ISSN 2464-4617 (print)
ISSN 2464-4625 (online)
Computer Science Research Notes - CSRN
http://www.wscg.eu
WSCG 2025 Proceedings

Improving 3D Monocular Object Detection with Dynamic Loss Function Adjustments

Alexandre Evain
Université Rouen Normandie, ESIGELEC, IRSEEM, France (FRA), 76000 Rouen,
alexandre.evain@groupe-esigelec.org

Redouane Khemmar
Université Rouen Normandie, ESIGELEC, IRSEEM, France (FRA), 76000 Rouen
redouane.khemmar@esigelec.fr

Mathieu Orzalesi
SEGULA Technologies
19 rue d'Arras
France (FRA), 92000, Nanterre
mathieu.orzalesi@segula.fr

Sofiane Ahmedali
Universite d'Evry Val d'Essonne
2 Rue du Facteur Cheval
France (FRA), 91000, Evry-Courcouronnes
sofiane.ahmedali@univ-evry.fr

ABSTRACT

In 3D monocular object detection, optimizing the loss function is crucial for balancing multiple competing metrics, such as depth estimation, orientation, and object dimensions. Traditional approaches use a weighted sum of individual losses, allowing metric prioritization but risking training instability due to competition between terms. To address this, we first experimented with different loss function configurations to see how different loss interactions could emphasize specific metrics. These initial results demonstrated that abrupt changes in loss functions cause significant precision drops, therefore we decided to try dynamic loss functions adjustment, using transition functions to gradually shift metric emphasis over the training process. Among the tested transition functions, the Smoothstep function had the best balance across all metrics, followed by the Linear function, while the Smootherstep function provided strong initial performance but was eventually outperformed. Our results suggest that controlled, smooth transitions between different loss functions can enhance training stability and final detection accuracy, providing a way to improve 3D object detection models without overhauling their architecture.

Keywords

Monocular 3D Object Detection, Loss Function Optimization, Dynamic Loss Adjustment, Transition Functions, Deep Learning, Computer Vision, Training Stability

1 INTRODUCTION

Monocular 3D object detection is a challenging task in computer vision, as it requires estimating an object's position, orientation, and dimensions using only a single image. This contrasts with stereo or LiDAR-based methods that have access to additional sensor data, making monocular methods particularly difficult. In this paper, we explore improvements to the loss function of a modified version of YOLOv7[1] adapted for 3D monocular object detection, focusing on dynamic adjustments to enhance performance across multiple key metrics.

Our current approach involves formulating the loss function as a weighted sum of individual terms, each corresponding to a critical detection metric: 2D object score, depth, orientation, dimensions, and central position. Each of these terms is assigned a coefficient, which is refined during training using an evolutionary adjustment process. This design has the advantage of flexibility, allowing the model to prioritize specific detection metrics at different stages of training. For example, the model can prioritize depth accuracy early in training and focus more on orientation or localization as training progresses.

While this modular approach offers significant flexibility, it also has potential drawbacks. Specifically, there may be competition between the different loss terms. For example, optimizing one metric (such as depth) may inadvertently cause a degradation in another (such as orientation or center position). Moreover, as certain metrics may perform poorly, the overall loss can become misleading, indicating progress even when some critical components of the model's performance have not improved.

One potential alternative to this modular loss design is the adoption of a unified 3D loss function, which has been used by some recent methods. This strategy can help mitigate the competition between individual metrics, but it comes at the cost of reduced modularity and flexibility in prioritizing specific metrics. Given the trade-offs associated with this approach, we have opted to retain our modular loss design, as it offers the flexibility to prioritize different aspects of detection. However, to further enhance this framework, we have decided to experiment with dynamic coefficient adjustment throughout the training process. This allows the model to adaptively prioritize specific components, such as depth and dimensions in the early stages, and localization and orientation in the later stages. This adaptive strategy enables the model to better balance the competing requirements of different metrics, ultimately maximizing overall performance while preserving the advantages of a modular loss structure.

2 RELATED WORK

2.1 Loss Functions for 2D Object Detection

Intersection over Union (IoU) is the standard metric for assessing localization performance in object detection, focusing on the overlap between predicted bounding boxes and ground-truth bounding boxes. Loss IoU directly optimizes this metric but suffers from problems such as reaching a plateau in cases where boxes do not overlap, limiting its usefulness for model learning in challenging scenarios.

Loss Generalized IoU (GIoU) [2] improves on IoU by addressing its limitations in the case of non-overlapping boxes. It incorporates additional geometric properties of bounding boxes, such as the bounding area of their union, ensuring a more consistent optimization process. GIoU loss has demonstrated performance improvements in popular detection frameworks such as Faster R-CNN, Mask R-CNN, and YOLO.

Traditional losses such as cross-entropy for classification and smooth L1 for localization don't always manage to establish an adequate correlation between the two tasks, which can lead to sub-optimal performance. The IoU-Balanced (IoU-B) [3] loss remedies this problem by focusing on examples with a high utility index, thus strengthening the link between classification and localization. This approach has proved particularly effective for single-stage detectors, significantly improving their localization accuracy without sacrificing efficiency.

[4, 5] introduce different Alternating Losses (AL), adding dynamic switching between loss functions as a function of training conditions, targeting universal adverse disturbances. The 3 alternating loss functions for UAP training are : Batch Alternating Loss (B-AL), Epoch-Batch Alternating Loss (EB-AL), and Progressive Alternating Loss Training (P-AL). We will take up this idea of switching and alternating loss functions for monocular 3D detection.

Distance-based IoU variants. [6] proposes both Distance IoU loss (DIoU) as well as Complete IoU loss (CIoU). DIoU introduces a penalty term that minimizes the normalized distance between the centers of the predicted bounding boxes and the ground truth. By incorporating this distance, DIoU achieves faster convergence and better accuracy than IoU and GIoU. Its incorporation into algorithms such as YOLO v3 and SSD has led to significant performance gains, and DIoU has also been used in non-maximal suppression (NMS) for more robust occlusion management. Based on DIoU, CIoU loss takes into account three geometric factors: overlap area, distance from center point and aspect ratio. By integrating these factors, CIoU provides a comprehensive optimization strategy for bounding box regression, enabling faster learning and improved performance.

Mahalanobis loss IoU (MIoU) [7] extends DIoU by incorporating the Mahalanobis distance between predicted and target bounding boxes. This modification solves gradient inconsistency problems and improves localization accuracy. MIoU has shown notable improvements in applications such as night-time vehicle detection, where traditional IoU-based losses are difficult to achieve.

In addition to this work, [8] exhaustively analyzes 31 classic loss functions in machine learning, classifying them into the traditional and deep learning domains. With regard to deep learning, it focuses on specific tasks such as object detection and face recognition, and provides detailed descriptions of the formula, meaning and algorithmic impact of each loss function.

2.2 Loss Functions for 3D Object Detection

Among works dedicated to loss functions for 3D object detection, [9] introduces global geometric constraints into monocular 3D object detection by exploiting the relationships between objects in a scene. It uses homography to reinforce consistency between the image plane and the bird's-eye view, guaranteeing precise relative positions between objects. This loss integrates 2D and 3D spatial information and is designed to be a plug-and-play module compatible with various 3D detectors.

MonoRUn [10] introduces robust KL loss, which minimizes uncertainty-weighted reprojection errors for 3D localization. By projecting dense 2D-3D correspondences onto the image plane, the loss function optimizes pose estimation in a self-supervised manner. This design improves pose depth and accuracy without the need for additional annotations.

YOLOFM [11] contains a Focal-SIoU loss function, which handles bounding box regression by combining SIoU and Focal L1 losses, effectively balancing angle, distance, shape and IoU parameters. This loss accelerates network convergence and improves localization accuracy, particularly for tasks requiring efficient feature fusion and robust detection in resource-limited environments.

Finally, [12] proposes a quasi-isometric (K, B, ε) loss function that uses metric learning to organize object descriptors according to actual depth while preserving geodesic distances in feature space. This loss preserves depth discrimination properties while minimizing interference with other detection tasks. An auxiliary depth estimation head further improves depth prediction during learning, without increasing model complexity or inference time.

3 METHOD

3.1 Initial Experiments

In order to explore how the prioritization of specific metrics affects training, we began by experimenting using a modified YOLOv7[1] model trained on the KITTI dataset[13] with various loss configurations. These initial attempts involved multiplying the depth loss term with others, or combining all loss terms multiplicatively. The aim was to determine whether certain loss interactions could better emphasize specific measures or improve overall training results. We used an 80% / 20% training/validation split, with strict separation between the two to ensure generalization and reduce the risk of overfitting. All reported evaluations were performed exclusively on the validation set, unless otherwise specified.

The formulas of the different loss functions are defined as follow:

- Function 1: Basic Loss

  $L = L_{box} + L_{obj} + L_{cls} + L_{cent} + L_{depth} + L_{dim} + L_{orient}$

  Here, L_{box} the loss associated with the 2D bounding box, it uses the Complete Intersection over Union (CIoU) loss as defined by [6], which can be expressed in the following way (Equation (1)):

  $$L_{\text{CIoU}} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{\text{gt}})}{c^2} + \alpha v \qquad (1)$$

  L_{obj} and L_{cls} use the Classification Head Label Smoothing introduced by [1], as well as Binary Cross Entropy (BCE) Loss. For the rest of the loss functions, we use the smooth L1 loss for L_{cent}, L_{depth}, L_{dim}, and L_{orient}.

- Function 2: Depth-weighted Sum

  $L = L_{depth}(L_{box} + L_{obj} + L_{cls} + L_{cent} + L_{dim} + L_{orient})$

  Prioritize depth estimation, so that depth errors have more influence on the total loss.

- Function 3: Multiplicative Loss

  $L = L_{box} \cdot L_{obj} \cdot L_{cls} \cdot L_{cent} \cdot L_{depth} \cdot L_{dim} \cdot L_{orient}$

  If one of the terms is large, the total loss increases considerably. This approach favors balanced learning between all measurements but could suffer from instability if one of the terms becomes too small or too large.

- Function 4: Euclidean Loss

  $$L = \sqrt{\begin{aligned} &L_{box}^2 + L_{obj}^2 + L_{cls}^2 + L_{cent}^2 \\ &+ L_{depth}^2 + L_{dim}^2 + L_{orient}^2 \end{aligned}}$$

  This method reduces the effect of outliers compared to a simple sum, smoothing out large errors. It is commonly used in distance-based optimization, where squared errors emphasize larger deviations.

- Function 5: Multiplicative Shift Loss

  $L = (1 + L_{box}) \cdot (1 + L_{obj}) \cdot (1 + L_{cls}) \cdot (1 + L_{cent}) \cdot (1 + L_{depth}) \cdot (1 + L_{dim}) \cdot (1 + L_{orient})$

  This variant of loss #3 avoids the collapse of zero losses (when a loss term becomes too small and reduces the impact of learning). The offset of 1 prevents any term from reaching 0 completely, while still applying multiplicative interactions between the different terms.

The evaluation itself is done using usual 2D metrics (Precision, Recall, and Average Precision (AP) at IoU thresholds of 0.5 and 0.95) combined with the Depth Error, the Center offset & Dimension Score defined by [14] and the Orientation Score. We can see the results of training the different loss functions below in figures 1, 2 and 3.

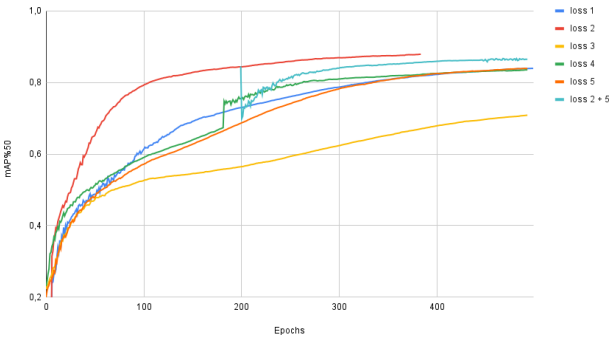

Figure 1: mAP@0.5 Score of the different loss functions during training

According to figure 1, loss function 2 performed best at mAP@0.5 compared to the base function, while functions 4 and 3 performed worst.

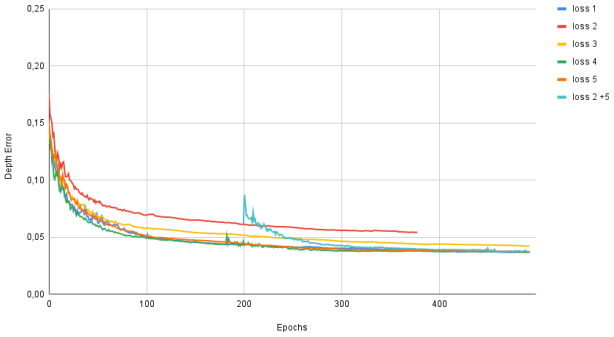

Figure 2: Depth error of the different loss functions during training

For depth scores, (Fig. 2) functions 1, 4, and 5 have the best scores, while loss function 2 has the worst depth score, despite supposedly theoretically having to prioritize depth loss over the other functions. Its elevated score at mAP@0.5 seems to indicate that multiplying all the loss terms by the depth loss decreases the latter's impact rather than increasing it.

Finally, there is not much difference in the behavior of the functions for the CS score (Fig. 3), except for function 2, which fails to learn these predictions. Worse still, their scores decline during training. Changes made by switching to function 5 after training with function 2 compensate for these problems, as seen in function 2+5, however, it causes a sudden map@0.5 score collapse.

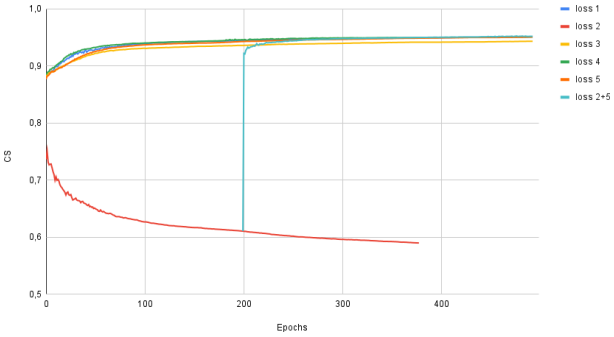

Figure 3: Center Score of the different loss functions during training

During these first experiments, we observed that:

- The abrupt change in the loss function during training led to a significant drop in accuracy, which disrupted the learning stability of the model.
- Despite this pronounced drop in accuracy, the maximum performance achieved during training exceeded that of a normal training configuration. This suggests that dynamic modification of the loss function holds promise for improving the overall training process.

Based on these results, we concluded that sudden transitions between loss configurations are too disruptive. We therefore proposed a gradual approach: the implementation of an epoch-dependent loss function that allows smooth transitions from one configuration to another. By employing a transition function, we sought to take advantage of the benefits of dynamic loss prioritization while maintaining drive stability and avoiding abrupt performance degradation.

3.2 Transition functions

In this section, we present the various transition functions used to dynamically adjust the coefficients of individual loss components during training. Our main objective is to find a method that allows loss coefficients to evolve adaptively throughout the learning process, prioritizing specific parameters at different stages. This approach aims to mitigate the challenge posed by competing metrics in our loss function, where one metric may improve while another deteriorates. By experimenting with transition functions, we explore the possibility of dynamically modifying the training orientation without sacrificing modularity and priority control of individual metrics.

To this end, we have introduced transition functions that dynamically modulate coefficient values. These transition functions interpolate between two extreme states (e.g., giving priority to one measure or another) as learning progresses. We describe several transition functions below, each designed to provide different levels of smoothness and control over the prioritization process.

These functions are the linear transition (Fig. 4), the sigmoid transition (Fig. 5), the smoothstep transition (Fig. 6), the smootherstep transition and the Gaussian transition (Fig. 7).

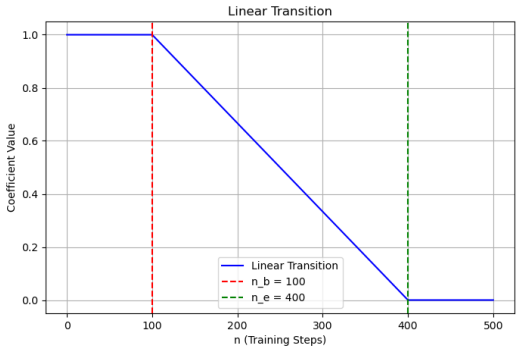

Figure 4: Linear Transition Function

Linear Transition. Linear transition (Fig. 4) provides a simple and constant rate of change between two states. This function guarantees a gradual transition of coefficient values from one metric to another as the drive progresses. Its expression is the equation 2:

$$x(n) = b \cdot \frac{n - n_b}{n_e - n_b} + a \cdot \frac{n_e - n}{n_e - n_b} \quad (2)$$

With a and b our two loss functions, n_b the epoch at the start of the transition and n_e the epoch at the end of the transition.

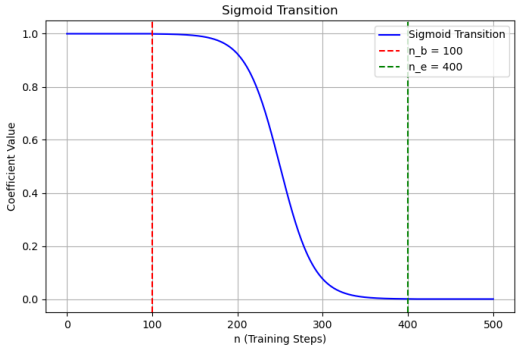

Figure 5: Sigmoid Transition Function

Sigmoid Transition. The sigmoid transition is characterized by a smooth curve (Fig. 5). It starts slowly, accelerates in the middle and slows down again towards the end, providing a smooth transition between the two functions to avoid a drop in accuracy. It is defined by the equation 3:

$$x(n) = a \cdot \sigma(n) + b \cdot (1 - \sigma(n)),$$
$$\sigma(n) = \frac{1}{1 + e^{-k(n - \frac{n_b + n_e}{2})}} \tag{3}$$

With $\sigma(n)$ the sigmoid function and k the slope of the transition around the midpoint $\frac{n_b+n_e}{2}$.

Smoothstep Transition The smoothstep function transitions between two values using a cubic polynomial. It ensures a smooth change of coefficient values with zero slope at both ends, offering a more gradual and natural transition than the linear function. We can see this function in figure 6 and it is defined by the equation 4:

$$x(n) = a \cdot (1 - 3t^2 + 2t^3) + b \cdot (3t^2 - 2t^3)$$
$$t = \frac{n - n_b}{n_e - n_b} \tag{4}$$

Smootherstep Transition. The smootherstep function is an extension of smoothstep with a quintic polynomial, offering an even smoother transition. The difference between the two functions can be seen in figure 6. It is defined by the equation 5:

$$x(n) = a \cdot (1 - 6t^5 + 15t^4 - 10t^3) + b \cdot (6t^5 - 15t^4 + 10t^3)$$
$$t = \frac{n - n_b}{n_e - n_b} \tag{5}$$

Gaussian transition

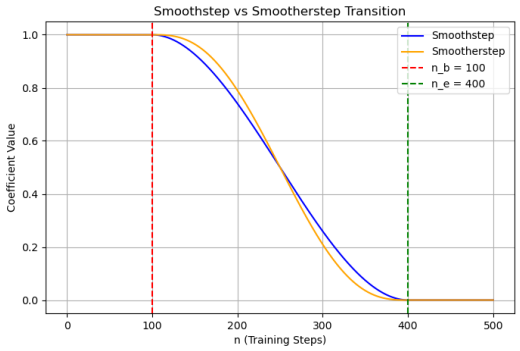

Figure 6: Comparison between Smoothstep and Smootherstep Transition Functions

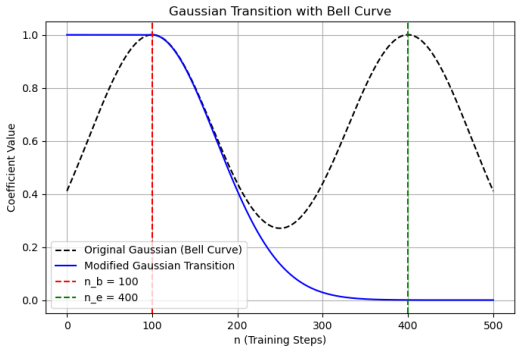

Figure 7: Gaussian Transition Function

The Gaussian transition (Fig. 7) uses a bell-shaped curve to blend between two coefficient values. It is defined by the equation 6 :

$$x(n) = a \cdot \exp\left(-\frac{(n - n_b)^2}{2\sigma^2}\right) + b \cdot \exp\left(-\frac{(n_e - n)^2}{2\sigma^2}\right) \tag{6}$$

With σ the transition control coefficient.

4 EXPERIMENTAL RESULTS

In order to evaluate the effectiveness of the various transition functions, we used several performance metrics : mAP@0.5, mAP@0.95, Depth Error, Orientation Score (OS), Center Score (CS), and Dimension Score (DS).

MAP 0.5 and mAP 0.95 metrics. For mAP@0.5, the Smoothstep transition function performed best, closely followed by the Smootherstep, Linear, Sigmoid, and Gaussian functions, with the reference method achieving the lowest score. Although Smootherstep showed the best initial performance, it was eventually surpassed by Smoothstep. The same ranking was observed for mAP@0.95, but with one notable difference: the linear transition also managed to outperform the Smootherstep transition at the end of training. Figures 8 illustrate these results.

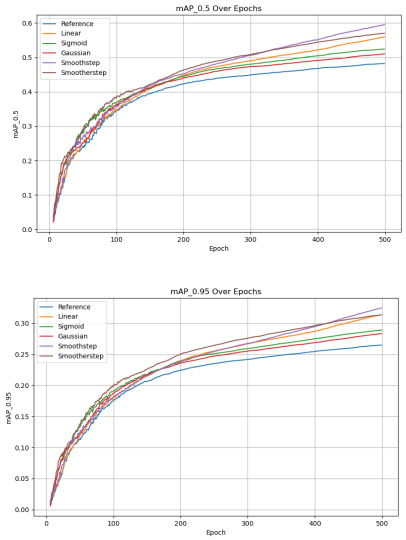

Figure 8: Comparison of mAP 50% and mAP 95% for the different transition functions

Depth errors. In terms of depth estimation (Fig. 9), the linear transition produced the lowest depth error, outperforming all other transition functions. Smoothstep and the reference method followed closely, while Smoothstep, Gaussian, and Sigmoid showed higher error rates.

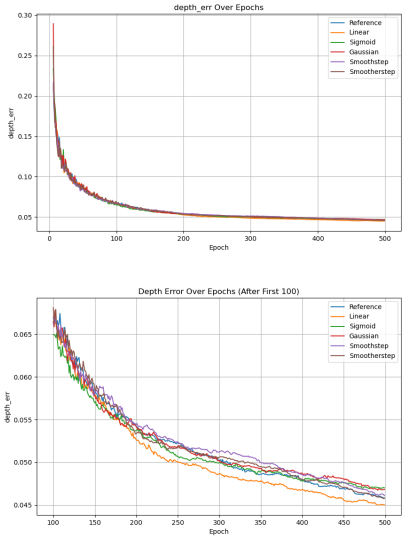

Figure 9: Comparison of depth errors between the different transition functions

Orientation Score (OS). For orientation estimation (Fig. 10), the linear function again achieved the best results, followed by the Smoothstep function. Smootherstep and Gaussian achieved similar results, while Sigmoid and the reference model obtained the lowest scores.

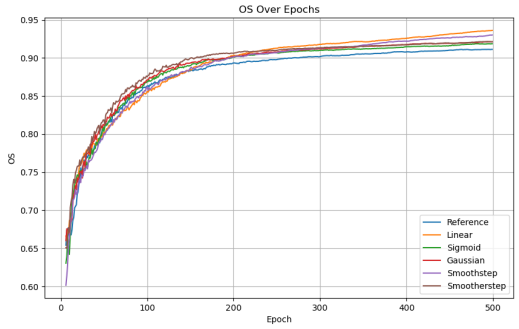

Figure 10: OS metric scores for the different transition functions

Center Score (CS). The ranking of center position accuracy (Fig. 11) was similar to that of OS : Linear transition performed best, followed by Smoothstep and then Smootherstep. The sigmoid and Gaussian models were tied, with the reference model performing least well.

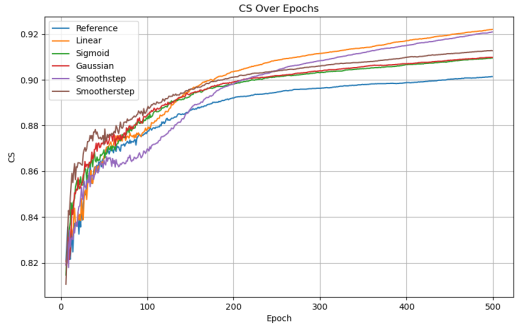

Figure 11: CS metric scores for the different transition functions

Dimension Score (DS). For object dimension estimation (Fig. 12), the Smoothstep transition provided the best performance, followed by the linear transition and then the Gaussian transition. The Smoothstep and Sigmoid transitions performed less well, with the reference model again ranking last.

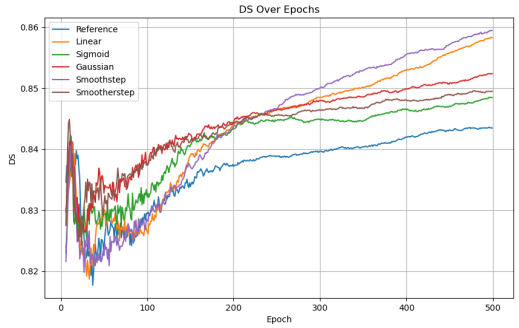

Figure 12: DS metric scores for the different transition functions

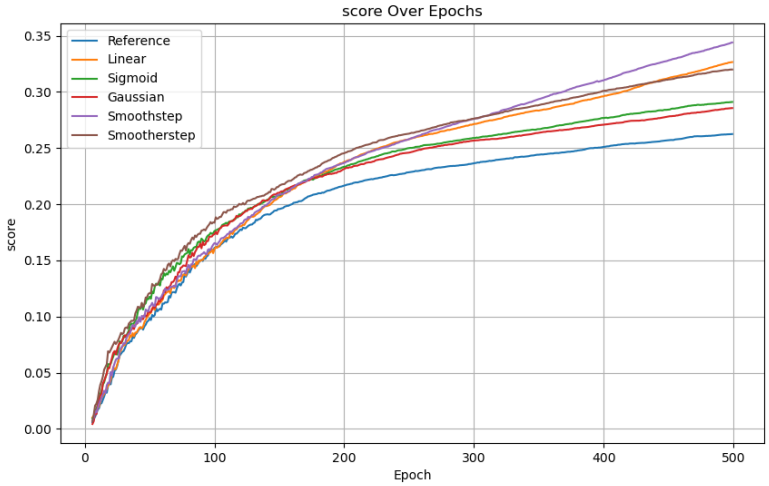

Figure 13: Overall performance of the various transition functions

For OS, CS, and DS metrics, the first 100 Epochs of training showed significant instability for all functions. In all cases, scores initially peaked before declining as training progressed. This can be explained by the fact that the focus shifted to other loss components at the start of training. Since there is not yet a transition between functions during these Epochs, all transition functions follow the same pattern during this initial phase, variations in performance are due to training noise rather than inherent differences in the functions themselves.

Overall performance: To evaluate overall performance, we use a global score following the equation 7, the mAP3D evaluation being too long to be included in the training.

$$\text{Global Score} = \text{mAP@0.5} \times \text{CS} \times \text{OS} \times \text{DS} \times \left(1 - \sqrt{\text{Depth Err}}\right) \quad (7)$$

From the Score results (Fig. 13), Smoothstep emerged as the best-performing transition function, followed by Linear and Smootherstep. Sigmoid and Gaussian functions performed less well, and the reference model obtained the lowest overall score. These results suggest that Smoothstep provides the most balanced optimization between the different metrics, making it the most efficient loss transition function in our experiments.

Among the transition functions, Loss 1 performed better than most, while Gaussian achieved the worst overall score.

5 CONCLUSION

In this paper, we investigated various loss transition functions to enhance the learning process for monocular 3D object detection. Our experimental findings demonstrated that dynamically adjusting the loss function over time can lead to improvements in overall performance when compared to a static learning approach.

Among the transition functions evaluated, Smoothstep demonstrated the most balanced performance across all key metrics, attaining the highest overall score. It exhibited particular strengths in mAP and dimension estimation while maintaining adequate performance in orientation and center position accuracy. The linear transition also performed well, particularly in depth estimation, orientation, and center position accuracy, establishing it as a competitive alternative. Smootherstep initially exhibited promising performance but was ultimately surpassed by Smoothstep and Linear in subsequent training phases. Conversely, Sigmoid and Gaussian functions demonstrated suboptimal performance across most metrics, underscoring the constraints imposed by certain smooth transition methodologies in this setting.

The findings of this study demonstrate that the adoption of gradual transitions between disparate loss functions enhances learning stability and final detection accuracy. Notably, the efficacy of Smoothstep as a transition function is particularly pronounced. Future research endeavors could involve the exploration of advanced refinements, such as adaptive transition functions that respond dynamically to fluctuations in performance in real-time, as opposed to adhering to a predetermined transition schedule.

6 ACKNOWLEDGMENTS

This research is funded and supported by SEGULA Technologies. We would like to thank SEGULA Technologies for their collaboration and for allowing us to conduct this research. We would like to thank also the engineers of the Autonomous Navigation Laboratory (ANL) of IRSEEM for their support. In addition, this work was performed, in part, on computing resources provided by CRIANN (Centre Regional Informatique et d'Applications Numeriques de Normandie, Normandy, France).

7 REFERENCES

[1] Z. Zhang, T. He, H. Zhang *et al.*, "Bag of freebies for training object detection neural networks," 2019.

[2] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 658–666.

[3] S. Wu, J. Yang, X. Wang, and X. Li, "Iou-balanced loss functions for single-stage object detection," *Pattern Recognition Letters*, vol. 156, pp. 96–103, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865522000289

[4] D. Sen, B. T. Karli, and A. Temizel, "Training universal adversarial perturbations with alternating loss functions," in *The AAAI-22 Workshop on Adversarial Machine Learning and Beyond*, 2022.

[5] D. Şen, "Universal adversarial perturbations using alternating loss functions," Master's thesis, Middle East Technical University, 2022.

[6] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 12 993–13 000.

[7] S. Jiang, H. Qin, B. Zhang, and J. Zheng, "Optimized loss functions for object detection and application on nighttime vehicle detection," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 236, no. 7, pp. 1568–1578, 2022.

[8] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A comprehensive survey of loss functions in machine learning," *Annals of Data Science*, pp. 1–26, 2020.

[9] J. Gu, B. Wu, L. Fan, J. Huang, S. Cao, Z. Xiang, and X.-S. Hua, "Homography loss for monocular 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1080–1089.

[10] H. Chen, Y. Huang, W. Tian, Z. Gao, and L. Xiong, "Monorun: Monocular 3d object detection by reconstruction and uncertainty propagation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 379–10 388.

[11] X. Geng, Y. Su, X. Cao, H. Li, and L. Liu, "Yolofm: an improved fire and smoke object detection algorithm based on yolov5n," *Scientific Reports*, vol. 14, no. 1, p. 4543, 2024.

[12] W. Choi, M. Shin, and S. Im, "Depth-discriminative metric learning for monocular 3d object detection," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 80 165–80 177. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/fda257e65f46e21dbc117b20fd0aba3c-Paper-Conference.pdf

[13] A. Geiger, P. Lenz *et al.*, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[14] H.-N. Hu, Q.-Z. Cai *et al.*, "Joint monocular 3d vehicle detection and tracking," 2019.