Supplementary

A Overview

In this supplementary material, we provide more quantitative results, technical details and additional qualitative test examples. Section B presents more implementation and training details of SOUL, in Section C the efficacy of the GVD algorithm is discussed, while Section D conducts ablation studies to investigate the impact of different loss functions on the model's performance. At last, Section E demonstrates the performance improvement achieved by employing multiple-scale computation of geometric features compared to a single-scale approach. A video showcasing the performance of SOUL is available at this link: here.

B Additional Implementation Details

As a supplement to Sections 3.4 and 4.4, we provide additional details here.

LiDAR scanner. Table 3 presents a summary of the distinguishing laser characteristics between ULS and TLS. The acquisition of LiDAR data is profoundly impacted by atmospheric characteristics, the LiDAR extinction coefficients exhibit a low impact to atmospheric humidity at both 905 nm and 1550 nm wavelengths (Wojtanowski et al. [47]). ULS uses 905 nm as wavelength, at which the scanning device hast the minimum energy consumption (Brede et al. [3]). Meanwhile, the TLS system operates at a wavelength of 1550 nm, which is more susceptible to the fog impact (Wojtanowski et al. [47]). This poses the issue that the spectral reflectance of leaf and wood is less contrasted (Brede et al. [3]) at the 905 nm wavelength, thereby leading us to employ only point coordinates as input for avoiding the use of intensity information.

Table 3: Laser sensor characteristics

	TLS	ULS
RIEGL Scanner	VZ400	miniVux-1UAV
Laser Wavelength (nm)	1550	905
Beam divergence (mrad)	≤ 0.25	≤ 1.6*0.5
Footprint diameter (cm@100m)	3.5	$16*5$
Pulse duration (ns)	3	6
Range resolution (m)	0.45	0.9

Training details. In addition to the content already illustrated in the main paper, we provide further details regarding the training parameters. First and foremost, sufficient training time is required for the model to achieve desirable performance. Our observation indicates that models trained less than 2,000 epochs are inadequately trained to achieve desirable performance. During the training process, three metrics, AUROC, MCC and Specificity, are considered more informative and insightful. Especially, the metric of specificity is crucial as it measures the ability to accurately discriminate wooden points, which is the primary requirement of our method. Another thing to note is that, contrary to the mentioned practice in the main text of increasing the batch size by a factor of two every 1,000 epochs, we did not strictly follow this restriction during the actual training process. Often, the increase in batch size occurred around 800-900 or 1100-1200 epochs for better using the GPU resources. However, theoretically, this offset should not affect the final performance.

Architecture of DL model. We have made several modifications to the architecture of PointNet++ (MSG) [8]. Firstly, we observed that Adam (Kingma & Ba [48]) outperformed Nesterov SGD (Liu & Belkin [49]) in this task, and ELU (Clevert et al. [50]) activation function was better than ReLU (Nair & Hinton [51]). The fraction of the input units to drop for dropout layer was changed from 0.5 to 0.3, that means the probability of an element to be zeroed is 30% (Paszke et al. [52]). We also decreased the number of hidden neurons and added two more fully connected layers at the end.

Test with error bar. We have calculated a confidence interval at 95% confidence level to indicate the uncertainty of our method. In the main body of the paper, it was mentioned that we obtained

data from four flights, and each data collection from these flights allowed us to obtain a labeled ULS data set through label transfer from TLS data. The test data set is composed by 40 trees from the same positions across these four labeled ULS data sets, resulting in a total of 160 trees. We divided further these 160 trees based on their unique treeID to 16 smaller test sets. However, trees with the same treeID are not placed together in the same test set. The calculation of confidence intervals was derived from these 16 test sets. The result is summarized in Table 4 and Table 5. It is clear that SOUL outperforms all other methods on ULS data and achieves state-of-the-art performance on metrics such as Specificity, Balanced Accuracy and G-mean.

Table 4: Comparison of different methods

Methods	Specificity	G-mean	BA[1]
FSCT [9]	0.13	0.356	0.554
FSCT + retrain	0.01	0.1	0.505
LeWos [4]	0.069	0.259	0.520
LeWos (SoD[2]) [22]	0.069	0.260	0.523
SOUL (focal loss [13])	0.395	0.615	0.677
SOUL (rebalanced loss)	**0.576 ± 0.063**	**0.651 ± 0.030**	**0.720 ± 0.027**

[1] BA (Balanced Accuracy) $BA = \frac{1}{2}(Recall + Specificity)$.
[2] SoD (Significance of Difference).

Table 5: Comparison of different methods

Methods	Accuracy	Recall	Precision
FSCT [9]	0.974	0.977	0.997
FSCT + retrain	0.977	1.0	0.977
LeWos [4]	0.947	0.97	0.975
LeWos (SoD) [22]	0.953	0.977	0.975
SOUL (focal loss [13])	0.942	0.958	0.982
SOUL (rebalanced loss)	0.857 ± 0.014	0.865 ± 0.015	0.988 ± 0.002

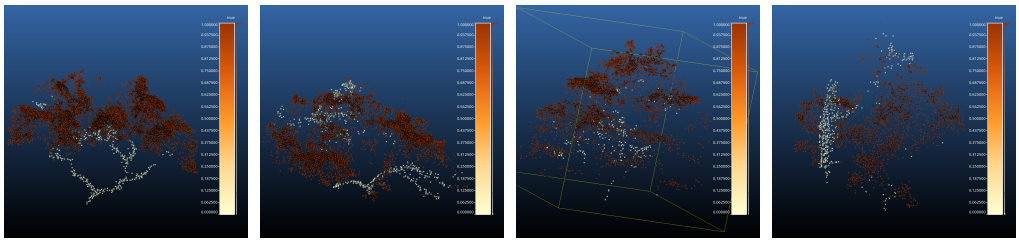
Output contains redundant points. To proceed with further data analysis and usage, it is necessary to remove duplicate points present in the output.

C Assessing the efficacy of GVD

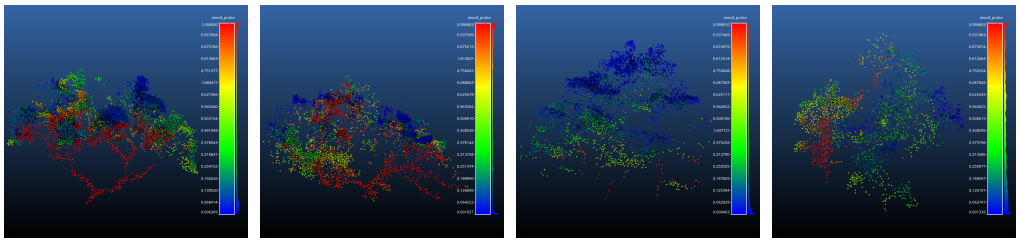
The ULS data often covers several hectares, or even hundreds of hectares. The situation in tropical forests is also highly complex, with various types and sizes of trees densely packed together. The significant memory demand makes it nearly impossible to process all the data at once, leading us to adopt a spatial split scheme approach as a necessary compromise.

We can select data randomly from the whole scene, but selecting data randomly can result in a sparse and information-poor sample. An alternative is to employ a divide and conquer strategy to handle the chaotic, big volume, and complex ULS data. That is why we propose GVD, a method that involves breaking down the data into more manageable subsets (see Figure 8), allowing us to handle the intricacies and extract meaningful insights in a more systematic manner. This approach enables us to retain the information-rich aspects of the data while overcoming computational challenges associated with the sheer volume of data.

Prior to employing the GVD method, we initially adopted a more intuitive approach. The data was partitioned in unit of voxel, serving as component for batch preparation through down-sampling. However, this approach gave rise to border effects, particularly impeding SOUL's focus on the meticulous segmentation of intricate branch and leaf within tree canopy (see Figure 9(c)). The segmentation of cubes led to the emergence of noise point clusters along the voxel edges. ULS have more points on tree canopy, so the presence of noise point clusters on voxel edges is more severe on tree canopy, which imposes bigger obstacles to our leaf/wood segmentation task.

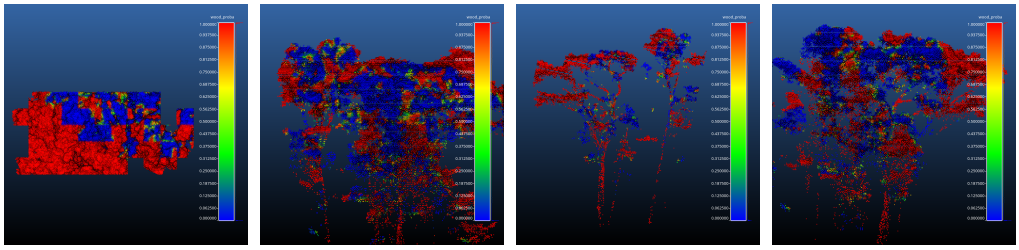


(a) Comp N°1148 (b) Comp N°575 (c) Comp N°1137 (d) Comp N°552



(e) Comp N°1148 (f) Comp N°575 (g) Comp N°1137 (h) Comp N°552

Figure 8: The figure depict distinct samples subjected to GVD processing and presents qualitative results within each sample. The upper row illustrates the ground truth for each sample (where brown represents leaves and white represents wood), whereas the lower row exhibits the predictive results generated by SOUL (where blue represents leaves and red represents wood incorporating gradient colors for transitional probability).

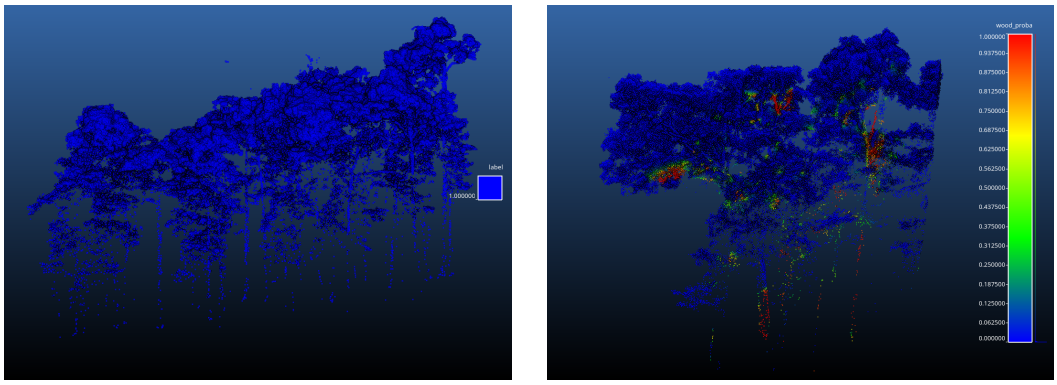


(a) Top view (b) Middle layer (c) Thin layer (d) Back view

Figure 9: Downgrade version's output with sliding window. The use of GVD eliminates observed edge effects in this context.

We experimented with cuboids, a choice aimed at preserving greater semantic information within each component sample and expanding the spatial range for batch selection. Similar to a "sliding window", we can systematically traverse the entire forest with overlapping coverage in this way. But border effects persisted (see Figure 9), prompting the introduction of the GVD method, which led to a substantial improvement. Modifying parameters τ and γ adjusts the coverage scope of each component in GVD segmentation. Additionally, tweaking the minimum accepted number of voxels and points, two GVD's configurable thresholds, within each component effectively manages component size. This process aids in the elimination of low-information content outlier point clusters to a certain extent.

A comparison between the result of the downgraded version using sliding window as spatial split schema and the individual-component point performance of full version SOUL is illustrated in Figure 8. Notably, in component N°552, SOUL effectively discriminates trunk points from leaf points that were previously entirely intertwined, surpassing our initial expectations.



(a) Cross entropy loss (b) Focal loss

Figure 10: The predictions of SOUL model on raw ULS data training with cross entropy loss function and focal loss function. Blue represents wood points with high probability, while red represents leaf points with high probability. Despite utilizing the focal loss, the model remains overwhelmed by leaf points.

D Ablation study of rebalanced loss

In this section, we mainly discuss the significant benefits introduced by the rebalanced loss. The network is biased towards leaf points when using cross-entropy loss function and the use of focal loss function (Lin et al. [13]) does not yield substantial improvements to the task, as its performance remains unsatisfactory when tested on the raw data set (see Figure 10(a) and Figure 10(b)).

Through a comparison of the specificity curves (see focal loss specificity in Figure 11 versus rebalanced loss specificity in Figure 12) and the training/validation loss curves of focal loss and rebalanced loss (see loss curve of the focal loss in Figure 13 versus loss curve of the rebalanced loss in Figure 14), we observed that focal loss failed to address the issue of class imbalance in our task. Plus, it is important to note that in the loss curve of rebalanced loss, the validation loss curve showed significant fluctuations in Figure 14. As foreseen, this variation occurred because the training process employed the rebalanced loss for model training, while the validation process utilized the cross-entropy loss function. Consequently, the loss value used for backpropagation is derived from the rebalanced loss function, which inherently does not consider the majority of leaf points. So once using cross entropy to calculate the loss, all points within a batch are taken into account, which can lead to fluctuations in the validation loss curve. However, it was expected that the validation loss curve would eventually converge on the validation set to demonstrate that using rebalanced loss for backward propagation effectively balances the representation of leaf and wood points. In fact, the final results surpassed focal loss by a significant margin, as evidenced by the convergence of the validation loss curve for rebalanced loss (see Figure 14). For the loss curve of focal loss in Figure 13, both the training and validation processes employ the focal loss function.

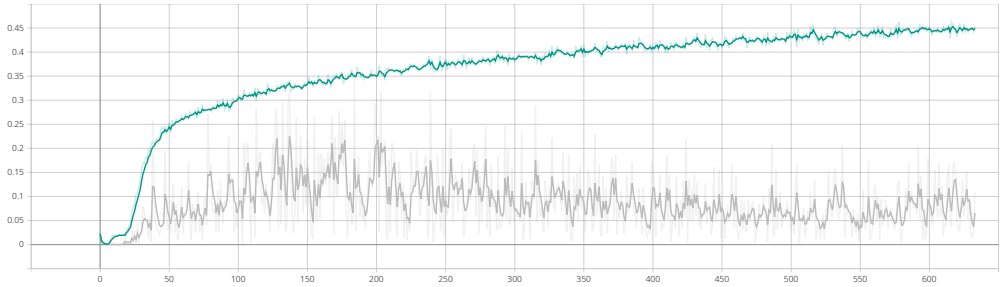


Figure 11: Specificity - focal loss (single-scale feature calculation). In the figure, the cyan curve represents the specificity curve of the training data set, while the gray curve represents the specificity curve of the validation data set.

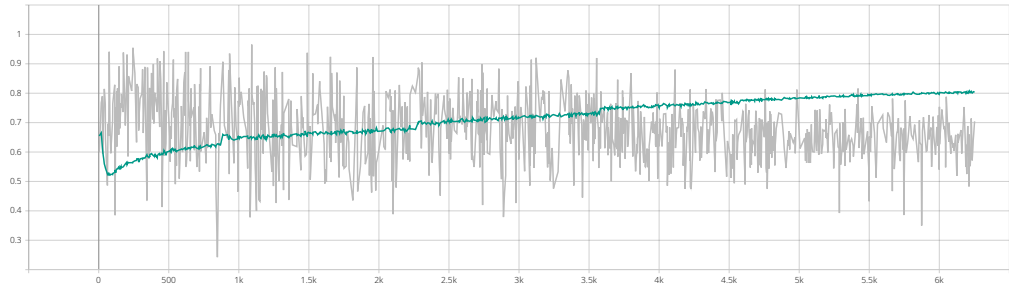


Figure 12: Specificity - rebalanced loss (single-scale feature calculation), cyan curve - training data set, gray curve - validation data set.

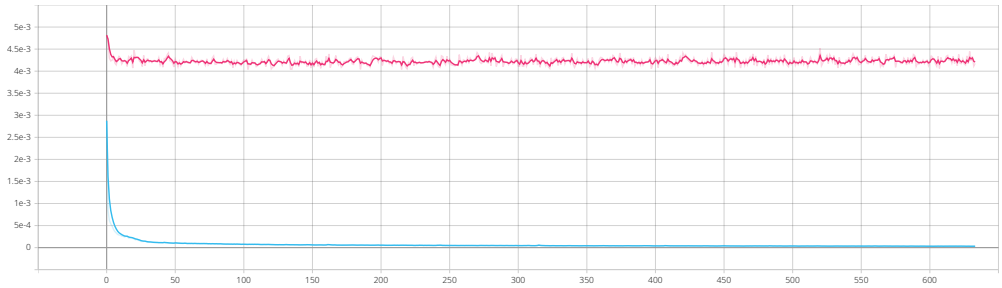


Figure 13: Loss - focal loss (single-scale feature calculation). The blue curve represents the loss curve of the training data set, while the red curve represents the loss curve of the validation data set.

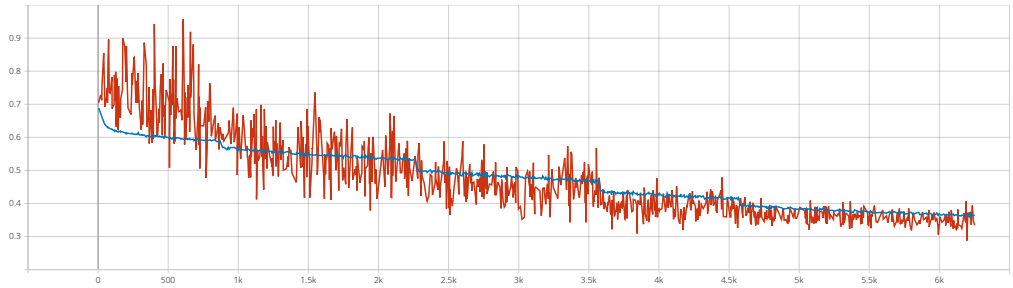


Figure 14: Loss - rebalanced loss (single-scale feature calculation), blue curve - training data set, red curve - validation data set.

In Figure 12, Figure 14, Figure 15, Figure 16 and Figure 17, we can clearly observe the presence of "sharp fluctuations", which are a result of the operation mentioned on main paper and Section B, where we increased the batch size by a factor of two approximately every 1,000 epochs. We followed this practice proposed by Smith et al. [36], which involves increasing the batch size instead of decaying the learning rate.

E Single-scale vs Multiple-scale

In this section, we mainly showcase multiple-scale geometric features calculation outperform single-scale at our task. After applying the rebalanced loss as the loss function, we observed that computing geometric features at multiple scales ultimately improved the performance of SOUL. When comparing the evolution of specificity with the number of epochs between single-scale and multiple-scale (see Figure 16 and Figure 17), multiple-scales not only exhibit higher values but also converge more effectively over time, same for loss convergence (see Figure 14 and Figure 15).

The Matthews Correlation Coefficient (MCC) (see Yao & Shepperd [37]) and AUROC are both effective metrics to evaluate the model performance under class imbalance. Therefore, besides the specificity, we provide the MCC and AUROC values for both single-scale and multiple-scale cases

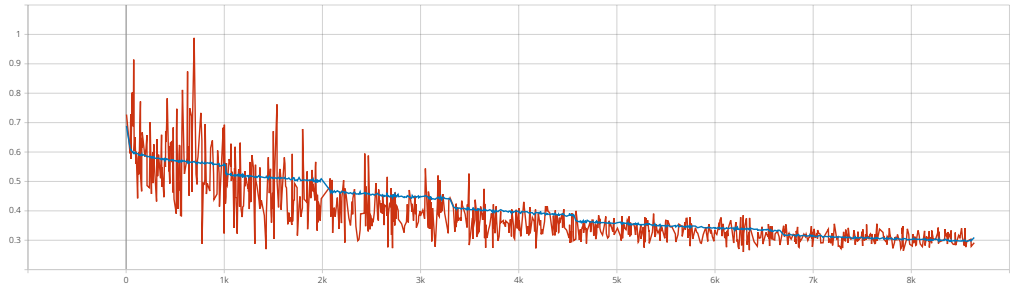


Figure 15: Loss - rebalanced loss (multiple-scale feature calculation), blue curve - training data set, red curve - validation data set.

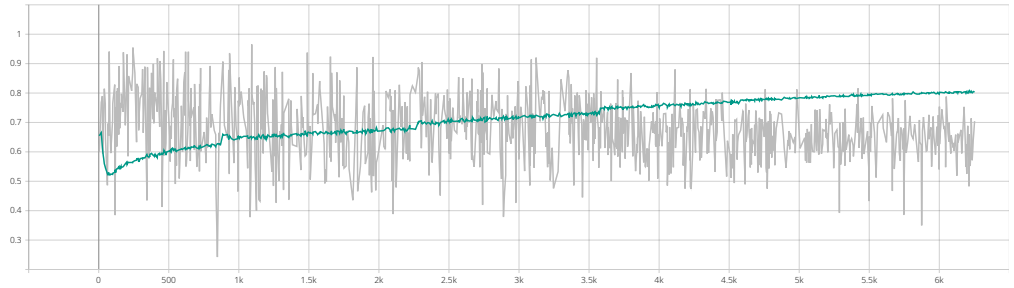


Figure 16: Specificity - Single-scale geometric features calculation, cyan curve - training data set, gray curve - validation data set.

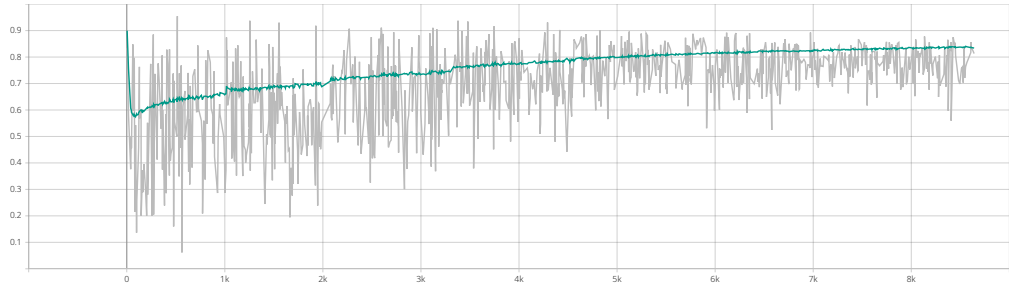


Figure 17: Specificity - Multiple-scale geometric features calculation, cyan curve - training data set, gray curve - validation data set.

during the training process. Upon analyzing the comparative results of MCC in Figure 18 and Figure 19, and AUROC in Figure 20 and Figure 21, we consistently observe that the multiple-scale computation of geometric features output higher values compared to the single-scale. This strongly supports the superior performance of multiple-scale feature computation over single-scale.

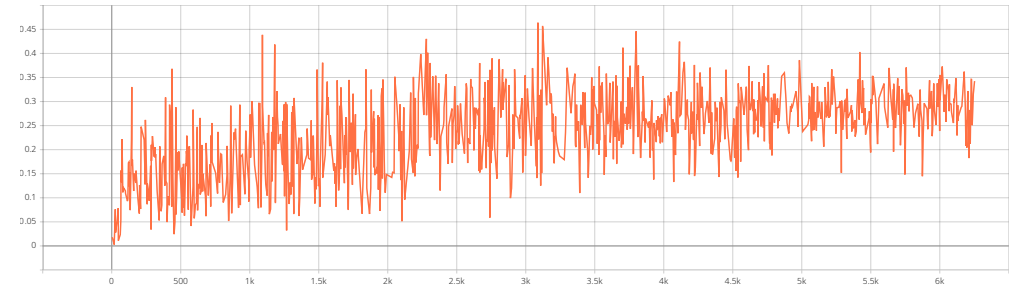


Figure 18: During the training process, the MCC (single-scale) values change with the number of epochs.

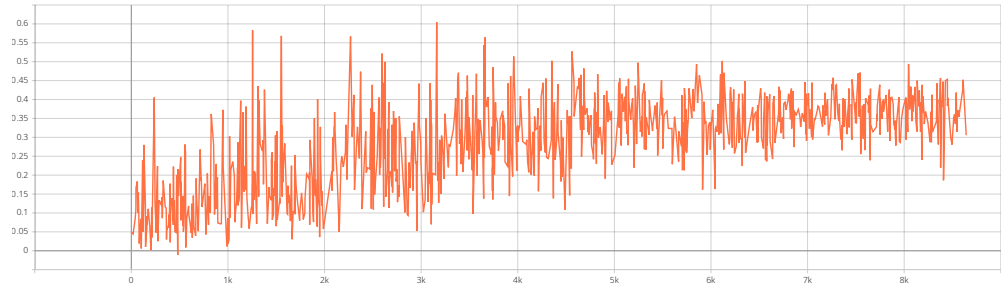


Figure 19: MCC (multiple-scale) values change with the number of epochs.

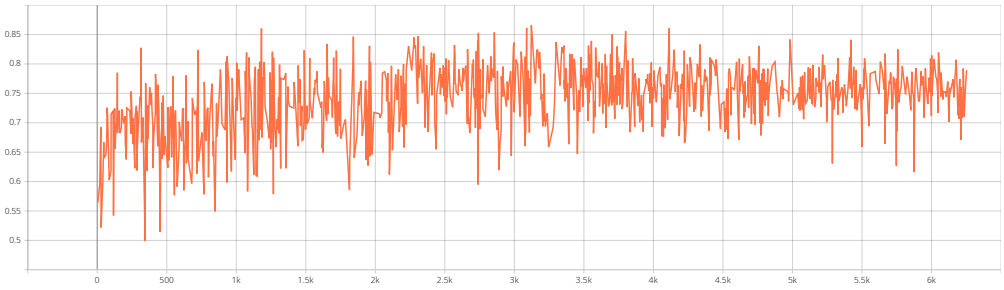


Figure 20: AUROC (single-scale) values change with the number of epochs.

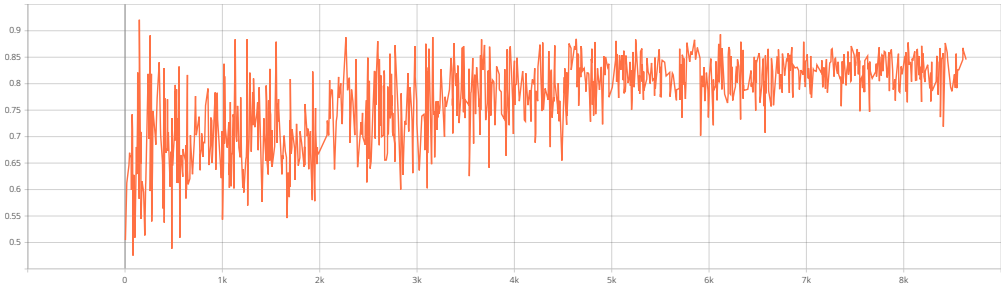


Figure 21: AUROC (single-scale) values change with the number of epochs.

MCC and AUROC provide valuable insights for selecting the final model state (checkpoint). For example, up to now, the best-performing model state is the one obtained at the 3161st epoch with the multiple-scale geometric features computation, as mentioned in the main paper. The model achieves an MCC value of 0.605 and an AUROC value of 0.888, these results generally outperform all single-scale metrics. As MCC is a discrete case of Pearson correlation coefficient, we can conclude that there exists a strong positive relationship exists between the model's predictions and the ground truth by using the interpretation of Pearson correlation coefficient.