Using Different Loss Functions with YOLACT++ for Real-Time Instance Segmentation

Selin Koles, Selami Karakas, Alain P. Ndigande, and Sedat Ozer

Ozer Lab, Department of Computer Science, Ozyegin University, Istanbul, Turkiye

2023 46th International Conference on Telecommunications and Signal Processing (TSP) | 979-8-3503-0396-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/TSP59544.2023.10197832

***Abstract*—In this paper, we study and analyze the performance of various loss functions on a recently proposed real-time instance segmentation algorithm, YOLACT++. In particular, we study the loss functions, including Huber Loss, Binary Cross Entropy (BCE), Mean Square Error (MSE), Log-Cosh-Dice Loss, and their various combinations within the YOLACT++ architecture. We demonstrate that we can use different loss functions from the default loss function (BCE) of YOLACT++ for improved real-time segmentation results. In our experiments, we show that a certain combination of two loss functions improves the segmentation performance of YOLACT++ in terms of the mean Average Precision (mAP) metric on Cigarettes dataset, when compared to its original loss function.**

***Keywords*—Instance segmentation, Loss function, YOLACT++, real time segmentation.**

I. Introduction

Real-time instance segmentation is an important field where the goal is completing the instance segmentation task as in [1]–[8] at real-time (or near real-time) speed. YOLACT++ [7], is a recent algorithm proposed for real-time instance segmentation. When the goal is performing instance segmentation at real-time, there are many computational constraints which usually enforce researchers to use smaller architectures. As such, the segmentation performance of real time algorithms might be lower than offline (or non-real-time) algorithms in terms of various segmentation based metrics. To avoid further differences in the segmentation accuracy, it is important to use the most suitable loss function. YOLACT++ uses mainly binary cross entropy (BCE) loss function. However, for real time segmentation, while being sufficient, that might not be the best loss function. Therefore, in this study, we analyze the effect of using different loss functions on the segmentation performance of YOLACT++ algorithm.

We analyze the effect of training YOLACT++ at different batch sizes first, and then use the best batch size for the rest of the experiments in this paper. We combine various loss functions and retrain YOLACT++ with each of those combinations separately to better understand which loss function might better perform with YOLACT++. Our preliminary experiments show that the combination of MSE and BCE losses yields better segmentation performance when compared to the segmentation performance of the original loss function.

The rest of this paper is organized as follows; in Section II, we discuss related work including the YOLACT and YOLACT++ architectures. Section III lists the loss functions that are used in our analysis. Section IV provides our experimental details including the used datasets, performance metrics and experimental results, respectively. We conclude the paper with Section V.

II. Related Work

Image segmentation has always been an important task in computer vision and earlier segmentation studies mainly focused on classical approaches such as [9]. However, deep learning based techniques showed success over classical approaches in many domains and they have taken the attention of the researchers in many fields as in [10]–[12]. Similar to those fields, recently proposed relevant segmentation studies have mainly focused on developing deep learning based approaches in the form of semantic or instance segmentation. In 2020, the work in [13] summarized 14 well-known loss functions used for image (semantic) segmentation in the form of a survey. The authors of that work also introduced a new loss function called "Log-Cosh-Dice loss" (LCDL). In 2021, another similar work was conducted in [14] where the authors focused on studying the performance of different loss functions especially for medical image segmentation applications. Additionally, they proposed a new loss function "Log-Cosh-BCE-Dice" (LCBD) loss, which was inspired by LCDL. In both works, they concluded that none of the loss functions had the best performance, and that the performance varied according to the task.

In 2020, another approach for instance segmentation called CondInst was proposed in [1]. It adapts the weights of the instance-aware fully convolutional networks (FCNs). Following the success of CondInst, BoxInst [3], a single-stage instance segmentation method, was also introduced along with two new loss terms enabling high-quality instance segmentation by using only box annotations. Next, we briefly describe YOLACT and YOLACT++, as we study the effect of loss functions on YOLACT based networks. **YOLACT [6]:** It is one of the recent methods to achieve real-time ($\geq$ 30 fps) instance segmentation due to its parallelized and lightweight structure. It performs two tasks: (1) generating a set of global prototype masks over the entire image; and (2) predicting the linear combination of coefficients per instance. The coefficients are used to encode the representation of each instance in the prototype mask space. After performing non-max-suppression (NMS) to suppress duplicate detections, the prediction is made by linearly combining the outputs of

Authorized licensed use limited to: Ozyegin Universitesi. Downloaded on November 18,2024 at 11:06:57 UTC from IEEE Xplore. Restrictions apply.

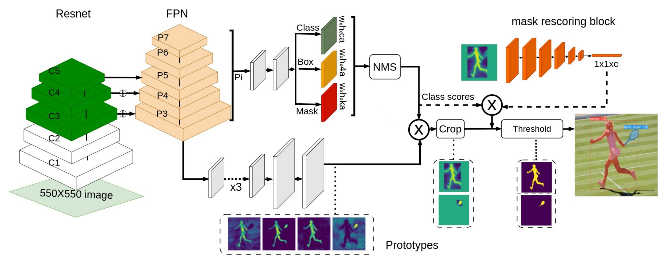

Fig. 1. This figure shows the architecture of the YOLACT++ algorithm. The green layers represent 3x3 deformable convolutional layers in the backbone. In the figure, c is for classes, a for anchors for feature layer Pi, k for prototypes, and WxH for input dimensions of the image. This figure is a re-illustration of the original architecture from [7].

the two subtasks, where a simple matrix multiplication is applied and followed by a cropping operation. For improved accuracy during the evaluation, final masks are cropped with the predicted bounding boxes; during the training, they are cropped with the ground truth boxes. Cropping works fine when the bounding box is accurate, otherwise ”leakage”, noise from outside of the cropped region, can show up in the predicted instance mask. They also applied semantic segmentation loss during training to increase feature richness. Their experiments showed that YOLACT outperformed in generating better masks with its speed in inference which makes it a better candidate for autonomous vehicles, object detection, face segmentation.

YOLACT++ [7]: Following YOLACT, the same authors proposed YOLACT++ [7] by introducing several improvements to the existing architecture. A fast mask re-scoring branch is added after the cropping operation. It contains a 6-layer FCN where each convolution layer uses a ReLU activation with a global pooling layer. It predicts the mask IoU for each object type (category) which is then used to re-score each segmentation mask by computing the product between the inferred (predicted) mask IoU and the corresponding classification confidence. Another novelty they introduced was using deformable convolution layers in the YOLACT backbone, which resulted in performance gain. For better detection results, they optimized the prediction head by choosing the hyper-parameters for the anchor including aspect ratios and scales. Overall, YOLACT++ showed a considerable boost compared to YOLACT while still performing in real-time. YOLACT++ architecture is shown in Figure 1.

III. Loss functions

Next, we summarize the used loss functions in our analysis.
Binary Cross Entropy (BCE): Cross-entropy [15] is used to compute the difference between two probability distributions. In YOLACT++, pixel-wise binary cross entropy between predicted masks and ground truths is computed and used as the mask loss. The BCE is a loss function that penalizes the predicted probabilities based on their deviations from the expected value, and it is calculated as the negative average of the logarithms. The formula for a single instance is:

$$BCE = -(ylog\hat{y}) - (1-y)log(1-\hat{y}) \tag{1}$$

where y is the ground-truth, $\hat{y}$ is the predicted value.
Mean Squared Error (MSE): It computes the mean squared error between the ground truth and its predicted value over all the samples. MSE considers outliers by putting a large weight on the outlier errors via its quadratic nature and is defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y-\hat{y})^2 \tag{2}$$

where N is the total sample-number.
Huber Loss [16]: It can be seen as a combination of MSE and mean absolute error (MAE). MAE takes the average of absolute (non-negative) differences between the model’s predictions and the ground truth. In setups where we need to focus on outliers, MAE alone becomes less effective because both small and large errors roughly get similar weights. Huber loss effectively undergoes both small and large errors by combining MSE and MAE. Based on whether the absolute difference is greater than a given threshold (δ), Huber loss routes the loss calculations as being quadratic or linear, accordingly. It is defined as follows:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2, & \text{if } |a| \geq \delta \\ \delta|a| - \frac{1}{2}\delta, & \text{otherwise} \end{cases} \tag{3}$$

where $(a = y - f(x))$, f(x) is the predicted value and δ is a threshold.
Log-Cosh-Dice Loss [13]: Dice loss (DL) [17] is calculated based on dice coefficient (DC), which measures a form of similarity between two sets, and is commonly used for addressing the data imbalance problem. The authors of [13] proposed log-cosh-dice loss ($LCDL$) to tackle the non-convex nature of the dice coefficient. They utilized cosh(x), since hyperbolic functions are easily differentiable. However, cosh(x) can go to infinity, and to keep that in a range log space is used. They managed to get a loss function whose derivative is continuous and finite, while encapsulating the properties of the dice coefficient. $LCDL$ is defined as follows:

$$DC(y,\hat{y}) = \frac{2\sum_{i=1}^{N} p_i g_i}{\sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} g_i^2} \tag{4}$$

$$LCDL = log(cosh(DL)) \tag{5}$$

where $DL = 1 - DC$, p_i, and g_i represent the prediction and ground-truth, respectively, at the i^{th} pixel.

IV. Experiments

Next, we present our experimental results. Our experiments were conducted to first compute an appropriate batch size for training, then to figure out the effect of individual loss functions on the segmentation performance of YOLACT++, and then the weighted combinations of loss functions in the

Authorized licensed use limited to: Ozyegin Universitesi. Downloaded on November 18,2024 at 11:06:57 UTC from IEEE Xplore. Restrictions apply.

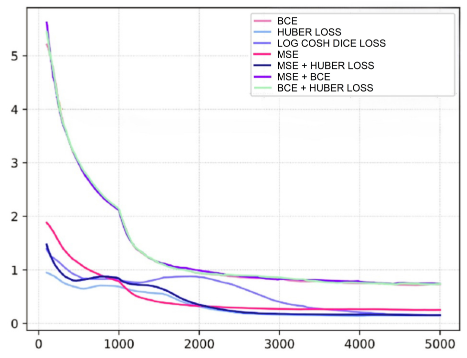

Fig. 2. This plot shows how the loss (y-axis) changes for various loss functions as the number of iterations (x-axis) increases during the training.

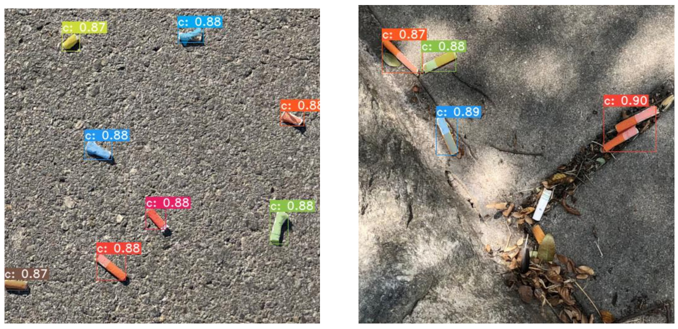

Fig. 3. Sample results are shown on two images from the Cigarettes dataset in the form of confidence scores, segmentation masks (overlayed over each instance) and bounding boxes (around each instance).

form of: $\alpha Loss_1 + \beta Loss_2$ where α and β are the weights of the chosen loss functions: $Loss_1$ and $Loss_2$, respectively. We conducted our experiments on 2 x NVIDIA RTX A6000 with CUDA version 12.0. In our preliminary experiments, we first trained YOLACT and YOLACT++ algorithms on MSCOCO [18] without making changes to the original code. Our goal was to see if we would be able to reproduce their reported results and we observed that we could reproduce the reported results on the COCO dataset. After that, we trained the model on Cigarettes dataset provided by Adam Kelly[1] (accessed on March 2023). The Cigarettes dataset is annotated for object detection and segmentation according to the COCO format. The dataset consists of 2200 images of cigarettes dropped on the ground, where 2000 images are included in the training set, and 200 images in the validation set. We used a maximum image size of 512 pixels.

In all of our experiments, we adopted ResNet101-FPN with deformable convolutions as the backbone, and utilized stochastic gradient descent as the optimization algorithm. We set the learning rate of YOLACT++ to $1e^{-4}$. The following performance metrics were used in our experiments:
(i) Intersection over Union (IOU): It measures the overlap between the target (ground truth) mask A and the predicted mask B as follows: $IOU = \frac{|A \bigcap B|}{|A \bigcup B|}$ and **(ii) mAP (Mean Average Precision):** All the AP values are averaged over different classes/categories. $mAP = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i+FP_i}$, where TP is true positives and FP is false positives.

In our experiments, we first performed tests to figure out the best batch size. Once obtained, we used the same batch size in all of our other experiments. As a batch size, the values of 2, 8 and 16 are used individually on the Cigarettes dataset. In this experiment, we did not change other parameters of the original YOLACT++, except the batch size. Table I summarizes the results of those experiments, where D indicates "detection", and M indicates "mask". In the table, we show results for both detection and segmentation (Mask) tasks by using four different metrics: mAP, AP_{50}, AP_{75} and AP_{95}. The best values, in the table, are shown in bold. In this table, our main metric was mAP and according to mAP value, best batch size values were as follows: for the detection task it was 8, and for the segmentation task, it was 16.

Next, we studied the performance of the individual loss functions. We studied how each loss function changes the results on YOLACT++ individually. In those experiments, we kept the batch size to 16. Table II shows the results of our experiments where we replaced the original mask loss function: the binary cross-entropy with each of the other loss functions as listed in the table. In the table, the best values are shown in bold. For the sake of simplicity, we use the following acronyms: BCE for binary cross-entropy, HL for Huber loss, LCDL for Log-Cosh-Dice loss, and MSE for mean squared error. As the results demonstrate, the results of using other loss functions individually did not introduce any improvements. Huber Loss with 1 as a delta value yielded the 80.53% mAP which was the highest mAP value that we obtained in this experiment. As shown in the table, experimentation with different delta values resulted in worse mask mAPs when values are increased.

Figure 2 plots how the loss value changes over iteration numbers for multiple loss functions including BCE, MSE, Huber loss, Log-Cosh-Dice loss, MSE + Huber loss, MSE + BCE and BCE + Huber loss. As seen from Table II and Figure 2, the use of Log-Cosh-Dice loss resulted in insufficient mAP value, the main reason for that, the algorithm with that loss function stucks at local optimum value. As a result, the model couldn't learn properly. Other loss functions yielded better performance compared to Log-Cosh-Dice loss.

Given the suboptimal results in mAP obtained from replacing the original mask loss function with several options, we decided to study the effect of using combinations of two loss functions. The results are summarized in Table III. We obtained a higher value in segmentation in terms of the mAP value with the combination of αMSE + βBCE, where α, and β were set to 0.5 and 1, respectively. Overall, we observed that the hybrid losses showed better performance than the single loss functions.

[1]https://www.immersivelimit.com/datasets/cigarette-butts

Authorized licensed use limited to: Ozyegin Universitesi. Downloaded on November 18,2024 at 11:06:57 UTC from IEEE Xplore. Restrictions apply.

TABLE I. TRAINING RESULTS OF YOLACT++ WITH DIFFERENT BATCH SIZES ON THE CIGARETTES DATASET

Batchsize	Det / Seg	mAP	AP_{50}	AP_{75}	AP_{95}
2	Detection	75.91	98.84	96.59	**0.28**
2	Segmentation	82.19	98.63	97.53	1.18
8	Detection	**79.91**	**99.99**	**98.99**	0.22
8	Segmentation	83.65	**99.99**	98.01	1.82
16	Detection	77.27	98.94	97.91	0.10
16	Segmentation	**84.67**	98.95	**98.95**	**2.23**

TABLE II. TRAINING RESULTS OF YOLACT++ BY USING DIFFERENT INDIVIDUAL LOSS FUNCTIONS

Losses	mAP	AP_{50}	AP_{55}	AP_{65}	AP_{75}	AP_{95}
BCE	**83.65**	**99.99**	**99.99**	**99.99**	**98.01**	**1.82**
$\text{HL}_{(\delta=1)}$	80.53	99.96	99.96	98.97	97.00	1.05
$\text{HL}_{(\delta=2)}$	79.03	99.90	98.89	98.89	97.38	0.00
$\text{HL}_{(\delta=3)}$	76.74	98.82	98.82	98.78	96.75	0.00
LCDL	71.56	98.96	98.96	98.94	95.86	0.00
MSE	79.21	98.95	98.95	97.96	96.94	0.02

TABLE III. SEGMENTATION RESULTS OF HYBRID LOSS FUNCTIONS WITH DIFFERENT WEIGHTS ON YOLACT++

Hybrid Losses	α	β	mAP	AP_{50}	AP_{65}	AP_{75}	AP_{95}
HL + BCE	1	1	81.36	99.00	99.00	97.00	0.33
HL + MSE	1	1	78.39	99.85	98.87	97.80	0.00
MSE + BCE	1	1	80.54	99.78	99.78	98.79	0.05
MSE + BCE	2	1	80.53	99.50	98.52	98.52	0.13
MSE + BCE	3	1	79.01	99.02	98.07	96.95	0.00
MSE + HL	2	1	75.44	99.02	97.13	95.19	0.00
MSE + HL	3	1	78.00	98.99	98.00	95.97	0.00
HL + BCE	2	1	82.37	99.96	98.71	96.28	0.82
HL + BCE	3	1	83.24	98.91	98.91	98.89	0.38
MSE + BCE	0.5	1	**84.43**	**99.98**	**99.98**	**98.98**	**2.08**
MSE + HL	0.5	1	71.37	93.93	92.71	89.55	0.00
HL + BCE	0.5	1	82.35	99.86	99.86	98.89	0.05

V. CONCLUSION

Real time instance segmentation is an important task in computer vision. In this paper, we tackled the problem of improving the performance of the recently proposed YOLACT++ algorithm for better segmentation performance without changing its architecture. We used various loss functions, as used in medical imaging, and studied their segmentation performance in our base model: YOLACT++. In particular, we studied the performance of MSE, BCE, Huber and Log-Cosh-Dice loss functions. We analyzed how they affect the segmentation performance individually and as a combination. We observed that hybrid (combination) loss functions yield better performance when compared to using individual losses. As a result, in this paper, we demonstrate that the performance of a real time instance segmentation algorithm can be improved without changing its architecture. In particular, we observed that by using the combination of MSE and BCE losses, we could get better segmentation results (see our results in Table III). Future work might include studying more combinations of loss functions and deriving new loss functions for segmentation.

ACKNOWLEDGMENT

This paper has been produced benefiting from the 2232 International Fellowship for Outstanding Researchers Program of TÜBİTAK (Project No:118C356). However, the entire responsibility of the paper belongs to the owner of the paper.

REFERENCES

[1] Z. Tian, C. Shen, and H. Chen, "Conditional convolutions for instance segmentation," in *European conference on computer vision*, pp. 282–298, Springer, 2020.

[2] O. Sahin, F. E. Doğanay, S. Ozer, and C. H. Chen, "Segmentation of covid-19 infected lung area in ct scans with deep algorithms," in *Computational Intelligence and Image Processing in Medical Applications*, pp. 33–48, World Scientific, 2022.

[3] Z. Tian, C. Shen, X. Wang, and H. Chen, "Boxinst: High-performance instance segmentation with box annotations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5443–5452, 2021.

[4] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo, "Polarmask: Single shot instance segmentation with polar representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12193–12202, 2020.

[5] F. E. Doğanay, O. Sahin, S. Ozer, and C. H. Chen, "Deep learning based 3d brain tumor segmentation with multispectral mri," in *Computational Intelligence and Image Processing in Medical Applications*, pp. 119–133, World Scientific, 2022.

[6] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9157–9166, 2019.

[7] D. B., C. Zhou, F. Xiao, and Y. J. Lee, "Yolact++: Better real-time instance segmentation," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[8] Y. Gürses, M. Taşpınar, M. Yurt, and S. Özer, "Grjointnet: Synergistic completion and part segmentation on 3d incomplete point clouds," in *2021 29th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2021.

[9] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, pp. 167–181, 2004.

[10] R. Valiente, M. Zaman, Y. P. Fallah, and S. Ozer, "Connected and autonomous vehicles in the deep learning era: A case study on computer-guided steering," in *Handbook Of Pattern Recognition And Computer Vision*, pp. 365–384, World Scientific, 2020.

[11] H. E. Ilhan, S. Ozer, G. K. Kurt, and H. A. Cirpan, "Offloading deep learning empowered image segmentation from uav to edge server," in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 296–300, IEEE, 2021.

[12] O. Sahin and S. Ozer, "Yolodrone: Improved yolo architecture for object detection in drone images," in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 361–365, IEEE, 2021.

[13] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pp. 1–7, IEEE, 2020.

[14] J. Xin and G. Sun, "Learn from each other: Comparison and fusion for medical segmentation loss," in *2021 7th International Conference on Computer and Communications (ICCC)*, pp. 662–666, IEEE, 2021.

[15] M. Yi-de, L. Qing, and Q. Zhi-Bai, "Automated image segmentation using improved pcnn model based on cross-entropy," in *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.*, pp. 743–746, IEEE, 2004.

[16] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73 – 101, 1964.

[17] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," pp. 240–248, Springer, 2017.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

Authorized licensed use limited to: Ozyegin Universitesi. Downloaded on November 18,2024 at 11:06:57 UTC from IEEE Xplore. Restrictions apply.