Electronics and Information Technologies, 2026, 34, 95–112
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC: 004.93

EVOLUTION OF IMAGE SEGMENTATION METHODS: CLASSICAL ALGORITHMS AND DEEP LEARNING

Viktor Vdovychenko
Ivan Franko National University of Lviv,
50 Drahomanova St., 79005 Lviv, Ukraine

Vdovychenko V. M. (2026). Evolution of Image Segmentation Methods: Classical Algorithms and Deep Learning. *Electronics and Information Technologies*, 33, 95–112. https://doi.org/10.30970/eli.34.6.

ABSTRACT

Background. Text image binarization is an important preprocessing task in document analysis, optical character recognition, and automated information extraction. The task becomes challenging when images contain blur, noise, low contrast, or background degradation.

Materials and Methods. This study compares classical and lightweight neural methods for binarizing degraded text images. The evaluation was conducted on a paired text binarization dataset comprising degraded input images and their corresponding clean binary target masks. The evaluated methods included simple global thresholding, Otsu binarization, adaptive thresholding, K-means clustering, Fuzzy C-means clustering, watershed segmentation, region growing, region splitting and merging, Canny edge detection, Laplacian of Gaussian filtering, and a U-Net model. Tunable parameters were selected on the validation subset, while final results were reported on the independent test subset. The methods were assessed using Intersection over Union (IoU, Jaccard index), Dice similarity coefficient (DSC), Pixel Accuracy (PA), Precision, Recall, and execution time.

Results and Discussion. Adaptive thresholding achieved the best performance among classical methods: IoU = 0.785, DSC = 0.877, PA = 0.961, Precision = 0.851, and Recall = 0.923. Simple global thresholding was the fastest method, with a runtime of 0.0015 ms/image, but lower Precision. K-means, Fuzzy C-means, Otsu, and region growing demonstrated high Recall values above 0.97, but lower Precision, indicating that degraded background regions were often classified as foreground. Edge-based methods produced lower IoU and DSC because they mainly detected character contours. The Tiny U-Net model achieved the highest overall quality: IoU = 0.959, DSC = 0.979, PA = 0.994, Precision = 0.979, and Recall = 0.979.

Conclusion. The results show that different binarization methods are suitable for different scenarios, depending on the required accuracy, runtime constraints, and available resources. Classical methods remain useful for low-latency processing, while lightweight neural models provide higher accuracy when sufficient computational resources are available. Pixel Accuracy should be interpreted carefully because background pixels dominate document images.

Keywords: text binarization, image segmentation, thresholding, clustering, U-Net, evaluation metrics.

INTRODUCTION

Image segmentation is one of the fundamental tasks in computer vision and digital image processing. It aims to divide an image into meaningful regions or classes and is

widely used as a preprocessing or decision-support stage in many computer vision systems. The quality of segmentation directly affects the reliability of subsequent tasks such as object recognition, measurement, classification, and visual analysis. Despite significant progress in this field, accurate and computationally efficient segmentation remains challenging, especially under conditions of noise, blur, uneven illumination, and low contrast.

One practically important segmentation problem is text image binarization. In this task, text image is transformed into a binary representation, where text pixels are separated from the background. Text binarization is an essential preprocessing step for optical character recognition, document analysis, historical document restoration, and automated information extraction. Although the task may appear simple in high-contrast images, its complexity increases significantly when the input contains degradation artifacts such as blur, background noise, low contrast, or non-uniform illumination [1-2].

Over the decades, many classical methods have been proposed for image segmentation and binarization. Threshold-based approaches, such as simple global thresholding [3], Otsu binarization [4], and adaptive thresholding [5], remain widely used because of their simplicity and low computational cost. Clustering-based methods, including K-means [6] and Fuzzy C-means [7], separate pixels according to intensity similarity and can be applied without supervised training. Region-based methods, such as watershed segmentation [8], region growing [9], and region splitting and merging [10], use spatial or structural information to form image regions. Edge-based methods, including Canny edge detection [11] and Laplacian of Gaussian filtering [12], focus on detecting intensity transitions and object boundaries.

Each of these methods has its own advantages and limitations. Global thresholding methods are fast but sensitive to illumination and contrast variation. Adaptive thresholding can handle local intensity changes but strongly depends on the choice of block size and threshold offset. Clustering-based methods may detect most text pixels but can also assign noisy background regions to the foreground when intensity distributions overlap. Region-based methods can incorporate spatial structure but are sensitive to seed selection, region homogeneity criteria, or marker generation. Edge-based methods are effective for detecting character contours but do not naturally produce filled foreground masks. Therefore, the choice of a segmentation method depends not only on accuracy but also on robustness, parameter sensitivity, and computational cost.

Recent deep learning methods have significantly improved segmentation performance in many computer vision tasks and have become a dominant direction in modern segmentation research [13]. However, neural models require training data, computational resources, and careful validation. For small text images, lightweight architectures may be more appropriate than deep backbone-based models because excessive downsampling can remove fine character details. Therefore, compact encoder-decoder architectures such as lightweight U-Net [14] models are of particular interest for text binarization.

A common limitation of comparative studies is that segmentation methods are often evaluated under limited experimental conditions or on a small number of examples. In such cases, conclusions about the superiority of a method may be strongly influenced by image characteristics, parameter choices, or class imbalance. In document binarization, this is especially important because background pixels usually dominate the image area, making Pixel Accuracy (PA) [15] less informative than foreground-oriented metrics such as IoU [16], Dice similarity coefficient (DSC) [17], Precision, and Recall.

This study presents a comparative experimental evaluation of classical segmentation algorithms and a lightweight neural baseline for text image binarization. The evaluated methods include simple global thresholding, Otsu binarization, adaptive thresholding, K-means clustering, Fuzzy C-means clustering, watershed segmentation, region growing, region splitting and merging, Canny edge detection, Laplacian of Gaussian filtering, and a compact U-Net model. The experiments are conducted on a paired text binarization dataset

containing degraded input images and corresponding clean binary target masks. To ensure a fair comparison, tunable parameters are selected using a validation subset, while final results are reported on an independent test subset.

The main contribution of this study is a unified benchmark of classical and lightweight neural methods for degraded text binarization. The comparison considers not only segmentation accuracy but also parameter sensitivity, runtime efficiency, and differences between metrics. This allows a more detailed analysis of why methods with similar Pixel Accuracy may differ substantially in foreground segmentation quality and practical applicability.

MATERIALS AND METHODS

This study evaluates classical and lightweight neural methods for text image binarization, formulated as a binary segmentation task where text pixels represent the foreground class and background pixels represent the background class. The experimental protocol includes dataset preparation, validation-based parameter selection, final evaluation on an independent test subset, and runtime analysis.

Dataset

The experiments were conducted using the Text Binarization dataset available on Kaggle [18]. The dataset contains 20,000 paired examples of randomly generated multiline text images, where each degraded input image is matched with a clean binarized target mask. The input images contain blur and noise, making the dataset suitable for evaluating binarization methods under degraded image conditions. An example of the input images is shown in **Fig. 1**.

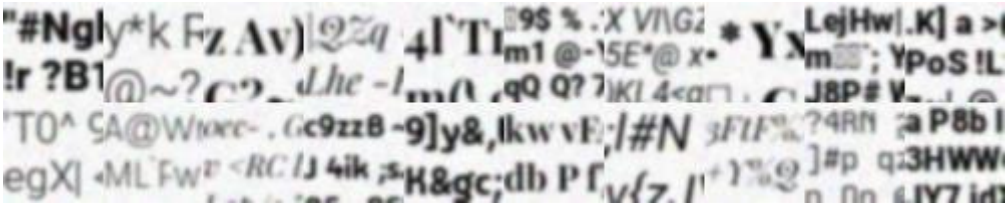

Fig. 1. Example of the dataset images

All images were processed in grayscale, and the provided binary target images were used as ground-truth masks. The dataset was divided into training, validation, and test subsets using a 70/15/15 split. The training subset was used only for the neural network model, while the validation subset was used for parameter selection in classical methods and threshold selection for the neural baseline. Final results were reported only on the independent test subset to avoid test-set-based parameter tuning.

Evaluated Methods

The study evaluated classical and lightweight neural approaches for degraded text image binarization. The classical methods included simple global thresholding, Otsu binarization, adaptive thresholding, K-means clustering, Fuzzy C-means clustering, watershed segmentation, region growing, region splitting and merging, Canny edge detection, and Laplacian of Gaussian filtering. These methods cover threshold-based, clustering-based, region-based, and edge-based segmentation strategies. [3-12]

A compact two-level U-Net [14] was additionally trained as a neural baseline. This architecture was selected because the input images have a small spatial resolution of 50×50 pixels, where deeper backbone-based models may lose fine text details due to excessive downsampling. The model consists of two encoder blocks, a bottleneck, and two decoder blocks with skip connections, producing a single-channel probability map. The

architecture of the model is shown in **Fig. 2**. It was trained using a combined binary cross-entropy and Dice loss, and the best checkpoint was selected according to validation IoU.

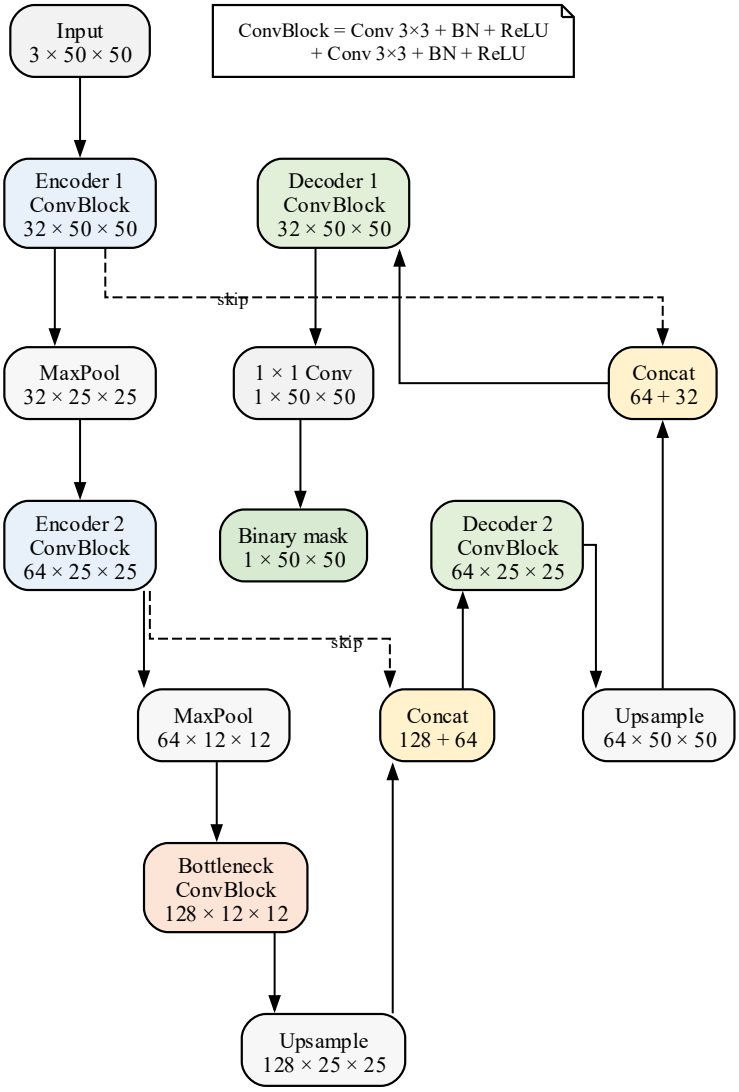

Fig. 2. Visualization of model architecture

Parameter Selection Protocol

All tunable parameters were selected on the validation subset using a predefined parameter grid. For each method, the configuration with the highest mean IoU was selected and then fixed for final evaluation on the independent test subset. Otsu binarization was the only exception, since its threshold was estimated automatically for each image. For the neural baseline, the probability threshold used to convert the output probability map into a binary mask was also selected on the validation subset.

Evaluation Metrics

IoU and Dice were used as the main foreground segmentation metrics because they directly measure the overlap between predicted and ground-truth text regions. Precision and Recall were used to analyze over-segmentation and under-segmentation behavior.

Pixel Accuracy was reported as an auxiliary metric, since background pixels dominate document images and may increase this metric even when the foreground mask is not accurately segmented.

Runtime Measurement

Computational efficiency was evaluated as execution time in milliseconds per image. For classical algorithms, runtime included only image processing time and excluded file loading and result saving. For the neural network, inference time included the forward pass of the model. Final runtime values are reported as mean milliseconds per image.

Experimental Environment

All experiments were implemented in Python 3.9. Classical methods were implemented using OpenCV and NumPy, while the neural network model was implemented using PyTorch. The experiments were conducted on a workstation equipped with an Apple M1 CPU and 8 GB of RAM, running macOS 26.4.1.

RESULTS AND DISCUSSION

Simple Global Thresholding

The global threshold value was selected on the validation subset, with the best result obtained at T = 165. As shown in **Fig. 3**, increasing the threshold improved text coverage up to an optimal range, after which additional background pixels started to be classified as foreground. On the test subset, the method achieved IoU = 0.661, DSC = 0.784, PA = 0.933, Precision = 0.751, and Recall = 0.861. The high Recall indicates that most text pixels were detected, while the lower Precision shows that some background degradation was also included in the foreground mask. The visual examples in **Fig. 4** and **Fig. 5** confirm this behavior. The method was extremely fast, with an average runtime of 0.0015 ms/image, but its robustness is limited by the use of a single global threshold.

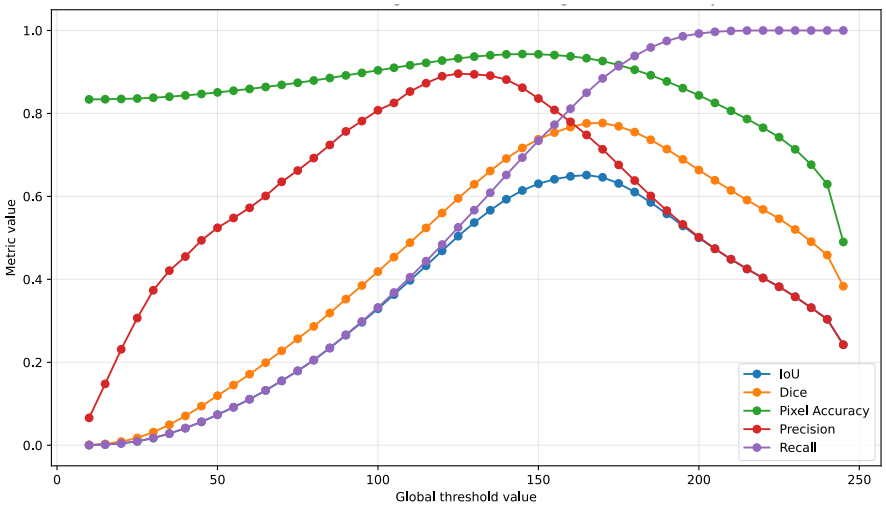

Fig. 3. Effect of the global threshold value on validation metrics for simple thresholding

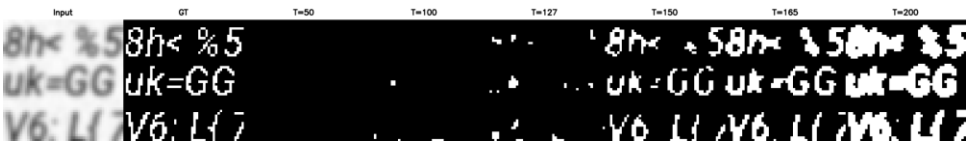

Fig. 4. Visual comparison of simple thresholding results with different threshold values on a test image

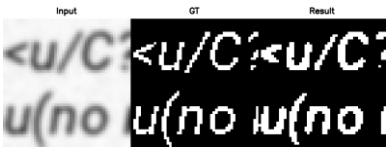

Fig. 5. Example of the best simple thresholding result on the test subset using the validation-selected threshold T=165

K-means Clustering

For K-means clustering, the number of clusters was fixed to K = 2, and the maximum number of iterations was selected on the validation subset. The best validation result was obtained with max_iter = 5, although the validation curves in **Fig. 6** show that increasing this value had almost no effect on the metrics, indicating early convergence. On the test subset, K-means achieved IoU = 0.587, DSC = 0.730, PA = 0.886, Precision = 0.596, and Recall = 0.984. The very high Recall shows that the method detected almost all text pixels, while the lower Precision indicates that many degraded background pixels were also assigned to the foreground cluster. The visual examples in **Fig. 7** and **Fig. 8** confirm this tendency toward over-segmentation. The average runtime was 0.305 ms/image, which is higher than simple thresholding but still computationally efficient.

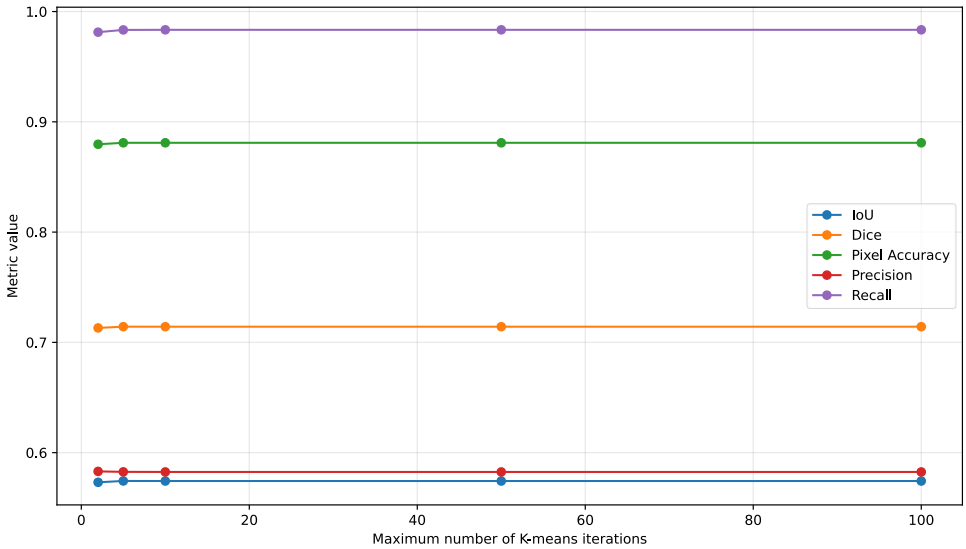

Fig. 6. Effect of the maximum number of K-means iterations on validation metrics

Fig. 7. Visual comparison of K-means results with different max_iter values on a test image

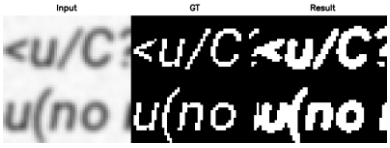

Fig. 8. Example of the best K-means result on the test subset using the validation-selected parameter max_iter = 5

Otsu Binarization

Otsu binarization was applied as an automatic thresholding method, so no validation-based parameter selection was required. On the test subset, it achieved IoU = 0.587, DSC = 0.725, PA = 0.884, Precision = 0.595, and Recall = 0.984. The very high Recall shows that Otsu detected almost all text pixels, but the lower Precision indicates that background noise and degraded regions were often included as foreground. As shown in **Fig. 9**, the automatically selected thresholds varied across images, with an average threshold of 181.853. A representative qualitative result of Otsu binarization is shown in **Fig. 10**. The average runtime was 0.0073 ms/image, confirming that Otsu remains a very fast classical baseline.

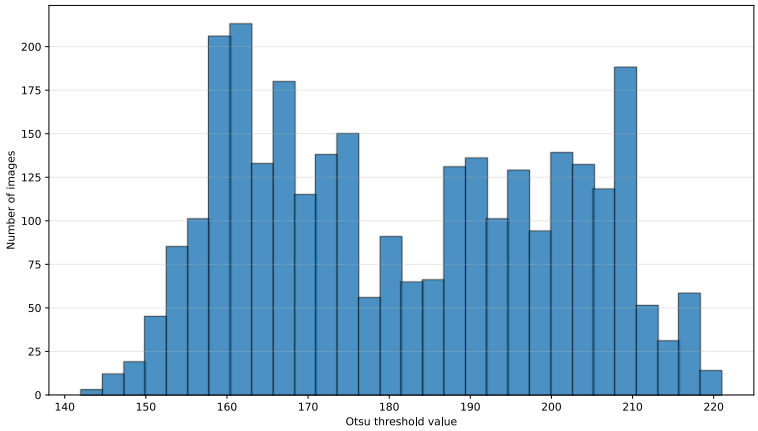

Fig. 9. Distribution of automatically selected Otsu threshold values on the test subset

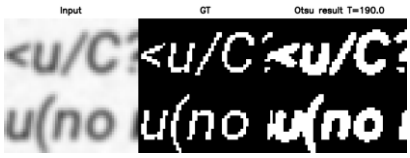

Fig. 10. Example of Otsu binarization on the test subset with the automatically selected threshold

Watershed Segmentation

For watershed segmentation, the distance threshold ratio for foreground marker generation was selected on the validation subset. The best result was obtained with ratio = 0.4. As shown in **Fig. 11**, increasing the ratio improved Precision but reduced Recall, since stricter foreground markers limited the segmented text regions. The comparison in **Fig. 12** illustrates how different ratio values affect the resulting masks. On the test subset, watershed achieved IoU = 0.567, DSC = 0.718, PA = 0.905, Precision = 0.721, and Recall = 0.756. Compared with K-means and Otsu, watershed produced more balanced Precision and Recall, but it missed more text pixels. The best test example obtained with the selected ratio is shown in **Fig. 13**. The average runtime was 0.111 ms/image, remaining relatively low for a region-based method.

Fuzzy C-means Clustering

For Fuzzy C-means, the number of clusters was fixed to 2, and the fuzziness coefficient was selected on the validation subset. The best result was obtained with m = 1.7. As shown in **Fig. 14**, changing m had only a minor effect on the validation metrics, indicating that the method behaved similarly across the tested fuzziness values. The comparison in **Fig. 15** illustrates how different values affect the resulting masks. On the test subset, Fuzzy C-means achieved IoU = 0.590, DSC = 0.727, PA = 0.885,

Precision = 0.599, and Recall = 0.983. Similar to K-means and Otsu, the method detected almost all text pixels, but also included degraded background regions as foreground. The best test example obtained with the selected parameters is shown in **Fig. 16**. The average runtime was 2.965 ms/image, making it noticeably slower than K-means due to iterative soft membership estimation.

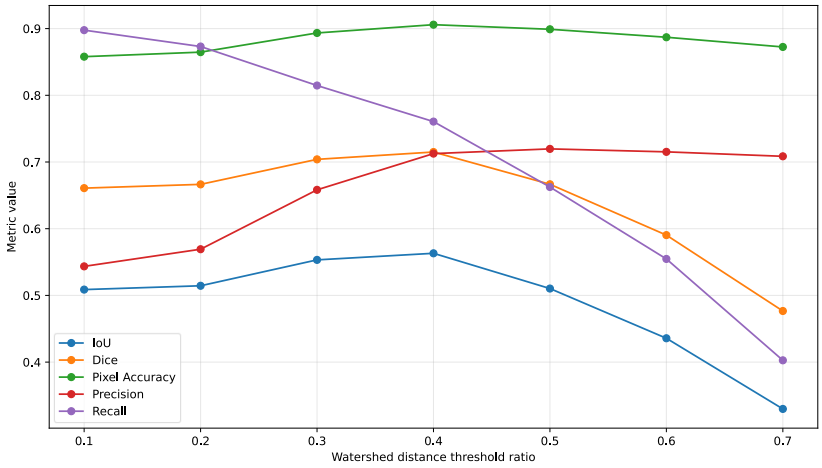

Fig. 11. Effect of the watershed distance threshold ratio on validation metrics

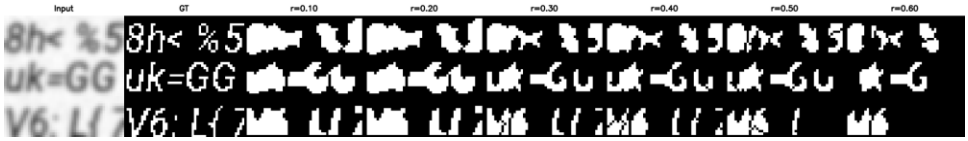

Fig. 12. Visual comparison of watershed results with different distance threshold ratios on a test image

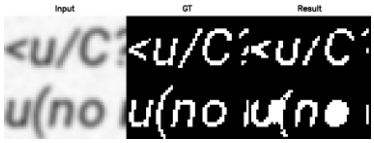

Fig. 13. Example of the best watershed result on the test subset using the validation-selected ratio $r = 0.4$

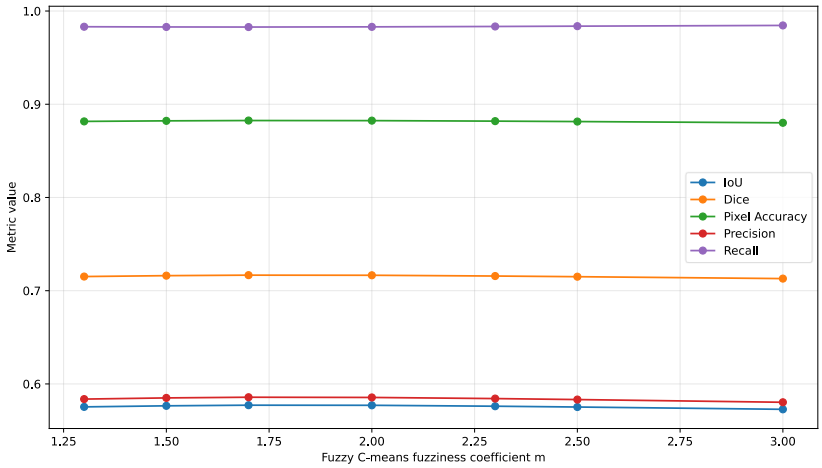

Fig. 14. Effect of the fuzziness coefficient m on validation metrics for Fuzzy C-mean

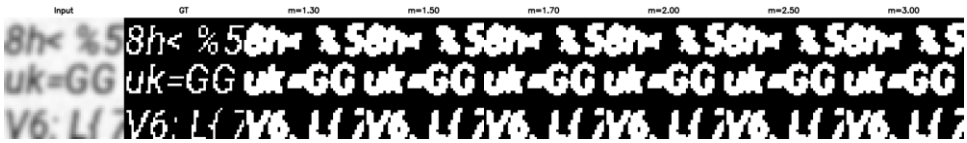

Fig. 15. Visual comparison of Fuzzy C-means results with different *m* values on a test image.

Fig. 16. Example of the best Fuzzy C-means result on the test subset using the validation-selected value *m* = 1.7

Adaptive Thresholding

For adaptive thresholding, the best validation configuration was Gaussian adaptive thresholding with block size = 3 and C=5. As shown in **Fig. 17**, the method was highly sensitive to the local window size and C value, confirming the importance of validation-based parameter selection. The comparison in **Fig. 18** illustrates how different values affect the resulting masks. On the test subset, adaptive thresholding achieved IoU = 0.785, DSC = 0.877, PA = 0.961, Precision = 0.851, and Recall = 0.923. Compared with global thresholding, it provided a better balance between Precision and Recall, indicating stronger robustness to local intensity variation and noise. The best test example obtained with the selected parameters is shown in **Fig. 19**. The average runtime was 0.022 ms/image, which is still very low and suitable for lightweight processing.

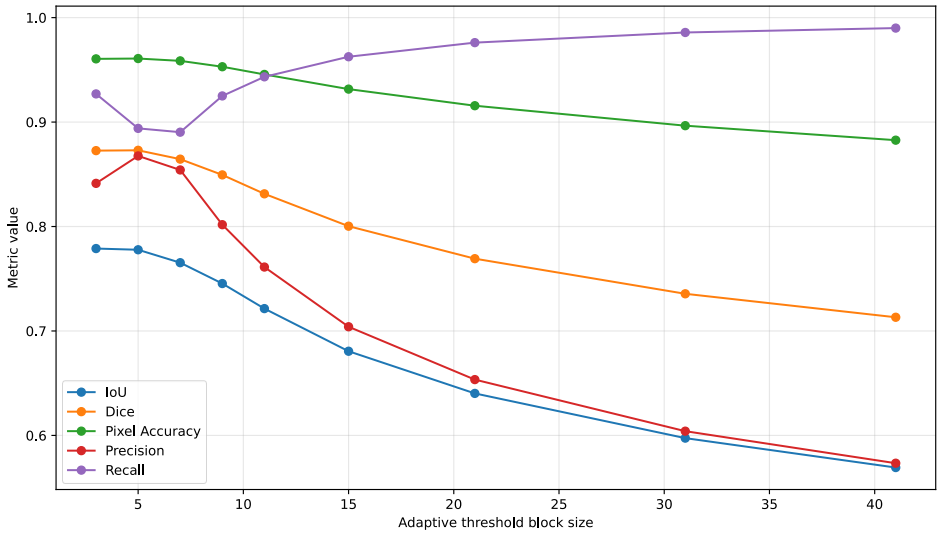

Fig. 17. Effect of adaptive thresholding parameters on validation metrics

Fig. 18. Visual comparison of adaptive thresholding results with different parameter configurations on a test image

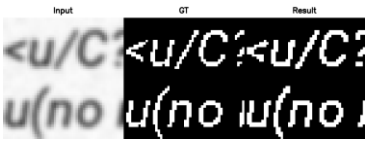

Fig. 19. Example of the best adaptive thresholding result on the test subset using Gaussian adaptive thresholding with block size = 3 and C = 5

Region Growing

For region growing, the growth margin was selected on the validation subset, with the best result obtained at grow_margin = 0. As shown in **Fig. 20**, increasing the growth margin did not improve the segmentation quality, because additional expansion mainly introduced background artifacts rather than useful text pixels. The comparison in **Fig. 21** illustrates how different values affect the resulting masks. On the test subset, the method achieved IoU = 0.590, DSC = 0.727, PA = 0.886, Precision = 0.601, and Recall = 0.978. Similar to clustering-based methods, Region Growing detected almost all text pixels but also included degraded background regions as foreground. The best test example obtained with the selected parameters is shown in **Fig. 22**. The average runtime was 0.073 ms/image, making it relatively fast among region-based methods.

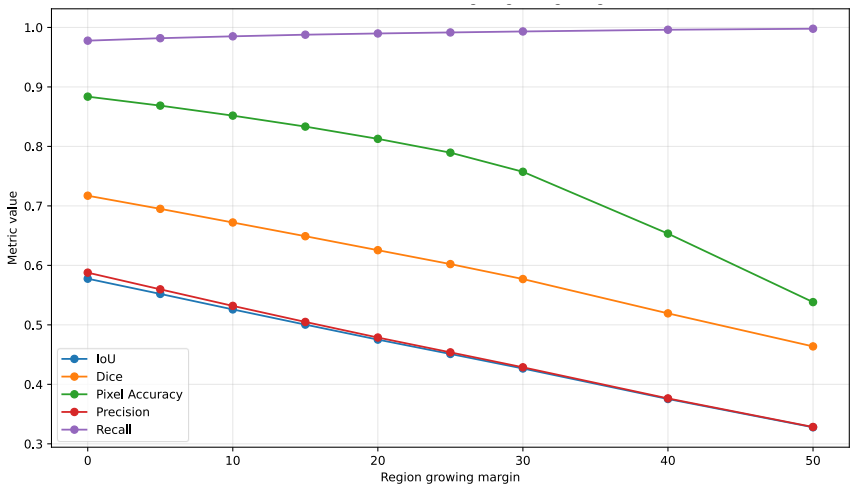

Fig. 20. Effect of the growth margin on validation metrics for Region Growing.

Fig. 21. Visual comparison of Region Growing results with different growth margin values on a test image

Fig. 22. Example of the best Region Growing result on the test subset using the validation-selected value grow_margin = 0

Region Splitting and Merging

For Region Splitting and Merging, the split threshold was selected on the validation subset, with the best result obtained at split_threshold = 5. As shown in **Fig. 23**, increasing the split threshold reduced IoU, Dice, and Recall because larger regions were no longer divided sufficiently and more text pixels were missed. The comparison in **Fig. 24** illustrates how different values affect the resulting masks. On the test subset, the method achieved IoU = 0.460, DSC = 0.620, PA = 0.830, Precision = 0.500, and Recall = 0.868. The relatively high Recall indicates that much of the text was detected, but the low Precision shows that the region-level decisions also introduced many false foreground pixels. The best test example obtained with the selected parameters is shown in **Fig. 25**. The average runtime was 5.836 ms/image, making it slower than threshold-based and clustering-based methods.

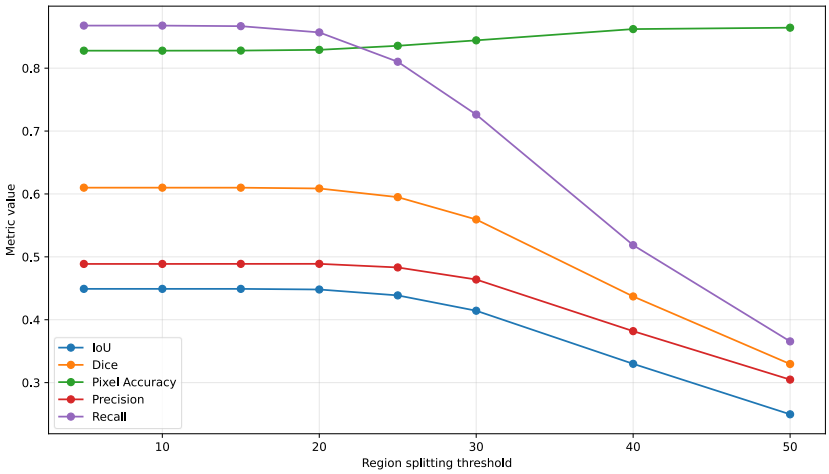

Fig. 23. Effect of the split threshold on validation metrics for Region Splitting and Merging

Fig. 24. Visual comparison of Region Splitting and Merging results with different split threshold values on a test image

Fig. 25. Example of the best Region Splitting and Merging result on the test subset using the validation-selected value split_threshold = 5

Edge-Based Detection

For edge-based detection, the best validation configuration was Canny low threshold = 100, high threshold = 250, and dilation iterations = 1. As shown in **Fig. 26**, changing the Canny low threshold had only a minor effect on the validation metrics. As shown in **Fig. 27**, dilation improved overlap with the ground-truth masks, but the method remained limited because it detects character contours rather than filled text regions. On the test subset, it achieved IoU = 0.313, DSC = 0.469, PA = 0.646, Precision = 0.320, and

Recall = 0.955. The high Recall indicates that most text boundaries were detected, while the low Precision shows that many non-text edges and background artifacts were also included. The best test example obtained with the selected parameters is shown in **Fig. 28**. The average runtime was 0.058 ms/image.

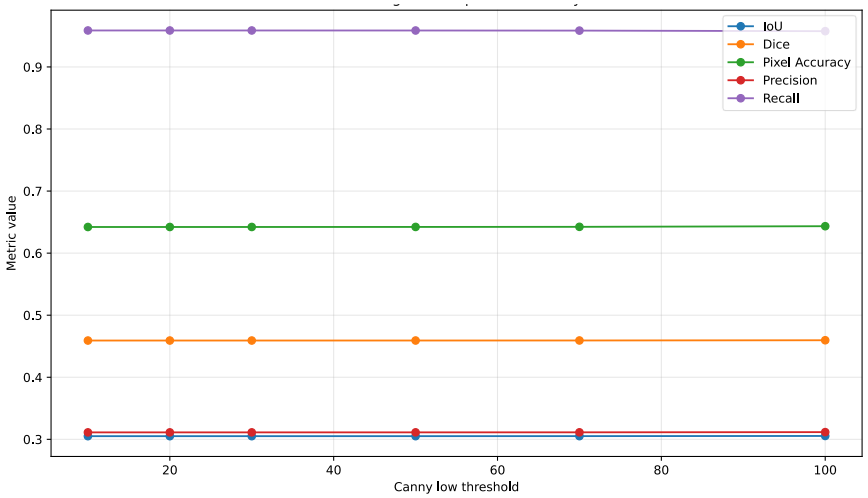

Fig. 26. Effect of Canny edge detection parameters on validation metrics

Fig. 27. Visual comparison of edge-based detection results with different parameter configurations on a test image

Fig. 28. Example of the best edge-based result on the test subset using Canny thresholds 100/250 and one dilation iteration

Laplacian of Gaussian Filtering

For Laplacian of Gaussian filtering, the best validation configuration was Gaussian kernel size = 7, sigma = 1.5, LoG threshold = 40, and dilation iterations = 0. **Fig. 29** shows the influence of LoG threshold on metrics. As shown in **Fig. 30**, the method was sensitive to the response threshold, since low thresholds introduced more noisy edges, while high thresholds removed weak text structures. On the test subset, it achieved IoU = 0.415, DSC = 0.584, PA = 0.823, Precision = 0.492, and Recall = 0.742. These results confirm that LoG captures part of the text structure, but as an edge-based method, it does not fully recover complete foreground regions. The best test example obtained with the selected parameters is shown in **Fig. 31**. The average runtime was 0.036 ms/image.

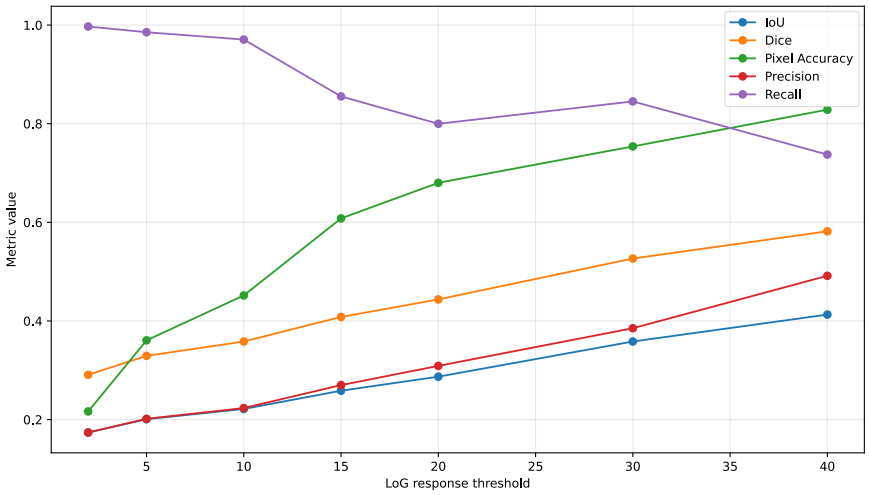

Fig. 29. Effect of Laplacian of Gaussian parameters on validation metrics.

Fig. 30. Visual comparison of Laplacian of Gaussian results with different parameter configurations on a test image

Fig. 31. Example of the best Laplacian of Gaussian result on the test subset using Gaussian kernel size = 7, sigma = 1.5, and LoG threshold = 40

Tiny U-Net

For the neural baseline, the probability threshold was selected on the validation subset, with the best result obtained at threshold = 0.55. The training curve in **Fig. 32** shows that both training and validation loss decreased and stabilized, indicating successful model training. As shown in **Fig. 33**, the validation metrics remained high across a wide threshold range, indicating stable model predictions and good separation between text and background probabilities. The comparison in **Fig. 34** illustrates how different values affect the resulting masks. On the test subset, Tiny U-Net achieved IoU = 0.959, DSC = 0.979, PA = 0.994, Precision = 0.979, and Recall = 0.979. These results substantially outperform the classical methods, showing that the model successfully learned spatial text structure and degraded background patterns. The best test example obtained with the selected threshold is shown in **Fig. 35**. The average inference time was 3.191 ms/image, which is slower than most classical methods but still practical for lightweight processing.

Overall Comparison of Methods

Table 1 summarizes the final test results for all evaluated methods. The reported parameters were selected on the validation subset, while the metrics were computed on the independent test subset.

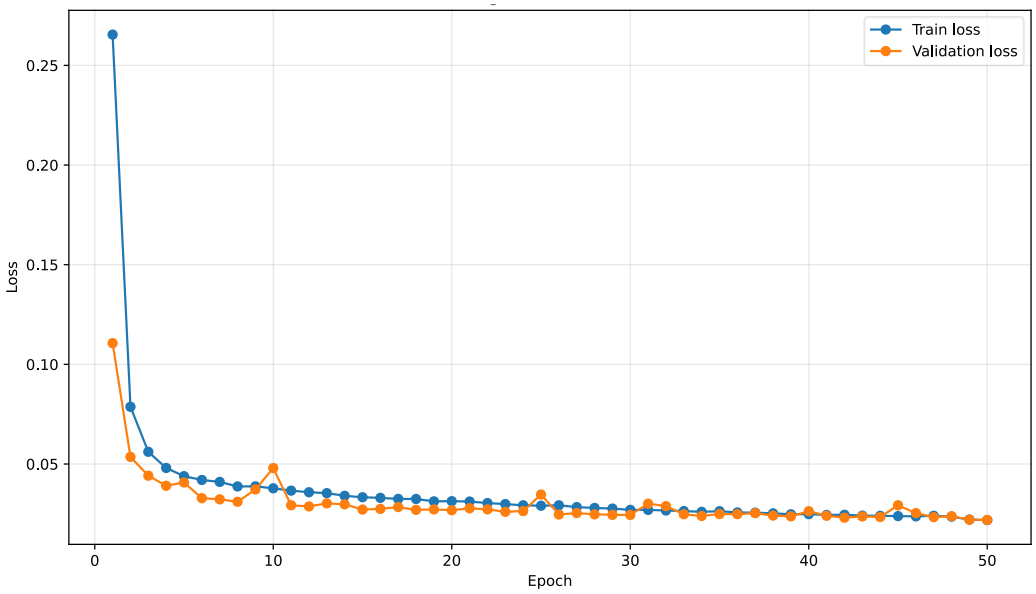

Fig. 32. Training and validation loss during Tiny U-Net training

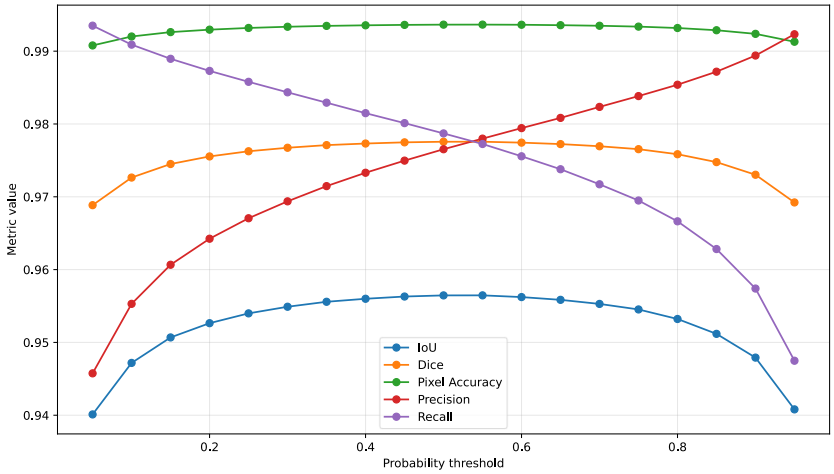

Fig. 33. Effect of the probability threshold on validation metrics for Tiny U-Net

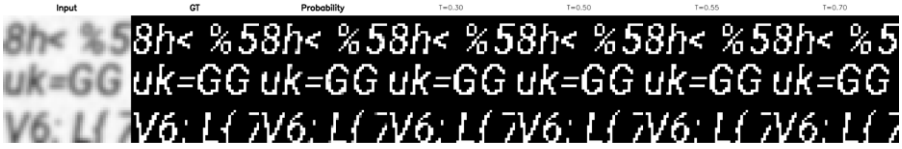

Fig. 34. Visual comparison of Tiny U-Net predictions with different probability thresholds on a test image

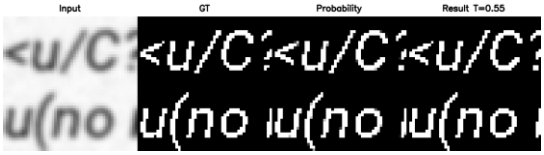

Fig. 35. Example of the best Tiny U-Net result on the test subset using the validation-selected threshold T=0.55

Table 1. **Final comparison of segmentation methods on the test subset.**

Method	Selected parameters	IoU	Dice	Pixel Accuracy	Precision	Recall	Runtime, ms/image
Simple Global Thresholding	(T = 165)	0.661	0.784	0.933	0.751	0.861	0.0015
Otsu Binarization	automatic threshold	0.587	0.725	0.884	0.595	0.984	0.0073
Adaptive Thresholding	Gaussian, block size = 3, (C = 5)	0.785	0.877	0.961	0.851	0.923	0.022
K-means Clustering	(K = 2), max_iter = 5	0.587	0.730	0.886	0.596	0.984	0.305
Fuzzy C-means Clustering	(c = 2), (m = 1.7)	0.590	0.727	0.885	0.599	0.983	2.965
Watershed Segmentation	distance ratio = 0.4	0.567	0.718	0.905	0.721	0.756	0.111
Region Growing	grow_margin = 0	0.590	0.727	0.886	0.601	0.978	0.073
Region Splitting and Merging	split_threshold = 5	0.460	0.620	0.830	0.500	0.868	5.836
Edge-Based Detection	Canny 100/250, dilation = 1	0.313	0.469	0.646	0.320	0.955	0.058
Laplacian of Gaussian	(k = 7), (\sigma = 1.5), threshold = 40, dilation = 0	0.415	0.584	0.823	0.492	0.742	0.036
Tiny U-Net	input size = 50×50, threshold = 0.55	0.959	0.979	0.994	0.979	0.979	3.191

CONCLUSION

This study presented a comparative evaluation of classical segmentation methods and a lightweight neural baseline for binarizing degraded text images. The task was formulated as binary semantic segmentation, where text pixels were treated as the foreground class and background pixels as the background class. To ensure a fair comparison, all tunable parameters were selected on the validation subset, while the final quantitative results were reported on the independent test subset.

The experimental results showed that the evaluated methods differ substantially in terms of segmentation quality, robustness, and computational efficiency. Among the classical approaches, adaptive thresholding achieved the best overall performance, with IoU = 0.785, DSC = 0.877, and PA = 0.961. This confirms that local threshold estimation is more suitable for degraded text images than a single global threshold. At the same time, simple global thresholding demonstrated the lowest runtime, requiring only 0.0015 ms/image, which makes it an attractive option for extremely resource-constrained scenarios despite its lower robustness.

Clustering-based methods, including K-means and Fuzzy C-means, achieved very high Recall values, indicating that they detected most text pixels. However, their lower Precision showed that background degradation and noise were often incorrectly assigned to the foreground class. Region-based methods demonstrated mixed performance. Watershed segmentation provided a more balanced Precision–Recall trade-off, while Region Splitting and Merging suffered from lower foreground accuracy due to coarse region-level decisions. Edge-based methods, including Canny edge detection and Laplacian of Gaussian filtering, were less effective for full-text mask extraction because they primarily detected character contours rather than complete foreground regions.

The Tiny U-Net neural baseline achieved the highest segmentation quality among all evaluated methods, with IoU = 0.959, DSC = 0.979, PA = 0.994, Precision = 0.979, and Recall = 0.979. These results indicate that even a compact neural architecture can effectively learn spatial text structures and distinguish text from degraded background patterns. However, this improvement comes at the cost of higher inference time compared with most classical algorithms.

The results also confirmed that Pixel Accuracy should not be used as the only evaluation metric for text binarization. Due to the dominance of background pixels, Pixel Accuracy can remain high even when the foreground text mask is incomplete or contains many false positives. Therefore, foreground-oriented metrics such as IoU, DSC, Precision, and Recall provide a more reliable interpretation of segmentation quality.

Overall, the results show that different binarization methods are suitable for different application scenarios, depending on the required accuracy, runtime constraints, and available computational resources. Classical methods remain useful when simplicity, interpretability, and low runtime are the main requirements. Adaptive thresholding provides the strongest classical baseline, while Tiny U-Net offers the best segmentation quality when training data and computational resources are available. Future research will focus on extending this comparative evaluation framework to 3D point cloud segmentation and classification, where irregular data structure, noise, varying point density, and complex object geometry introduce additional methodological challenges.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The author received no financial support for the research, writing, and/or publication of this article

COMPLIANCE WITH ETHICAL STANDARDS

The author declares that the research was conducted in the absence of any conflict of interest.

AUTHOR CONTRIBUTIONS

The author has read and agreed to the published version of the manuscript.

REFERENCES

[1] Sauvola, J., & Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern Recognition*, 33(2), 225–236. https://doi.org/10.1016/S0031-3203(99)00055-2

[2] Gatos, B., Pratikakis, I., & Perantonis, S. J. (2006). Adaptive degraded document image binarization. *Pattern Recognition*, 39(3), 317–327. https://doi.org/10.1016/j.patcog.2005.09.010

[3] Sahoo, P. K., Soltani, S., & Wong, A. K. (1988). A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41(2), 233-260. https://doi.org/10.1016/0734-189X(88)90022-9

[4] Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1), 62-66. https://doi.org/10.1109/TSMC.1979.4310076
[5] Bradley, D., & Roth, G. (2007). Adaptive Thresholding Using the Integral Image. *Journal of Graphics Tools*, 12(2), 13-21. https://doi.org/10.1080/2151237X.2007.10129236
[6] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1(14), 281-297. https://projecteuclid.org/euclid.bsmsp/1200512992
[7] Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press. https://doi.org/10.1007/978-1-4757-0450-1
[8] Beucher, S., & Lantuéjoul, C. (1979). Use of Watersheds in Contour Detection. *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation*. https://people.cmm.minesparis.psl.eu/users/beucher/publi/watershed.pdf
[9] Adams, R., & Bischof, L. (1994). Seeded Region Growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6), 641-647. https://doi.org/10.1109/34.295913
[10] Horowitz, S. L., & Pavlidis, T. (1974). Picture segmentation by a directed graph of regions and edges. *Proceedings of the 2nd International Joint Conference on Pattern Recognition*, 424-433.
[11] Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6), 679-698. https://doi.org/10.1109/TPAMI.1986.4767851
[12] Marr, D., & Hildreth, E. (1980). Theory of Edge Detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167), 187-217. https://doi.org/10.1098/rspb.1980.0020
[13] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., & Garcia-Rodriguez, J. (2018). A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70, 41–65. https://doi.org/10.1016/j.asoc.2018.05.018
[14] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28
[15] Müller, D., Soto-Rey, I., & Kramer, F. (2022). Towards a guideline for evaluation metrics in medical image segmentation. BMC Research Notes, 15, 210. https://doi.org/10.1186/s13104-022-06096-y
[16] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 303-338. https://doi.org/10.1007/s11263-009-0275-4
[17] Dice, L. R. (1945). Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3), 297-302. https://doi.org/10.2307/1932409
[18] Fellarrusto. Text Binarization Dataset. Kaggle. Available online: https://www.kaggle.com/datasets/fellarrusto/text-binarization

ЕВОЛЮЦІЯ МЕТОДІВ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ: КЛАСИЧНІ АЛГОРИТМИ ТА ГЛИБИННЕ НАВЧАННЯ

Віктор Вдовиченко
Львівський національний університет імені Івана Франка,
вул. Драгоманова 50, 79005 Львів, Україна

АНОТАЦІЯ

Вступ. Бінаризація текстових зображень є важливим етапом попередньої обробки в задачах аналізу документів, оптичного розпізнавання символів та автоматизованого вилучення інформації. Завдання стає складним, коли зображення містять розмиття, шум, низьку контрастність або деградацію фону.

Матеріали та методи. У цьому дослідженні порівнюються класичні та нейронні методи бінаризації деградованих текстових зображень. Оцінювання проводилося на парному наборі даних для бінаризації тексту, який містить деградовані вхідні зображення та відповідні чисті бінарні цільові маски. Оцінювальні методи включали просте глобальне порогове значення, бінаризацію Оцу, адаптивне порогове значення, кластеризацію K-середніх, метод нечіткої класифікації C-середніх, водороздільна сегментація, нарощування областей, розбиття та злиття областей, детектор границь Кенні, лапласіан гауссіана і модель U-Net. Методи оцінювалися за наступними метриками: коефіцієнтом Жаккара (КЖ), коефіцієнтом подібності Дайса (КПД), точністю пікселів (ТП), влучністю, повнотою та часом виконання.

Результати та обговорення. Адаптивне порогове значення продемонструвало найкращу якість серед класичних методів, досягнувши КЖ = 0,785. Просте глобальне порогове значення було найшвидшим методом із часом виконання 0,0015 мс/зображення, але з нижчою точністю. Метод K-середніх, нечіткий метод C-середніх бінаризація Оцу та нарощування областей продемонстрували високі значення повноти понад 0,97, проте нижчі значення влучності показали, що деградовані ділянки фону часто класифікувалися як передній план. Методи на основі границь показали нижчі значення КЖ та КПД, оскільки вони переважно виявляли контури символів. Модель U-Net досягла найвищої загальної якості: КЖ = 0,959.

Висновки. Результати дослідження показують, що різні методи бінаризації є доцільними для різних умов застосування, залежно від вимог до точності, швидкодії та обчислювальних ресурсів. Класичні методи залишаються привабливими для сценаріїв із низькою затримкою, тоді як легкі нейронні моделі забезпечують суттєво вищу точність сегментації за наявності достатніх обчислювальних ресурсів. Результати також показують, що точність пікселів потрібно інтерпретувати обережно через домінування фонових пікселів у зображеннях документів.

Ключові слова: бінаризація тексту, сегментація зображень, порогове значення, кластеризація, U-Net, показники оцінювання.

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
09 April, 2026	25 May, 2026	27 May, 2026	29 June, 2026