A DISTANCE TRANSFORM-BASED LOSS FUNCTION FOR SEMANTIC IMAGE SEGMENTATION WITH DEEP NEURAL NETWORKS

by

Furkan Gül

Submitted to
the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

SABANCI UNIVERSITY

December, 2023

A Distance Transform-Based Loss Function for Semantic Image Segmentation with Deep Neural Networks

Furkan Gül

Mechatronics Engineering, MSc Thesis, 2023

Thesis Supervisor: Assoc. Prof. Dr. Erchan Aptoula

Keywords: Deep learning, computer vision, image segmentation, semantic segmentation, boundary loss, distance transform, distance map, loss function

Abstract

In recent decades, there has been a tremendous enlargement of semantic segmentation datasets across diverse complex domains, including autonomous driving, satellite imaging, and medical imaging. Despite numerous advancements in solving complex semantic segmentation problems with these datasets, certain challenges, such as the precise segmentation of object boundaries in complexly structured objects, persist. Traditional loss functions like Cross-Entropy and Intersection over Union (IoU), which are typically based on integrals over segmentation regions, often fall short in these scenarios. These functions perceive objects regionally rather than contour-based, assigning equal importance to all object contours such as boundaries and inner parts. This approach overlooks the fact that segmentation at object boundaries is both more challenging and more critical. To address this, this thesis introduces a distance transform-based loss function, specifically designed to enhance the alignment between predicted and ground-truth boundaries during training, a feature not explicitly enforced in commonly used image segmentation losses. This proposed loss function is model-agnostic and can be integrated into the training of any segmentation models to enhance boundary details. Our loss was evaluated using two segmentation datasets: CelebAMask-HQ for single-class, and Cityscapes for multi-class segmentation. Experiments were conducted using two models, U-Net and DeepLabv3+, and three encoders, ResNet-34, ResNet-50, and MobileNetV2, to demonstrate the adaptability and effectiveness of our loss across various network architectures. Our evaluations and comparisons of different loss functions revealed that our loss surpassed other commonly used loss functions by 0.0561 for the Cityscapes dataset with U-Net models in terms of boundary IoU, a metric specifically designed to assess the boundary quality of objects in images. Furthermore, our loss demonstrated superior performance by using 2.4% less GPU memory, a significant factor when training larger neural networks with big datasets.

Mesafe Dönüşümü Tabanlı Yitim Fonksiyonu ile Derin Sinir Ağları Kullanılarak Görüntü Bölütleme

Furkan Gül

Mekatronik Mühendisliği, Master Tezi, 2023

Tez Danışmanı: Doç. Dr. Erchan Aptoula

Anahtar kelimeler: Derin öğrenme, bilgisayarlı görü, görüntü segmentasyonu, semantik segmentasyon, mesafe dönüşümü, mesafe haritası, yitim fonksiyonu

Özet

Son yıllarda, otonom sürüş, uydu görüntüleme ve tıbbi görüntüleme gibi çeşitli alanlarda semantik segmentasyon veri setlerinde büyük bir artış olmuştur. Bu veri setleriyle semantik segmentasyon problemlerini çözmede sayısız ilerleme olmasına rağmen, karmaşık yapıdaki nesnelerin sınırlarının doğru segmentasyonu gibi zorluklar devam etmektedir. Çapraz entropi ve IoU gibi geleneksel yitim fonksiyonları segmentasyon bölgeleri üzerinden integral alınmasına dayanır ve bu senaryolarda genellikle yetersiz kalır. Bu fonksiyonlar nesneleri bölgesel olarak algılarlar, sınırlar ve iç kısımlar gibi tüm nesne konturlarına da eşit önem verirler. Bu yaklaşım, nesne sınırlarındaki segmentasyonun hem daha zorlu hem de daha kritik olduğunu göz ardı eder. Bu tez bu sorunu çözmek için, tahmin edilen sınırlar ve gerçek sınırlar arasındaki hizalamayı artırmak amacıyla tasarlanmış mesafe dönüşüm tabanlı bir yitim fonksiyonu önermektedir. Yaygın olarak kullanılan segmentasyon yitim fonksiyonlarında bu özellik bulunmamaktadır. Önerdiğimiz yitim fonksiyonu, modelden bağımsızdır ve herhangi bir modelinin eğitimine sınır detaylarını geliştirmek için kolayca entegre edilebilir. Yitim fonsiyonumuz, tek sınıflı segmentasyon için CelebAMask-HQ ve çok sınıflı segmentasyon için Cityscapes olmak üzere iki veri seti kullanılarak değerlendirildi. U-Net ve DeepLabv3+ olmak üzere iki model ve ResNet-34, ResNet-50 ve MobileNetV2 olmak üzere üç kodlayıcı kullanılarak, yitim fonksiyonumuzun çeşitli ağ mimarileri arasında adaptasyon yeteneğini ve etkinliğini göstermek için deneyler yapıldı. Cityscapes veri seti için farklı yitim fonksiyonlarının değerlendirmeleri ve karşılaştırmaları sonucunda yitim fonksiyonumuzun sınır IoU (bIoU) açısından U-Net modelleri bazında diğer yaygın olarak kullanılan yitim fonksiyonlarını 0.0561 kadar geride bıraktığını gösterdi. Ayrıca, yitim fonsiyonumuz %2.4 daha az GPU belleği kullanarak üstün performans sergiledi. Bu durum daha büyük sinir ağları ile büyük veri setleri eğitirken önemli bir faktördür.

Acknowledgments

First and foremost, I would like to express my profound gratitude to my thesis advisor, Associate Professor Dr. Erchan Aptoula, and the Computer Vision and Pattern Analysis Lab. Our enlightening discussions sparked my creativity and critical thinking, pushing me to form a comprehensive and objective critique. His endless support in overcoming challenges, along with his encouraging guidance and warm attitude, paved the way for my achievement. Throughout the thesis writing period, his extraordinary patience and support were invaluable. I could not have imagined a better advisor for my master's thesis.

I would also like to extend my sincere thanks to the members of my thesis jury, Associate Professor Dr. Meltem Elitaş and Assistant Professor Dr. Yakup Genç. Their valuable time and insightful feedback significantly contributed to the improvement of my thesis. I consider myself privileged to have such distinguished professors on my thesis jury. I am also indebted to Sabancı University for providing me with a full tuition fee waiver for the duration of my studies.

I must express my profound gratitude to my parents and friends. Their unfailing support and continuous encouragement throughout my years of study have been invaluable. This achievement would not have been possible without them. Thank you.

Lastly, I would like to express my deepest love and gratitude to my wife. Her endless faith, love, and belief in my success have been my pillars of strength. This accomplishment would not have been possible without her. Thank you.

Contents

List of Figures

List of Tables

List of Abbreviations

bIoU Boundary Intersection over Union.

CE Cross-Entropy.

CNN Convolutional Neural Network.

CV Computer Vision.

DNNs Deep Neural Networks.

DSC Dice Similarity Coefficient.

GPU Graphics Processing Unit.

IoU Intersection over Union.

PCIe Peripheral Component Interconnect express.

RAM Random Access Memory.

ReLU Rectified Linear Unit.

vCPUs Virtual Central Processing Units.

VRAM Video Random Access Memory.

Chapter 1

Introduction

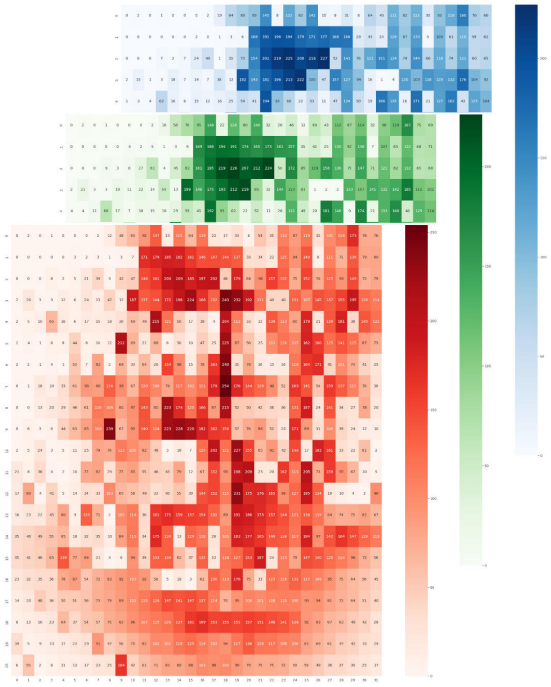
1.1 Background

Artificial Intelligence has gathered significant interest from researchers for decades, with the aim of replicating and expanding human intelligence in machines. While the achievement of Artificial General Intelligence remains a distant goal, there have been notable advancements in image and text understanding [9], [10], as well as in reinforcement learning for gaming [11], [12]. These breakthroughs have occurred in a relatively short span of time, driven by substantial investments from research laboratories and technology companies. AI applications now enhance various aspects of daily life, including virtual assistants (e.g. Alexa [1]), automatic translation (e.g. DeepL [2]), speech recognition, object tracking, clinical diagnoses (e.g. miisking [3]), and face recognition (Know Your Customer (KYC) applications e.g. Verifai [4]).

Considering that approximately 80% of human perception is visual, Computer Vision (CV) emerges as a pivotal field within AI [13]. Fundamentally, CV aims to replicate and extend human vision capabilities, automating processes related to the

[1] `https://developer.amazon.com/en-US/alexa`
[2] `https://www.deepl.com/en/translator`
[3] `https://miiskin.com/app/`
[4] `https://www.verifai.com/en/`

(A) What humans perceive



(B) What computers recognize

FIGURE 1.1: Divergence in Perception: Humans vs. Computers. Image sourced from the COCO dataset [1]

acquisition, processing, analysis, and understanding of digital images. The surge in internet usage has led to an exponential growth in visual data worldwide, with over 500 hours of video uploaded to YouTube every minute [14], and an endless stream of visual content on platforms such as Instagram, Facebook, and TikTok. The manual processing of this vast visual data becomes practically impossible, highlighting the necessity for the development of efficient methods for processing and understanding visual data.

Image recognition, a foundational step for computers to comprehend context within images, has been a persistent and challenging problem in the field of CV. Computers perceive the world differently from humans. For a computer, an image is a large matrix (or tensor) of numbers, as visually illustrated in Figure 1.1.

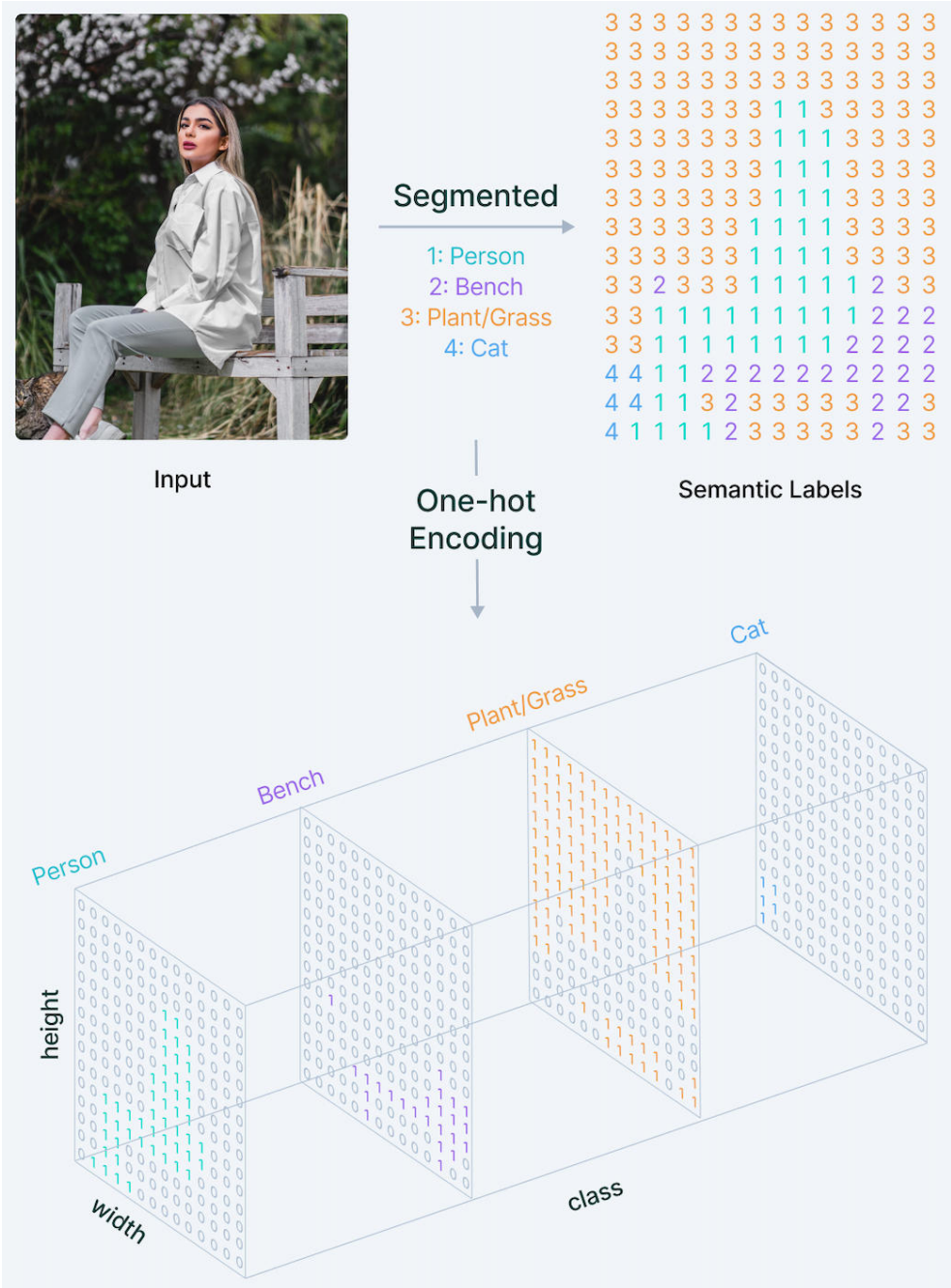


FIGURE 1.2: **An overview of the semantic image segmentation process [2]**: The entire input image is semantically segmented into regions, classifying each pixel into one of the four categories: person, bench, plant/grass, and cat.

1.2 Motivation

Despite significant advancements in image recognition, the challenge of image understanding continues to pose a significant challenge. It is insufficient to merely identify which objects are present in the image; there is a need to extract more

information from images, such as the location of the objects and their boundaries. Essentially, the goal is to segment the image into distinct regions, a process known as image segmentation. Historically, this task has been one of the most challenging aspects of Computer Vision. There are various types of image segmentation tasks, each differing in how regions are segmented. The most common type is semantic segmentation task, where regions within images are segmented semantically, as depicted in Figure 1.2.

The emergence of deep learning has led to significant progress in the field of image segmentation. Semantic segmentation, which involves associating every pixel in an image with a class (e.g., car, bus, sky, face), has particularly benefited from the rise of deep learning, larger datasets, increased computational power, and enhanced models. The evolution of this research field is driven by a wide range of practical applications, including autonomous vehicles [15], scene understanding [16], and medical image analysis [17]. However, several challenges still persist and require attention.

Our primary focus will be on addressing one of the major challenges associated with the use of deep learning-based models for semantic segmentation - the definition of appropriate loss functions. The role of the loss function is crucial in training semantic segmentation models as it guides the model to produce results that accurately reflect the shapes and geometry of objects. A poorly chosen loss function can lead to inaccurate segmentation, misclassification of objects, and ultimately, a failure in the model's ability to understand the image. Therefore, the importance of selecting an appropriate loss function cannot be underestimated. This thesis aims to delve into this critical aspect by designing a new loss function for semantic segmentation tasks.

1.3 Contributions

Existing loss functions proposed for re-balancing class prior distributions are region-based. In contrast, this study aims to introduce a boundary loss that manifests

as a distance transform-based metric on the contour or shape space, rather than regions. Instead of utilizing unbalanced integrals over regions, a boundary loss employs integrals over the boundary or interface between regions. Moreover, a boundary loss offers information that complements regional losses.

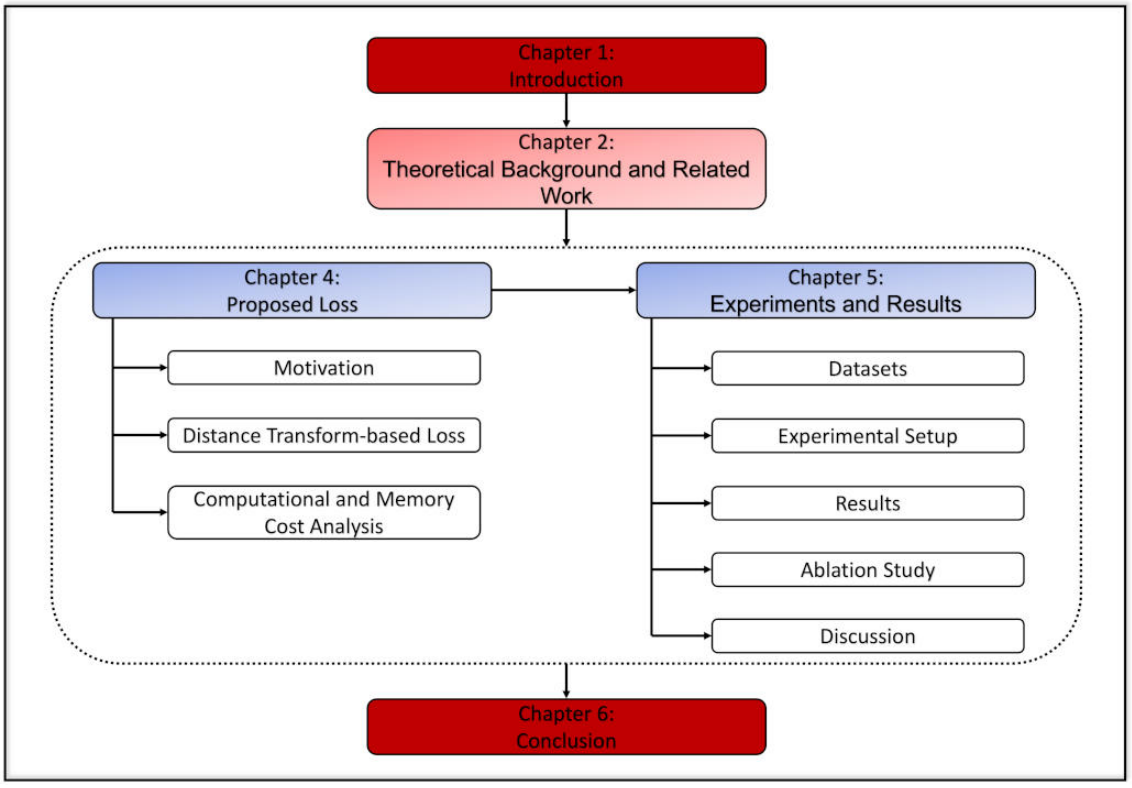


FIGURE 1.3: Outline of the Thesis

We introduce a novel loss function, termed the distance transform-based loss function, designed to more effectively constrain segmentation models. As discussed in the preceding section, the loss function typically employed for semantic segmentation is a simplistic extension of the loss function used for image classification. This conventional loss function presents several issues, a significant one being its inability to measure the structural similarity, such as shapes and geometries of objects, between the predicted mask and the ground truth mask, as it treats each pixel independently. To address this issue, we propose the distance transform-based loss function. This function is defined by implementing several steps, including a distance transformation on the ground-truth segmentation mask. We validate

the distance transform-based loss on two diverse datasets, ranging from an autonomous vehicle dataset to a face dataset, and compare it with other leading loss functions.

The thesis conducted six experiments using two different model architectures (U-Net and DeepLabv3+) and three distinct encoders (ResNet-34, ResNet-50, and MobileNetV2). The results showed that the proposed distance transform-based loss function significantly improved segmentation performance, resulting in higher bIoU, IoU, and Dice scores. The distance transform-based loss function consistently ranked first in terms of superiority bIoU across all six experiments, with the exception of the U-Net with MobileNetV2 encoder. The loss function also demonstrated superior performance with the DeepLabv3+ model in terms of bIoU score. Furthermore, the loss function consistently surpassed others in terms of GPU memory usage without any spikes observed across all experiments. The ablation study confirmed the consistent success of the proposed loss function under different seed values. Additionally, we offer a comprehensive analysis of the different loss functions and provide visualizations to better understand the loss function's behavior.

1.4 Outline

This thesis is organized as depicted in Figure 1.3. The subsequent chapter establishes the fundamental concepts necessary for understanding this thesis and provides a concise review of relevant research in the field of semantic segmentation, while also underscoring certain limitations in segmentation-related tasks. Chapter 3 introduces a novel boundary loss for datasets with highly intricate shapes, and elaborates on the methodology we employed. Chapter 4 illuminates the datasets, experimental setup, experiments, and their respective results and discussions. In the final chapter, we draw conclusions from our work and suggest potential avenues for future research.

Chapter 2

Theoretical Background and Related Work

This chapter provides an overview of the theoretical background and a literature review. The objective of this chapter is to equip the reader with crucial foundational knowledge, including Convolutional Neural Network (CNN), model architectures, various loss functions, and evaluation metrics for semantic segmentation, alongside an exploration of related works. By detailing these related works, we strive to offer a thorough understanding of the research presented in this thesis, while avoiding redundant information.

2.1 Convolutional Neural Networks (CNNs)

Deep Neural Networks (DNNs) [18] are models composed of numerous successive layers, aiming to transform input data into output while progressively learning higher-level features. CNNs stand out as highly effective models, especially for image recognition and image segmentation. A CNN architecture has various components, including convolutional layers, pooling layers, Rectified Linear Unit (ReLU) layers as an activation function, dropout layers, loss functions, and evaluation metrics. In this section, these basic components of CNNs are explained.

2.1.1 Convolutional Layer

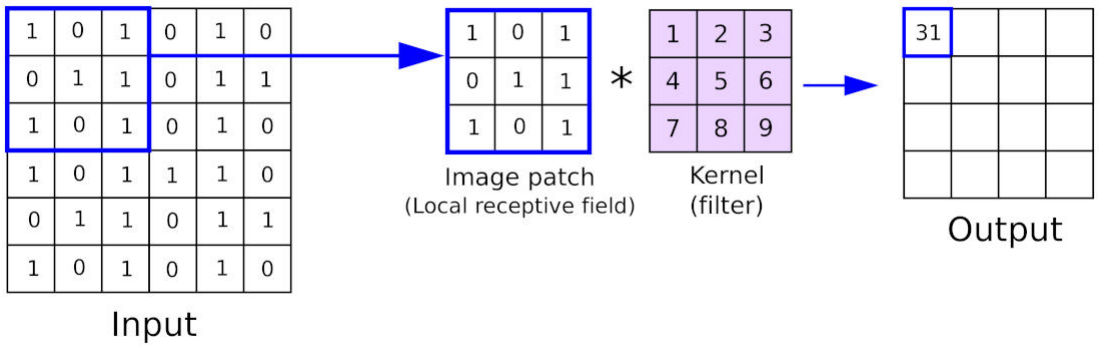


FIGURE 2.1: Convolution operation with 3×3 filter [3]

The convolutional layer serves as the fundamental building block of CNNs, specifically designed to extract hierarchical features from an image [19]. Typically, the initial input is a feature map, with the first layer representing the original image. Multiple filters are applied to the image in a convolutional layer. As depicted in Figure 2.1, a convolutional kernel can be visualized as a small matrix that identifies features within the image by sliding across it, computing an element-wise product with the underlying image patch, and generating a feature map. Intuitively, each kernel can extract low-level features such as lines, circles, and triangles in earlier layers, progressing to more complex (high-level) features like faces and cars like in deeper layers. Unlike classical computer vision, where filters are manually crafted, CNNs learn filters through optimization algorithms during the training process.

2.1.2 Pooling Layer

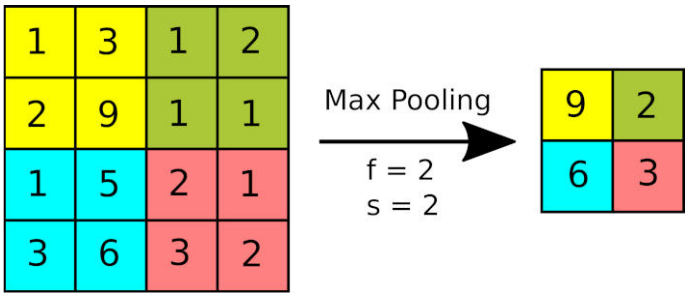


FIGURE 2.2: Max-pooling operation [3]

Another crucial component in CNNs is the pooling operation. Pooling layers play a vital role in reducing the size of the extracted feature maps, retaining only the most significant features. The most common type of pooling layers used is max-pooling. The pooling operation is applied in a manner similar to the convolution operator, involving the sliding of a small matrix of size $f \times f$ across the input image with a stride of s. Padding is typically not used. The max-pooling operation involves taking the max value. An example of the max-pooling operation applied to a 4×4 matrix with $f = 2$ and $s = 2$ can be observed in Figure 2.2. Since there are no parameters to be learned in a pooling layer, it is often not considered when counting the total number of layers in a CNN. The reduction in the dimensions of the feature map contributes to a decrease in the number of parameters in the network and equips the model with partial translational invariance.

2.1.3 ReLu Layer as an Activation Function

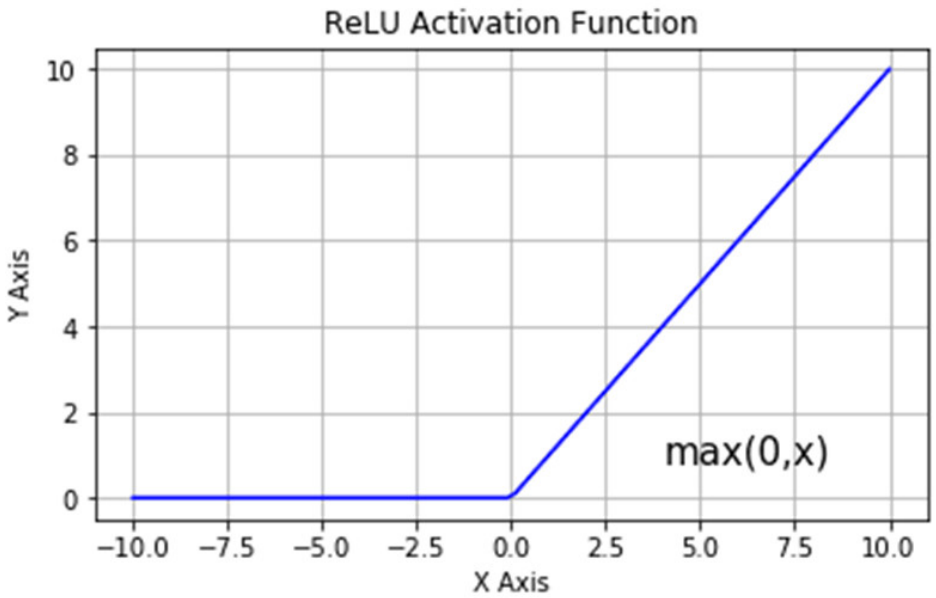


FIGURE 2.3: Rectified Linear Unit (ReLU) activation function graph

Convolution is a linear operator, but the relationship between the independent and dependent variables is probably non-linear [20]. To introduce non-linearity to this relationship, an activation function is applied to feature maps. One of the most commonly used and practical activation functions for CNNs is ReLU.

Mathematically, it can be expressed as $f(x) = max(0, x)$ (see Fig. 2.3). It is suggested to add the ReLU layer right after each convolutional layer except the last one. This practice enhances the flexibility and complexity of the model by introducing additional non-linearity.

2.1.4 Dropout Layer

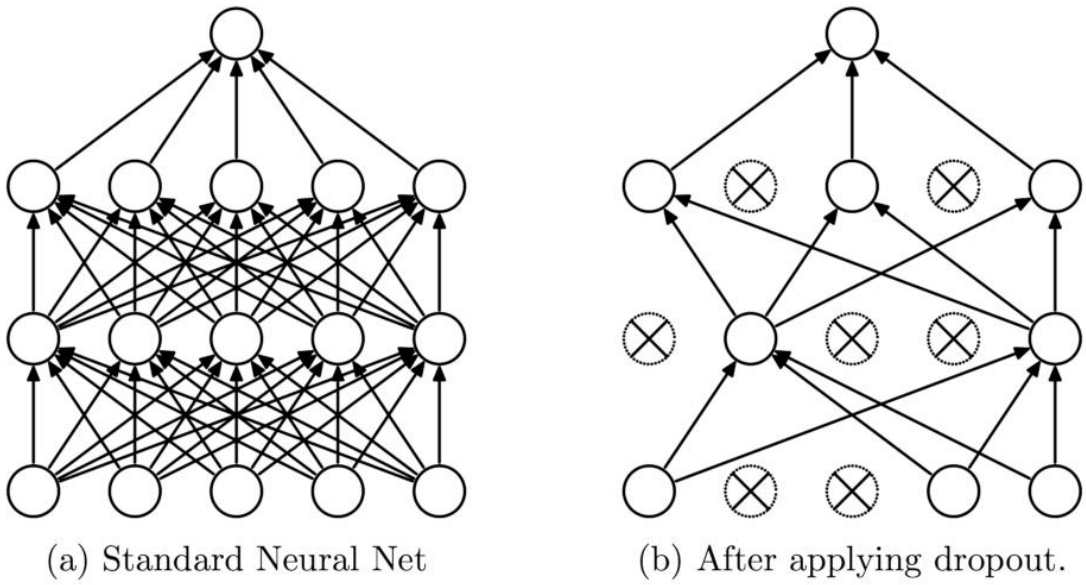


FIGURE 2.4: Difference in Neural Network with (a) and without (b) dropout method applied. Image taken from the original paper [4].

CNNs consist of several layers including convolution, pooling, and ReLU, with numerous parameters to be updated during the training process. However, the issue of overfitting may arise especially when dealing with a smaller training dataset. Overfitting leads to poor model prediction performance, making the model less effective on unseen data. To address this problem, dropout, introduced by Nitish Srivastava et al. [4], serves as a simple method to prevent DNNs from overfitting. Figure 2.4 visualizes the dropout concept where specific neurons in predefined layers are set to zero, effectively dropped out. It's crucial to note that this technique is applied only during the training process, and at test time, all neurons are used.

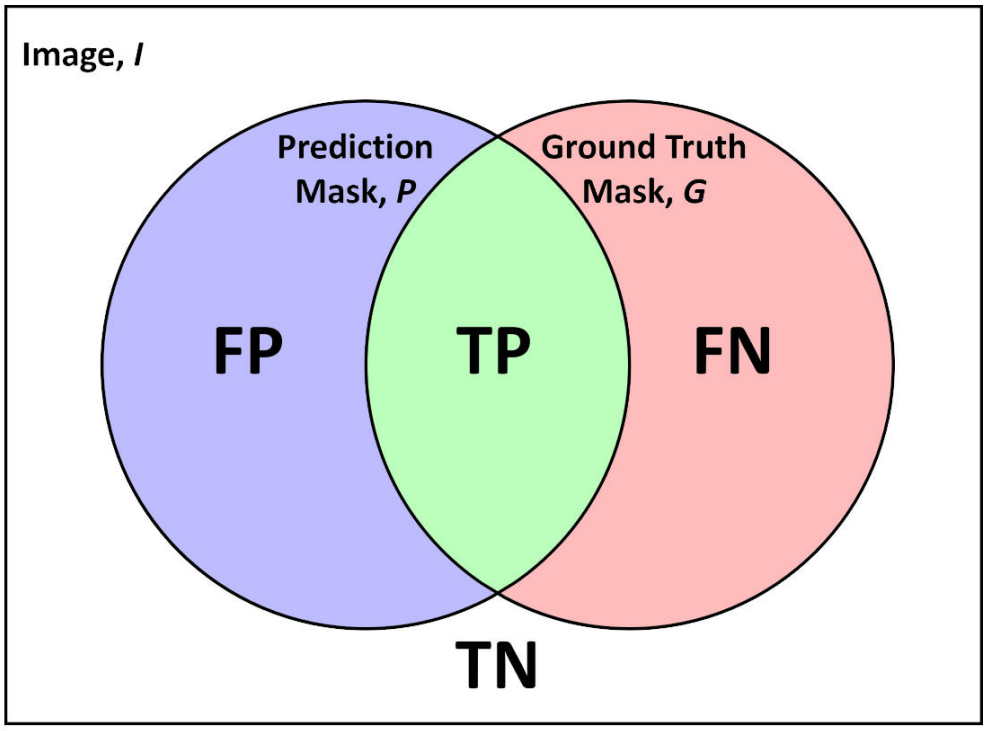


FIGURE 2.5: Ground truth and prediction binary masks on Image.

2.1.5 Loss Functions

In the process of training any DNNs, a loss function is minimized for successful learning. This loss function can be intuitively understood as the errors a model makes on the training dataset. It provides the model with an understanding of the discrepancy between the prediction and the ground-truth, enforcing the model to adjust its weights accordingly. A wide range of loss functions can be used to train CNNs, with the choice depending on the specific use case and the task at hand. Currently, the most commonly used loss functions for image segmentation are Cross-Entropy (CE), Dice, and Intersection over Union (IoU), which are extensively utilized in platforms such as Kaggle and in academic literature.

Let an image $\boldsymbol{I}$ exist in $\mathbb{R}^2$ or $\mathbb{R}^3$ domain. $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$ represent the ground-truth segmentation mask and prediction mask, respectively. y_i and $\hat{y}_i$ denote ground-truth segmentation and prediction probability for i^{th} voxel (a location/position in $\mathbb{R}^3$) respectively. N represents the total number of voxels in an image, and C represents the total number of classes in the segmentation task.

2.1.5.1 Categorical CE

CE is defined as a measure of dissimilarity between two probability distributions, specifically the predicted probability distribution generated by a machine learning model and the true probability distribution, represented by ground-truth data or labels. It stands as one of the most frequently preferred loss functions for classification tasks. Since segmentation tasks can be considered as binary classification at the pixel level, cross-entropy serves as a good choice.

Cross-Entropy is mathematically defined as:

$$L_{CE}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = -\frac{1}{N} \sum_{c=1}^{C} \sum_{i=1}^{N} y_i^c \log(\hat{y}_i^c) \tag{2.1}$$

Where y_i^c denotes the ground-truth binary segmentation value of class label c for i^{th} voxel, and $\hat{y}_i^c$ represents the prediction probability of class label c for i^{th} voxel. It can be adapted to binary classification tasks by taking C as 1. Then, it is called binary cross-entropy.

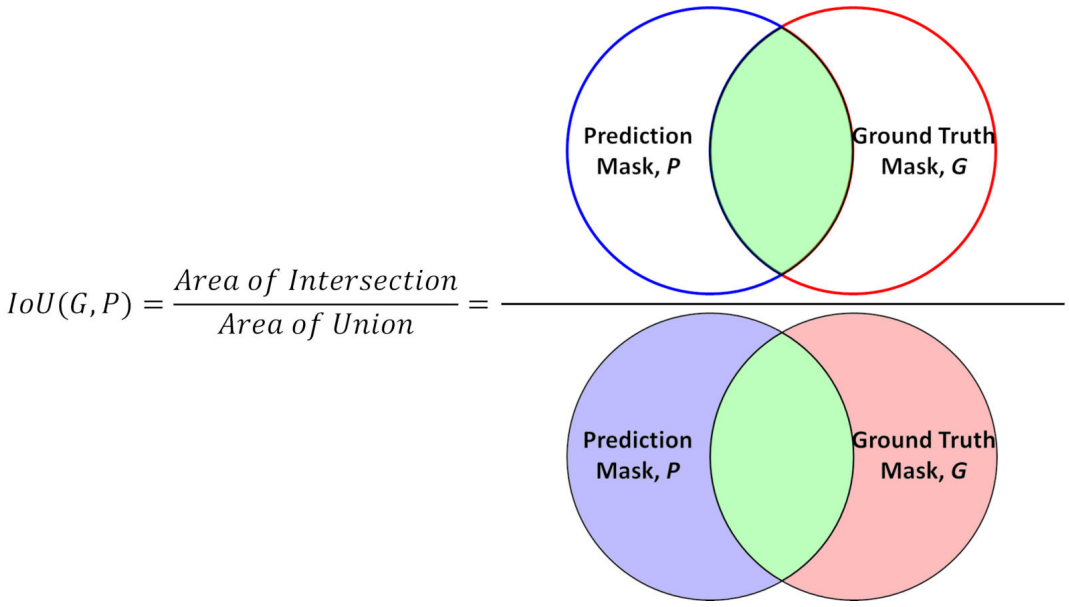


FIGURE 2.6: Computation visualization for IoU Metric.

2.1.5.2 Dice Loss

Dice Similarity Coefficient (DSC), also known as F1 score, is the most commonly used evaluation metric for image segmentation tasks. Dice has been proposed as a loss function to directly optimize DSC by Sudre et al. in 2016 [21].

$$L_{Dice}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = 1 - \frac{1 + 2\sum_{c=1}^{C}\sum_{i=1}^{N} y_i^c \hat{y}_i^c}{1 + \sum_{c=1}^{C}\sum_{i=1}^{N} y_i^c + \sum_{c=1}^{C}\sum_{i=1}^{N} \hat{y}_i^c} \tag{2.2}$$

Dice loss function is not defined when either the numerator or the denominator equals zero. To avoid the issue of undefined dice loss in such edge case scenarios, a value of 1 is added to both the numerator and denominator. Two specific edge-case scenarios result in undefined Dice loss when this adjustment is not made. The first edge case arises when a ground-truth segmentation mask is fully-empty i.e. $\sum_{c=1}^{C}\sum_{i=1}^{N} y_i^c = 0$. The second edge case takes place when a prediction mask is entirely empty i.e. $\sum_{c=1}^{C}\sum_{i=1}^{N} \hat{y}_i^c = 0$.

2.1.5.3 Intersection over Union / Jaccard Loss

IoU, also known as Jaccard similarity coefficient, stands as one of the most commonly used evaluation metrics for image segmentation problems. It is defined as the number of pixels in the intersection between the ground-truth segmentation mask and the prediction mask divided by the number of pixels in their union. IoU has been adapted as a loss function for the direct optimization of IoU by Atiqur and Yand in 2016 [22].

$$L_{IoU}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = 1 - \frac{1 + \sum_{c=1}^{C}\sum_{i=1}^{N} y_i^c \hat{y}_i^c}{1 + \sum_{c=1}^{C}\sum_{i=1}^{N} (y_i^c + \hat{y}_i^c - y_i^c \hat{y}_i^c)} \tag{2.3}$$

To guarantee the IoU loss function remains defined in edge case scenarios, a value of 1 is added to both the numerator and denominator. These edge-case scenarios are completely identical to those mentioned in Section 2.1.5.2.

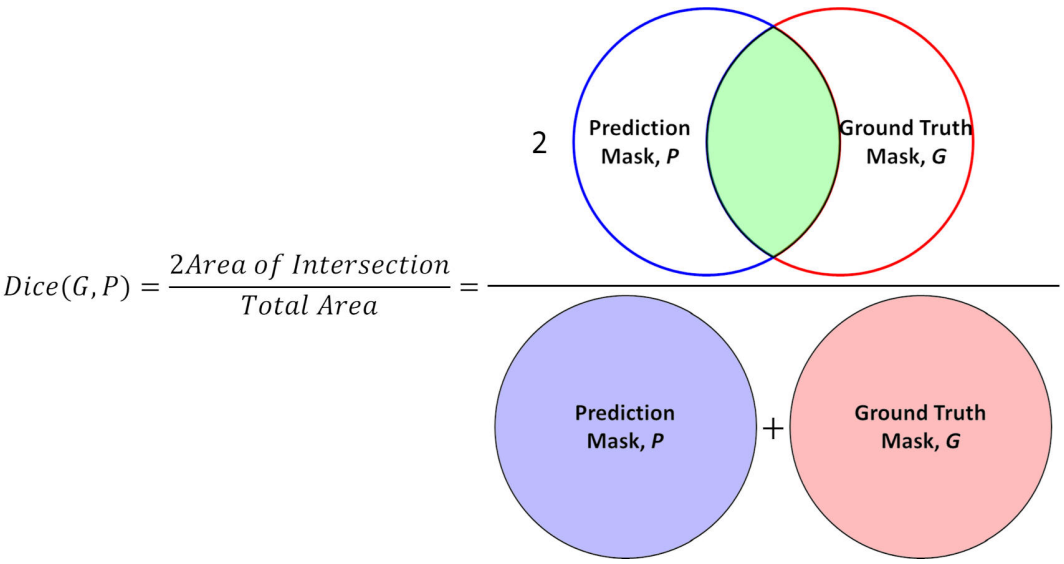


FIGURE 2.7: Computation visualization for Dice Metric.

2.1.5.4 Combo Loss

Combo loss, as proposed by Taghanaki et al. in 2019 [23], is defined as an equally weighted sum of both Dice Loss and Cross-Entropy.

$$L_{Combo}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = L_{Dice}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) + L_{CE}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) \tag{2.4}$$

where $L_{Dice}(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$ and $L_{CE}(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$ are defined in Eqs. (2.2) and (2.1), respectively.

2.1.5.5 Dice Loss and Focal Loss

Focal Loss [24] originated from Cross-Entropy is specifically designed to focus training on challenging data samples by assigning less weight to the classification of easy-to-learn data samples. A modulating factor, $(1 - \hat{y}_i^c)^\gamma$ where γ represents a tunable focusing parameter, is introduced to dynamically adjust the weights in accordance with the learning difficulty of data samples. Thus, it is well-suited for scenarios with high class imbalances. Focal Loss is defined as below:

$$L_{Focal}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \overbrace{-\frac{1}{N}\sum_{c=1}^{C}\sum_{i=1}^{N} y_i^c \log(\hat{y}_i^c)}^{L_{CE}(\boldsymbol{Y},\hat{\boldsymbol{Y}})} \underbrace{(1-\hat{y}_i^c)^{\gamma}}_{modulating factor} \tag{2.5}$$

When γ is set to 0, Focal Loss becomes Cross-Entropy. In the Focal Loss paper, the best performance is achieved when $\gamma = 2$.

A hybrid loss is utilized as an equally weighted combination of both Dice Loss and Focal Loss to overcome highly imbalanced organ segmentation task [25].

$$L_{Dice\&Focal}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = L_{Dice}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) + L_{Focal}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) \tag{2.6}$$

where $L_{Dice}(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$ and $L_{Focal}(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$ are defined in Eqs. (2.2) and (2.5), respectively.

2.1.5.6 Application of Distance Transformation in Loss Calculation

The application of distance transformation in loss computation can be categorized into two distinct methods, both of which do not exclusively rely on distance transform-based loss. The first approach integrates the distance transformation of the ground-truth mask as an additional component to the primary loss function. Hoel Kervadec et al. [26] proposed a boundary loss in the form of a distance metric to address significant imbalances in datasets. This boundary loss was integrated with standard regional losses such as Dice, cross-entropy, and focal loss during their experiments. In 2022, Chi Wang et al [27] introduced an active boundary loss for semantic segmentation to improve a better alignment between predicted and ground-truth boundaries during the training process. However, this active boundary loss is not used independently; it is an additional term to the overall loss, the sum of cross-entropy and IoU.

The second approach involves DNNs that concurrently predict the ground-truth mask and its corresponding signed distance map. A regression loss, such as the

mean square error between the distance-transformed ground-truth mask and the predicted distance-transformed mask is added into the regular loss function such as Dice loss. This method is generally applied in the medical imaging domain, as evidenced in [28], [29], [30], [31], [32], [33], [34], [35], and [36].

TABLE 2.1: Notation Table

$\boldsymbol{G}$	$\triangleq$	Ground truth binary mask
$\boldsymbol{P}$	$\triangleq$	Prediction binary mask
d	$\triangleq$	Pixel width of the boundary region
$\boldsymbol{G}_d$	$\triangleq$	Ground truth boundary mask: Set of pixels within a distance d from ground truth binary mask
$\boldsymbol{P}_d$	$\triangleq$	Prediction boundary mask: Set of pixels within a distance d from prediction binary mask
FP	$\triangleq$	False positives
TP	$\triangleq$	True positives
FN	$\triangleq$	False negatives
TN	$\triangleq$	True negatives
FP_d	$\triangleq$	False positives within a distance d
TP_d	$\triangleq$	True positives within a distance d
FN_d	$\triangleq$	False negatives within a distance d
TN_d	$\triangleq$	True negatives within a distance d

2.1.6 Evaluation Metrics

$$IoU(\boldsymbol{G}, \boldsymbol{P}) = \frac{|\boldsymbol{G} \cap \boldsymbol{P}|}{|\boldsymbol{G} \cup \boldsymbol{P}|} = \frac{TP}{FP + TP + FN} \tag{2.7}$$

$$Dice(\boldsymbol{G}, \boldsymbol{P}) = \frac{2|\boldsymbol{G} \cap \boldsymbol{P}|}{|\boldsymbol{G}| + |\boldsymbol{P}|} = \frac{2TP}{FP + 2TP + FN} \tag{2.8}$$

$$bIoU(\boldsymbol{G}_d, \boldsymbol{P}_d) = \frac{|\boldsymbol{G}_d \cap \boldsymbol{P}_d|}{|\boldsymbol{G}_d \cup \boldsymbol{P}_d|} = \frac{TP_d}{FP_d + TP_d + FN_d} \tag{2.9}$$

Evaluation metrics play a crucial role in assessing the performance of deep learning models. The most common metric is accuracy, defined as the ratio of correct predictions to the total number of predictions. However, in the context of semantic segmentation, accuracy may not serve as the most effective performance measure

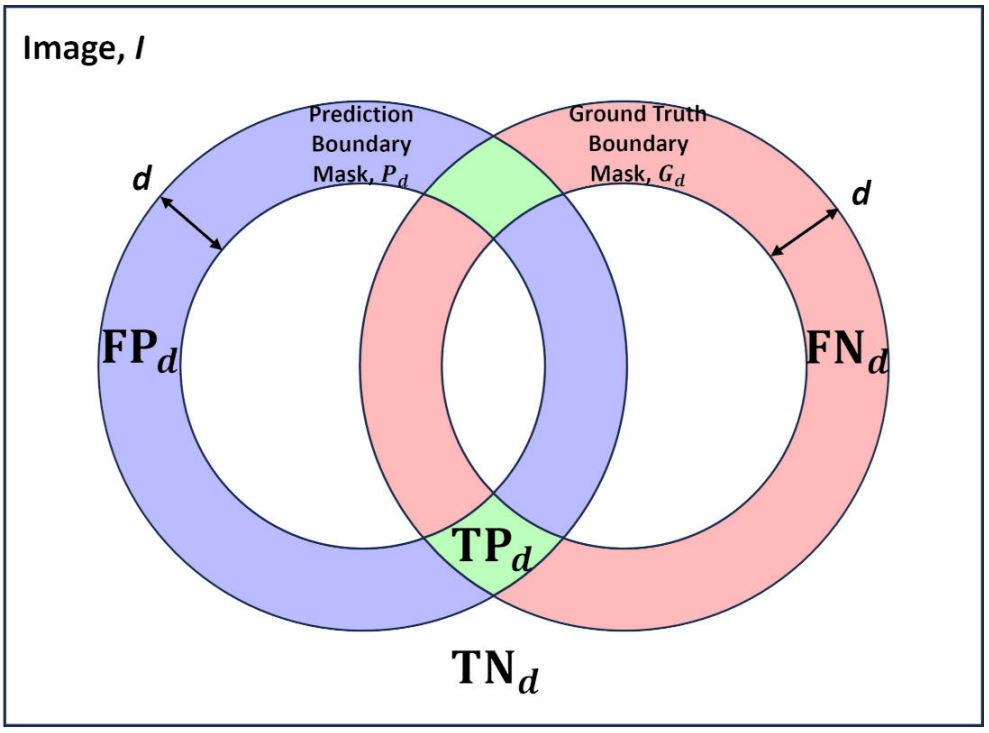


FIGURE 2.8: Ground truth and prediction binary masks within a distance d on Image.

due to the high likelihood of unbalanced class frequency in segmentation datasets. In such datasets, a model may demonstrate high accuracy by performing well only in the most frequent class without learning anything about the remaining classes. Therefore, for semantic image segmentation tasks, the DSC (i.e. Dice), IoU, and Boundary IoU (bIoU) are considered standard metrics. While the Dice and IoU are discussed in Sections 2.1.5.2 and 2.1.5.3, respectively, bIoU is explained in the subsequent chapter. These three metrics are evaluated over the entire dataset for all classes. The mean of each metric is calculated over all the classes.

For the visualization and formulation of these evaluation metrics, the notations used in this paper are presented in Table 2.1. We visualize the ground truth binary mask ($\boldsymbol{G}$) and prediction binary mask ($\boldsymbol{P}$) alongside the image ($\boldsymbol{I}$) in Figure 2.5. The definitions of Dice and IoU, along with their corresponding illustrations for computation, are provided in Equations (2.8) and (2.7), and Figures 2.7 and 2.6, respectively.

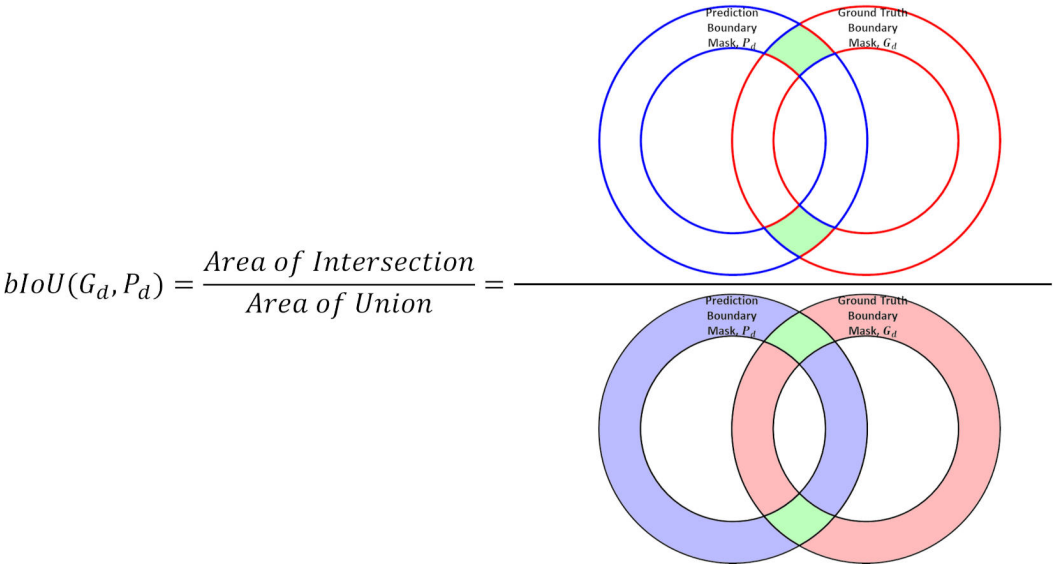


FIGURE 2.9: Computation visualization for Boundary IoU Metric.

Boundary IoU, as proposed by Cheng et al. in 2021 [37], represents a boundary-sensitive image segmentation evaluation metric, specifically designed to capture the boundary quality of objects in images. When applied to two masks, $\boldsymbol{G}$ and $\boldsymbol{P}$, Boundary IoU initially calculates the set of pixels within a distance of d pixels from each mask's contour. These sets are abbreviated as $\boldsymbol{G}_d$ and $\boldsymbol{P}_d$, respectively. Subsequently, it computes the IoU between $\boldsymbol{G}_d$ and $\boldsymbol{P}_d$, as visually illustrated in Figure 2.9. The visualization of ground truth boundary mask ($\boldsymbol{G}_d$) and prediction boundary mask ($\boldsymbol{P}_d$) alongside the image ($\boldsymbol{I}$) can be observed in Figure 2.8. The formula for bIoU is provided in Eq. (2.9).

In addition to these metrics, we introduce a novel evaluation criterion, Superiority bIoU (%), which represents the percentage of achieving the best bIoU score for the experiment with corresponding loss among all epochs. This criterion is specifically applied to the Cityscapes validation dataset, as the performance of Superiority bIoU (%) on the Cityscapes validation dataset can be calculated after the entire training process is completed.

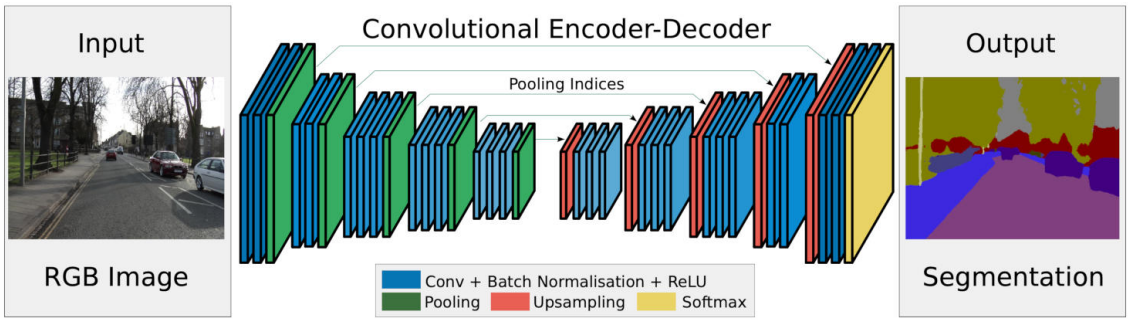


FIGURE 2.10: The visualization of the SegNet, encoder-decoder model architecture from the original paper [5]

2.2 Encoder-Decoder Model Architecture

The encoder-decoder modeling architectures are commonly employed in semantic segmentation tasks, featuring both an encoder and a decoder. Classic image semantic segmentation algorithms such as DeepLab [38], U-net [6], and FCN [39] are all built based on the encoder-decoder structure. Typically, the encoder is a network from a family of models such as ResNet [40], EfficientNet [41], VGG [42], and ResNeXt [43], which includes a deconvolution layer and an upper sampling layer. Downsampling is employed to capture semantic information, while upsampling is used to recover spatial information. Figure 2.10 illustrates the structure of encoder-decoder networks, as seen in the SegNet architecture [5].

2.2.1 Backbone Models

The two commonly preferred backbone model architectures for image recognition and segmentation are utilized to justify the effectiveness of our proposed loss function: U-Net and DeepLabv3+.

2.2.1.1 U-Net Architecture

The U-Net model architecture [6], developed by Ronneberger et al. in 2015, is a CNN specifically designed for image segmentation tasks. As shown in Fig. 2.11,

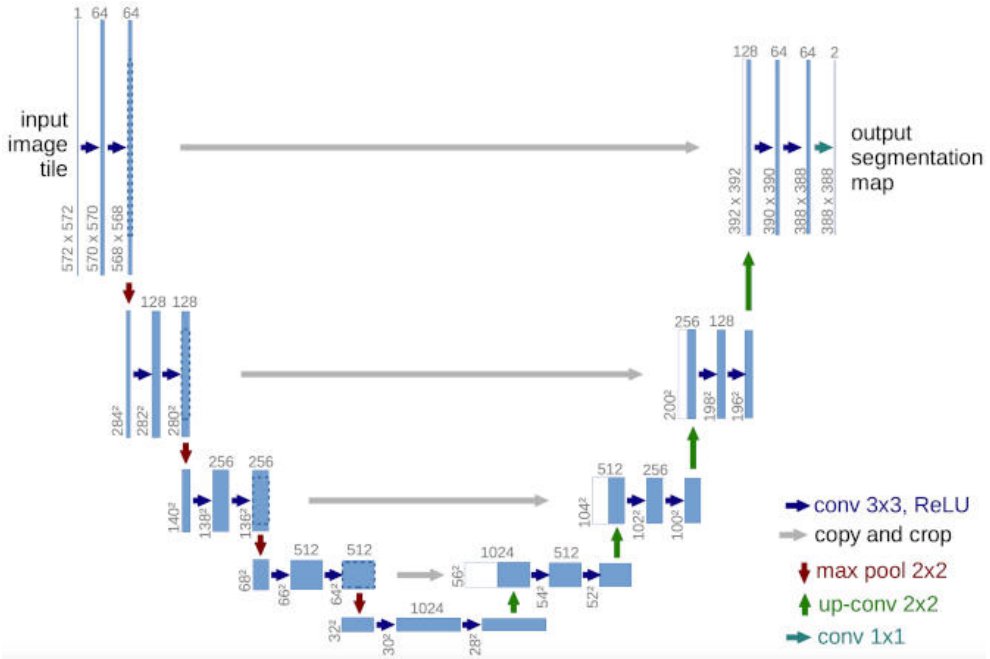


FIGURE 2.11: Original U-Net model architecture as presented in the original paper [6]

the architecture has a U-shaped structure, which consists of a symmetric encoder-decoder network and skip connections. The encoder network extracts features from input images, while the decoder network, with the assistance of skip connections, generates fine-grained segmentation masks by facilitating the flow of information between the encoder and decoder.

2.2.1.2 DeepLabv3+ Architecture

The DeepLabv3+ model architecture [7] is another convolutional neural network, but it is specifically designed for semantic image segmentation. As depicted in Figure 2.12, it extends the original DeepLab model architecture [38] by integrating the Atrous Spatial Pyramid Pooling module. This module enables the extraction of multi-scale contextual information from images. The model is composed of an encoder-decoder network and skip connections, which help to identify fine-grained details within images. It is this encoder-decoder network structure that sets DeepLabv3 [44] apart from DeepLabv3+.

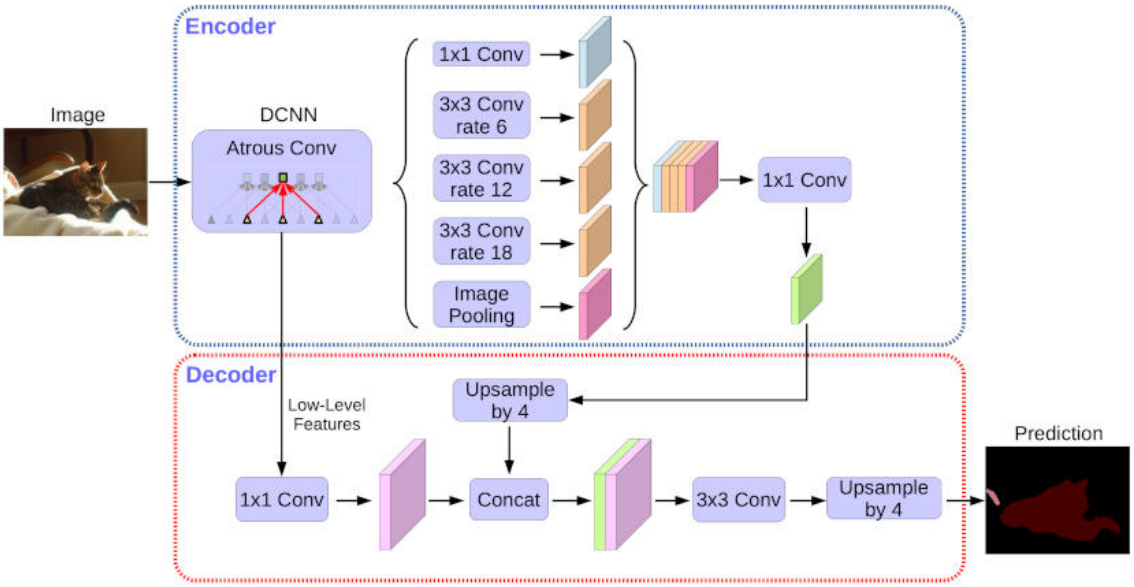


FIGURE 2.12: Original DeepLabv3+ model architecture as presented in the original paper [7]

2.2.2 Encoders

The three most common encoder architectures are utilized to indicate the adaptability and effectiveness of our distance-based loss function across diverse network architectures: ResNet-34 [40], ResNet-50, and MobileNetV2 [45].

2.3 Discussion

The related work presented in this thesis provides a comprehensive overview of the key components and methodologies in the field of semantic segmentation using Convolutional Neural Network (CNN). The field is characterized by rapid progress, prioritizing the development of models that are both accurate and efficient in learning from complex image data, rather than the advancement in new segmentation-specific loss functions.

An in-depth analysis of CNN components, loss functions, and evaluation metrics reveals that the field has evolved to address specific challenges such as feature extraction, non-linearity, overfitting, and the evaluation of segmentation quality.

The use of various loss functions, such as categorical cross-entropy, Dice loss, IoU loss, and their combinations, reflects the ongoing efforts to optimize model performance, particularly in the presence of class imbalances and the need for precise segmentation at object boundaries.

Several problems have been addressed in detail, such as the extraction of hierarchical features through convolutional layers, the introduction of non-linearity with activation functions like ReLU, and the prevention of overfitting with dropout strategy. Encoder-decoder architectures have been successful in capturing both semantic and spatial information, which is crucial for accurate segmentation.

However, not all issues have been resolved. One major problem that persists is the handling of highly imbalanced datasets, where certain classes are underrepresented. This can lead to models that perform well on dominant classes but fail to accurately segment less frequent classes. Additionally, the precise segmentation of object boundaries remains a challenge, as segmentation models may struggle to capture the fine details of object contours, which is critical in applications such as medical image analysis.

The major problems in the field include the need for more effective loss functions that can handle class imbalances and improve boundary segmentation, the development of models that can generalize well on unseen data, and the reduction of computational complexity to enable the deployment of segmentation models on devices with limited processing power.

In conclusion, while significant progress has been made in the field of semantic segmentation, there are still some research directions for improvement, particularly in the areas of class imbalance, boundary segmentation, and model efficiency. Future research should focus on these focal points to develop more robust and versatile segmentation models.

Chapter 3

Proposed Loss

In this chapter, we introduce our proposed loss function, designed for seamless integration into any semantic segmentation model, ensuring respect for geometric constraints in object segments. Initially, we describe the inspiration behind the idea of our loss function, followed by an in-depth exploration of the mathematical framework and a comprehensive explanation of the proposed methodology.

3.1 Motivation

The loss function, also referred to as the objective function, is used as a measure of a model's performance, steering the predictions we aim for the model to produce. The selection of an appropriate loss function for semantic segmentation is a complex task, as different tasks require different loss functions. The cross-entropy loss function is typically employed for training models for image classification. Given that segmentation can be perceived as a pixel-wise classification problem, the pixel-wise cross-entropy loss function is the most intuitive choice for segmentation. This function treats each pixel with equal importance, averaging the performance across all pixels. However, this loss function has several drawbacks, such as data imbalance issues. In addition to cross-entropy, other loss functions like

[1] https://www.freepik.com/free-photo/pizza-pizza-filled-with-tomatoes-salami-olives_6087618.htm

(A) Overlay of ground-truth mask on input image sourced from [1]

(B) Segmentation prediction mask 1: Polygonal shape predicted as Pizza

(C) Segmentation prediction mask 2: pizza enclosed within the rectangle predicted mask

(D) Segmentation prediction mask 3: predicted mask enclosed within the pizza

FIGURE 3.1: Illustration of three segmentation predictions (1, 2, and 3). Prediction 1 exhibits a polygonal boundary, while predictions 2 and 3 have rectangular boundaries.

IoU and Dice are frequently used for semantic segmentation. However, these commonly used loss functions often overlook the unique shape characteristics of each segmented region. Furthermore, they lack shape-awareness, leading to incomplete object coverage, and they tend to disregard geometric constraints within object segments.

Consider a simple example where the image contains a pizza, and the ground truth segmentation mask is an ellipse as depicted in Figure 3.1. Assume there are three

different predictions 1, 2, and 3 as shown in Figure 3.1. Prediction 1 has a polygonal boundary but mostly aligns with the ground-truth mask. Both predictions 2 and 3 are rectangles. Prediction 2 encompasses the entire ground-truth mask with many wrong predictions outside of pizza, i.e. false positive predictions. Prediction 3 could not predict all pixels of pizza correctly, i.e. there are many false negatives, and it also has many false positives. Despite all three predictions largely aligning with the ground-truth mask, except for boundaries, our loss function penalizes more for the boundary shape of the object by assigning higher values, as shown in Table 3.1. Since defining suitable loss functions is a crucial issue for segmentation models and can assist the model in respecting the shapes and geometry of objects, we proposed a distance transform-based loss function that regards the unique shape characteristics of each segmented objects.

TABLE 3.1: Comparisons of different loss function values with the ground-truth mask and three distinct predictions, as illustrated in Figure 3.1.

	Prediction 1	Prediction 2	Prediction 3
Cross-Entropy	0.565	0.619	0.597
IoU	0.037	0.214	0.166
Dice	0.019	0.120	0.091
Ours	0.056	0.296	0.238

3.2 Proposed Distance Transform-based Loss

Distance-based loss is specifically designed to enhance segmentation details along the object boundaries and to obey geometric constraints of objects by applying the distance transform method and additional processes to ground-truth segmentation masks. In general, loss functions utilize ground-truth binary segmentation, $\boldsymbol{Y}$, and prediction probability, $\hat{\boldsymbol{Y}}$, for their mathematical formulations. Our loss calculation is also based on prediction probability masks and ground-truth segmentation masks that undergo the distance transform method and additional processes.

For simplicity in explanation, let's assume there is just a single class in ground-truth segmentation masks. All seven steps can be seen in detail as an example

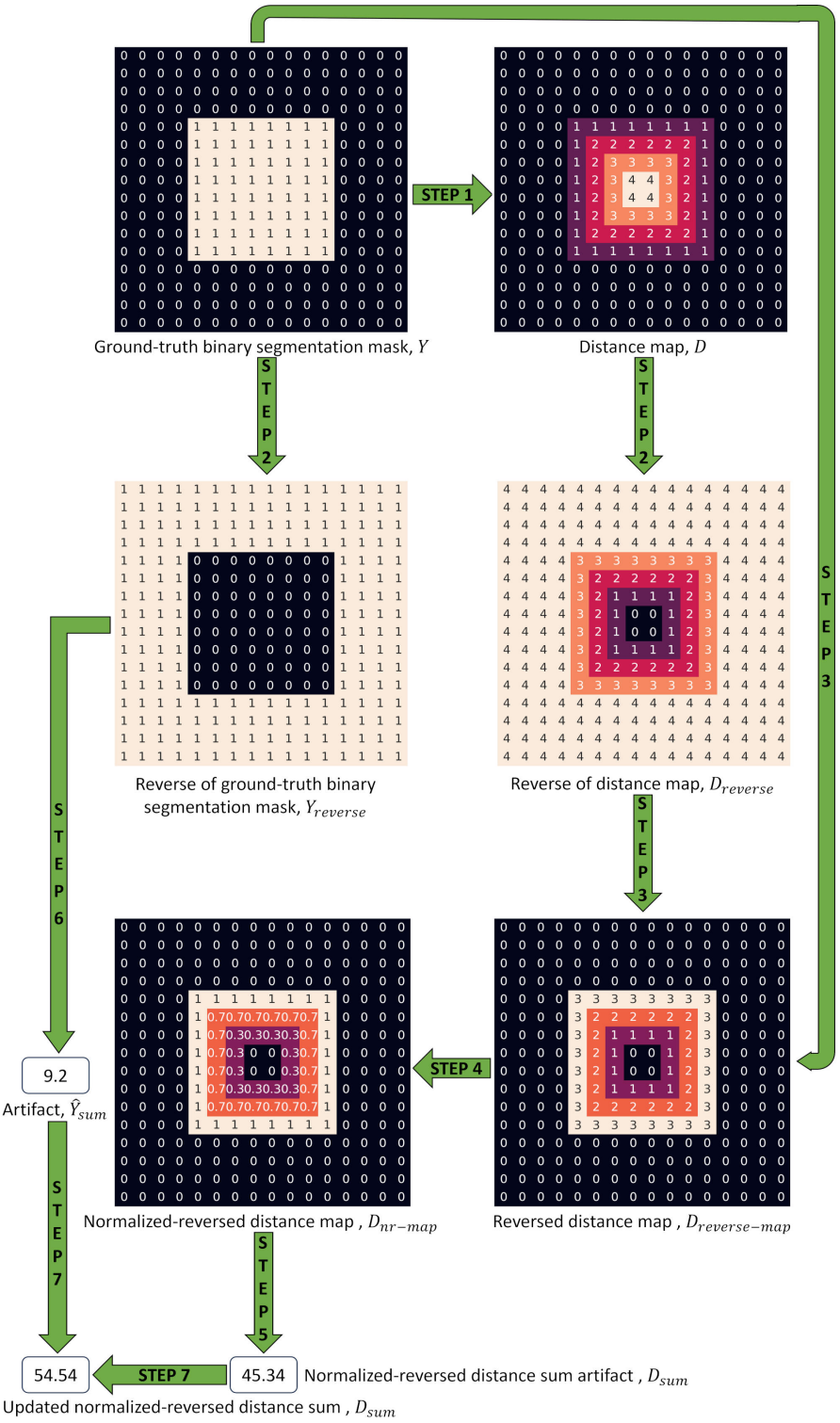


FIGURE 3.2: Example visualization of algorithmic pipeline: it takes two inputs, $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$, and produces two outputs: D_{sum} and $\boldsymbol{D}_{nr-map}$.

applied to the ground-truth binary segmentation mask in Figure 3.2. The entire process, including the distance transformation applied to the ground-truth segmentation masks, is outlined in the following seven steps. This algorithmic

pipeline takes two inputs, the ground-truth binary segmentation mask and the prediction probability, and produces two outputs: a normalized-reversed distance map, $\boldsymbol{D}_{nr-map}$, and a normalized-reversed distance sum, D_{sum}.

Step 1: Apply distance transformation to ground-truth binary segmentation mask, $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$, where m and n are the dimensions of the image, to obtain corresponding distance map, $\boldsymbol{D} \in \mathbb{R}^{m \times n}$.

Step 1.1: Classify pixels in the ground-truth binary segmentation mask into two sets: the foreground set, $\mathscr{S}_f$, and the background set, $\mathscr{S}_b$.
Where $\mathscr{S}_f = \{i \mid y_i = 1\}$ and $\mathscr{S}_b = \{i \mid y_i = 0\}$. i denotes a pixel (x, y) in $\mathbb{R}^2$.

Step 1.2: Generate the distance map $\boldsymbol{D}$ by finding each pixel, d_i. Firstly, calculate d_i pixel value where $i \in \mathscr{S}_f$ as the minimum Euclidean distance from this pixel to every pixel in $\mathscr{S}_b$:
$d_i = \min\{ED(i, i_\alpha) \mid i_\alpha \in \mathscr{S}_b, i \in \mathscr{S}_f\}$
Secondly, calculate d_i pixel value where $i \in \mathscr{S}_b$ as $d_i = \{0 \mid i \in \mathscr{S}_b\}$.
ED represents Euclidean distance between two pixel locations in $\mathbb{R}^2$.

Step 2: Take the reverse of the distance map and ground-truth binary segmentation mask; note that the maximum pixel value in $\boldsymbol{Y}$ is 1, and max represents an element-wise operation:
$\boldsymbol{D}_{reverse} = \max\{\boldsymbol{D}\} - \boldsymbol{D}$,
$\boldsymbol{Y}_{reverse} = 1 - \boldsymbol{Y}$.

Step 3: Generate the reversed distance map by making pixels that are outside of objects in $\boldsymbol{D}_{reverse}$ zero, $\times$ represents an element-wise operation:
$\boldsymbol{D}_{reversed-map} = \boldsymbol{D}_{reverse} \times \boldsymbol{Y}$

Step 4: Apply the min-max scaling algorithm to normalize the reversed distance map; note that the minimum pixel value in $\boldsymbol{D}_{reversed-map}$ is 0, and max represents an element-wise operation:
$\boldsymbol{D}_{nr-map} = \frac{\boldsymbol{D}_{reversed-map}}{max\{\boldsymbol{D}_{reversed-map}\}}$

Step 5: Sum all pixel values in the normalized-reversed distance map, $\boldsymbol{D}_{nr-map}$; note that sum represents an element-wise operation:

$D_{sum} = sum\{\boldsymbol{D}_{nr-map}\}$

Step 6: Sum all pixel values in $\boldsymbol{Y}_{reverse} \times \hat{\boldsymbol{Y}}$; note that sum represents an element-wise operation:

$\hat{Y}_{sum} = sum\{\boldsymbol{Y}_{reverse} \times \hat{\boldsymbol{Y}}\}$, where $\hat{\boldsymbol{Y}} \in \mathbb{R}^{m \times n}$ is the predicted binary segmentation mask.

Step 7: Add these two scalar values to update D_{sum}:

$D_{sum} = D_{sum} + \hat{Y}_{sum}$

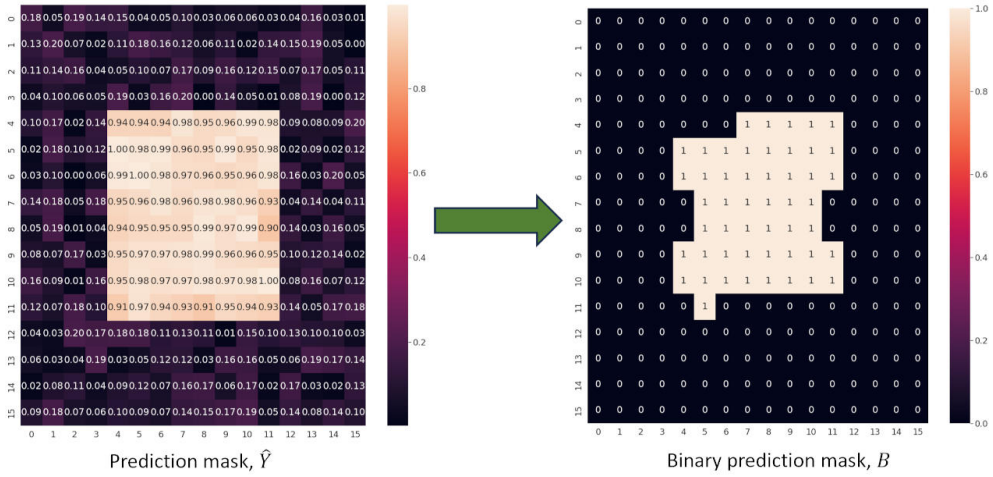


FIGURE 3.3: Example visualization of applying a threshold of 0.95 to $\hat{\boldsymbol{Y}}$: it takes $\hat{\boldsymbol{Y}}$ as a single input and produces $\boldsymbol{B}$ as the output.

Subsequently, a prediction mask ($\hat{\boldsymbol{Y}}$), a probability distribution generated by a machine learning model, is transformed into a binary prediction mask ($\boldsymbol{B}$) by applying a threshold of 0.95. Note that this threshold is larger than the value of 0.5, which is usually preferred, to ensure confident predictions. This process is visually demonstrated with an example applied to a prediction mask in Figure 3.3. Finally, the two components: a normalized-reversed distance map, $\boldsymbol{D}_{nr-map}$ and a normalized-reversed distance sum, D_{sum} are utilized to calculate the distance loss, as defined below:

$$L_{Dist}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = 1 - \sum_{c=1}^{C} \frac{\sum_{i=1}^{N} (d_{nr-map})_i^c \hat{y}_i^c}{\sum_{i=1}^{N} (d_{sum})_i^c} \tag{3.1}$$

Here, $(d_{nr-map})_i^c$ represents the normalized-reversed distance value of class label c for i^{th} voxel, $\hat{y}_i^c$ denotes the prediction probability value of class label c for i^{th} voxel, and $(d_{sum})_i^c$ stands for the normalized-reversed distance scalar sum value of class label c for i^{th} voxel.

3.3 Computational and Memory Cost Analysis

The computational and memory cost of the proposed distance transform-based loss function is an important aspect to consider, especially when dealing with large-scale datasets and complex models. The computational cost is primarily associated with the mathematical operations involved in the calculation of the loss function, while the memory cost is related to the storage requirements of the intermediate results and final outputs.

The proposed loss function involves several mathematical operations, including distance transformation, reverse operation, element-wise multiplication, min-max scaling, and summation. The distance transformation operation, which is applied to the ground-truth binary segmentation mask, is the most computationally intensive part of the process. This operation involves calculating the minimum Euclidean distance from each pixel in the foreground set to every pixel in the background set. The computational complexity of this operation is $O(N^2)$, where N is the number of pixels in the image. However, efficient algorithms exist for distance transformation that can reduce the computational complexity to $O(N)$, [46], [47], and [48].

The reverse operation, element-wise multiplication, and min-max scaling are all element-wise operations and have a computational complexity of $O(N)$, where N is the number of pixels in the image. The summation operation, which is used to calculate the sum of all pixel values in the normalized-reversed distance map and the sum of all pixel values in the product of the reversed ground-truth

binary segmentation mask and the prediction probability, also has a computational complexity of $O(N)$.

In terms of memory cost, the proposed loss function requires storage for the ground-truth binary segmentation mask, the distance map, the reversed distance map, the normalized-reversed distance map, and the final normalized-reversed distance sum value. The storage requirements for these elements are proportional to the size of the image, with the exception of the final normalized-reversed distance sum value, which is a scalar. Therefore, the memory cost of the proposed loss function is $O(N)$, where N is the number of pixels in the image.

In conclusion, the proposed distance-based loss function has a reasonable computational and memory cost, making it suitable for use in semantic segmentation models. However, it is important to note that the actual computational and memory costs may vary depending on the specific implementation and hardware used.

Chapter 4

Experiments and Results

This chapter primarily focuses on an in-depth exploration of the experiments, including the datasets and the experimental setup. Subsequently, we will present the results and engage in a comprehensive discussion.

4.1 Datasets

This study used two distinct image segmentation datasets to evaluate the effect of the proposed distance transform-based loss function: CelebAMask-HQ and Cityscapes. While both CelebAMask-HQ and Cityscapes datasets correspond to multi-class segmentation challenges, we will convert CelebAMask-HQ into a single-class segmentation (i.e. binary segmentation that assigns one of two classes to each pixel) dataset to validate our hypothesis for both binary and multi-class segmentation in the experiments. Each dataset possesses its unique characteristics, outlined briefly in Sections 4.1.1 and 4.1.2.

4.1.1 CelebAMask-HQ

CelebAMask-HQ [49] is a large-scale facial dataset with high-quality mask annotations. This dataset has fine-grained annotations for 30000 facial images, split into

FIGURE 4.1: A sample from the CelebAMask-HQ dataset is transformed from multi-class segmentation to single-class/binary segmentation, face label.

23200 training images, 5800 validation images, and 1000 test images. All images are 512×512 pixels resolution. The CelebAMask-HQ dataset has 19 label classes, including not only facial components but also accessories such as skin, cloth, hat, hair, eyebrows (right and left), eyes (right and left), ears (right and left), earrings (right and left), eyeglass, nose, lip (upper and lower), mouth, neck, and necklace.

Both CelebAMask-HQ and Cityscapes are multi-class image segmentation datasets. In our pursuit to demonstrate the efficacy of distance loss in both single-class and multi-class image segmentation, all 19 label classes in the CelebAMask-HQ dataset are unified into a single-class, specifically face label, as shown in Figure 4.1. Consequently, the CelebAMask-HQ is used as a single-class image segmentation dataset in our research.

4.1.2 Cityscapes

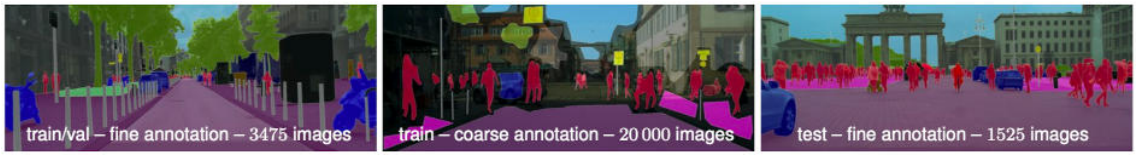


FIGURE 4.2: Examples from train, validation, and test datasets for both fine and course annotation as presented in the original paper [8]

Cityscapes is a large-scale, high-resolution dataset for semantic understanding of urban scenes [8]. This dataset has high-quality, pixel-level fine annotations for 2975 training images, 500 validation images, and 1525 test images as demonstrated in

Figure 4.2. Additionally, it includes 20000 coarsely annotated images, which are not used in this paper. All images in this dataset are in 1024×2048 pixels resolution. The Cityscapes dataset consists of 19 distinct classes such as road, sidewalk, building, wall, fence, pole, traffic light, traffic sign, vegetation, terrain, sky, person, rider, car, truck, bus, train (i.e. on rails), motorcycle, and bicycle. As the test images and their corresponding masks are private and not publicly available, only the results of the validation dataset will be reported in our experiments.

4.2 Experimental Setup

In this section, we explain the setup of our experiments, including data pre-processing and model training details.

4.2.1 Data Pre-processing

Prior to training all models with both datasets (Cityscapes and CelebAMask-HQ), a pre-processing step was performed for each dataset. It is best to use original/source images and masks for training, validation, and testing phases without changing their original sizes. This means avoiding resizing or cropping, ensuring no loss of information during the training phase. However, due to the large resolution of original images and their corresponding masks (512×512 pixels for CelebAMask-HQ and 1024×2048 pixels for Cityscapes), it demands a significant amount of GPU memory (VRAM). Even with a high-memory GPU, using the original sizes of images and masks would considerably increase the training time required to achieve satisfactory results. Certain model architectures, such as U-Net, will not train unless the resolutions (both height and width) of images and masks are divisible by a downsampling factor of 32.

Therefore, our data preprocessing consists of three steps for all training, validation, and testing phases. Firstly, the largest size (height or width) of images and masks is set to 256 pixels for CelebAMask-HQ and 512 pixels for Cityscapes while

maintaining the aspect ratio. Secondly, images and masks are padded with borders if needed to ensure that all images and masks have a size of 256×256 pixels for CelebAMaskHQ and 512×512 pixels for Cityscapes. Thirdly, images are normalized using the mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225) of Imagenet [50] dataset, a common practice due to the large scale of the dataset. Apart from these three steps, no data augmentation is applied during our experiments since we want to observe the effect of only loss functions and not data augmentation methods.

4.2.2 Implementation Details

We used the Segmentation Models (smp) codebase[1] for training all models. In addition to two distinct types of model architectures: U-Net and DeepLavV3+, we used pre-trained encoders of ResNet-34, ResNet-50, and MobileNetV2 on the ImageNet dataset and randomly initialized the decoders of models. Throughout the training process, no data augmentation was applied, as explained in the previous section. We trained all models using AdamW[51] optimizer for 60 epochs with a batch size of 32 for both CelebAMask-HQ and Cityscapes datasets. The learning rate was set to a constant value of 0.0003, and no learning rate scheduler was applied to isolate the impact of hyperparameters other than the loss function choice in our experiments. Distance maps, D, were calculated using the standard SciPy function[2]. Seed values for all libraries, including Python, Numpy [52], and PyTorch [53], were set to 42 for all experiments, except those within the ablation study. We report the semantic segmentation performance of our experiments using bIoU, IoU, and Dice for both datasets, along with the percentage superiority of bIoU for the Cityscapes dataset.

[1] `https://github.com/qubvel/segmentation_models.pytorch`

[2] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.distance_transform_cdt.html`

4.2.3 Hardware Environment

The models and training code were implemented in Python 3.7.13 and PyTorch 1.11.0, running on Linux. The experiments were conducted using 2 Tesla T4 GPUs provided in Kaggle's free-tier environment. These GPUs, having 15 GB VRAM in total, were used for all experiments involving the CelebAMask-HQ dataset to assess the performance of the loss functions. Note that Kaggle upgraded their 2 Tesla T4's VRAM to 30 GB in total at the time of writing my thesis. Additionally, for training both U-Net and DeepLabv3+ models with Cityscapes dataset, NVIDIA A100 80GB PCIe (80 GB VRAM in total) on Azure server (Standard_NC24ads_A100_v4[3] size with 24 vCPUs and 220 GB RAM) was used, as this dataset required a larger VRAM capacity than CelebAMask-HQ to fit into GPU memory. In both Kaggle and Azure platforms, GPUs were configured to train using half-precision (16-bit) to optimize and accelerate the training process in our experiments.

4.3 Results

In total, there are seven experiments with two different model architectures (U-Net and DeepLabv3+) and three distinct encoders (ResNet-34, ResNet-50, and MobileNetV2). The results of these experiments are consolidated in Tables 4.1, 4.2, 4.3, 4.4, 4.5, and 4.6. In this context, bIoU, IoU, and CE are abbreviations for Boundary Intersection over Union, Intersection over Union, and Cross-Entropy, respectively. Each of the seven experiments involves CE loss, IoU loss, Dice loss, Combo loss, Dice & Focal loss, and our proposed distance transform loss (as defined in 3.1). In our experiments, we aim to compare the effects of six loss functions on various evaluation metrics: bIoU, IoU, and Dice scores. Given that bIoU offers a superior evaluation of object boundary quality in segmentation tasks, it is used as the primary metric for comparisons across our experiments.

[3] `https://learn.microsoft.com/en-us/azure/virtual-machines/nc-a100-v4-series`

TABLE 4.1: The performance of the U-Net model with a ResNet-34 encoder, trained using the proposed distance transform-based loss and other loss functions, on the validation dataset of Cityscapes and the test dataset of CelebAMask-HQ.

U-Net Architecture with ResNet-34 Encoder							
	Cityscapes				**CelebAMask-HQ**		
Loss Functions	bIoU	IoU	Dice	Superiority bIoU (%)	bIoU	IoU	Dice
Cross-Entropy	0.5008(+0.0769)	0.6943(+0.0443)	0.8145(+0.0293)	15(+53.33)	0.6932(+0.0513)	0.9633(+0.0068)	0.9809(+0.0036)
IoU	0.5216(+0.0561)	0.6873(+0.0513)	0.8105(+0.0333)	10(+58.33)	**0.7451**(-0.0006)	**0.9704**(-0.0003)	**0.9846**(-0.0001)
Dice	0.5157(+0.062)	0.6859(+0.0527)	0.809(+0.0348)	5(+63.33)	0.7438(+0.0007)	0.9693(+0.0008)	0.984(+0.0005)
Combo	0.4217(+0.156)	0.623(+0.1156)	0.763(+0.0808)	0(+68.33)	0.7053(+0.0392)	0.9646(+0.0055)	0.9816(+0.0029)
Dice & Focal	0.4454(+0.1323)	0.6452(+0.0934)	0.7781(+0.0657)	1.67(+66.66)	0.6715(+0.073)	0.9607(+0.0094)	0.9796(+0.0049)
Ours	**0.5777**	**0.7386**	**0.8438**	**68.33**	0.7445	0.9701	0.9845

4.3.1 Results for Single-Class/Binary Segmentation using CelebAMask-HQ Dataset

Tables 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 present the evaluation criteria, including bIoU, IoU, and Dice scores for our experiments on the CelebAMask-HQ test dataset. Our distance transform-based loss notably enhances segmentation performance, resulting in significantly improved bIoU, IoU, and Dice scores.

In the performance on the CelebAMask-HQ dataset, our distance transform-based loss ranks third three times, ranks second twice, and ranks first once in the DeepLabv3+ with MobileNetV2 experiment, as depicted in Table 4.6. In general, models trained with our loss outperform those trained with Dice & Focal loss, CE loss, and Combo loss by a considerably large margin, ranging from 0.0018 (as observed in the comparison between Combo loss and our loss in Table 4.4) to 0.0119 (as observed in the comparison between Dice & Focal loss and our loss in Table 4.6) in terms of bIoU metric. However, models trained with Dice loss and IoU loss occasionally demonstrate slightly better performance than those trained with our loss, with margins ranging from 0.0001 (as observed in the comparison between IoU loss and our loss in Table 4.1) to 0.0005 (as observed in the comparison between Dice loss and our loss in Table 4.3) in terms of the bIoU score. Therefore, while our loss demonstrates robust performance compared to other loss functions, it is surpassed only in the fourth significant figure in terms of bIoU metric.

TABLE 4.2: The performance of the U-Net model with a ResNet-50 encoder, trained using the proposed distance transform-based loss and other loss functions, on the validation dataset of Cityscapes and the test dataset of CelebAMask-HQ.

U-Net Architecture with ResNet-50 Encoder							
	Cityscapes				**CelebAMask-HQ**		
Loss Functions	bIoU	IoU	Dice	Superiority bIoU (%)	bIoU	IoU	Dice
Cross-Entropy	**0.6001**(-0.0155)	**0.7639**(-0.0141)	**0.8632**(-0.01)	26.67(+25)	0.6777(+0.0655)	0.9613(+0.0082)	0.9799(+0.0042)
IoU	0.5622(+0.0224)	0.7305(+0.0193)	0.8399(+0.0133)	13.33(+38.34)	0.7426(+0.0006)	0.9686(+0.0009)	0.9836(+0.0005)
Dice	0.5503(+0.0343)	0.725(+0.0248)	0.8367(+0.0165)	3.33(+48.34)	**0.745**(-0.0018)	**0.9698**(-0.0003)	**0.9843**(-0.0002)
Combo	0.4766(+0.108)	0.683(+0.0668)	0.8075(+0.0457)	1.67(+50)	0.6997(+0.0435)	0.9644(+0.0051)	0.9815(+0.0026)
Dice & Focal	0.4762(+0.1084)	0.6664(+0.0834)	0.7939(+0.0593)	3.33(+48.34)	0.6568(+0.0864)	0.9589(+0.0106)	0.9786(+0.0055)
Ours	0.5846	0.7498	0.8532	**51.67**	0.7432	0.9695	0.9841

4.3.2 Results for Multi-Class Segmentation using Cityscapes Dataset

To evaluate the efficacy of our distance transform-based loss in multi-class segmentation challenges, we conducted training and validation using the same models and encoders as in the Cityscapes dataset. Tables 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 present the evaluation criteria, including bIoU, IoU, Dice scores, and superiority bIoU in percentage for our experiments on the Cityscapes validation dataset.

Our distance transform-based loss consistently ranks first in terms of superiority bIoU (%) across all the seven experiments, with the exception of the U-Net with MobileNetV2 encoder, as indicated in Table 4.3. As demonstrated in Figure 4.7, among the seven experiments, our loss ranked first four times, while IoU loss, Dice loss, and CE loss ranked first once in terms of bIoU score. However, when compared with the best-performing loss for the experiments involving the U-Net model, our loss underperformed by 0.0155 and 0.129 in terms of bIoU score, as shown in Tables 4.2 and 4.3 respectively. Despite this, our loss significantly outperformed other loss functions in terms of bIoU for the experiments utilizing the DeepLabv3+ model architecture. Our loss functions ranked first for the experiment with the ResNet-34 encoder (Table 4.4) and the ResNet-50 encoder (Table 4.5) and third for the experiment with the MobileNetV2 encoder (Table 4.6). In Table 4.4, our loss exceeded the second-best loss, IoU, by 0.0198 in terms of bIoU score. In Table 4.5, our loss surpassed the second-best loss, Dice, by 0.0323 in terms of bIoU score. However, our loss underperformed the best-performing loss, Dice, by 0.029 in terms of bIoU in Table 4.6. When considering the experiments where IoU, Dice, CE, and our loss ranked first, our loss exhibited a greater performance boost than other losses in the bIoU score. Our loss demonstrated superior performance with the DeepLabv3+ model compared to the U-Net model in terms of bIoU score and consistently excelled in terms of superiority bIoU (%).

TABLE 4.3: The performance of the U-Net model with a MobileNetV2 encoder, trained using the proposed distance transform-based loss and other loss functions, on the validation dataset of Cityscapes and the test dataset of CelebAMask-HQ.

U-Net Architecture with MobileNetV2 Encoder							
	Cityscapes				**CelebAMask-HQ**		
Loss Functions	bIoU	IoU	Dice	Superiority bIoU (%)	bIoU	IoU	Dice
Cross-Entropy	0.4692(-0.0587)	0.6316(-0.0585)	0.7714(-0.0458)	5(**+38.33**)	0.6389(**+0.1052**)	0.9561(**+0.0135**)	0.9771(**+0.0071**)
IoU	**0.5395**(-0.129)	**0.7197**(-0.1466)	**0.8339**(-0.1083)	36.67(**+6.66**)	0.7452(-0.0011)	0.9699(-0.0003)	0.9843(-0.0001)
Dice	0.4236(-0.0131)	0.5944(-0.0213)	0.743(-0.0174)	1.67(**+41.66**)	**0.7455**(-0.0014)	**0.9705**(-0.0009)	**0.9847**(-0.0005)
Combo	0.4435(-0.033)	0.6659(-0.0928)	0.7959(-0.0703)	1.67(**+41.66**)	0.6635(**+0.0806**)	0.9583(**+0.0113**)	0.9782(**+0.006**)
Dice & Focal	0.4786(-0.0681)	0.6733(-0.1002)	0.7998(-0.0742)	11.67(**+31.66**)	0.5907(**+0.1534**)	0.9483(**+0.0213**)	0.973(**+0.0112**)
Ours	0.4105	0.5731	0.7256	**43.33**	0.7441	0.9696	0.9842

TABLE 4.4: The performance of the DeepLabv3+ model with a ResNet-34 encoder, trained using the proposed distance transform-based loss and other loss functions, on the validation dataset of Cityscapes and the test dataset of CelebAMask-HQ.

DeepLabv3+ Architecture with ResNet-34 Encoder							
	Cityscapes				**CelebAMask-HQ**		
Loss Functions	bIoU	IoU	Dice	Superiority bIoU (%)	bIoU	IoU	Dice
Cross-Entropy	0.3846(**+0.1101**)	0.5908(**+0.0845**)	0.7389(**+0.0625**)	6.67(**+70**)	0.6907(**+0.0425**)	0.9628(**+0.0057**)	0.9807(**+0.0029**)
IoU	0.4749(**+0.0198**)	0.6703(**+0.005**)	0.7981(**+0.0033**)	8.33(**+68.34**)	0.7341(-0.0009)	0.969(-0.0005)	**0.9839**(-0.0003)
Dice	0.4331(**+0.0616**)	0.631(**+0.0443**)	0.7691(**+0.0323**)	3.33(**+73.34**)	**0.7367**(-0.0035)	**0.9691**(-0.0006)	**0.9839**(-0.0003)
Combo	0.3761(**+0.1186**)	0.6316(**+0.0437**)	0.7703(**+0.0311**)	5(**+71.67**)	0.7046(**+0.0286**)	0.9649(**+0.0036**)	0.9818(**+0.0018**)
Dice & Focal	0.4182(**+0.0765**)	0.6235(**+0.0518**)	0.7618(**+0.0396**)	0(**+76.67**)	0.6741(**+0.0591**)	0.9603(**+0.0082**)	0.9793(**+0.0043**)
Ours	**0.4947**	**0.6753**	**0.8014**	**76.67**	0.7332	0.9685	0.9836

4.3.3 GPU Memory and Run-time for the Entire End-to-End Process

In addition to comparing the performance of different loss functions, it is also beneficial to examine the GPU memory usage and run-time for the entire process, from data loading to the end of the testing phase.

Firstly, we investigated the result of the experiment of binary segmentation where the DeepLabv3+ models are used together with the ResNet-50 encoder on the CelebAMaskHQ dataset on Kaggle platform. As shown in Figure 4.3, there were no significant spikes in the memory usage of GPUs (note that two GPUs on Kaggle: GPU 0 and GPU 1 were used) during the training phase for the binary segmentation task. For GPU 1, the percentage of allocated GPU memory remained consistent at 28.1% across all loss functions. However, our loss function utilized the least memory in GPU 0, at 49.32%, which was 2.0% less than the second-best loss function in terms of memory usage. As depicted in Figure 4.4, there were no significant differences in run-time; almost all loss functions performed similarly, with the exception of Dice & Focal loss, which underperformed.

Secondly, we focused on a multi-class segmentation case using the Cityscapes dataset, where both the U-Net model and ResNet-50 encoder were utilized on an Azure server. Contrary to binary segmentation experiments, there were crucial spikes in GPU memory usage for multi-class segmentation, as seen in Figure 4.5. This is a critical issue as full GPU memory usage can halt the training process. It is worth noting that these spikes may stem from the implementation in smp codebases. Therefore, it is advisable to use loss functions that do not cause any spikes, such as Dice & Focal loss, Dice loss, and our loss. Among these three, our loss function required the least GPU memory, using 2.4% less than the second-best loss function, Dice. However, the run-time for our loss, IoU loss, and Dice & Focal loss was relatively longer than the run-time for Dice loss, CE loss, and Combo loss.

TABLE 4.5: The performance of the DeepLabv3+ model with a ResNet-50 encoder, trained using the proposed distance transform-based loss and other loss functions, on the validation dataset of Cityscapes and the test dataset of CelebAMask-HQ.

DeepLabv3+ Architecture with ResNet-50 Encoder							
	Cityscapes				**CelebAMask-HQ**		
Loss Functions	bIoU	IoU	Dice	Superiority bIoU (%)	bIoU	IoU	Dice
Cross-Entropy	0.41(**+0.1194**)	0.6402(**+0.0713**)	0.777(**+0.0508**)	3.33(**+83.34**)	0.6848(**+0.0515**)	0.9622(**+0.0069**)	0.9803(**+0.0036**)
IoU	0.4409(**+0.0885**)	0.6329(**+0.0786**)	0.7696(**+0.0582**)	6.67(**+80**)	**0.7388**(-0.0025)	0.9693(-0.0002)	0.984(-0.0001)
Dice	0.4971(**+0.0323**)	0.6938(**+0.0177**)	0.8144(**+0.0134**)	0(**+86.67**)	0.7387(-0.0024)	**0.9697**(-0.0006)	**0.9843**(-0.0004)
Combo	0.3986(**+0.1308**)	0.6268(**+0.0847**)	0.7674(**+0.0604**)	3.33(**+83.34**)	0.6995(**+0.0368**)	0.9648(**+0.0043**)	0.9817(**+0.0022**)
Dice & Focal	0.4779(**+0.0515**)	0.6702(**+0.0413**)	0.7971(**+0.0307**)	0(**+86.67**)	0.6596(**+0.0767**)	0.9588(**+0.0103**)	0.9786(**+0.0053**)
Ours	**0.5294**	**0.7115**	**0.8278**	**86.67**	0.7363	0.9691	0.9839

TABLE 4.6: The performance of the DeepLabv3+ model with a MobileNetV2 encoder, trained using the proposed distance transform-based loss and other loss functions, on the validation dataset of Cityscapes and the test dataset of CelebAMaskHQ.

DeepLabv3+ Architecture with MobileNetV2 Encoder							
	Cityscapes				**CelebAMask-HQ**		
Loss Functions	bIoU	IoU	Dice	Superiority bIoU (%)	bIoU	IoU	Dice
Cross-Entropy	0.3916(**+0.1126**)	0.6137(**+0.0718**)	0.7574(**+0.0515**)	1.67(**+8.33**)	0.6354(**+0.0994**)	0.9557(**+0.0136**)	0.977(**+0.007**)
IoU	0.5116(-0.0074)	0.7192(-0.0337)	0.8326(-0.0237)	36.67(-26.67)	0.7344(**+0.0004**)	0.9689(**+0.0004**)	0.9838(**+0.0002**)
Dice	**0.5332**(-0.029)	**0.7296**(-0.0441)	**0.8402**(-0.0313)	**50**(-40)	0.7344(**+0.0004**)	0.9687(**+0.0006**)	0.9837(**+0.0003**)
Combo	0.4047(**+0.0995**)	0.6465(**+0.039**)	0.7817(**+0.0272**)	1.67(**+8.33**)	0.667(**+0.0678**)	0.9594(**+0.0099**)	0.9789(**+0.0051**)
Dice & Focal	0.4399(**+0.0643**)	0.651(**+0.0345**)	0.7831(**+0.0258**)	0(**+10**)	0.5822(**+0.1526**)	0.9466(**+0.0227**)	0.9721(**+0.0119**)
Ours	0.5042	0.6855	0.8089	10	**0.7348**	**0.9693**	**0.984**

Hence, our loss function consistently surpassed others in terms of GPU memory usage without any spikes observed across all experiments. This is a significant factor, especially when training larger neural networks on big datasets. However, it's important to note that there was no clear winner in terms of run-time among the evaluated loss functions.

TABLE 4.7: A comprehensive table summarizing the results, indicating the number of experiments in which each loss function performed at its best.

	Cityscapes				**CelebAMask-HQ**		
Loss Functions	bIoU	IoU	Dice	Superiority bIoU (%)	bIoU	IoU	Dice
Cross-Entropy	1	1	1	0	0	0	0
IoU	1	2	2	1	2	1	2
Dice	1	1	1	0	**3**	**5**	**5**
Combo	0	0	0	0	0	0	0
Dice & Focal	0	0	0	0	0	0	0
Ours	**4**	**3**	**3**	**6**	2	1	1

TABLE 4.8: Legend Table for Figures 4.3 and 4.5

Red color and dashed one	$\triangleq$	GPU 0 and GPU 1 Memory Allocated (%) for Distance transform-based loss respectively
Green color and dashed one	$\triangleq$	GPU 0 and GPU 1 Memory Allocated (%) for IoU loss respectively
Blue color and dashed one	$\triangleq$	GPU 0 and GPU 1 Memory Allocated (%) for Dice & Focal loss respectively
Purple color and dashed one	$\triangleq$	GPU 0 and GPU 1 Memory Allocated (%) for Dice loss respectively
Orange color and dashed one	$\triangleq$	GPU 0 and GPU 1 Memory Allocated (%) for Cross-Entropy loss respectively
Brown color and dashed one	$\triangleq$	GPU 0 and GPU 1 Memory Allocated (%) for Combo loss respectively

4.4 Ablation Study

4.4.1 Effect of Seed Value

In this subsection, we offer a detailed examination of the results obtained by varying the seed values for Python, Numpy, and PyTorch. While the seed value was

FIGURE 4.3: The percentage of allocated GPU memory for the experiment involving the DeepLabv3+ model with the ResNet-50 encoder on the CelebAMaskHQ dataset, conducted on Kaggle. Color legends are explained in Table 4.8.

set to 42 in our initial six experiments, we adjusted it to 1234 for the experiments presented in this section. This change was implemented to demonstrate the consistent success of our loss under different seed values. The ablation study is conducted on the CelebAMaskHQ and Cityscapes datasets with an ImageNet pre-trained DeepLabv3+ and ResNet-50 model. The results, evaluated using previously employed evaluation metrics, are presented in Table 4.12.

In Section 4.3.2, we examined the results of single-class segmentation with a seed value of 42 using the CelebAMaskHQ dataset. In the experiment with the DeepLabv3+ and ResNet-50 model and a seed value of 42, our loss ranked third in terms of all three evaluation metrics: bIoU, IoU, and Dice as shown in Table 4.5.

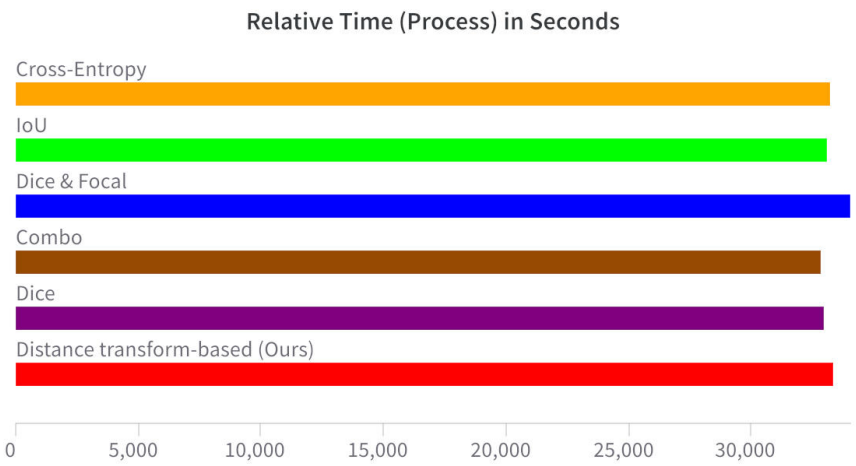


FIGURE 4.4: The total run-time for the entire process including data loading, training, and testing phases for the experiment involving the DeepLabv3+ model with the ResNet-50 encoder on the CelebAMaskHQ dataset, conducted on Kaggle

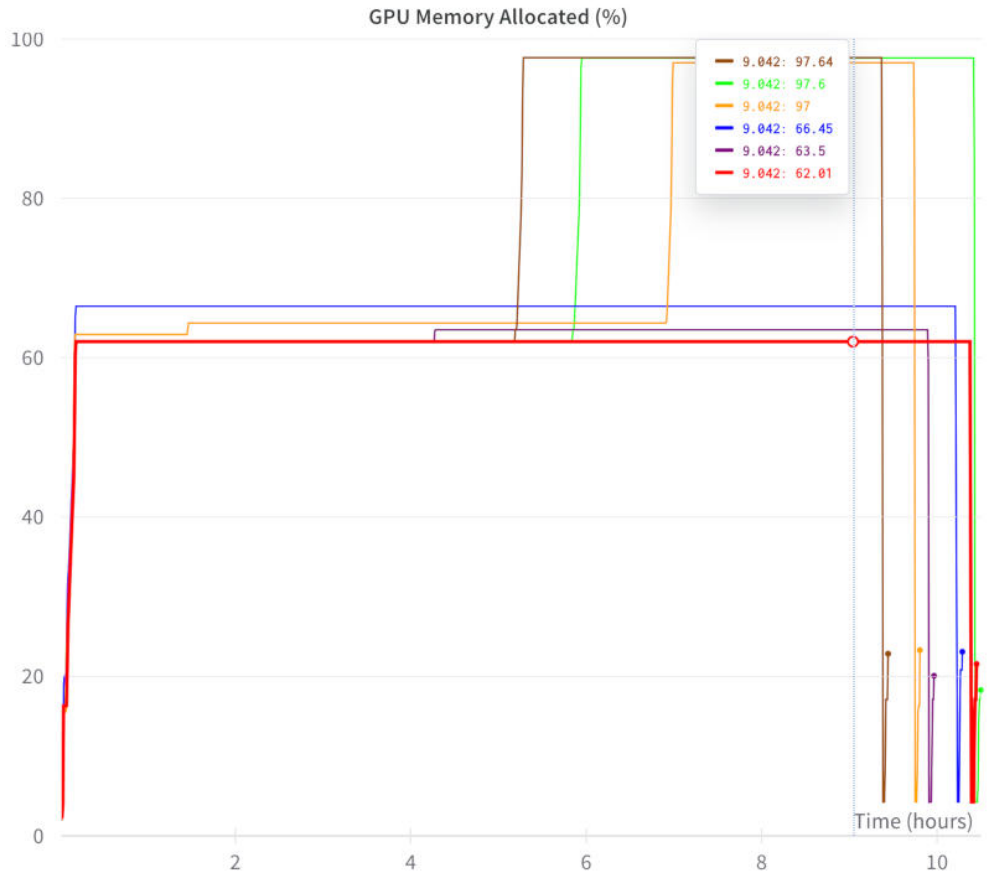


FIGURE 4.5: Allocated GPU memory in percentage for the experiment involving the U-Net model with the ResNet-50 encoder on the Cityscapes dataset, carried out on Azure Server. Color legends are explained in Table 4.8.

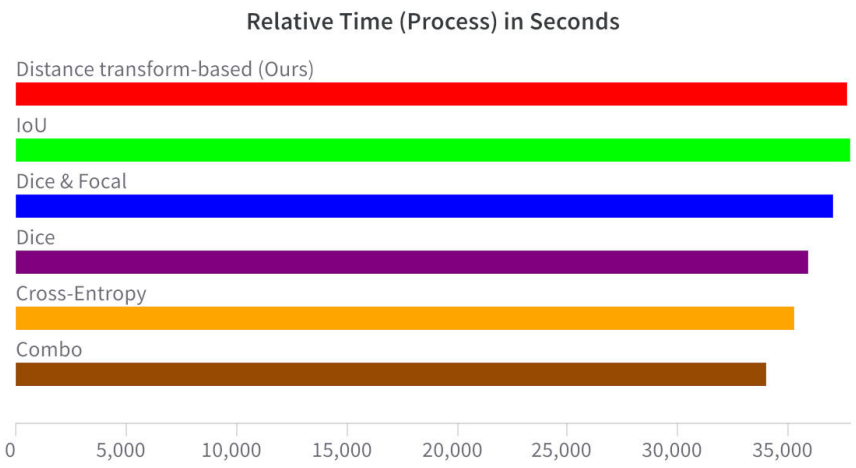


FIGURE 4.6: Run-time for the whole process including data loading, training, and testing for the experiment including the U-Net model with the ResNet-50 encoder on the Cityscapes dataset, conducted on Azure Server with A100 GPU

When the same model architecture and encoder, with all training implementation settings identical except for the seed value, are utilized in the ablation study, our loss ranks first for bIoU score and second for IoU and Dice, as depicted in Table 4.12. Therefore, changing the seed from 42 to 1234 for the corresponding experiment with the CelebAMaskHQ dataset resulted in a performance boost for our loss.

The result of the DeepLabv3+ and ResNet-50 model trained with the Cityscapes dataset using the seed value of 42 was examined under Section 4.3.2 with the assistance of Table 4.5. Our loss ranked first for all three evaluation metrics: bIoU, IoU, and Dice, as seen in 4.5. When using the seed value of 1234 instead of 42, our loss again ranked first for the bIoU score, as shown in Figure 4.12.

In summary, changing the seed value for one model architecture, the DeepLabv3+ and ResNet-50, with both the CelebAMaskHQ and Cityscapes datasets did not cause any decrease in the performance of our loss; it even demonstrated a performance increase for the experiment with the CelebAMaskHQ dataset.

TABLE 4.9: The number of training epochs required for training to achieve a convergence at 0.72 bIoU score on the CelebAMaskHQ validation dataset.

	Loss Functions					
Model & Encoder	CE*	IoU	Dice	Combo	Dice & Focal	Ours
U-Net & ResNet-34	-	**2**	5	-	-	**2**
U-Net & ResNet-50	-	8	6	-	-	**5**
U-Net & MobileNetV2	-	4	**3**	-	**-**	4
DeepLabv3+ & ResNet-34	-	11	15	-	-	**10**
DeepLabv3+ & ResNet-50	-	**8**	10	-	-	**8**
DeepLabv3+ & MobileNetV2	-	14	16	-	-	**10**
DeepLabv3+ & ResNet-50**	-	**6**	8	-	-	8

*Cross-Entropy **Experiment with the seed of 1234.

4.4.2 Convergence Performance Comparison of Loss Functions

This subsection focuses on comparing the convergence rates of different loss functions during training on the validation sets of the CelebAMaskHQ and Cityscapes datasets, using the bIoU metric as a benchmark. To ensure a fair comparison across all loss functions, we selected bIoU scores of 0.72 and 0.47 as convergence targets for the CelebAMaskHQ and Cityscapes datasets, respectively. As shown in Table 4.9, among seven experiments conducted on the CelebAMaskHQ dataset, our loss function achieved the fastest convergence to a bIoU score of 0.72 in five instances. The IoU loss function was the second-best performer, showing rapid convergence in three out of seven experiments. For the Cityscapes dataset, as demonstrated in Table 4.10, our loss function was the quickest to converge in four out of seven experiments, with Cross-Entropy being the closest competitor. However, Cross-Entropy only outperformed the others in two out of seven experiments. An examination of both tables suggests that our loss function is superior among the six evaluated loss functions in terms of convergence speed to the specified bIoU scores on the validation sets of both the CelebAMaskHQ and Cityscapes datasets.

TABLE 4.10: The number of training epochs required for training to achieve a convergence at 0.47 bIoU score on the Cityscapes validation dataset.

	Loss Functions					
Model & Encoder	CE*	IoU	Dice	Combo	Dice & Focal	Ours
U-Net & ResNet-34	**3**	20	43	-	-	31
U-Net & ResNet-50	**2**	32	33	56	36	46
U-Net & MobileNetV2	-	37	-	-	**11**	-
DeepLabv3+ & ResNet-34	-	50	-	-	-	**7**
DeepLabv3+ & ResNet-50	-	-	29	-	48	**15**
DeepLabv3+ & MobileNetV2	-	7	12	-	-	**5**
DeepLabv3+ & ResNet-50**	-	47	-	-	-	**12**

*Cross-Entropy **Experiment with the seed of 1234.

4.4.3 Cross-Validation

In this subsection, we employ k-fold cross-validation to further demonstrate the consistent performance of our loss function compared to other loss functions across different data splits. Due to computational constraints, we selected the U-Net model with the ResNet-50 encoder on the CelebAMaskHQ dataset for cross-validation, as it represents a commonly used architecture in image segmentation tasks. We conducted a 5-fold cross-validation, allocating 70% of the data for training, 10% for validation, and 20% for testing, following the methodology proposed by Kanta Miura et al. [54]. All other experimental parameters remained consistent with previous experiments.

Table 4.11 shows that our loss function achieved an average bIoU score of 0.7461 across the folds, which is a slight improvement over the 0.7432 score reported in the previous experiment (see Table 4.2). When comparing the average bIoU scores for all loss functions, our loss function exhibits the best performance. The second-highest average bIoU score is 0.7401, achieved by the IoU loss as seen in Table A.2. In terms of IoU and Dice scores, our loss function ranks second by a small margin. The comprehensive analysis in Table 4.11 confirms that our loss function stands out among the six evaluated loss functions in terms of all three evaluation scores across the 5-fold validation on the CelebAMaskHQ dataset for the U-Net model with a ResNet-50 encoder.

TABLE 4.11: Performance of the U-Net model with a ResNet-50 encoder using all loss functions: average and standard deviation of 5-fold cross-validation results on the CelebAMaskHQ test dataset.

Evaluation Metrics						
	bIoU		IoU		Dice	
Loss Functions	Avg.*	Std.*	Avg.	Std.	Avg.	Std.
Cross-Entropy	0.6786	0.0013	0.9609	0.0018	0.9766	0.0053
IoU	0.7401	0.0016	0.9647	0.0032	0.9771	0.0035
Dice	0.7362	0.0018	0.9691	0.0048	**0.9893**	0.0038
Combo	0.7040	0.0020	**0.9728**	0.0043	0.9838	0.0038
Dice & Focal	0.6592	0.0019	0.9643	0.0039	0.9830	0.0034
Ours	**0.7461**	**0.0012**	0.9701	**0.0016**	0.9889	**0.0030**

*Average **Standard deviation

Table 4.12: The performance of the DeepLabv3+ model with a ResNet-50 encoder with different SEED value, trained using the proposed distance transform-based loss and other loss functions, on the validation dataset of Cityscapes and the test dataset of CelebAMask-HQ.

DeepLabv3+ Architecture with ResNet-50 Encoder							
	Cityscapes				**CelebAMask-HQ**		
Loss Functions	bIoU	IoU	Dice	Superiority bIoU (%)	bIoU	IoU	Dice
Cross-Entropy	0.3327(**+0.1874**)	0.5233(**+0.1727**)	0.6844(**+0.133**)	5(**+66.67**)	0.6908(**+0.0471**)	0.9636(**+0.0057**)	0.9811(**+0.0029**)
IoU	0.5011(**+0.019**)	**0.7057**(-0.0097)	**0.8228**(-0.0054)	21.67(**+50**)	0.7371(**+0.0008**)	0.9692(**+0.0001**)	0.984(**0**)
Dice	0.4606(**+0.0595**)	0.6782(**+0.0178**)	0.8043(**+0.0131**)	0(**+71.67**)	0.7376(**+0.0003**)	**0.9695**(-0.0002)	**0.9842**(-0.0002)
Combo	0.3833(**+0.1368**)	0.6047(**+0.0913**)	0.7482(**+0.0692**)	0(**+71.67**)	0.7016(**+0.0363**)	0.965(**+0.0043**)	0.9818(**+0.0022**)
Dice & Focal	0.4615(**+0.0586**)	0.6636(**+0.0324**)	0.7927(**+0.0247**)	1.67(**+70**)	0.6572(**+0.0807**)	0.958(**+0.0113**)	0.9781(**+0.0059**)
Ours	**0.5201**	0.696	0.8174	**71.67**	**0.7379**	0.9693	0.984

4.5 Discussion

In this section, we delve into the examination of object boundaries segmented using both the best-performing loss function and our proposed loss function for each experiment. We conduct a comparative analysis of the qualitative results across various model networks in our experiments, focusing on the effectiveness of our proposed loss qualitatively.

4.5.1 Qualitative Evaluation

In Figures 4.7, 4.9, 4.11, 4.13, 4.15, 4.17, and 4.19, we present the ground-truth mask and the segmentation mask for a selection of samples from the CelebAMask-HQ dataset, trained using diverse model architectures and encoders. Correspondingly, Figures 4.8, 4.10, 4.12, 4.14, 4.16, 4.18, and 4.20 demonstrate examples from the Cityscapes dataset. The ground-truth mask and the segmentation mask predicted by the model are overlayed with the input image in all these Figures to facilitate a comprehensive visual assessment and effective comparisons.

When the single-class segmentation task with the CelebAMask-HQ dataset is considered, we observed that segmentation masks generated using our loss and the other loss exhibit striking similarities, particularly in terms of boundary quality. Compared loss functions demonstrate excellent performance in segmenting the single class, specifically the face label.

Upon reviewing the qualitative results on the Cityscapes dataset, it becomes evident that predicted segmentation masks exhibit improved conformity to geometric boundaries for objects when our loss is employed during training. For instance, in the initial image in Figure 4.16, our loss produced more accurately shaped predictions for the car in the red rectangle compared to the second-best loss, Dice, which generates fewer correctly labeled pixels in the boundary. This enhancement in the boundary of small and large objects persists across various images from the Cityscapes dataset, particularly when utilizing both the U-Net and the DeepLabv3+ model architectures.

Our loss Ground-truth IoU loss

FIGURE 4.7: **Qualitative results on CelebAMask-HQ** using the U-Net model with ResNet-34 encoder.

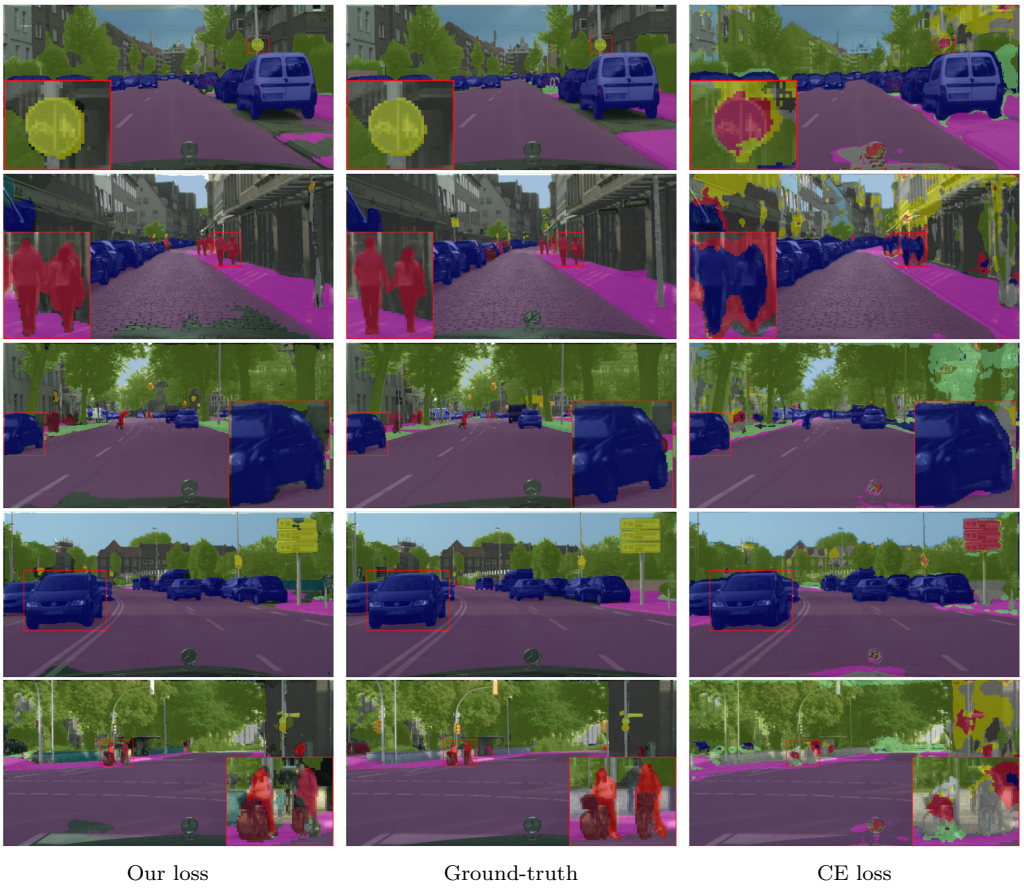


FIGURE 4.8: **Qualitative results on Cityscapes** using the U-Net model with ResNet-34 encoder.

Our loss Ground-truth Dice loss

FIGURE 4.9: **Qualitative results on CelebAMask-HQ** using the U-Net model with ResNet-50 encoder.

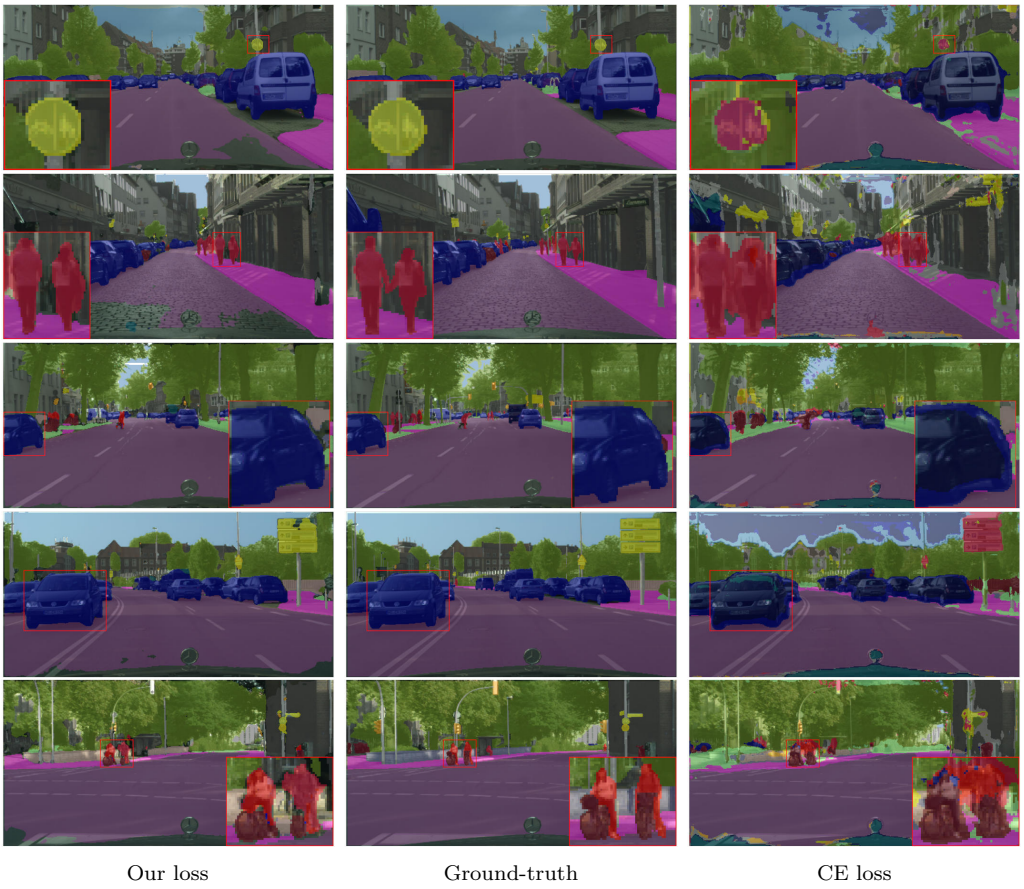


FIGURE 4.10: **Qualitative results on Cityscapes** using the U-Net model with ResNet-50 encoder.

Our loss Ground-truth Dice loss

FIGURE 4.11: **Qualitative results on CelebAMask-HQ** using the U-Net model with MobileNetV2 encoder.

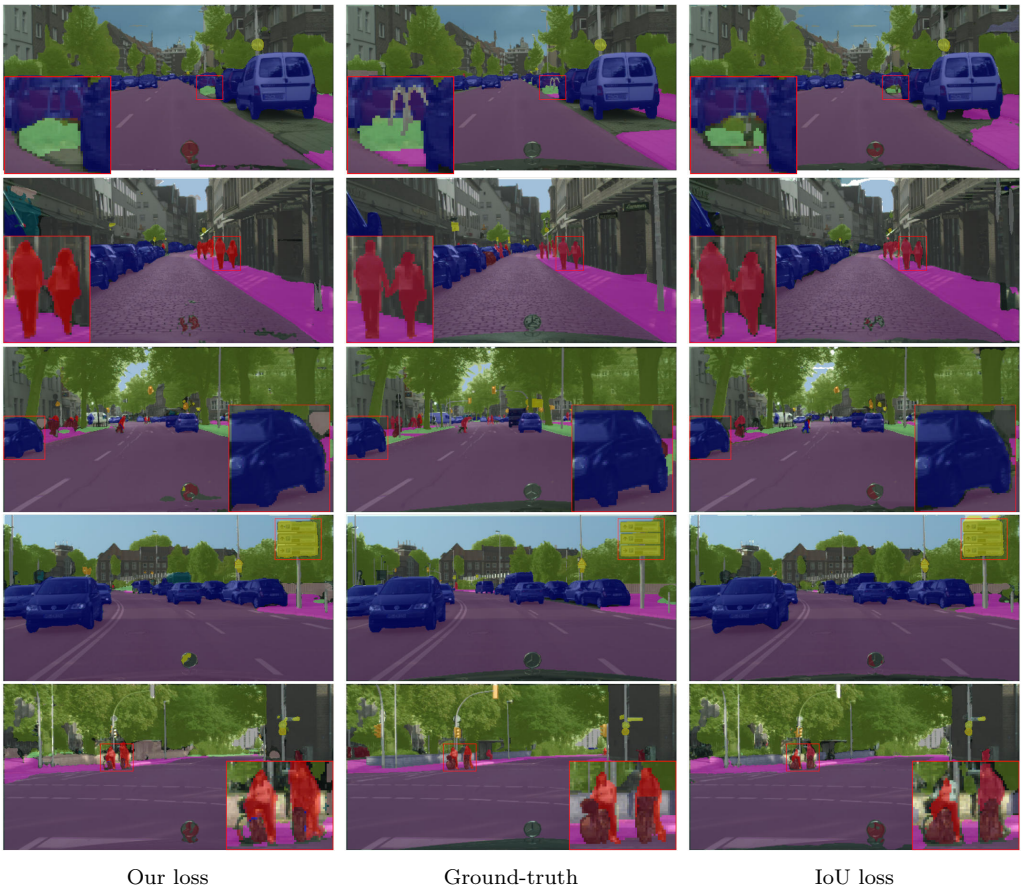


FIGURE 4.12: **Qualitative results on Cityscapes** using the U-Net model with MobileNetV2 encoder.

Our loss Ground-truth Dice loss

FIGURE 4.13: **Qualitative results on CelebAMask-HQ** using the DeepLabv3+ model with ResNet-34 encoder.

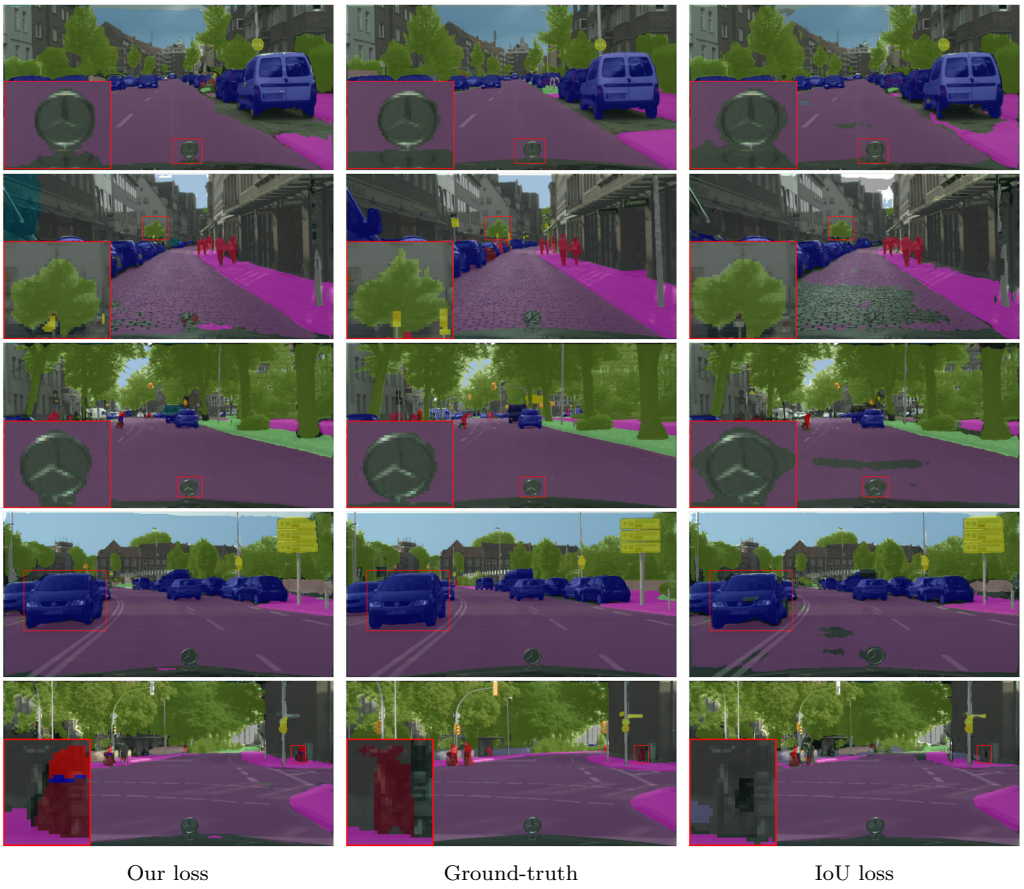


FIGURE 4.14: **Qualitative results on Cityscapes** using the DeepLabv3+ model with ResNet-34 encoder.

Our loss Ground-truth IoU loss

FIGURE 4.15: **Qualitative results on CelebAMask-HQ** using the DeepLabv3+ model with ResNet-50 encoder.

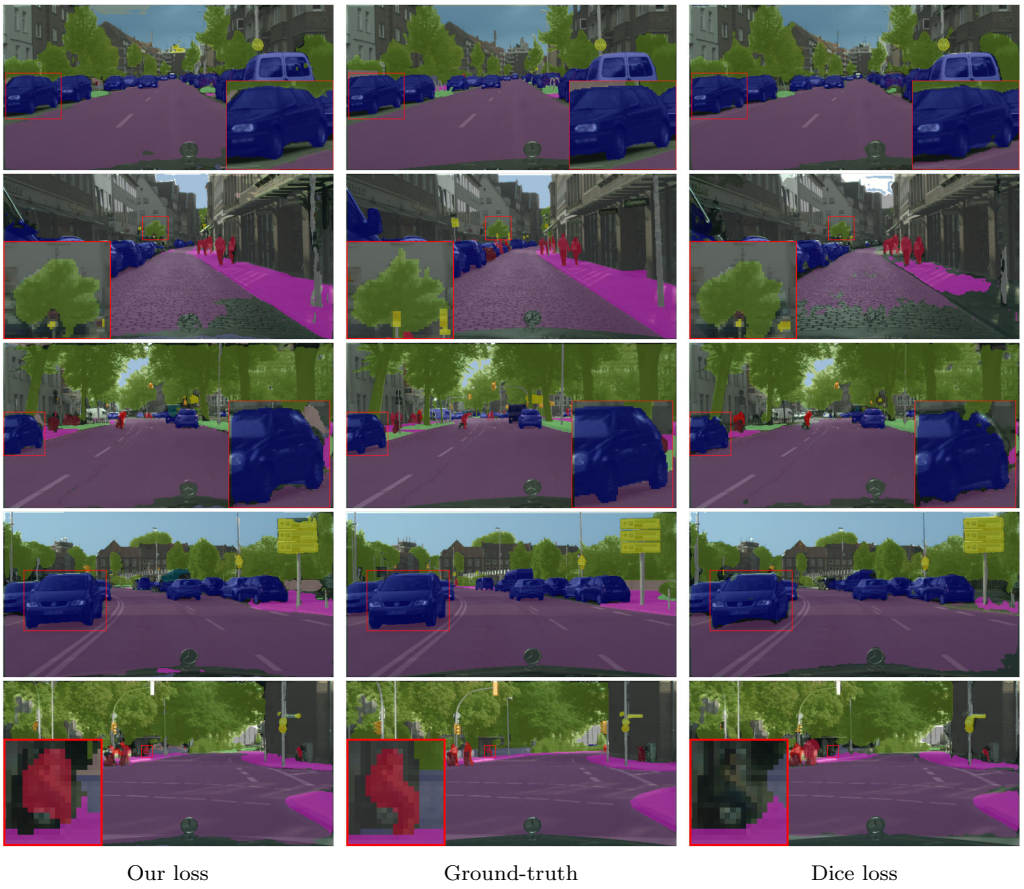


FIGURE 4.16: **Qualitative results on Cityscapes** using the DeepLabv3+ model with ResNet-50 encoder.

Our loss Ground-truth IoU loss

FIGURE 4.17: **Qualitative results on CelebAMask-HQ** using the DeepLabv3+ model with MobileNetV2 encoder.

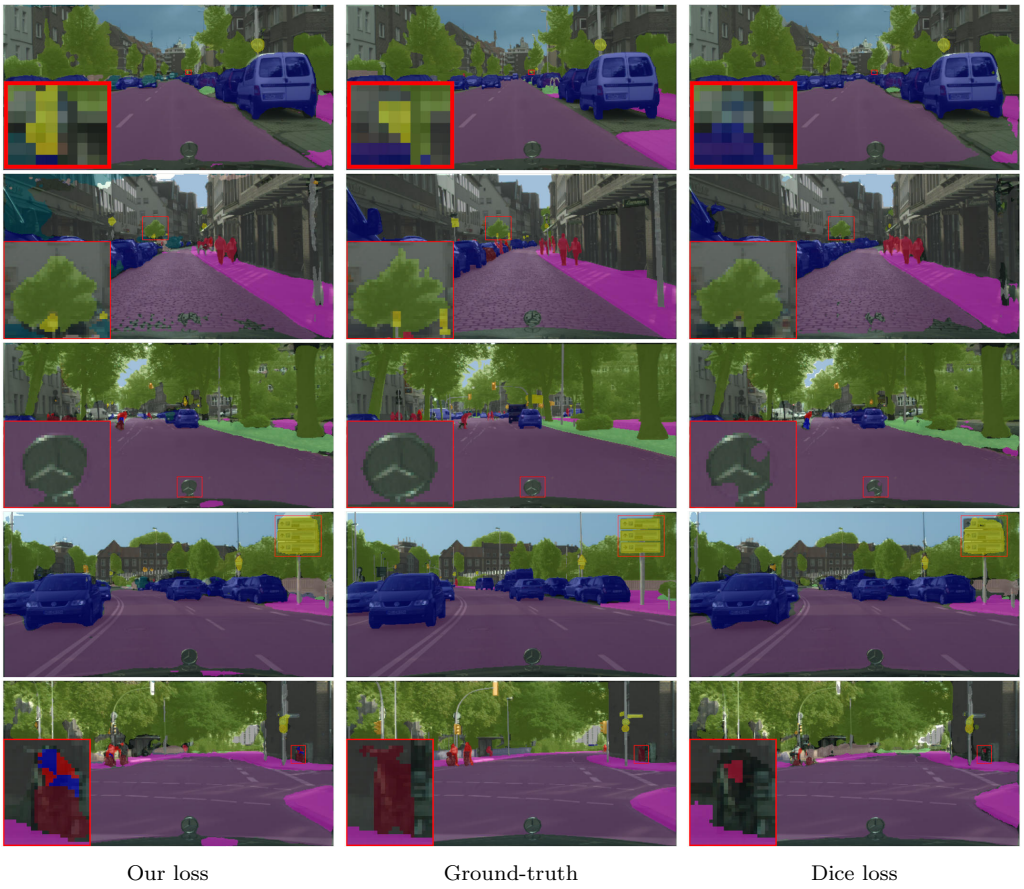


FIGURE 4.18: **Qualitative results on Cityscapes** using the DeepLabv3+ model with MobileNetV2 encoder.

Our loss Ground-truth Dice loss

FIGURE 4.19: **Qualitative results on CelebAMask-HQ** using the DeepLabv3+ model and ResNet-50 encoder with different seed value for the Ablation Study in Section 4.4.

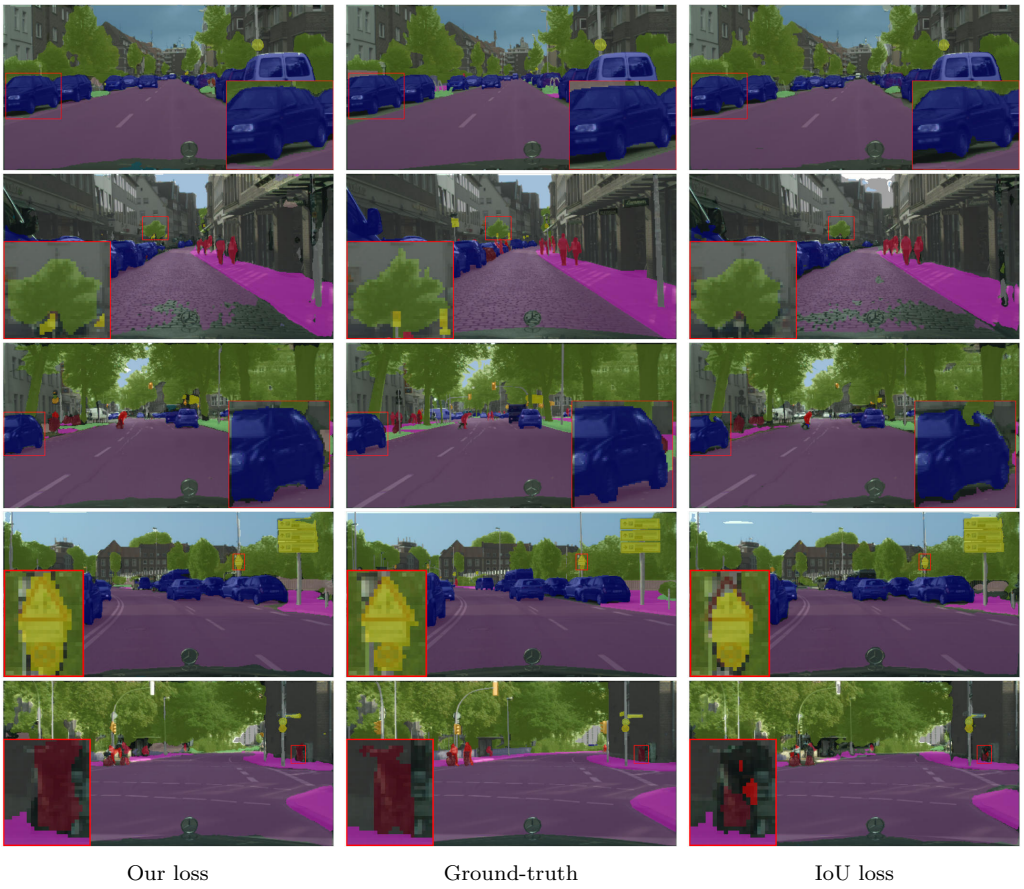


FIGURE 4.20: **Qualitative results on Cityscapes** using the DeepLabv3+ model and ResNet-50 encoder with different seed value for the Ablation Study in Section 4.4.

Chapter 5

Conclusion

In this work, we tackled the challenging task of accurately segmenting object boundaries in complexly structured objects using existing semantic segmentation loss functions such as Cross-Entropy, Intersection over Union (IoU), Dice, Combo, and Dice & Focal. Our objective was to develop a loss function that enhances the alignment between predicted and ground-truth boundaries during training while outperforming other loss functions. Drawing on our understanding of distance transformation and various loss functions, we proposed a loss function that trains any semantic segmentation model to improve boundary details with significantly reduced GPU memory usage. Our method employs a semantic boundary-aware loss to implicitly integrate geometric information into segmentation predictions. We argue that our distance transform-based loss can alleviate the problems associated with other losses in segmenting objects with shape-awareness.

Through comprehensive evaluation of both binary and multi-class segmentation datasets, our experiments demonstrated that our loss usually enhances the overall performance of semantic segmentation models both qualitatively and quantitatively for both binary and multi-class segmentation tasks. Moreover, the improvements achieved with our loss are consistent across different evaluation metrics such as boundary IoU, IoU, and Dice scores. This work prompts us to reconsider how structural information, such as semantic boundaries, can be integrated into traditional segmentation loss functions to improve boundary details.

For future work, we aim to validate our approach using other widely used semantic segmentation datasets such as ADE20K [55]. We also intend to carry out experiments with commonly used data augmentation techniques and learning rate schedulers like cosine annealing [56]. Given sufficient GPU resources, we could also increase the number of epochs to observe the long-term effects of our loss and explore the loss function's potential with other types of segmentation: panoptic, instance, etc.

Appendix A

Supplementary Results

Tables A.1 through A.6 present the average and standard deviation of each evaluation metric across the 5-fold cross-validation for experiments, studied in Section 4.4.3, using Cross-Entropy, IoU, Dice, Combo, Dice & Focal, and our proposed loss, respectively.

Table A.1: Performance of the U-Net model with a ResNet-50 encoder using Cross-Entropy loss: 5-fold cross-validation results on the CelebAMaskHQ test dataset.

	Evaluation Metrics		
5-Folds	bIoU	IoU	Dice
Fold-1	0.6779	0.9630	0.9802
Fold-2	0.6782	0.9613	0.9772
Fold-3	0.6810	0.9622	0.9768
Fold-4	0.6783	0.9593	0.9677
Fold-5	0.6778	0.9587	0.9811
Average	0.6786	0.9609	0.9766
Standard Deviation	0.0013	0.0018	0.0053

Table A.2: Performance of the U-Net model with a ResNet-50 encoder using IoU loss: 5-fold cross-validation results on the CelebAMaskHQ test dataset.

	Evaluation Metrics		
5-Folds	bIoU	IoU	Dice
Fold-1	0.7414	0.9652	0.9811
Fold-2	0.7413	0.9669	0.9761
Fold-3	0.7407	0.9669	0.9726
Fold-4	0.7398	0.9653	0.9803
Fold-5	0.7375	0.9592	0.9756
Average	0.7401	0.9647	0.9771
Standard Deviation	0.0016	0.0032	0.0035

Table A.3: Performance of the U-Net model with a ResNet-50 encoder using Dice loss: 5-fold cross-validation results on the CelebAMaskHQ test dataset.

	Evaluation Metrics		
5-Folds	bIoU	IoU	Dice
Fold-1	0.7358	0.9741	0.9831
Fold-2	0.7389	0.9705	0.9917
Fold-3	0.7370	0.9701	0.9885
Fold-4	0.7345	0.9611	0.9925
Fold-5	0.7349	0.9697	0.9906
Average	0.7362	0.9691	0.9893
Standard Deviation	0.0018	0.0048	0.0038

Table A.4: Performance of the U-Net model with a ResNet-50 encoder using Combo loss: 5-fold cross-validation results on the CelebAMaskHQ test dataset.

	Evaluation Metrics		
5-Folds	bIoU	IoU	Dice
Fold-1	0.7076	0.9675	0.9893
Fold-2	0.7026	0.9742	0.9861
Fold-3	0.7031	0.9782	0.9807
Fold-4	0.7038	0.9696	0.9819
Fold-5	0.7032	0.9745	0.9809
Average	0.7040	0.9728	0.9838
Standard Deviation	0.0020	0.0043	0.0038

Table A.5: Performance of the U-Net model with a ResNet-50 encoder using Dice & Focal loss: 5-fold cross-validation results on the CelebAMaskHQ test dataset.

	Evaluation Metrics		
5-Folds	bIoU	IoU	Dice
Fold-1	0.6595	0.9622	0.9779
Fold-2	0.6578	0.9643	0.9838
Fold-3	0.6569	0.9614	0.9868
Fold-4	0.6616	0.9709	0.9848
Fold-5	0.6601	0.9625	0.9816
Average	0.6592	0.9643	0.9830
Standard Deviation	0.0019	0.0039	0.0034

Table A.6: Performance of the U-Net model with a ResNet-50 encoder using our proposed loss: 5-fold cross-validation results on the CelebAMaskHQ test dataset.

	Evaluation Metrics		
5-Folds	bIoU	IoU	Dice
Fold-1	0.7465	0.9704	0.9909
Fold-2	0.7441	0.9681	0.9882
Fold-3	0.7461	0.9693	0.9855
Fold-4	0.7467	0.9705	0.9869
Fold-5	0.7471	0.9724	0.9929
Average	0.7461	0.9701	0.9889
Standard Deviation	0.0012	0.0016	0.0030

Appendix B

List of Publications

- F. Gul, E. Aptoula, "A Distance Transform Based Loss Function for the Semantic Segmentation of Very High Resolution Remote Sensing Images", 2024 IEEE International Geoscience and Remote Sensing Conference (**Under Review**)

Bibliography

[1] T.-Y. Lin, M. Maire, S. J. 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*, 2014.

[2] N. Barla, "The beginner's guide to semantic segmentation." `https://www.v7labs.com/blog/semantic-segmentation-guide`, 2021. [Online; accessed 28-November-2023].

[3] A. H. REYNOLDS, "Convolutional neural networks (cnns)." `https://anhreynolds.com/blogs/cnn.html`, 2019. [Online; accessed 27-November-2023].

[4] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 2481–2495, 2015.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *ArXiv*, vol. abs/1505.04597, 2015.

[7] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European Conference on Computer Vision*, 2018.

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3213–3223, 2016.

[9] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164, 2014.

[10] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, 2015.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *ArXiv*, vol. abs/1312.5602, 2013.

[12] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. P. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, pp. 350 – 354, 2019.

[13] T. Politzer, "Vision is our dominant sense." `https://www.brainline.org/article/vision-our-dominant-sense`, 2018. [Online; accessed 12-December-2023].

[14] M. Mohsin, "10 youtube stats every marketer should know in 2023." `https://www.oberlo.com/blog/youtube-statistics`, 2023. [Online; accessed 19-November-2023].

[15] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6526–6534, 2016.

[16] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *ArXiv*, vol. abs/1511.02680, 2015.

[17] D. S. Kermany, D. S. Kermany, M. H. Goldbaum, W. Cai, C. C. S. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. K. Prasadha, J. Pei, J. Pei, M. Y. L. Ting, J. Zhu, C. M. Li, S. Hewett, S. Hewett, J. Dong, I. Ziyar, A. Shi, R. Zhang, L. Zheng, R. Hou, W. Shi, X. Fu, X. Fu, Y. Duan, V. A. N. Huu, V. A. N. Huu, C. Wen, E. Zhang, E. Zhang, C. L. Zhang, C. L. Zhang, O. Li, O. Li, X. Wang, M. A. Singer, X. Sun, J. Xu, A. R. Tafreshi, M. A. Lewis, H. Xia, and K. Zhang, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, pp. 1122–1131.e9, 2018.

[18] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[19] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks : the official journal of the International Neural Network Society*, vol. 61, pp. 85–117, 2014.

[20] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, 2017.

[21] C. H. Sudre, W. Li, T. K. M. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," *Deep learning in medical image analysis and multimodal learning for clinical decision support : Third International Workshop,*

DLMIA 2017, and 7th International Workshop, ML-CDS 2017, held in conjunction with MICCAI 2017 Quebec City, QC,..., vol. 2017, pp. 240–248, 2017.

[22] M. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *International Symposium on Visual Computing*, 2016.

[23] S. A. Taghanaki, Y. Zheng, S. K. Zhou, B. Georgescu, P. S. Sharma, D. Xu, D. Comaniciu, and G. Hamarneh, "Combo loss: Handling input and output imbalance in multi-organ segmentation," *Computerized medical imaging and graphics : the official journal of the Computerized Medical Imaging Society*, vol. 75, pp. 24–33, 2018.

[24] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017.

[25] W. Zhu, Y. Huang, L. Zeng, X. Chen, Y. Liu, Z. Qian, N. Du, W. Fan, and X. Xie, "Anatomynet: Deep learning for fast and fully automated whole-volume segmentation of head and neck anatomy," *Medical Physics*, vol. 46, p. 576–589, 2018.

[26] H. Kervadec, J. Bouchtiba, C. Desrosiers, E. Granger, J. Dolz, and I. B. Ayed, "Boundary loss for highly unbalanced segmentation," *Medical image analysis*, vol. 67, p. 101851, 2018.

[27] C. Wang, Y. Zhang, M. Cui, J. Liu, P. Ren, Y. Yang, X. Xie, X. Hua, H. Bao, and W. Xu, "Active boundary loss for semantic segmentation," in *AAAI Conference on Artificial Intelligence*, 2021.

[28] S. Li, C. Zhang, and X. He, "Shape-aware semi-supervised 3d semantic segmentation for medical images," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2020.

[29] Y. Xue, H. Tang, Z. Qiao, G. Gong, Y. Yin, Z. Qian, C. Huang, W. Fan, and X. Huang, "Shape-aware organ segmentation by predicting signed distance maps," *ArXiv*, vol. abs/1912.03849, 2019.

[30] B. Murugesan, K. Sarveswaran, S. M. Shankaranarayana, K. Ram, and M. Sivaprakasam, "Psi-net: Shape and boundary aware joint multi-task deep network for medical image segmentation," *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 7223–7226, 2019.

[31] Y. Wang, X. Wei, F. Liu, J. Chen, Y. Zhou, W. Shen, E. K. Fishman, and A. L. Yuille, "Deep distance transform for tubular structure segmentation in ct scans," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3832–3841, 2019.

[32] S. Dangi, Z. R. Yaniv, and C. A. Linte, "A distance map regularized cnn for cardiac cine mr image segmentation," *Medical physics*, 2019.

[33] F. Navarro, S. Shit, I. Ezhov, J. C. Paetzold, A. Gafita, J. C. Peeken, S. E. Combs, and B. H. Menze, "Shape-aware complementary-task learning for multi-organ segmentation," *ArXiv*, vol. abs/1908.05099, 2019.

[34] Y. Tang, Y. Tang, Y. Zhu, J. Xiao, and R. M. Summers, "E2net: An edge enhanced network for accurate liver and tumor segmentation on ct scans," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2020.

[35] Y. Qin, H. Zheng, Y. Gu, X. Huang, J. Yang, L. Wang, F. Yao, Y. M. Zhu, and G.-Z. Yang, "Learning tubule-sensitive cnns for pulmonary airway and artery-vein segmentation in ct," *IEEE Transactions on Medical Imaging*, vol. 40, pp. 1603–1617, 2020.

[36] X. Li, Y. Wang, Q. sheng Tang, Z. Fan, and J. Yu, "Dual u-net for the segmentation of overlapping glioma nuclei," *IEEE Access*, vol. 7, pp. 84040–84052, 2019.

[37] B. Cheng, R. B. Girshick, P. Doll'ar, A. C. Berg, and A. Kirillov, "Boundary iou: Improving object-centric image segmentation evaluation," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15329–15337, 2021.

[38] L.-C. Chen, G. Papandreou, I. Kokkinos, K. P. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 834–848, 2016.

[39] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2014.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[41] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *ArXiv*, vol. abs/1905.11946, 2019.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[43] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, 2016.

[44] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *ArXiv*, vol. abs/1706.05587, 2017.

[45] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[46] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink, "A general algorithm for computing distance transforms in linear time," in *International Symposium on Mathematical Morphology and Its Application to Signal and Image Processing*, 2000.

[47] C. R. Maurer, R. Qi, and V. Raghavan, "A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 265–270, 2003.

[48] T. Strutz, "The distance transform and its computation," *ArXiv*, vol. abs/2106.03503, 2021.

[49] C.-H. Lee, Z. Liu, L. Wu, and P. Luo, "Maskgan: Towards diverse and interactive facial image manipulation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5548–5557, 2019.

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[51] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2017.

[52] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Neural Information Processing Systems*, 2019.

[54] K. Miura, T. Miyamoto, K. Sakurai, K. Ito, and T. Aoki, "Eyeglass frame segmentation for face image processing," in *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1572–1576, 2022.

[55] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5122–5130, 2017.

[56] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv: Learning*, 2016.