BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **25**, No 3
Sofia • 2025 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2025-0022

Unification of Semantic and Instance Segmentation with BoundaryX

Teodor Boyadzhiev, Krassimira Ivanova

Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, 1113 Sofia, Bulgaria
E-mails: t.boyadzhiev@math.bas.bg kivanova@math.bas.bg

Abstract**:** *Semantic segmentation is a field of image content recognition in which each pixel is classified according to the type of object it belongs to, while instance segmentation distinguishes individual object instances. A novel method, BoundaryX, is proposed to unify both tasks without relying on bounding boxes. Each pixel is classified, and boundaries are drawn around separate instances, enabling easy bounding box calculation without shape constraints or region proposals. Both instanced objects* (*like people*) *and non-instanced ones* (*like the sky*) *are handled by BoundaryX, without hardcoded exceptions. The quality of the method was evaluated on the COCO dataset for the class "people" by measuring Intersection over Union* (*IoU*) *for the semantic segmentation and bounding boxes recall and precision. The method achieved 0.774 IoU for semantic segmentation, 75% recall, and 83% precision for bounding box quality. Segmentation pipelines are simplified through the unified solution and flexible boundary-based representation provided by BoundaryX.*

Keywords**:** *Semantic segmentation, Instance segmentation.*
2020 Mathematics Subject Classification*: 68T07, 68T45.*

1. Introduction

Image content recognition remains a challenging task, yet advances in machine learning have enabled effective solutions across a wide range of applications, including medical imaging, traffic surveillance, infrastructure monitoring, and scene understanding. Recognition can be performed at several levels: image classification provides a global label; semantic segmentation assigns a class to each pixel; instance segmentation identifies and distinguishes individual object instances by their type, location, and boundaries. Combining instance and semantic segmentation offers a richer scene understanding, capturing both distinguishable objects and amorphous regions like sky or grass. The result is a detailed representation where each object is outlined and each pixel is semantically labeled.

Instance segmentation has gained significant interest in recent years, driven by deep learning techniques such as reinforcement learning and transformers [18], with

CNN-based solutions continuing to show strong performance [5]. UNet [17] has been a dominant architecture for the purposes of semantic segmentation in many fields such as remote sensing [1] and medicine [14], and has been enhanced with self-attention mechanisms [15, 21]. Meanwhile, two-stream networks using high- and low-resolution paths have also shown promise [13, 19]. Instance segmentation is commonly tackled using bounding box regression (e.g., YOLO [16]) or region proposal methods [3, 4, 6]. Our work is inspired by the Discriminative Feature Network (DFN) [20], which addresses intra-class inconsistency and inter-class confusion in semantic segmentation by combining a Smooth Network for masks and a Border Network for boundaries.

In this paper, BoundaryX is proposed as a novel instance segmentation framework that uses semantic masks in combination with object boundaries, predicted via a fully convolutional network. Our contributions are twofold:

- A new approach to instance segmentation that leverages object boundaries;
- A demonstration that semantic segmentation models can be easily extended for instance-level tasks.

By incorporating boundaries, precise object localization and shape representation are enabled, while remaining adaptable to any semantic segmentation backbone. We illustrate this approach using the basic UNet architecture.

The rest of this paper is organized as follows. Related works that catalyse ideas for the proposed approach and its architectural solutions are reviewed in Section 2. The proposed framework, BoundaryX, is introduced in Section 3. The components used in the pipeline are explained in Section 4. The methods employed in the pipeline are detailed in Section 5. Experimental results are presented in Section 6, and the findings and future research directions are discussed in Section 7.

2. Related works

In image segmentation tasks, capturing both fine-grained details and global context is very important. A widely used architecture, UNet, achieves this through an encoder–decoder structure with skip connections: the encoder extracts features, while the decoder restores spatial information. The encoder is a sequence of encoder blocks, typically containing two convolutional layers with some activation, such as ReLU. Each encoder block is followed by a resolution-reducing operation such as max-pool. After the last max-pool operation, there is a block, which here is called “bottom”, followed by the decoder. The decoder is a sequence of resolution-increasing operations, each of which is followed by a decoder block, typically two convolutions. After the last decoder block, a 1×1 convolution acts as a pixel classifier. In this paper, the overall architecture of all the networks is structured like UNet; however, two decoders are used – one to derive the object boundaries and another to derive the semantic labels.

Although UNet provides a solid foundation for image segmentation, recent work has shown that extending the backbone with specialized modules can significantly improve performance and reduce computational burden. The goal is to include mechanisms that better capture semantic context or adaptively focus on

informative regions of an image. In this direction, parallel convolutions with different dilation factors [2] are proposed, which increase the receptive field and allow multiple levels in the UNet architecture to be replaced with a single Atrous pyramid level. Recent research [11] also demonstrates that applying of channel attention mechanism [8] that adaptively recalibrates channel-wise features by modelling inter-channel dependencies enables channels to be emphasized or suppressed via learned weights. Architectures with different types of blocks for the encoder-decoder architecture are explored, based on Atrous and Channel attention mechanisms, and a new type of block, called Weighted Channel Attention, is proposed, which is inspired by the gates in the Long-Short-Term Memory (LSTM) networks. This block contains a convolutional feature map transformation, for example, two convolutional layers, and has a skip connection. Then a weight is extracted for each channel, which is used as a soft switch between the same channel before and after the transformation. This way, each block can "choose" for each channel to what extent to pass down the old features or the newly derived ones.

3. Our approach: BoundaryX

In the method presented here, the problem of instance segmentation is addressed by classifying each pixel of an image according to the object it belongs to, and semantic segmentation is achieved by drawing a boundary around each object. Firstly, the boundaries are drawn around each segment, regardless of whether they are part of the same object or not, and later it is determined whether they belong to the same object. The method consists of three steps:

- **Boundary and semantic mask generation.** A convolutional neural network, called Boundary Generator, is used to predict the semantic masks for each object class, and the boundaries around each object segment are drawn.
- **Morphologic analysis.** Semantic masks and boundaries are used to extract the individual segments of each object by class.
- **Segment association.** The convolutional network, called Segment Associator, is used for determining whether individual segments belong to the same object. An equivalence relation is used for optimization, reducing the number of necessary comparisons.

3.1. Boundary and semantic mask generation

The boundaries and the semantic mask generation are approached as a pixel classification task. For an input image, $x \in \mathbb{R}^{H \times W \times C}$, where H, W, and C are the height, width, and the number of channels, respectively, a semantic probability mask, $\hat{s} \in [0,1]^{H \times W \times K}$, where K is the number of semantic categories, and boundary probabilities $\hat{b} \in [0,1]^{H \times W}$ are generated. For example, $\hat{s}_{hwk} = P(x_{hw} \text{ is class } k | x)$ is the probability of pixel x_{hw} to be part of an object of class k and $\hat{b}_{hw} = P(x_{hw} \text{ is the boundary} | x)$ is the probability of pixel x_{hw} to be part of the boundary of an object.

These conditional probabilities are modelled by a fully convolutional neural network, which has one encoder, followed by a convolutional block called bottom, and two decoders, Fig. 1.

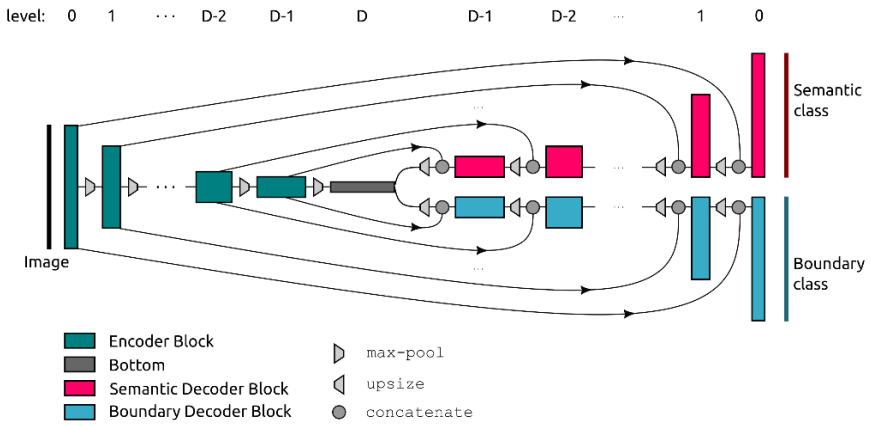

Fig. 1. Generic architecture of Boundary Generator

Each encoder or decoder is a series of convolutional blocks, block_l, $l = 0, \ldots, D-1$, where D is the depth of the network. The bottom is considered to be at level D. Each block in the encoder has a skip connection to each block of the decoder at the same level.

After each encoder block, there is a max-pool operation, which halves the height and the width. Before each decoder block, there is an upsize operation that doubles the height and the width. Each block consists of several convolutional layers, and each convolutional layer has $\min(f2^l, m)$ filters, where f is the number of filters in the level $l = 0$, and m is the maximum number of filters.

The effects of different types of convolutional blocks are explored in the paper. Details of these blocks can be found in Section 4.

Data representation and error functions

The training data consists of triplets $\mathcal{D} = \{(x, s, b)\}$, where $x \in \{0, 1, \ldots, 255\}^{H \times W \times 3}$ is the input RGB image, $s \in \{1, \ldots, K\}^{H \times W}$ is the semantic label, and $b \in \{0, 1\}^{H \times W}$ is the boundary label. A boundary pixel has label 1 – otherwise 0. The semantic label is one-hot encoded, with an additional background category. If there are three categories (e.g., "person", "cat", "dog"), then $K = 4$. The boundary pixels do not overlap with the segment pixels, avoiding issues during morphological analysis with segments that are just several pixels thin.

The boundary loss ε_b is calculated using binary cross-entropy:

$$\varepsilon_b = -\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} b_{hw} \ln \hat{b}_{hw} + (1 - b_{hw}) \ln(1 - \hat{b}_{hw}).$$

The semantic loss ε_s is computed using categorical cross-entropy

$$\varepsilon_s = -\frac{1}{HWK} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{k=1}^{K} s_{hwk} \ln \hat{s}_{hwk}.$$

The total error is the sum of both:

$$\mathcal{E} = \mathcal{E}_b + \mathcal{E}_s.$$

3.2. Morphological analysis

This step uses boundary and semantic information to derive object segments for each class. A semantic mask is computed from the semantic probability map:

$$m^{(s)}{}_{hwk} = f(x) = \begin{cases} 1 & \text{if } \hat{s}_{hwk} > 0.5 \\ 0 & \text{otherwise} \end{cases},$$

and a boundary mask from the boundary probability map

$$m^{(b)}{}_{hw} = f(x) = \begin{cases} 0 & \text{if } \hat{b}_{hw} > \tau_{bg} \\ 1 & \text{otherwise} \end{cases}.$$

For each class k, the segment regions are calculated as

$$r(k)_{hw} = m^{(s)}{}_{hwk} \times m^{(b)}{}_{hw}.$$

The multiplication separates segments from the semantic maps, where boundaries are 0 and non-boundaries are 1. Finally, each connected region of class k, denoted as $\text{seg}(k)^{(i)}$, is extracted from the segment region map $r(k)$.

3.3. Segment Association

After identifying the segments of each object, they are grouped based on whether they belong to the same object. This is done using the Segment Associator network, which classifies pairs of segments as "same object" or "different objects". The input to the network is the elementwise sum of the two segments:

$$\text{seg}(k)^{(ij)}{}_{hw} = \text{seg}(k)^{(i)}{}_{hw} + \text{seg}(k)^{(j)}{}_{hw}.$$

In addition to the merged segments $\text{seg}(k)^{ij}$, the original image x is also used as input. These inputs are concatenated, resulting in a network input of shape $H \times W \times 4$.

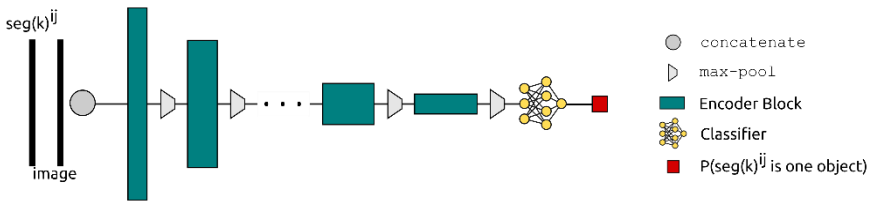

Fig. 2. The principal architecture of Segment Associator

The encoder is composed of convolutional blocks followed by max-pooling, and a fully connected layer with a weighted sum and sigmoid activation (Fig. 2). The encoder blocks use $\min(f2^l, m)$ filters, where f is the number of filters at level $l = 0$, and m is the maximum number of filters. The effects of different convolutional blocks are explored experimentally. A separate Segment Associator network, $\text{SA}(k)$, is trained for each category k.

This network is computationally expensive to use; therefore, it is desirable to apply an algorithm for reducing the number of comparisons. The algorithm represents the set of segments as $S(k) = \{\text{seg}(k)_i\}$, $i = 1, \ldots, N$, and defines an

equivalence relation $\mathrm{E} \subseteq S \times S$, where two segments are related if their similarity score $E = \left\{\left(\mathrm{seg}_i,\ \mathrm{seg}_j\right) \middle|\ \mathrm{SA}\left(\mathrm{seg}_{ij},\ x\right)\right\}$ exceeds a threshold τ_{SA}. Each equivalence class $e_{\mathrm{p}} \subseteq S$ corresponds to a separate object, and their total number equals the number of detected objects in the image.

The algorithm operates on two undirected graphs, $R\left(E_R, V_{R,}\right)$ and $T(E_T, V_T)$, where each vertex represents a segment, and both graphs have N vertices. Initially, R has no edges, and T is fully connected. The algorithm iteratively removes a random edge from T and tests whether the corresponding segments belong to the same object. If they do, it adds the edge (and transitive connections) to R and removes these from T. Otherwise, it removes from T all edges between the connected components containing the segments. The number of comparisons ranges from $O(n)$ when all segments form one object, to $O(n^2)$ when each segment is separate.

4. Encoder and decoder block types

We explore four types of encoder-decoder blocks: UNet, Atrous, Channel Attention (CA), and Weighted Channel Attention (WCA), shown in Fig. 3.

- UNet block follows the classic structure with 3×3 convolution followed by BatchNormalization and ReLU activation twice.
- Atrous block has 4 parallel 3×3 convolutions with dilations of 1 px, 3 px, 5 px, and 7 px. Each of these convolutions is followed by BatchNormalization and ReLU activation. In parallel to these four convolutions, there is a global average pool followed by fully connected layers, batch normalization with ReLU activation, and a replicate operation. Finally, the results of all the parallel operations are concatenated over the channels and linearly projected with a 1×1 convolutional layer.
- CA block uses a Squeeze-and-Excitation style channel attention mechanism: global average pooling followed by two fully connected layers generates channel-wise weights, which modulate the output of the convolutional layers. In encoder blocks, this output is downsampled; in decoder blocks, it is upsampled and concatenated with the corresponding encoder feature map.
- WCA block extends CA by combining the input and transformed feature maps via a learned convex combination with channel-specific weights, inspired by residual learning. Since the two inputs may differ in channel size, a 1×1 convolution is used to match dimensions before fusion.
- The experiments explore the effects of all four types of blocks. When using the UNet, CA, and WCA, the corresponding type is used for every convolutional block. However, when using the Atrous block, it is used only for the bottom block while the rest blocks in the encoder and decoder are of type UNet. In this case, the network has fewer levels.
- The experiments with Segment Associator only explore the effects of the blocks UNet and WCA. In these experiments, all the corresponding block type is used for each block in the encoder part.

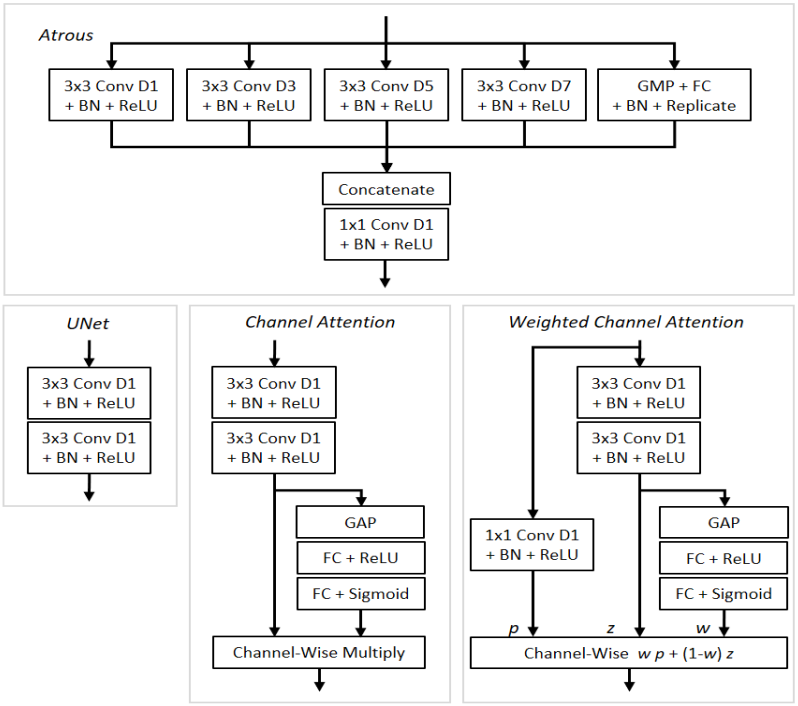

Fig. 3. Architectures of different types of blocks

5. Methods

This section shows implementation details for Boundary Generator and Segment Associator.

All experiments, data preprocessing, model training, and evaluation were conducted using Wolfram Mathematica 13.0. The source code and scripts are available upon request.

5.1. Boundary Generator

All networks in this study were trained on the COCO dataset [10], using only images labelled under the class “people”. A total of 59,420 images were extracted and resized to 256×256 pixels. The input resolution of 256×256 pixels was selected as a compromise to enable faster training and evaluation across multiple architectures and parameter settings. The primary objective of the experiments was to compare the proposed methods against each other under consistent conditions, rather than to achieve direct comparability with external benchmarks. This resolution was found sufficient to demonstrate the functional viability of the models and to observe their relative performance differences.

In addition to resizing, further preprocessing steps were applied to refine the dataset and ensure data consistency. Objects smaller than 256 pixels in area were removed, and the resulting gaps were filled using image in-painting. If the removed objects covered less than 25% of the image, the “Texture Synthesis” function was

used; otherwise, the "Fast Matching" method was applied. As COCO labels neckties as a separate class, the neckties were reassigned to the "people" class to avoid segmentation gaps.

Boundary masks were generated using morphological 1 px dilation, edge detection, and additional 2 px dilation.

The training set was augmented using random zoom (scale between 1 and 1.2), horizontal and vertical flips, and rotations within ±90º, each applied with a probability of 0.25. When zoom or rotation was applied, special care was taken to recover object boundaries at the image edges.

The dataset was randomly split into training and testing subsets using a fixed random seed to ensure reproducibility. 80% of the images were used for training, and 20% for testing.

The research in this work is mainly focused on the development and comparison of different architectural variants of the model, where the hyperparameters (including regularization and thresholds) were optimized empirically. Several architectures were evaluated for the boundary generation network, with the best-performing one selected and regularized to reduce overfitting in the final pipeline. All architectures were configured to perform roughly the same number of multiplications – approximately 16 billion for a 256×256 image. The hyperparameters of the networks are summarised in 0.

Table 1. Hyperparameters of the networks for Boundary Generator

Architecture	Levels	f	m	r
UNet	6	24	512	
Atrous	4	26	512	
CA	6	23	512	2
WCA	6	23	512	2

The quality of the semantic segmentation is measured using the Intersection Over Union (IoU) using the derived semantic mask, $m^{(\mathrm{s})}$, and the label, s. The quality of the boundary generation is measured by IoU using the reversed boundary mask, $1 - m^{(\mathrm{b})}$, and the label, b. For the boundary IoU, a threshold of $\tau_{\mathrm{bg}} = 0.5$ was used.

The networks were initialized using the Xavier method, with the weights drawn from a normal distribution, and trained using the ADAM Algorithm with a learning rate of 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-5}$ for 200 iterations. Training was performed with a batch size of 32 images. When comparing the different architectures, no regularization or dropout was applied. However, L2 regularization was used when training the best-performing architecture.

5.2. Segment Associator

The Segment Associator network has 7 encoder blocks, where each block has two convolutions with BatchNormalization and ReLU activation. Each encoder block, l, has $\min(f2^l, m)$ filters, where f is the number of filters in level 0, and m is the maximum number of filters. The encoder blocks are followed by a classifier, which

is a fully connected network with two layers, where the hidden layer has d units with ReLU activation and the second layer has one unit and sigmoid activation.
The Segment Associator networks are trained with the ADAM Algorithm, with a learning rate of 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-5}$ for 100 iterations. The batch size is 128 samples. Each network is initialized with the Xavier method. The used hyperparameters for Segment Associator are $\text{levels} = 7$, $f = 16$, $d = 256$, $r = 2$ (for WCA).

The data was split randomly, with 70% used for training and 30% for testing. The same augmentation is used as the one used for training the Boundary Generator. The batch size is 128 images, where each image is selected randomly from the training data. When an image is selected, it may contain multiple segments. Therefore, from each image, two random segment masks are selected and merged. Then the label of the sample would depend on whether the two randomly selected segments are part of the same object or not. When calculating the accuracy of the F1 score, a threshold $\tau_{SA} = 0.5$ was used.

6. Results

This section shows the results from the experiments, including training the Boundary Generator, Segment Associator, adding regularization, and the complete pipeline.

6.1. Boundary Generator

The training and testing semantic IoU (left) and the boundary IoU (right) for the four architectures used for the Boundary Generator, trained without regularization, are presented in Fig. 4. The results are smoothed by a moving average with a window of 20 epochs. Inspection of the semantic IoU shows that the worst-performing architecture is Atrous, while the best performance is achieved by the WCA. A similar pattern is observed when inspecting the boundary IoU. The CA architecture is found to perform better than Atrous due to less overfitting.

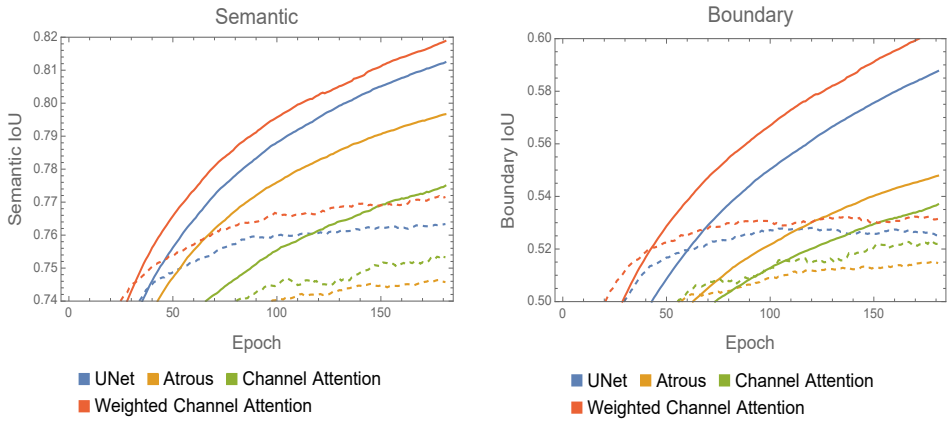

Fig. 4. Semantic and Boundary training IoU (solid lines) and testing IoU (dashed lines) of four architectures for Boundary Generator

6.2. Segment associator

The training and testing F1 score (left) and Accuracy (right) for the two architectures used for the Segment Associator are illustrated in Fig. 5. The results are smoothed with a moving average with 20-epoch window. The plots indicate slightly better training results for the WCA blocks. However, the difference becomes negligible for the testing results, which are around 0.85 F1 Score and 0.95 Accuracy.

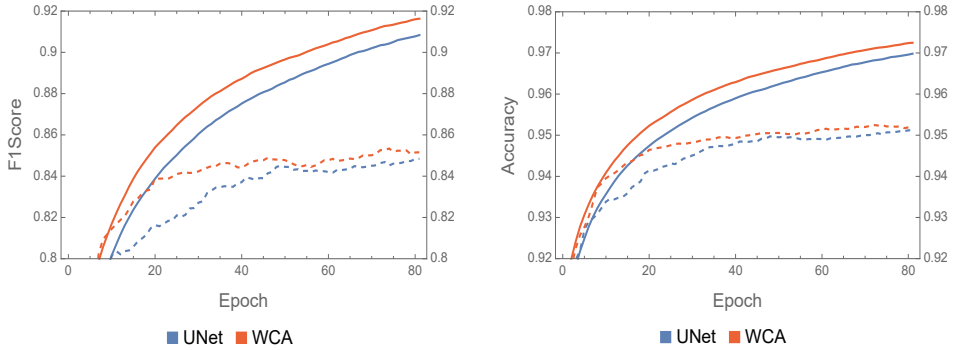

Fig. 5. Training (solid lines) and testing (dashed line) F1 Score and Accuracy for Segment Associator

6.3. Avoiding overfitting

The comparison of the different architectures for Boundary Generator and Segment Associator showed an advantage for WCA. However, all the architectures overfitted the training data, and therefore, L2 regularization was used. The regularization coefficients were chosen through the process of trial and error.

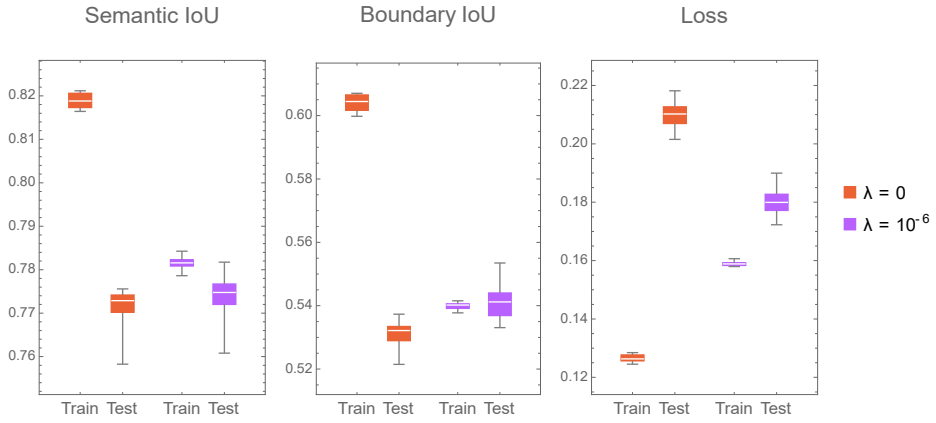

Fig. 6. Semantic IoU, Boundary IoU, and Loss with no regularization and L2 regularization with a coefficient of 10^{-6} for Boundary Generator

Box plots of training and testing metrics over the last 20 epochs for the Boundary Generator are presented in Fig. 6. Both the semantic and boundary IoU showed a notable reduction in overfitting when L2 regularization with a coefficient

of 10^{-6} was applied. Additionally, testing performance saw a slight improvement. Although the increase in testing IoU was modest, the testing loss dropped significantly with regularization. This discrepancy might be due to IoU being a discretized approximation of the loss function. Given the substantial loss improvement, the regularized model was selected for the final pipeline.

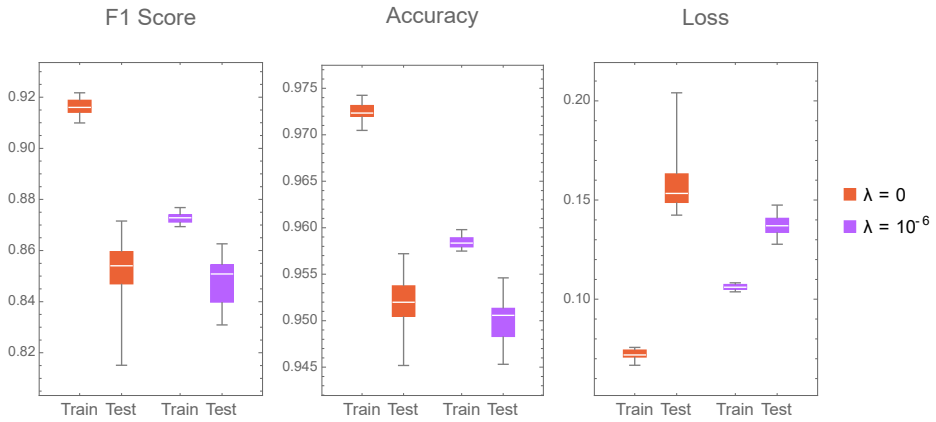

Fig. 7. F1-score, Accuracy, and Loss with no regularization and L2 regularization with a coefficient of 5×10^{-5} for Segment Associator

Training and testing results over the last 20 epochs for the Segment Associator are displayed in Fig. 7. The regularization L2 with a coefficient of 5×10^{-5} led to a clear reduction in overfitting across all metrics. However, F1 score and accuracy remained largely unchanged. A noticeable decrease in testing loss was observed, supporting the decision to use the regularized model in the final system.

6.4. Recall and Precision of the Pipeline

During boundary generation, each pixel is classified into a class "boundary" or "background". Therefore, the boundary generator is a binary classifier at the pixel level. To determine whether a pixel is categorized as boundary, the probability $P(x_{hw} \text{ is boundary} | x)$ is compared to the threshold of boundary generator τ_{bg}. Similarly, the segment associator is also a binary classifier, which uses a threshold τ_{SA} to determine if two segments are part of the same object.

The quality measurements shown in the experiments in this paper are derived assuming thresholds of 0.5; however, this may not be the most optimal value. To find good values for the thresholds, τ_{bg} and τ_{SA}, the complete pipeline was evaluated on all the samples from the testing data used during the training of Segment Associator. For each sample, bounding boxes were derived from each object. Then the COCO annotations of the same samples were used to calculate precision, P, and recall, R, for bounding boxes [12]. Two bounding boxes are determined to match if their IoU is larger than 0.5.

Recall and Precision of the pipeline

	τ_{sa}											
τ_{bg}		0.3		0.4		0.5		0.6		0.7		0.8	
	0.4	0.72	0.81	0.73	0.81	0.74	0.80	0.75	0.79	0.76	0.77	0.77	0.74
	0.5	0.73	0.84	0.74	0.83	0.75	0.82	0.76	0.81	0.77	0.80	0.77	0.78
	0.6	0.72	0.85	0.73	0.84	0.74	0.83	0.75	0.83	0.75	0.81	0.76	0.79

Fig. 8. The Recall (on the left) and the Precision (on the right) of the pipeline for different values of the thresholds τ_{bg} and τ_{SA}, calculated on test data extracted from COCO

The precision and recall for different values of the thresholds are shown in Fig. 8. The results show that the increase of τ_{SA} causes an increase in the recall and a decrease in the precision, which is an expected behaviour. The behaviour of the measurements is different when τ_{bg} varies. The larger values do not result in higher recall, because a missing boundary would merge two segments, resulting in larger bounding boxes.

1 2 3

Fig. 9. Examples of successful segmentation and object detection

1 2 3

Fig. 10. Examples of failures of semantic segmentation and segment association

Examples of successful segmentation and object detection are provided in Fig. 9. In the first image, the focus gradually decreases, but the segmentation algorithm correctly delineates the figures, and the Segment Associator has correctly merged the segments belonging to the fourth person in the column. In the second image, the objects are correctly recognized, even though one of the people is only partially visible and faces the camera with their back. In the third image, there is one small false positive segment, but two big segments are correctly associated with one person.

Examples of mistakes made by the BoundaryNet are presented in Fig. 10. In the first image, the tree and the boy are wrongly recognized as one segment. In the second image, the father and the kid are recognized as one segment from one side, and the segment corresponding to the hand of a woman is wrongly associated with the father. In the third image, a phantom person, consisting of two segments, is extracted from the background; also, the first and second figures are wrongly associated with one person.

7. Conclusion

In this work, BoundaryX, a novel instance segmentation framework, was introduced that combines semantic masks and object boundaries to identify individual instances within an image. The architecture includes Boundary Generator, Morphologic Analyzer, and Segment Associator. The Boundary Generator, based on a modified UNet with dual decoders, showed best performance when using a newly developed WCA module, which applies channel attention as a soft switch between input and transformed features. The Segment Associator, implemented as a convolutional classifier, achieved high accuracy but struggled with class imbalance, particularly in distinguishing between segments from the same object. Addressing this through data augmentation may improve performance. Effective instance segmentation for the class "people" was demonstrated by the method, with potential applicability to other classes. Future work includes improving the Segment Associator, training on higher-resolution and multi-class data, and exploring the use of large pre-trained encoders like DenseNet [9] or ResNet [7].

***Acknowledgements*:** The research is supported by "NGIC – National Geoinformation Center for monitoring, assessment and prediction of natural and anthropogenic risks and disasters" under the Program "National Roadmap for Scientific Infrastructure 2017-2027", financed by the Bulgarian Ministry of Education and Science. The authors acknowledge access to the e-infrastructure "National Centre for High Performance and Distributed Computing", funded by the same program.

References

1. Brand, A., A. Manandhar. Semantic Segmentation of Burned Areas in Satellite Images Using a U-Net-Based Convolutional Neural Network. – Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. **43**, 2021, No B3, pp. 47-53.
2. Chen, L.-C., G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected Crfs. 2014, arXiv, 1412.7062.
3. Girshick, R., J. Donahue, T. Darrell, J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. – In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580-587.
4. Gkioxari, G., J. Malik, J. Johnson. Mesh r-cnn. – In: Proc. of IEEE/CVF Int. Conference on Computer Vision, 2019, pp. 9785-9795.
5. Hafiz, A. M., G. M. Bhat. A Survey on Instance Segmentation: State of the Art. – International Journal of Multimedia Information Retrieval, Vol. **9**, 2020, pp. 171-189.
6. He, K., G. Gkioxari, P. Dollár, R. Girshick. Mask r-cnn. – In: Proc. of IEEE Int. Conference on Computer Vision, 2017, pp. 2961-2969.
7. He, K., X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. – In: Proc. of 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770-778.
8. Hu, J., L. Shen, G. Sun. Squeeze-and-Excitation Networks. – In: Proc. of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132-7141.
9. Huang, G., Z. Liu, L. Van Der Maaten, K. Weinberger. Densely Connected Convolutional Networks. – In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700-4708.

10. Lin, T. Y., M. Maire, , S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick. Microsoft Coco: Common Objects in Context. – In: Lecture Notes in Computer Science. Vol. **8693**. 2014, pp. 740-755.
11. Long, D. T. Efficient DenseNet Model with Fusion of Channel and Spatial Attention for Facial Expression Recognition. – Cybernetics and Information Technologies, Vol. **24**, 2024, No 1, pp. 171-189.
12. Padilla, R., S. L. Netto, E. A. B. da Silva. A Survey on Performance Metrics for Object-Detection Algorithms. – In: Proc. of 2020 Int. Conference on Systems, Signals and Image Processing, 2020, pp. 237-242.
13. Pan, H., Y. Hong, W. Sun, Y. Jia. Deep Dual-Resolution Networks for Real-Time and Accurate Semantic Segmentation of Traffic Scenes. – IEEE Transactions on Intelligent Transportation Systems, Vol. **24**, 2023, pp. 3448-3460.
14. Panchal, S., M. Kokare. Resmu-Net: Residual Multi-Kernel u-Net for Blood Vessel Segmentation in Retinal Fundus Images. – Biomedical Signal Processing and Control, 2024, 90:105859.
15. Peng, L., C. Zhu, L. Bian. U-Shape Transformer for Underwater Image Enhancement. – IEEE Transactions on Image Processing, Vol. **32**, 2021, pp.3066-3079.
16. Redmon, J., S. Divvala, R. Girshick, A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. – In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779-788.
17. Ronneberger, O., P. Fischer, T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. – In: Lecture Notes in Computer Sciences. Vol. **9351**. 2015, pp. 234-241.
18. Sharma, R., M. Saqib, C. Lin, M. Blumenstein. A Survey on Object Instance Segmentation. – SN Computer Science, Vol. **3**, 2022, No 499.
19. Xu, J., Z. Xiong, S. Bhattacharyya. Pidnet: A Real-Time Semantic Segmentation Network Inspired by Pid Controllers. – In: Proc. of 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 19529-19539.
20. Yu, C., J. Wang, C. Peng, C. Gao, G. Yu, N. Sang. Learning a Discriminative Feature Network for Semantic Segmentation. – In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1857-1866.
21. Zhang, W., S. Chen, Y. Ma, Y. Liu, X. Cao. Etunet: Exploring Efficient Transformer Enhanced UNet for 3d Brain Tumor Segmentation. – Computers in Biology and Medicine, Vol. **171**, 2024, 108005.

Received: 13.05.2025. First Revision: 14.06.2025.
Second revision: 25.08.2025. Accepted: 04.09.2025