmentation network M (here GU-Net) and is represented by Eq. 1.

$$M(I) = O(\sim I) \tag{1}$$

Skeleton Loss: The skeleton loss guides the GU-Net to choose key points from the neighbour of the skeleton. The skeleton of the training images is provided to the GU-Net as a segmentation GT. Given an input image I, the skeleton S is calculated using Zhang's algorithm. The BCE loss is then calculated between O and S (see Eq. 2).

$$L_{skel}(O, S) = -W * [S \cdot log(O) + (1 - S) \cdot log(1 - O)] \tag{2}$$

Budget Loss: The budget loss works towards minimizing the number of selected key points from I. The number of selected pixels in O is calculated by binarizing O using a global threshold value. Given threshold t, binarized version B_o of O is calculated using Eq. 3.

$$B_o(x,y) = \begin{cases} 0, & \text{if } O(x,y) \leq t \\ 1, & \text{otherwise} \end{cases} \tag{3}$$

During model training, $t = 0.3$ is set. Thus, $n = \sum\sum B_o(x,y)$ is the number of selected key points in O. Next, we calculate the $L1$ norm (see Eq. 4) between n and a small integer n', user input representing the selected key points.

$$\sum B_o = L_{budget} = |n - n'| \tag{4}$$

Shape Loss: The shape loss is an essential component of our method that aims to preserve the shape of the input image when reconstructed. This loss serves as a counterbalance to the budget loss and helps create a compact representation resembling the input image. We use a predefined convolutional filter to generate O to achieve this. For this purpose, B_o is convolved by a 5×5 kernel to create the reconstructed version (say, R_o). By spreading the effect of a pixel to its neighboring area, we ensure that the semblance of the original image is maintained. Next, we then calculate the Structural Similarity Index Measure (SSIM) based loss between R_o and I (see Eq. 5) to ensure that the restored image attains a close resemblance to I.

$$SSIM(R_o, I) = L_{shape} = \frac{(2 \cdot \mu_{R_o}\mu_I + c_1)\cdot(2 \cdot \sigma_{R_oI} + c_2)}{(\mu_{R_o}^2 + \mu_I^2 + c_1)(\sigma_{R_o}^2 + \sigma_I^2 + c_2)} \tag{5}$$

In Eq. 5, μ_{R_o}, and μ_I represent the pixel sample mean of R_o, and I respectively while standard deviations of all pixel intensities present in R_o, and I are represented by σ_{R_o}, and σ_I respectively. Also, σ_{R_oI} is the covariance between R_o and I. $c_1 = k_1L^2, c_2 = k_2L^2$ 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 Method of Reconstruction

To reconstruct the actual image from compact representations generated by GU-Net, we use a simple but efficient PSF kernel. We convolve this kernel with B_o to generate the R_o. However, we use two more kernels: the Gaussian kernel, and the Mean kernel to test the effectiveness of our choice i.e., the PSF kernel. A comparative result is shown in Fig. 4, which visually ensures the effectiveness of the use of the PSF kernel in the reconstruction process.

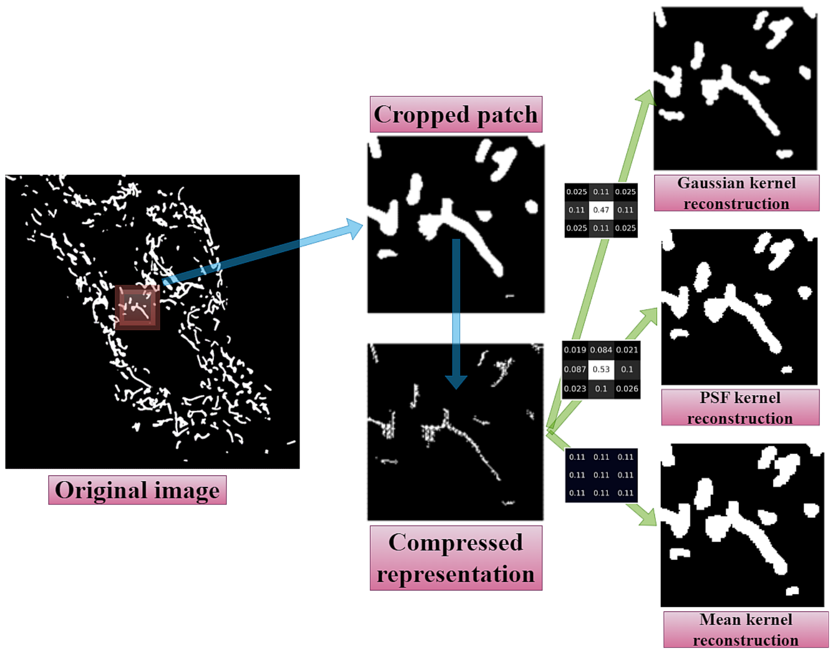


Fig. 4: Illustration of reconstruction process with varying kernels applied on a sample image taken from the UiTMito [13] dataset

4 Result and Discussion

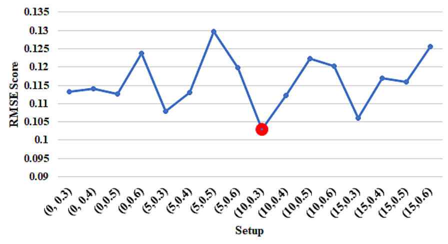
In this work, we have designed GU-Net, which tries to generate the compressed representation of an input image. We have evaluated its performance on four diversified datasets to test its effectiveness. Notably, GU-Net is trained only on one dataset, and the trained module generates the compressed representation on other datasets. We have provided qualitative as well as quantitative performance. Additionally, we have performed some classification tasks to test how the reconstructed images behave.

4.1 Database Description

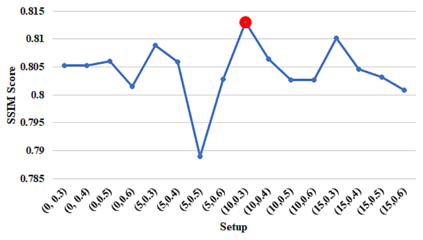
The datasets in use are described here. Two handwritten (digit and word images) and two fluorescence microscopy datasets are considered for testing GU-Net's applicability almost on contrasting domains. **MNIST** [16] is a well-known dataset of handwritten digits widely used to benchmark different classification problems. The dataset consists of 70,000 images of handwritten digits, each of which is grayscale and has a resolution of 28×28 pixels. The images are labeled with their corresponding digit. **CMATERdb2.1.2** [3] (CMATER) is a handwritten Bangla word recognition dataset. It contains handwritten words representing 120 popular city names of the state of West Bengal, India. Each city name has 150 different handwritten samples. In short, this database is used for 120 class classification problems, primarily used for holistic word recognition purposes [10]. The **UitMito** dataset [13] is a collection 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 seconds. The dataset is split into a training set of 800 images and a test set of 200 images. The **2D HeLa** [4] (HeLa)dataset consists of fluorescence microscopy images of HeLa cells that have been stained with organelle-specific fluorescent dyes. The dataset includes images of 10 organelles, including DNA (nuclei), endoplasmic reticulum (ER), cis/medial Golgi (Giantin), cis Golgi (GPP130), lysosomes (Lamp2), mitochondria, nucleoli (Nucleolin), actin, endosomes (TfR), and tubulin.

4.2 Model Training

For training the model, we use the MNIST dataset. The model is trained on the MNIST dataset. The training dataset was divided into the training and the validation set. The training set consists of 50,000 images, and the validation dataset consists of 10,000 images. The images are of size 28 × 28. The Adam optimizer is used with a learning rate of 0.001. A batch size of 512 is used for a total of 10 epochs. Here we present experimental results for the hyperparameters used in the model training as shown in Fig. 5. We plot the RMSE and SSIM scores corresponding to the hyperparametric setup: the pixel budget and the pixel intensity threshold. As evident from Fig. 5a, we get the least RMSE at the setup point (10, 0.3), and we also get the highest SSIM score at point (10, 0.3) as shown in Fig. 5b.



(a)



(b)

Fig. 5: Performance of the GU-Net in terms of (a) RMSE, and (b) SSIM scores with varying hyperparameters marked as (pixel budget, pixel intensity threshold)

4.3 Results

We assess the performance of GU-Net both qualitatively and quantitatively. We show some reconstructed images in Fig. 6. The figure shows that the proposed model can generate very close to the actual image while having less number of key points that are needed to store the images for future use (see Fig. 4).

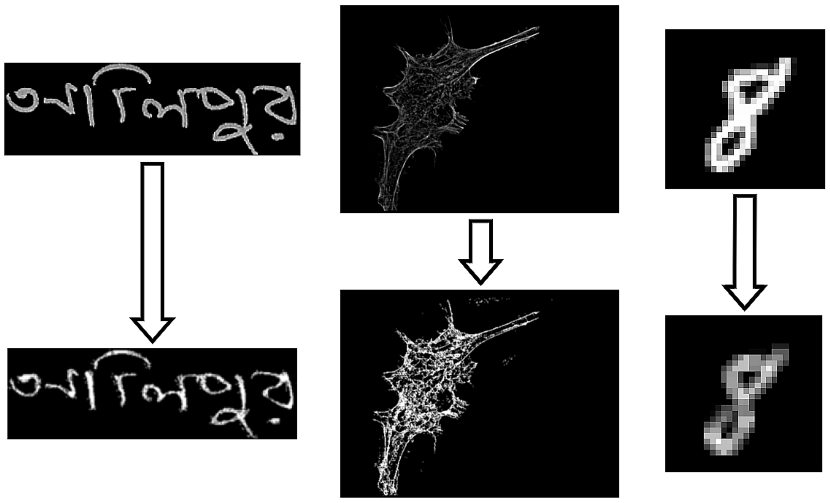


Fig. 6: Original (top) and reconstructed (bottom) images of the CMATER, HeLA, and MNIST datasets (left to right)

For quantitative analysis, we use two different evaluation strategies. We perform image-to-image comparisons and classification performance on reconstructed images. We use root mean squared error (RMSE) and SSIM metrics for image-to-image comparison. The results are shown in Table 2a. This table shows the GU-Net's performance with varying kernels used during the reconstruction process on the top of selected key points. The threshold values are chosen experimentally (results shown in Fig. 5). Using these threshold values, significant compression is achieved as shown in Fig. 7a while still maintaining high reconstruction similarity scores (see Table 2a). The total number of pixels in the photos, expressed in bytes, was used to compute the storage needed for the original photographs. The total number of pixels for the entire dataset was then determined. We only record the key points' coordinates in bytes for the compressed picture representations. The full dataset is gone through this process again. The dataset's photos' dimensions are stored in an additional 2 bytes. These results ensure that the reconstructed images retain the crucial features of the original images and the overall quality of the reconstructed images is closely maintained.

We also use classification accuracy to test the quality of the reconstructed images. Our goal does not involve generating the best scores for any particular dataset. Instead we test how the classification performance is affected by the proposed compressed representation of the images. We use a pre-trained EfficientNet

Table 2: Quantitative comparisons of different compact representation methods

(a) GU-Net with varying kernels. Here GU-Net selects keypoints and the mentioned kernels are used in the reconstruction process. $Th.$ represents the threshold value used for selecting keypoints in GU-Net. ↑, and ↓ represent larger, and smaller values mean better result respectively.

Dataset	**Th.**	**Kernel**	**SSIM (↑)**	**RMSE (↓)**
MNIST [16]	0.5	PSF	0.7943	0.0628
		Gaussian	**0.8397**	**0.0621**
		Mean	0.8041	0.0742
UiTMito [13]	0.005	PSF	**0.9526**	**0.0256**
		Gaussian	0.9431	0.0392
		Mean	0.9448	0.0364
CMATER [3]	0.5	PSF	**0.8344**	**0.0328**
		Gaussian	0.8267	0.0384
		Mean	0.8139	0.0455
HeLA [4]	0.05	PSF	**0.8885**	**0.0654**
		Gaussian	0.8592	0.0704
		Mean	0.8517	0.0744

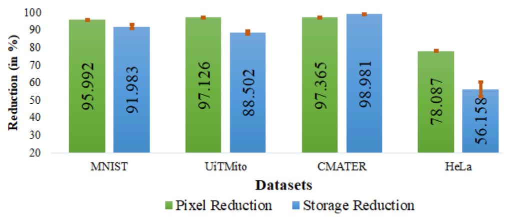
(b) GU-Net with other compact representation techniques

Dataset	**Method**	**SSIM (↑)**	**RMSE (↓)**
MNIST	GU-Net	**0.8397**	**0.0621**
	Skeletonization	0.7926	0.1063
	PSPU-SkelNet	0.7470	0.1501
	Canny Edge	0.7922	0.1104
UiTMito	GU-Net	**0.9526**	**0.0256**
	Skeletonization	0.9393	0.0710
	PSPU-SkelNet	0.9407	0.0705
	Canny Edge	0.9388	0.0710
CMATER	GU-Net	**0.8344**	**0.0328**
	Skeletonization	0.7918	0.0648
	PSPU-SkelNet	0.7974	0.0775
	Canny Edge	0.7916	0.0783
HeLA	GU-Net	**0.8885**	**0.0654**
	Skeletonization	0.8337	0.1287
	PSPU-SkelNet	0.8873	0.0916
	Canny Edge	0.8034	0.1499

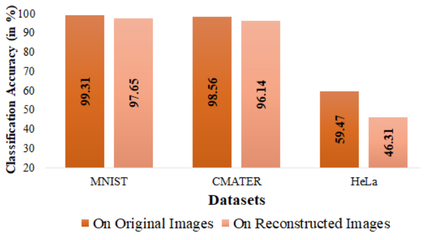
B0 [8] model for the classification tasks. The classification task is performed using both original and reconstructed images. This experiment was conducted on MNIST, CMATER, and HeLa datasets, and the results are presented in Fig. 7b. The classification accuracy obtained on reconstructed MNIST, CMATER, and UiTMito images has dropped by 1.66%, 2.41% , and 13.16% , respectively. From Figs. 7a and 7b, it can be observed that in the case of HeLa dataset, the model demonstrated relatively poorer performance compared to the other datasets. The reason is that this dataset's biological structures contain many scattered points, leaving little room for ample reduction. Further, there is much bleeding of labeling fluid around the cell organelles, leading to a less efficient compact representation. Overall, our approach demonstrates promising results across all four datasets, showcasing the effectiveness of our discussed method for data compression, visualization, and preservation tasks.

4.4 Comparison with Other Compact Image Representation

To test the effectiveness of GU-Net aided image compact representation concerning some existing image representation techniques, namely Skeletonization [7], PSPU-SkelNet [2], and Canny edge [5], we choose the reconstruction kernel that performs the best (see Table 2a) during the reconstruction process. The performances of other existing methods and GU-Net are shown in Table 2b. Our method provides better SSIM and RMSE scores than traditional (i.e., Skeletonization and Canny Edge) and deep-learning-based (i.e., PSPU-SkelNet)



(a) Pixel and storage reduction



(b) Classification performances

Fig. 7: The comparison between compact representation and classification performances. The UiTMito dataset is not applicable for classification tasks.

methods. Moreover, when considering the visual quality of the reconstructed images (see Fig. 1), a noticeable distinction further supports our approach's efficacy.

5 Conclusion

In the present work, we develop GU-Net that selects key points to compactly store an image. GU-Net uses budget, shape, and skeleton losses while using U-Net architecture in the backbone. The effectiveness of the compact representation of images using GU-Net has been evaluated on four datasets: MNIST, CMATER, UiTMito, and HeLa. The visual and quantitative findings are promising. Despite the success of GU-Net, there is still room for improvement. It has already been observed that GU-Net fails to achieve similar results compared to others due to the presence of scattered points. Therefore, fruitful techniques, at least an effective reconstruction process, need to be devised for images with ample scattered points like HeLa in the future. GU-Net could be applied to more data to test its generalization capabilities. Finally, the work can be extended to 3-channel images in the future.

Acknowledgments. This work was supported by the nanoAI Research Council of Norway Project No. 325741, H2020 Project(OrganVision) 964800, and VirtualStain(UiT) Cristin Project ID: 2061348.

References

1. Abu-Ain, W., Abdullah, S.N.H.S., Bataineh, B., Abu-Ain, T., Omar, K.: Skeletonization algorithm for binary images. Procedia Technology **11**, 704–709 (2013)
2. Atienza, R.: Pyramid U-network for skeleton extraction from shape points. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
3. Bhowmik, S., Malakar, S., Sarkar, R., Basu, S., Kundu, M., Nasipuri, M.: Off-line Bangla handwritten word recognition: a holistic approach. Neural Computing and Applications **31**, 5783–5798 (2019)

4. Boland, M.V., Murphy, R.F.: 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 (2001)
5. Canny, J.: A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence (6), 679–698 (1986)
6. Castleman, K.R.: Digital image processing. Prentice Hall Press (1996)
7. Guo, Z., Hall, R.W.: Fast fully parallel thinning algorithms. CVGIP: image understanding **55**(3), 317–328 (1992)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
9. Ko, D.H., Hassan, A.U., Majeed, S., Choi, J.: SkelGAN: A font image skeletonization method. Journal of Information Processing Systems **17**(1), 1–13 (2021)
10. Malakar, S., Sharma, P., Singh, P.K., Das, M., Sarkar, R., Nasipuri, M.: A holistic approach for handwritten Hindi word recognition. International Journal of Computer Vision and Image Processing (IJCVIP) **7**(1), 59–78 (2017)
11. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015)
12. Rosenfeld, A.: A characterization of parallel thinning algorithms. Information and control **29**(3), 286–291 (1975)
13. Sekh, A.A., Opstad, I.S., Godtliebsen, G., Birgisdottir, Å.B., Ahluwalia, B.S., Agarwal, K., Prasad, D.K.: Physics-based machine learning for subcellular segmentation in living cells. Nature Machine Intelligence **3**(12), 1071–1080 (2021)
14. Shen, W., Zhao, K., Jiang, Y., Wang, Y., Bai, X., Yuille, A.: Deepskeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images. IEEE Transactions on Image Processing **26**(11), 5298–5311 (2017)
15. Villegas-Hernández, L.E., Dubey, V., Nystad, M., Tinguely, J.C., Coucheron, D.A., Dullo, F.T., Priyadarshi, A., Acuña, S., Ahmad, A., Mateos, J.M., et al.: Chip-based multimodal super-resolution microscopy for histological investigations of cryopreserved tissue sections. Light: Science & Applications **11**(1), 43 (2022)
16. Yan, L., Corinna, C., Burges, C.: The MNIST dataset of handwritten digits (1998)
17. Zhang, T.Y., Suen, C.Y.: A fast parallel algorithm for thinning digital patterns. Communications of the ACM **27**(3), 236–239 (1984)