Optimized Hausdorff Distance Loss Function Based on a GPU-Accelerated Distance Transform

Dale Black, Wenbo Li, Sabee Molloi

June 30, 2023

Abstract

This paper presents a novel Hausdorff distance loss function for image segmentation, particularly in the field of medical imaging. The proposed Hausdorff distance loss function, based on the Felzenszwalb distance transform algorithm, addresses the computational complexity associated with previous Hausdorff distance loss function implementations, bringing it closer to the efficiency of the Dice loss function. The new method significantly reduces the computational time, making it only 11.2% slower than the Dice loss function, compared to the 125.9% slower rate of previous implementations. Furthermore, the proposed Hausdorff distance loss function improves the Dice similarity coefficient from 0.918 to 0.925 and reduces the Hausdorff distance from 4.583 to 3.464, demonstrating enhanced segmentation accuracy. The study's findings suggest that the proposed Hausdorff distance loss function can be a valuable tool for medical image segmentation, providing a balance between computational efficiency and segmentation precision. The code for the new Hausdorff distance loss function is publicly available for use.

1 Introduction

Delineating a region of interest in an image is known as image segmentation. The need to separate the region of interest for additional analysis is a key task in medical image analysis [1].

Numerous studies have addressed the topic of medical image segmentation [1]. Manual segmentations, created by an expert radiologist, are typically regarded as the gold standard. However, the speed and repeatability of the results can be improved by semi-automatic and fully-automatic segmentation techniques. Fully-automated segmentation techniques remove the inter and intra-observer variability. Furthermore, significant progress has been made in closing the performance gap between automatic and manual segmentation approaches, particularly with the recent emergence of segmentation algorithms based on convolutional neural networks [2].

Segmentation may be one step in a more involved procedure in some applications. For instance, several multimodal medical image registration techniques rely on segmenting a target organ in one or more images [3]. The Dice similarity coefficient (DSC), overlap, and Hausdorff distance (HD) are some metrics that are typically computed to assess the accuracy of the segmentation method relative to some ground truth segmentation [4, 5]. HD is one of the most useful metrics for quantifying boundary information. As measured by HD, the highest segmentation error can be a strong indicator of how effective the segmentations are. For two sets of points (X, Y), the HD is defined by Equation 1. Equation 1 shows the one-sided HD from X to Y, Equation 2 shows the one-sided HD from Y to X, and Equation 3 shows the bi-directional HD between both sets.

$$\text{hd}(X, Y) = \max_{x \varepsilon X} \; [\min_{y \varepsilon Y}(||x - y||_2)] \tag{1}$$

$$\text{hd}(Y, X) = \max_{y \varepsilon Y} \; [\min_{x \varepsilon X}(||x - y||_2)] \tag{2}$$

$$\text{HD}(X, Y) = \max[\text{hd}(X, Y), \text{hd}(Y, X)] \tag{3}$$

Recently proposed segmentation approaches, specifically deep learning-based methods, have focused on directly minimizing the HD during the training process via the HD loss function [6, 7, 8, 9, 10]. The most accurate form of the HD loss function utilizes a distance transform operation [6] to estimate the HD.

Although the HD loss function improves the segmentation, especially along the boundaries, this loss function is unstable, particularly in the early part of the training process [6]. A weighted combination of loss functions can solve this issue, like the combination of Dice loss function and the HD loss function [6].

Still, the HD loss function is more computationally expensive than the more common Dice loss function, which limits the usage of the HD loss function in deep learning [6]. Previous open source HD loss function implementations utilized central processing unit (CPU)-based distance transform operations since those methods are readily available in packages like SciPy [11]. The downside to this approach is that all practical deep learning training takes place on the graphics processing unit (GPU), which means the training must switch back and forth from CPU to GPU every time the HD loss function is utilized. Also, for large arrays, like those commonly found in medical imaging, GPU operations are faster than CPU operations. These reasons above likely explain the drastic increase in training time when utilizing the HD loss function over the more common Dice loss function [6]. Previous works have published GPU-optimized distance transform algorithms [12, 13], but no reports have taken advantage of these improvements for the HD loss function.

This study presents an HD loss function based on the Felzenszwalb distance transform algorithm [14, 15, 16]. This novel HD loss function decreases the computational complexity compared to previous HD loss function algorithms, making it comparable to the gold standard Dice loss function while directly optimizing DSC and HD. All the code is publicly available for use at https://github.com/Dale-Black/DistanceTransforms.jl

2 Methods

2.1 Distance Transform

To implement a new distance transform operation, the Julia programming language [17] was a logical choice since CPU and GPU code can be written in the same high-level language with speed similar to more common languages like C and Fortran.

We adapted the Felzenszwalb distance transform algorithm [14, 15]. The Felzenszwalb algorithm has several advantages that make it an excellent choice for our purposes. First, it is easily parallelizable, which allows for efficient multi-threaded operations on both CPUs and GPUs. Second, the algorithm is straightforward to adapt for GPU-accelerated computations, a feature that is crucial for handling the large data sets typically encountered in medical imaging. Most importantly, the Felzenszwalb algorithm is an O(n) algorithm, meaning its computational complexity grows linearly with the size of the input data. This is a significant advantage in our context, as distance transform computations are typically computationally expensive. By employing a linear algorithm, we can drastically reduce the computational resources required, thereby increasing the overall efficiency of our method. Algorithm 1 shows the one-dimensional Felzenszwalb method.

Previous HD loss functions utilized a readily available distance transform algorithm from SciPy [11]. The previous distance transform algorithm was directly ported to Julia using PythonCall.jl [18] and timed along with the CPU and GPU-accelerated Felzenszwalb distance transform algorithm across various-sized arrays in two and three-dimensions.

2.2 Hausdorff Loss Function

The HD loss function was previously described by [6] and can be seen in Equation 4

$$\mathrm{Loss}_{\mathrm{HD}}(q,p) = \frac{1}{|\Omega|}\sum_{\Omega}((p-q)^2 \circ (d_p^{\alpha} + d_q^{\alpha})) \tag{4}$$

Where p is the binary ground-truth segmentation, q is the binary prediction, d_p is the distance transform of p, d_q is the distance transform of q, Ω represents the set of all pixels, and $\circ$ represents the Hadamard product (i.e., the element-wise matrix product). The parameter α determines how strongly the larger errors are penalized. Previous reports show that values of α between 1 and 3 yielded good results.

The HD loss function is a good minimizer of HD, and Karimi et al. show that combining the HD loss function with the Dice loss function (Equation 5) helps avoid the instability issue associated with the HD loss function [6]. Thus, the resulting loss function used throughout this study directly optimizes for DSC and HD and is a combination of $\mathrm{Loss}_{\mathrm{HD}}$ and $\mathrm{Loss}_{\mathrm{DSC}}$, shown in Equation 6.

Algorithm 1 Felzenszwalb Distance Transform 1D

```
Require: A one-dimensional array f[1:n] with n elements
Ensure: An array D[1:n] where D[i] represents the transformed distance of f[i]
  Define an empty array of parabolas: parabolas[]
  Define an array of intersection points: intersection[] with same length as f,
  initialized to infinity
  parabolas[1] $\leftarrow$ 1
  k $\leftarrow$ 1
  for i = 2 to n do
    while k > 0 and f[i] + (i$^2$) < f[parabolas[k]] + intersection[k]$^2$ do
      k $\leftarrow$ k - 1
    end while
    if k > 0 then
      intersection[k+1] $\leftarrow$ (f[i] - f[parabolas[k]] + i$^2$ - parabolas[k]$^2$) / (2 $\times$ i -
      2 $\times$ parabolas[k])
    end if
    parabolas[k+1] $\leftarrow$ i
    k $\leftarrow$ k + 1
  end for
  k $\leftarrow$ 1
  for i = 1 to n do
    while intersection[k+1] ¡ i do
      k $\leftarrow$ k + 1
    end while
    D[i] $\leftarrow$ (i - parabolas[k])$^2$ + f[parabolas[k]]
  end for
  return D
```

$$\mathrm{Loss}_{\mathrm{DSC}}(q,p) = 1 - \frac{2\sum_{\Omega}(p \circ q)}{\sum_{\Omega}(p^2 + q^2)} \tag{5}$$

$$\begin{aligned}\mathrm{Loss}(q,p) =& \lambda\left(\frac{1}{|\Omega|}\sum_{\Omega}\left((p-q)^2 \circ (d_p + d_q)\right)\right) + \\ & (1-\lambda)\left(1 - \frac{2\sum_{\Omega}(p \circ q)}{\sum_{\Omega}(p^2 + q^2)}\right)\end{aligned} \tag{6}$$

Similarly to the pure distance transform timings, previously presented HD loss functions utilized the previously proposed SciPy distance transform which was compared to the HD loss function implemented in this study, which utilized the Felzenszwalb algorithm. Also, the gold standard Dice loss function was included in these timings to provide a baseline operation time - since the HD loss function is typically combined with the Dice loss function to provide stability during training. We determined each loss function's speed across various-sized arrays in two and three-dimensions.

2.3 Training

After timing the pure distance transforms and pure loss functions on various sized arrays, we then measured the average time per epoch on a simple but realistic deep learning training loop. We utilized the heart CT dataset from publicly available Medical Segmentation Decathlon [19]. For the trainig loop bencharms, we used a 3D U-Net model [20]. This U-Net model consists of a contracting path and an expanding path. The contracting path includes four layers, each composed of two 3D convolution blocks (each block with a leaky ReLU activation function embedded in a batch normalization operation) followed by a MaxPooling operation. The expanding path also contains four layers, each layer receiving input from both the corresponding contracting layer (via skip connections) and the preceding layer in the expanding path. The expanding layers employ Convolutional Transpose operations to gradually upsample the feature maps. The final layer of the model applies a 3D convolution operation with a softmax activation function for multi-class segmentation. We ran the training loop for 40 epochs and found the average training time per epoch for all three loss functions.

After determining the fastest Hausdorff distance (HD) loss function, we then ran a more exhaustive training process to validate the improved accuracy of the hybrid HD loss compared to the baseline DSC loss. The 3D U-Net model, identical in architecture to the one used in the training loop benchmarks, was trained for 500 epochs for each loss function.

This study's loss functions are available in both Python and Julia via juliacall/PythonCall.jl [18]. All the training code was implemented in Julia v1.8 and Flux v0.13.13 [21, 22] and run on Windows 10. An NVIDIA RTX A5000 GPU and CUDA 11 were used for training.

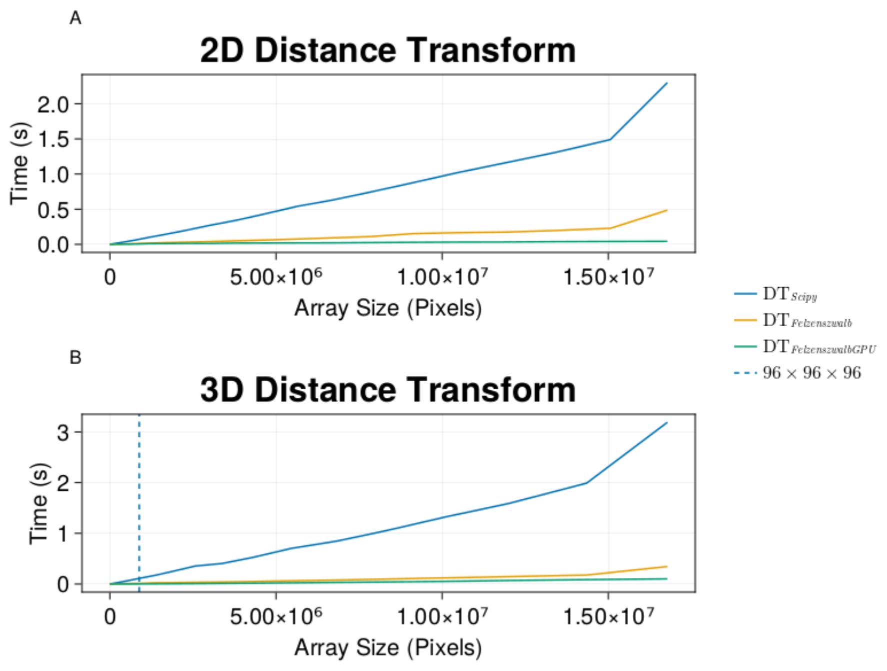

Figure 1: Distance transform benchmarks. (A) Shows the time to compute the distance transforms (in seconds) on various-sized two-dimensional arrays. (B) Shows the time to compute the distance transforms (in s) on various-sized three-dimensional arrays.

3 Results

3.1 Timings

3.1.1 Distance Transforms

Two and three-dimensional arrays containing between 16 and 16777216 elements were input into the distance transforms. The distance transform utilized in this paper $\mathrm{DT}_{FelzenszwalbGPU}$ was shown to be the fastest implementation for two and three-dimensional arrays within the size range of common medical images, like computed tomography scans. Specifically, Figure 1B shows a dashed vertical line of size $96 \times 96 \times 96$ which corresponds to the size of the images used during training. For the largest two-dimensional array, $\mathrm{DT}_{FelzenszwalbGPU}$ was 54.0 times faster than DT_{Scipy}. For the largest three-dimensional array, $\mathrm{DT}_{FelzenszwalbGPU}$ was 31.6 times faster than DT_{Scipy}. Figure 1 shows the two, and three-dimensional timings of the various distance transform algorithms.

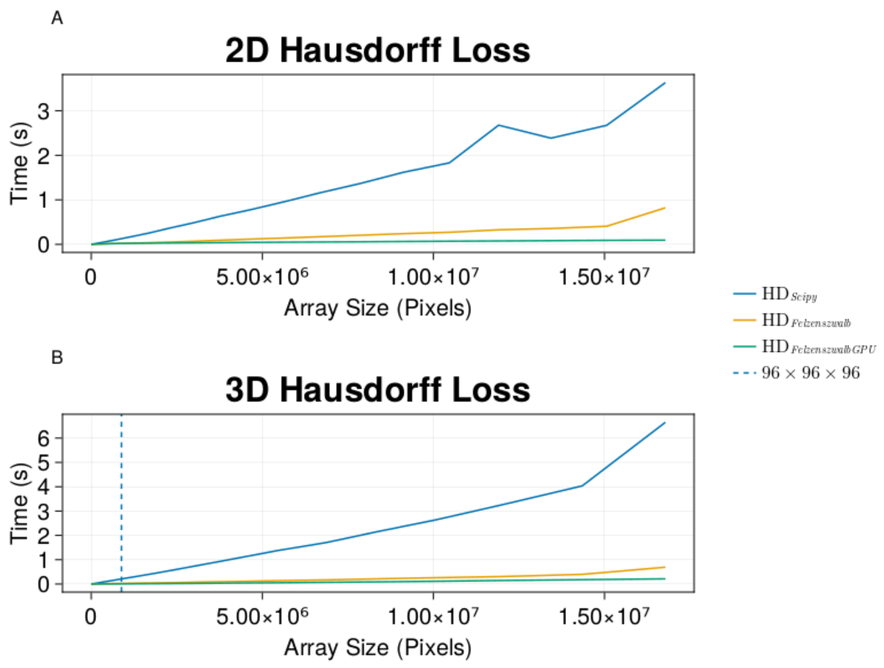

Figure 2: Loss function benchmarks. (A) Shows the time to compute the loss functions (in seconds) on various-sized two-dimensional arrays. (B) Shows the time to compute the loss functions (in seconds) on various-sized three-dimensional arrays.

3.1.2 Hausdorff Loss Functions

Two and three-dimensional arrays containing between 16 and 16777216 elements were input into the HD loss functions, with the only variation coming from the type of distance transform algorithm used in the Hausdorff loss function. The Hausdorff loss function, implemented in this paper HD$_{FelzenszwalbGPU}$, was faster than the previously implemented HD loss function HD$_{Scipy}$. Specifically, HD$_{FelzenszwalbGPU}$ was 38.7 times faster than HD$_{SciPy}$ on the largest two-dimensional arrays and 31.3 times faster on the largest three-dimensional arrays. The input arrays in the training loop were of size $96 \times 96 \times 96$ (Figure 2) and for this size, the HD$_{FelzenszwalbGPU}$ was 31.6 times faster than HD$_{SciPy}$. Figure 2 shows the two, and three-dimensional timings of the various loss functions.

3.1.3 Training Loop

A simplified training loop was run for 40 epochs, and the average epoch time was examined for each loss function. Based on previous reports, combining the HD loss function with the Dice loss function is recommended to avoid instabil-

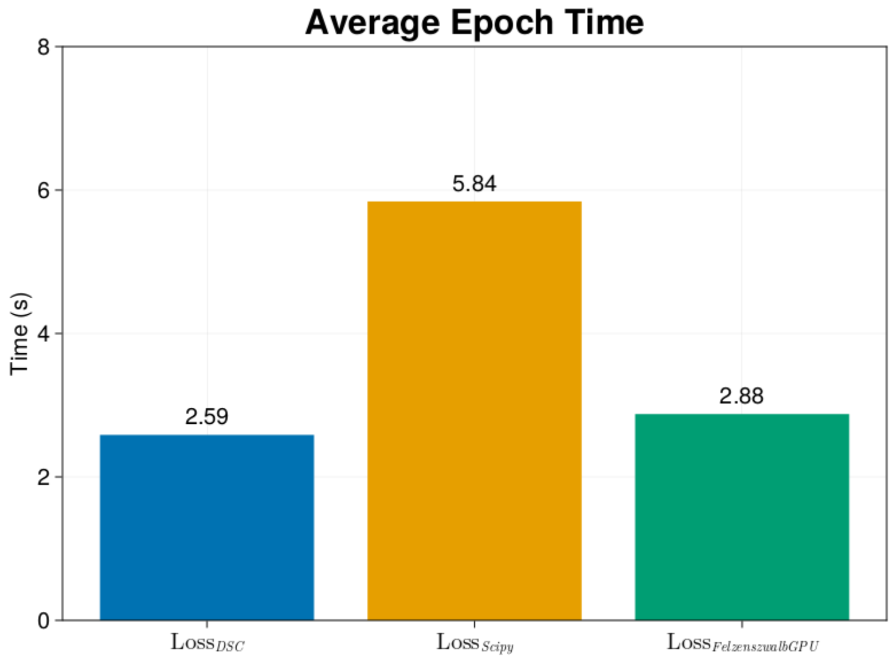

Figure 3: Average epoch time (s) for various loss functions per epoch

ity issues [6]. Therefore, three different loss functions were analyzed: 1) pure Dice loss function, Loss_{DSC}, which serves as the baseline, 2) a hybrid HD loss and Dice loss function, Loss_{Scipy}, which utilized the DT_{Scipy} for the distance transform algorithm within the Hausdorff loss function, and corresponds to the previously reported HD loss function, and 3) our new hybrid HD loss and Dice loss function, $\text{Loss}_{FelzenszwalbGPU}$, which runs entirely on the GPU and uses $\text{DT}_{FelzenszwalbGPU}$ within the Hausdorff loss function.

The average epoch time for the previously proposed HD loss function Loss_{Scipy} was 125.9% slower than the baseline Loss_{DSC} (Figure 3). The average step time for our new HD loss function $\text{Loss}_{FelzenszwalbGPU}$ was 11.2% slower than the baseline per epoch (Figure 3). Figure 3 shows the average epoch time for each loss function.

3.2 Accuracy

This study focused on solving the computational complexity issue associated with previously proposed HD loss functions [6]. Therefore, timing the HD loss functions was the main concern. But the purpose of the HD loss function is to improve upon the segmentation compared to the gold standard Dice loss function. Hence, we trained a deep learning model with the Dice loss, Loss_{DSC},

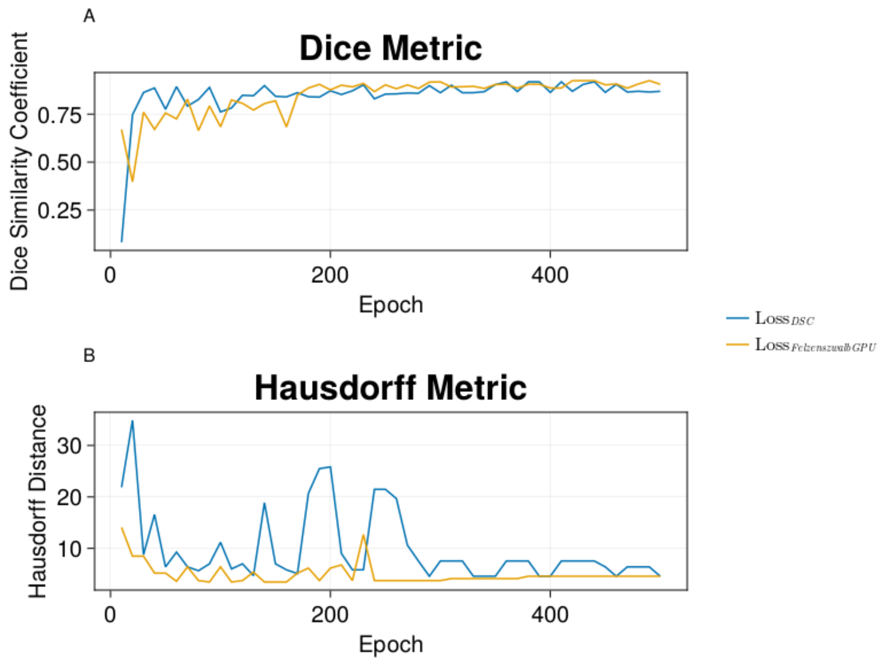

Figure 4: The HD and DSC of both models at every tenth epoch for the validation set. Both the DSC and HD are improved compared to the baseline DSC model, with less variation between each epoch

and our fast hybrid Dice-HD loss function, $\text{Loss}_{FelzenszwalbGPU}$, and compared the pertinent resulting metrics (DSC and HD) of each model to demonstrate the effectiveness of our proposed HD loss function. Figure 4 shows both models' DSC and HD metrics at every tenth epoch on the validation set. Table 1 shows both models' best DSC and lowest HD along with the 90th and 10th percentile DSC and HD metrics, respectively. $\text{Loss}_{FelzenszwalbGPU}$ improved upon the baseline model's DSC from 0.918 to 0.925. Similarly, $\text{Loss}_{FelzenszwalbGPU}$ improved upon the baseline model's HD from 4.583 to 3.464. Both the DSC and HD are improved compared to the baseline DSC model, with less variation between each epoch (Figure 4).

Figure 5 shows example slices of the heart dataset with the ground truth segmentation and predicted segmentation overlayed on the slice. The left column shows the predicted mask from Loss_{DSC}, and the right column shows the predicted mask from $\text{Loss}_{FelzenszwalbGPU}$.

Table 1: DSC and HD Metrics

Loss Function	DSC (Best)	DSC (90th %)	HD (Best mm)	HD (10th %)
Loss_{DSC}	0.919	0.907	4.583	4.583
$\text{Loss}_{FelzenszwalbGPU}$	0.925	0.917	3.464	3.591

4 Discussion

We introduced a new Hausdorff loss function, based on a GPU accelerated distance transform algorithm. This novel loss function reduces the computational complexity relative to previously proposed HD loss functions and reduces the time to be within 11.2% of the gold standard Dice loss function, while concurrently minimizing the DSC and HD better than the pure Dice loss function.

The HD metric is instrumental in quantifying boundary information [4], serving as an indicator of segmentation efficacy. The method proposed can enhance segmentations, particularly at the boundaries, and is applicable to a variety of image segmentation challenges, including those found in medical imaging [23].

In previous research, Karimi et al. [6] employed a distance transform corresponding to DT_{Scipy} within the HD loss function HD_{Scipy} and combined with the dice loss function to calculate a weighted loss. Our implementation of the HD loss function $\text{HD}_{FelzenszwalbGPU}$ was timed and compared to HD_{Scipy} within a realistic training loop. Given the variations in the data, deep learning model, and computational language, we re-implemented the previous approach directly in Julia using PythonCall.jl to minimize confounding variables. Thus, to ensure a robust comparison, we replicated Karimi et al.'s approach under identical conditions to those used for our implementation $\text{HD}_{FelzenszwalbGPU}$.

Our findings align with those of Karimi et al. in that we observed an increase in training when comparing the hybrid HD loss function $\text{Loss}_{FelzenszwalbGPU}$ to pure Dice loss function Loss_{DSC}. However, our optimized loss function only increased the total training time by 11.2% compared to 125.9% for Loss_{Scipy} (Figure 3). Direct comparisons to Karimi et al. are not feasible due to the aforementioned confounding variables and the lack of clear stopping criteria during training. As our study is focused on timing optimization, not accuracy, we applied identical stopping criteria to both the dice loss function and the HD loss function training loops.

Our results are in broad agreement with previous research [6], which also indicated improved performance when combining HD and Dice loss functions during training. However, as the stopping criteria during training were not specified for the various loss functions in the cited study, variables such as total epochs may differ, precluding direct comparisons. Therefore, while our findings concur with prior research regarding timing trends, further investigations are necessary to ensure comparability in terms of accuracy.

Our results demonstrate that $\text{HD}_{FelzenszwalbCPU}$ is 9.6 times faster than

HD_{Scipy} on three-dimensional arrays, on average. The HD loss function with Felzenszwalb's distance transform on the GPU $\text{HD}_{FelzenszwalbGPU}$ is 31.6 times faster than HD_{Scipy}. The average epoch time with the hybrid HD-Dice loss function $\text{Loss}_{FelzenszwalbGPU}$ is 103% faster than Loss_{Scipy}.

Prior research has concentrated on directly minimizing the HD during the training process via the HD loss function [8, 9]. However, the HD loss function is unstable, particularly in the early part of the training process, which limits its applicability. Although a weighted combination of loss functions, such as the Dice and HD loss functions, can solve this issue. Still, the HD loss function is more computationally expensive than the Dice loss function and further work needs to be done to investigate the use of more recently proposed distance transform algorithms for use in the HD loss function to further decrease time to compute the HD loss. Previous studies have looked at operations like convolutional kernels and morphological erosion/dilation in order to avoid the computational complexity of distance transforms within the HD loss function [6]. The results of such operations are lacking in the ability to directly minimize HD compared to distance transform based HD loss functions [6]. Our results, therefore, introduce an accurate and efficient approach to the HD loss function without sacrificing accuracy.

Our results diverge from previous studies that have focused solely on Dice loss functions for medical image segmentation [24]. In those studies, the emphasis was on achieving good overlap with the ground truth, ignoring the positional accuracy and shape of the segmented region. Our method, however, combines the Dice and HD loss functions, ensuring not only overlap but also accurate localization and shape preservation of the segmented regions. This can be particularly useful in clinical scenarios where precise delineation of the region of interest can significantly impact the subsequent steps, such as in tumor detection or organ segmentation [25, 26]. However, the potential drawback of increased sensitivity to noise and outliers due to the HD distance might cause issues. Future studies on more robust data, like centerline segmentation [27], are needed to investigate this. Nevertheless, given the overall improved performance in terms of segmentation precision and robustness to large discrepancies, the combined use of the Dice loss function and our optimized HD loss function could offer significant advantages while remaining computationally efficient.

Kervadec et al. proposed a Boundary loss function which is similar to the HD loss function in that it utilizes a distance transform [28]. Boundary loss functions can likely benefit directly from the proposed distance transform operation $\text{DT}_{FelzenszwalbGPU}$ and future studies are needed to elucidate the effect of a more efficient distance transform as it relates to boundary loss functions.

Prior attempts to implement HD loss functions were hampered by computational inefficiencies due to the CPU-based distance transform operations. This limitation had, until now, prevented the widespread adoption of HD loss functions in deep learning for at least two reasons - (1) the time increase of the HD loss function associated with the distance transform and (2) the code complexity associated with handling the back and forth of data between the CPU and GPU within a deep learning training loop. Our approach results in a GPU compatible HD loss function, which addresses these issues and brings the computational cost closer to that of the pure Dice loss function. This Dice loss function is widely used in the field and our approach can be a drop in replacement for the Dice loss function.

5 Conclusion

In this study, we present a novel Hausdorff distance loss function that is based on the Felzenszwalb distance transform operation. This Hausdorff distance loss function approaches the efficiency of the Dice loss function while directly optimizing Dice similarity coefficient and Hausdorff distance during training.

References

[1] P. Malhotra et al. "Deep Neural Networks for Medical Image Segmentation". In: *Journal of Healthcare Engineering* 2022 (Mar. 2022), p. 9580991. ISSN: 2040-2295. DOI: 10.1155/2022/9580991. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8930223/ (visited on 06/16/2023).

[2] X. Liu et al. "A Review of Deep-Learning-Based Medical Image Segmentation Methods". en. In: *Sustainability* 13.3 (Jan. 2021). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 1224. ISSN: 2071-1050. DOI: 10.3390/su13031224. URL: https://www.mdpi.com/2071-1050/13/3/1224 (visited on 06/17/2023).

[3] F. P. Oliveira and J. M. R. Tavares. "Medical image registration: a review". In: *Computer Methods in Biomechanics and Biomedical Engineering* 17.2 (Jan. 2014). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10255842.2012.670855, pp. 73–93. ISSN: 1025-5842. DOI: 10.1080/10255842.2012.670855. URL: https://doi.org/10.1080/10255842.2012.670855 (visited on 06/17/2023).

[4] M.-P. Dubuisson and A. Jain. "A modified Hausdorff distance for object matching". In: *Proceedings of 12th International Conference on Pattern Recognition* 1 (1994). Conference Name: 12th International Conference on Pattern Recognition ISBN: 9780818662652 Place: Jerusalem, Israel Publisher: IEEE Comput. Soc. Press, pp. 566–568. DOI: 10.1109/ICPR.1994.576361. URL: http://ieeexplore.ieee.org/document/576361/ (visited on 06/17/2023).

[5] K. H. Zou et al. "Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index". In: *Academic radiology* 11.2 (Feb. 2004), pp. 178–189. ISSN: 1076-6332. DOI: 10.1016/S1076-6332(03)00671-8. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1415224/ (visited on 06/17/2023).

[6] D. Karimi and S. E. Salcudean. “Reducing the Hausdorff Distance in Medical Image Segmentation with Convolutional Neural Networks”. In: *arXiv:1904.10030 [cs, eess, stat]* (Apr. 2019). arXiv: 1904.10030. URL: http://arxiv.org/abs/1904.10030 (visited on 11/02/2021).

[7] H. Asaturyan et al. “Advancing Pancreas Segmentation in Multi-protocol MRI Volumes Using Hausdorff-Sine Loss Function”. en. In: *Machine Learning in Medical Imaging*. Ed. by H.-I. Suk et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 27–35. ISBN: 978-3-030-32692-0. DOI: 10.1007/978-3-030-32692-0_4.

[8] S. Jadon. “A survey of loss functions for semantic segmentation”. In: *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. Oct. 2020, pp. 1–7. DOI: 10.1109/CIBCB48159.2020.9277638.

[9] J. Ma et al. “Loss odyssey in medical image segmentation”. en. In: *Medical Image Analysis* 71 (July 2021), p. 102035. ISSN: 1361-8415. DOI: 10.1016/j.media.2021.102035. URL: https://www.sciencedirect.com/science/article/pii/S1361841521000815 (visited on 06/17/2023).

[10] V. S. Fonov, P. Rosa-Neto, and D. L. Collins. “3D MRI Brain Tumour Segmentation with Autoencoder Regularization and Hausdorff Distance Loss Function”. en. In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Ed. by A. Crimi and S. Bakas. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 324–332. ISBN: 978-3-031-08999-2. DOI: 10.1007/978-3-031-08999-2_27.

[11] P. Virtanen et al. “SciPy 1.0: fundamental algorithms for scientific computing in Python”. eng. In: *Nature Methods* 17.3 (Mar. 2020), pp. 261–272. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2.

[12] Y. Chen et al. “GPU-Accelerated Incremental Euclidean Distance Transform for Online Motion Planning of Mobile Robots”. In: *IEEE Robotics and Automation Letters* 7.3 (July 2022). Conference Name: IEEE Robotics and Automation Letters, pp. 6894–6901. ISSN: 2377-3766. DOI: 10.1109/LRA.2022.3177852.

[13] F. de Assis Zampirolli and L. Filipe. “A Fast CUDA-Based Implementation for the Euclidean Distance Transform”. In: *2017 International Conference on High Performance Computing & Simulation (HPCS)*. July 2017, pp. 815–818. DOI: 10.1109/HPCS.2017.123.

[14] P. F. Felzenszwalb and D. P. Huttenlocher. “Pictorial Structures for Object Recognition”. In: *International Journal of Computer Vision* 61.1 (Jan. 2005). Publisher: Springer Nature, pp. 55–79. ISSN: 09205691. DOI: 10.1023/B:VISI.0000042934.15159.49. URL: https://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=14579036&site=ehost-live&scope=site (visited on 06/17/2023).

[15] P. F. Felzenszwalb and D. P. Huttenlocher. “Distance Transforms of Sampled Functions”. EN. In: *Theory of Computing* 8.1 (Sept. 2012). Publisher: Theory of Computing Exchange, pp. 415–428. ISSN: 1557-2862. DOI: 10.4086/toc.2012.v008a019. URL: http://www.theoryofcomputing.org/articles/v008a019 (visited on 11/16/2021).

[16] S. Gu, Y. Zheng, and C. Tomasi. “Linear time offline tracking and lower envelope algorithms”. In: *2011 International Conference on Computer Vision*. ISSN: 2380-7504. Nov. 2011, pp. 1840–1846. DOI: 10.1109/ICCV.2011.6126451.

[17] J. Bezanson et al. “Julia: A Fresh Approach to Numerical Computing”. In: *SIAM Review* 59.1 (Jan. 2017). Publisher: Society for Industrial and Applied Mathematics, pp. 65–98. ISSN: 0036-1445. DOI: 10.1137/141000671. URL: https://epubs.siam.org/doi/abs/10.1137/141000671 (visited on 06/17/2023).

[18] C. Rowley. *PythonCall.jl: Python and Julia in harmony.* 2022. URL: https://github.com/cjdoris/PythonCall.jl.

[19] M. Antonelli et al. “The Medical Segmentation Decathlon”. In: *arXiv:2106.05735 [cs, eess]* (June 2021). arXiv: 2106.05735. URL: http://arxiv.org/abs/2106.05735 (visited on 10/26/2021).

[20] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. en. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by N. Navab et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4. DOI: 10.1007/978-3-319-24574-4_28.

[21] M. Innes. “Flux: Elegant machine learning with Julia”. en. In: *Journal of Open Source Software* 3.25 (May 2018), p. 602. ISSN: 2475-9066. DOI: 10.21105/joss.00602. URL: https://joss.theoj.org/papers/10.21105/joss.00602 (visited on 06/19/2023).

[22] M. Innes et al. *Fashionable Modelling with Flux.* arXiv:1811.01457 [cs]. Nov. 2018. DOI: 10.48550/arXiv.1811.01457. URL: http://arxiv.org/abs/1811.01457 (visited on 06/19/2023).

[23] Z. Zhou et al. “UNet++: A Nested U-Net Architecture for Medical Image Segmentation”. en. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Ed. by D. Stoyanov et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 3–11. ISBN: 978-3-030-00889-5. DOI: 10.1007/978-3-030-00889-5_1.

[24] R. Zhao et al. “Rethinking Dice Loss for Medical Image Segmentation”. In: *2020 IEEE International Conference on Data Mining (ICDM)*. ISSN: 2374-8486. Nov. 2020, pp. 851–860. DOI: 10.1109/ICDM50108.2020.00094.

[25] T. Saba et al. “Brain tumor detection using fusion of hand crafted and deep learning features”. en. In: *Cognitive Systems Research* 59 (Jan. 2020), pp. 221–230. ISSN: 1389-0417. DOI: 10.1016/j.cogsys.2019.09.007. URL: https://www.sciencedirect.com/science/article/pii/S1389041719304735 (visited on 06/17/2023).

[26] M. Nazir, S. Shakil, and K. Khurshid. “Role of deep learning in brain tumor detection and classification (2015 to 2020): A review”. en. In: *Computerized Medical Imaging and Graphics* 91 (July 2021), p. 101940. ISSN: 0895-6111. DOI: 10.1016/j.compmedimag.2021.101940. URL: https://www.sciencedirect.com/science/article/pii/S0895611121000896 (visited on 06/17/2023).

[27] Y. Zheng, H. Tek, and G. Funka-Lea. “Robust and Accurate Coronary Artery Centerline Extraction in CTA by Combining Model-Driven and Data-Driven Approaches”. en. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013.* Ed. by K. Mori et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 74–81. ISBN: 978-3-642-40760-4. DOI: 10.1007/978-3-642-40760-4_10.

[28] H. Kervadec et al. “Boundary loss for highly unbalanced segmentation”. en. In: *Medical Image Analysis* 67 (Jan. 2021), p. 101851. ISSN: 1361-8415. DOI: 10.1016/j.media.2020.101851. URL: https://www.sciencedirect.com/science/article/pii/S1361841520302152 (visited on 06/17/2023).

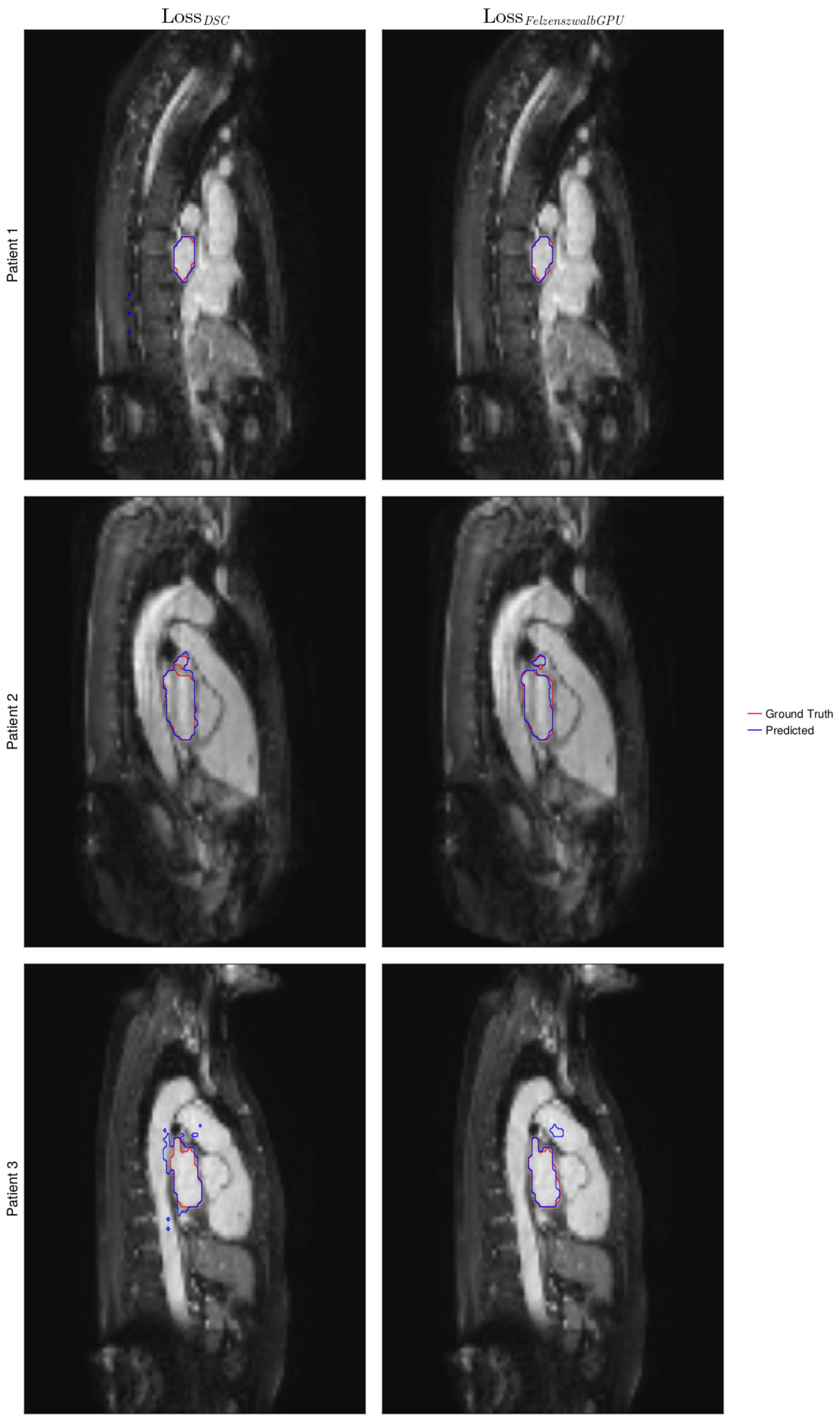

Figure 5: Selected slices of the heart dataset with ground truth and predicted boundaries overlayed. The left column shows the baseline Dice loss model and the right column shows the hybrid HD-Dice loss proposed in this study.