Reproducibility Study Of LaneSegNet: Map Learning with Lane Segment Perception for Autonomous Driving

Bart de Zwart Erik Meijer Lucas Ponticelli Luuk Versteeg Taiki Papandreou-Lazos

Tobie Werner

Abstract

We address the problem of scene reasoning by autonomously driving vehicles based on multi-view images and standard-definition maps. Developing models proficient at this task is beneficial for the safe operation of driverless vehicles. We participate in the Mapless Driving track of the Autonomous Grand Challenge, which stimulates the development of models for recognizing traffic elements and lane centerlines and understanding their topological relationships. The state-of-the-art LaneSegNet system does this in an end-to-end manner, using a lane attention module with a heads-to-regions mechanism and an identical initialization strategy of reference points. We explore the effectiveness of this system by evaluating alternative backbones in its encoding mechanism. Our analysis shows that using a backbone larger than the original ResNet-50 baseline degrades performance.

1. Introduction

An accurate perception of the environment is essential for autonomous vehicles to safely navigate. To provide the required information, efforts have been made to create High-Definition (HD) maps. These maps contain detailed information about road geometry and topology, and serve as a valuable resource. However, these maps are often unavailable due to high annotation costs associated with creating and keeping them up-to-date. Autonomous vehicle company Waymo, for instance, manually creates these maps by driving around in sensor-equipped vehicles, processing the data offline, and testing the maps before being deployed. Since the last decade, they have mapped around 25 cities in the USA, whereas it contains 336 cities with a population of over 100.000 people. The creation of HD maps is a process which does not scale easily.

To address this, research is being conducted on the automatic construction of HD maps in an online setting. The Mapless Driving track of the Autonomous Grand Challenge, using the OpenLanev2 benchmark, stimulates this research. In this challenge, the inputs are multi-view images from several vehicle-mounted cameras and a Standard-definition (SD) Map containing only topological and positional priors. Models submitted to the challenge should be able to use this information to find the drivable lane segments of the road around itself. One of the SOTA models that accomplishes this is LaneSegNet, which detects both centerlines and borders of lane segments, along with their topology relations.

In this paper, we seek to build off the LaneSegNet model and find ways to improve it. We experiment with replacing the ResNet-50 image object classifier used as the encoder backbone with two alternative backbones.

2. Related work

Previous research has been conducted into developing systems that can solve complex problems in traffic. To this end, extensive efforts have been made to detect traffic elements, such as traffic signs, lights, and lane centerlines. Recently, topology reasoning has gained interest due to its importance in downstream tasks such as planning [21]. Topology reasoning consists of two parts: linking centerlines to centerlines and connecting traffic elements to centerlines. Previous approaches include modular architectures and systems that operate in an end-to-end manner.

2.1. Centerline detection

Centerline perception, or lane graph learning, focuses on deciphering the structure of lanes from vehicle-mounted sensor data, a topic that has seen significant advancements recently. Various methodologies have emerged to address this challenge effectively. For instance, STSU [1] introduced a DETR-like network for centerline detection, which uses a multi-layer perceptron (MLP) to determine connectivity. Building on this foundation, they added minimal cycle queries to ensure the correct order of overlapping lines [2]. Another innovative approach is CenterLineDet [22], which treats centerlines as vertices within a graph and employs a graph-updating model refined through imitation learning. LaneGAP [13] introduced a path-wise method for lane graph reconstruction via an additional conversion algorithm, while TopoNet [8] aimed to explicitly model the connectivity of centerlines and incorporate various traffic elements into the network.

2.2. Traffic element detection

Shifting to map detection, recent trends emphasize moving from 2D camera plane detections to 3D space mappings to minimize projection errors. Bird-Eye-View (BEV) perception has set new benchmarks in this domain, focusing on high-definition maps through segmentation and vector-based methods. Despite the granularity provided by dense segmentation, challenges in representing overlapping elements persist. Addressing this, VectorMapNet [14] represents each map element as a sequence of points, facilitating accurate decoding of lane line locations. MapTR [12] uses a permutation-based model for points, improving performance and clarity, while PivotNet [3] offers a picot-based set prediction framework to minimize redundancy and improve accuracy. StreamMapNet [23] enhances detection stability through multi-point attention and temporal data. Our chosen model, LaneSegNet [13], leverages these advancements by adopting a unified lane segment representation for all HD map elements, integrating segment-level geometry and semantics to advance autonomous driving capabilities.

2.3. Topology reasoning before LaneSegNet

Liao et al. [12] propose MapTR, an end-to-end transformer designed to create high-definition (HD) maps using input images captured by sensors mounted on a vehicle. With their approach, a map element is represented by a set of points together with a group of equivalent permutations. The former is required because map elements often have dynamic shapes, which can not be captured well using bounding boxes. The latter is useful in fixing ambiguities in representing map elements as polylines and polygons. Furthermore, they use a hierarchical query embedding mechanism to encode information and to learn map elements.
TopoNet, released after MapTR, is the first end-to-end method capable of complex topology reasoning in traffic environments [9]. Initially, it extracts front-view features and bird's-eye-view features from the input multi-view images. Then, two separate branches of deformable decoders are used on the features to create instance-level embeddings for traffic elements and centerlines. A Scene Graph Neural Network is then used to refine the centerlines queries (in terms of position and topology). Lastly, heads transform the queries into the final predictions.
After TopoNet, TopoMLP [21] was introduced as a pipeline for topology reasoning. This method represents centerlines as Bézier curves, characterized by an anchor point and multiple control points. Detection is performed using a ResNet-50 backbone to generate feature maps from multi-view input images. The authors use 3D position embeddings encoded into visual features. Learnable 3D anchor points are initialized and updated using a stack of transformer decoder layers. Then, two MLPs are used to predict the positions of the control points. Traffic element detection uses query-based detection (deformable DETR) improved by YOLOv8 bounding box proposals. Lastly, lane-lane and lane-traffic element relationship predictions are made using MLP networks.

2.4. OpenLane-V2

OpenLane-V2 is the first dataset emphasizing the topological relationship between traffic elements and lane centerlines [20]. OpenLane-V2 expands upon the original OpenLane dataset, which solely contains images annotated with 3D centerlines. Added to this dataset are 2D annotations (bounding boxes) of traffic elements in the front-view images and their connection to centerlines. Besides the dataset, the authors propose a new "scene structure perception and reasoning" learning task with the corresponding OLS (OpenLane-V2 score) metric. For models to achieve a high score on this metric, they must be proficient at determining the drivability of lines given their state, influenced by the presence of traffic elements. In addition, traffic elements, centerlines and connections between centerlines must be determined correctly.

2.5. Snellius supercomputer

Snellius is a supercomputer located in the Netherlands used to perform scientific research hosted at SURF. The computer consists of nodes containing different computational resources. In total, the supercomputer can reach a performance of 14 petaflops/s[1].

3. Methodology

In this section, we explain how the LaneSegNet architecture works and what attempts were made to run LaneSegNet and improve its performance according to the OpenLane-V2 scoring metrics [20].

3.1. LaneSegNet

LaneSegNet is one of the traffic element and centerline detection methods provided, serving as a baseline for the CVPR Mapless Driving challenge. As LaneSegNet boasts a superior accuracy compared to the other provided methods, we have opted to expand upon LaneSegNet as our contribution to the Mapless Driving challenge [10].

[1] https://servicedesk.surf.nl/wiki/display/WIKI/Snellius

The LaneSegNet architecture consists of three main components:

- LaneSeg Encoder, the encoder converts the multiple-view input images to a BEV (birds-eye view) feature. In the original paper, a ResNet-50 backbone is used to obtain a feature map from the set of input images. The input images in conjunction with their feature maps are used in the BEVFormer [11] module to transform the images into a BEV feature.
- LaneSeg Decoder, the decoder refines Lane Segment queries through self-attention and cross-attention with the BEV features from the encoder. For Lane Attention, local and distant details in the image are important for decision-making. Thus, LaneSegNet proposes a two-part mechanism to capture all details:
  - A "heads-to-regions" mechanism that uniformly distributes samples within a lane segment.
  - A "multi-branch" mechanism where each attention head attends to a set of locations within a region from the heads-to-region mechanism.

  Mathematically, lane attention is calculated by:

  $$LaneAttn(q_i, p_i, B) = \sum_{m=1}^{M} W_m [\sum_{k=1}^{K} a_{i,m,k} \cdot$$
  $$W'_m Bi - Linear(B, p_{i,m} + \delta p_{i,m,k})]$$

  Where B is a BEV-feature, q_i is a lane segment query feature, p_i is a set of reference points, m is the attention head, k are the indices of the sampling locations, $W'_m and W_m$ are learnable weights.

  The positions of reference points need to be determined for the lane attention module to focus on sampling points. LaneSegNet uses an identical initialization mechanism to distribute these reference points. For each query, reference points are placed depending on the segment prediction of the previous layer. The heads of the first layer are adjusted to identical reference points generated from the positional query.
- LaneSeg Predictor, the predictor generates a predicted lane segment from the refined lane segment query. The predictor consists of two main branches:
  - A topology branch for centerline and lane offset prediction.
  - A topology branch that takes the query features and outputs a weighted adjacency matrix for the lane graph G.

3.2. LaneSegNet reproduction

Our first step to improving the architecture of LaneSegNet is to reproduce the results presented in the original paper [10] and use it to generate a valid submission file for the Mapless driving challenge.

3.2.1 OpenLane-V2 preprocessing

To train the LaneSegNet architecture, the OpenLane-V2 dataset is used [20]. The OpenLane-V2 paper introduces a GitHub repository that provides instructions for preprocessing the data into .pkl files. For LaneSegNet, we use the custom preprocessing provided by the authors to preprocess OpenLane-V2 test, train, and validation data into .pkl files. After preprocessing, we set up a symbolic link from LaneSegNet to the preprocessed OpenLane-V2 data.

3.2.2 Local

After creating a working custom Conda environment for LaneSegNet, we trained LaneSegNet for five epochs on a single Nvidia RTX 3080 GPU using a ResNet-34 backbone. Although the ResNet-34 training results have been generated for testing purposes only, we have decided to mention them in the paper as they serve as an interesting comparison to our results from the ResNet-101 backbone.

3.2.3 Snellius

We tried running the LaneSegNet architecture on the supercomputer cluster, Snellius [2], to further improve our results. Using the GPUs of this cluster which are A100s[3] on the GCN node, we knew that larger backbones would be able to fit as they have more VRAM than we had locally, 40GB vs 10 GB respectively. Using the same Conda environment as locally, we did not manage to run it due to package issues related to MMCV that were not recognizing the GPU compiler on Snellius.

3.2.4 UvA Robolab Workstation

We thus reverted to using a machine we could have easier access to, namely a workstation from the Robolab equipped with an RTX 3090 GPU. This GPU has 24GB of VRAM[4], more than we used locally. Subsequently, we installed all of the packages required to run LaneSegNet on this workstation, and we got it to work with ResNet-101 as the backbone.

3.3. Improving LaneSegNet

In this subsection, we will be introducing two improvements we considered to achieve a better score. The results and analysis of these results will later be presented in Section 4.

[2]https://www.surf.nl/diensten/snellius-de-nationale-supercomputer

[3]Hardware on Snellius: https://servicedesk.surf.nl/wiki/display/WIKI/Snellius+hardware#Snelliushardware-NodesOverview

[4]https://www.nvidia.com/en-my/geforce/graphics-cards/30-series/rtx-3090-3090ti/

3.3.1 PV-to-BEV backbones

One of the improvements we decided to look into was using different backbones for detecting objects and specifically lanes in our use case. We decided to experiment with different backbones.

- **EfficientNet** Mingxing & Quoc [18] introduce EfficientNets, which we wanted to experiment with as it achieved much better performance than ResNet-50 on the ImageNet dataset and transferred well to other datasets. Unfortunately, none of the different EfficientNet versions seemed to fit on the Workstation's GPU, as a result, we followed with experimenting on different backbones.

- **MobileNet** We thought of using a lighter version of EfficientNets which we found with MobileNetV2 and MobileNetV3 [17][5], but unfortunately the same issue arose, namely that it would not fit on the GPU.
- **MaxVit** The pytorch MaxVit model [19] did not fit on our GPU, similar to the EfficientNet and MobileNet models.
- **SwinTransformer** The format of the SwinTransformer [15] did not directly work with the LaneSegNet architecture, which is why we decided to look at different models first.
- **ResNet**. We looked at the larger ResNet models [4] that boast higher accuracies on the ImageNet dataset compared to the ResNet-50 that has been used in the original LaneSegnet architecture. While ResNet-152 appeared to be too large to fit on the workstation GPU, we were able to run LaneSegNet using ResNet-101.

Besides the models mentioned above, we have looked at the following models:

- **regnetx_400mf** [16]
- **DenseNet** [6]
- **NASNet** [25]

Unfortunately, none of these models were found in LaneSegNet's model registry so we were not able to use them.

3.4. Metrics

The primary metric used in the CVPR 2024 Mapless Driving challenge is the OLUS (OpenLane-V2 UniScore)[5]. OLUS is an average of the metrics that cover the primary task of Lane Segmentation and Topological reasoning:

$$OLUS = \frac{1}{5}[DET_l + DET_a + DET_t + f(TOP_u) + f(TOP_{lt})]$$

DET_l is the mAP of centerline perception performance, based on the Frèchet distance between the ground truth and recreated BEVs as seen in Figure 2. The lane attention role within the LaneSegNet decoder plays a significant role in improving the DET_l score.

DET_a is the mAP of map elements such as pedestrian crossings and road boundaries and are measured similarly using Chamfer distance to DET_l. To improve the DET_a score in LaneSegNet, an additional MapTR attention head is used in the LaneSegNet decoder to detect map elements [12].

DET_t is the mAP of traffic elements such as traffic lights and is measured using IoU. To improve the DET_t score, LaneSegNet uses a deformable DETR head to detect traffic elements [24].

TOP_{ll} is the mAP over vertices on the topology among lane segments. TOP_{lt} is the mAP over vertices on the topology between lane segments and traffic elements. The mAP between the GT vertices (V, E) and the predicted vertices $(\hat{V}', \hat{E}')$ is calculated using the following formula:

$$mAP = \frac{1}{|V|}\sum_{v \in V}\frac{\sum_{\hat{n} \in \hat{N}(v)} P(\hat{n})\mathbb{1}(\hat{n} \in N(v))}{|N(v)|}$$

Where $N(v)$ is the list of neighbors of vertex v and $P(v)$ is the precision of vertex v in the list [20].

To improve DET_l, DET_a, and DET_t scores individually, an option is to improve the attention heads mechanisms used to calculate these scores individually. A second option is to improve the LaneSegNet encoder that serves as an input for the Transformer-based detection in the decoder. TOP_{ll} and TOP_{lt} are directly influenced by the performance of the encoder as they use the lane predictions for the topology calculations. Thus, the primary focus of our efforts is to increase the DET_l and DET_a scores through data augmentation and backbone improvements.

[5] https://github.com/OpenDriveLab/OpenLane-V2/blob/master/docs/metrics.md#openlane-topology

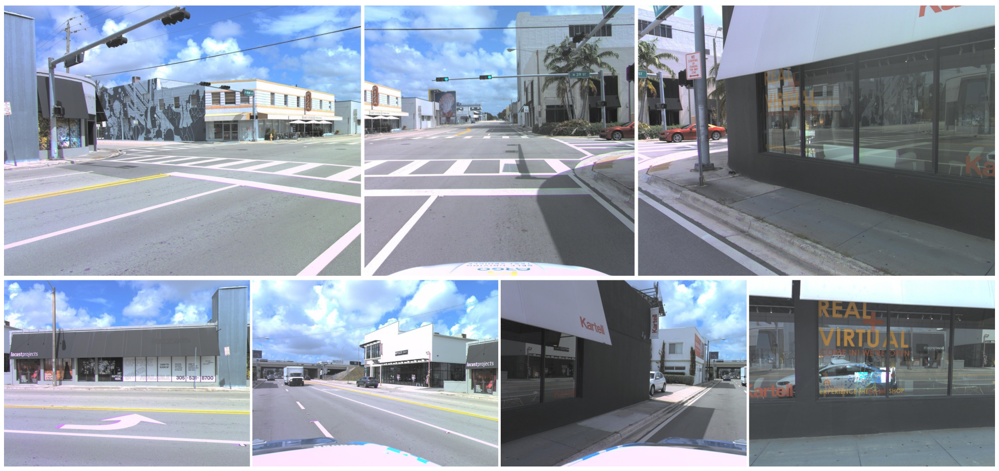
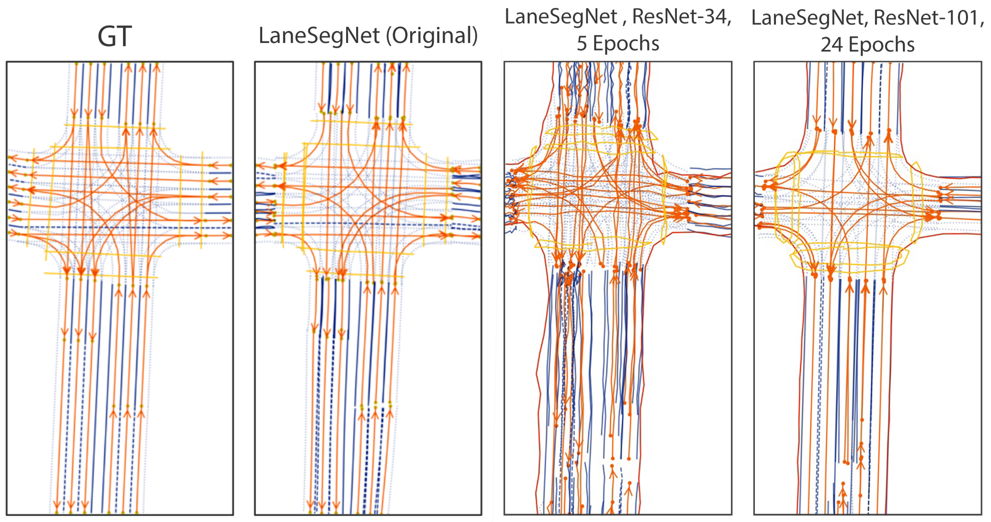


Figure 1. Example of the seven OpenLane-V2 camera angles taken at a crossroad .


GT
LaneSegNet (Original)
LaneSegNet , ResNet-34, 5 Epochs
LaneSegNet, ResNet-101, 24 Epochs
Figure 2. BEV Scene recreation of the input images from Figure 1. The two leftmost images are taken from the original LaneSegNet paper running on the ResNet-50 backbone. The two rightmost images are results from our ResNet-34 test training and ResNet-101 training respectively. Orange lines depict the center lines and lane direction. Blue lines depict lane separators. Yellow lines depict traffic elements such as crossroads.

4. Results & Analysis

Table 1. Test set scores from the original LaneSegNet GitHub page and our training on ResNet-34 and ResNet-101

Method	**OLUS**	DET_l	DET_a	DET_t	TOP_{ll}	TOP_{lt}
LaneSegNet (Original)	0.36	0.278	0.238	0.369	0.241	0.213
LaneSegNet-34, 5 Epochs	0.159	0.61	0.53	0.175	0.56	0.73
LaneSegNet-101, 24 Epochs	0.171	0.165	0.131	0.77	0.14	0.11

As seen in Table 1, our locally trained model expectantly shows a worse performance compared to the original LaneSegNet. Contrary to our expectations, LaneSegNet trained using a ResNet-101 backbone also does not outperform the original LaneSegNet and shows a similar score to our locally trained model. In particular, the DET_a and TOP_{lt} are significantly lower compared to the original LaneSegNet, with TOP_{lt} having an even lower score than our locally trained model. Despite both models we have trained having a similar $OLUS$ score, the reconstructed

Table 2. Test set scores from a selection of CVPR 2024 challengers

Method	**OLUS**	DET_l	DET_a	DET_t	TOP_{ll}	TOP_{lt}
LGmap	0.66	0.51	0.58	0.82	0.46	0.54
MapVision	0.58	0.39	0.40	0.80	0.38	0.48
BoschXCASW	0.51	0.39	0.30	0.76	0.32	0.32

BEV images are considerably more accurate for the LaneSegNet trained on ResNet-101 as seen in Figure 2. Some lanes, however, are missing or incorrect in the BEV from the ResNet-101 training.

One reason for our low overall scores may be related to the interruption of our training. During our 24-epoch training on the ResNet-101 backbone, training was interrupted at nine epochs. We used a checkpoint of epoch 9 to resume the training for the next 15 epochs. A possibility is that the checkpoint has not been loaded correctly with the desynchronized evaluation resulting from the interruption.
The limited performance of ResNet-101 as an object classifier, in comparison with more advanced models, could be another reason for our low score. As can be seen in [18], ResNet-101's performance falls between that of ResNet-50 and ResNet-152, and it performs similarly to, if not worse than, the smallest EfficientNet model in terms of accuracy on Imagenet. Finding a better-performing backbone would help get better results.

4.1. Comparison to other challengers[6]

Table 2 presents the public test server results from various teams that participated in the CVPR 2024 Mapless Driving Challenge.
The LGMap team, which secured first place, implemented a novel online mapping pipeline, LGMap, which excels in long-range temporal modelling. Key features of their approach include the Symmetric View Transformation (SVT) module, which we have also implemented but not had time to run. Their method uses depth perception and semantic driving prior information to address forward sparse feature representation limitations. Secondly, they proposed the Hierarchical Temporal Fusion (HTF) module, which leverages local and global temporal information for stable long-range HD map construction. Finally, their innovative pedestrian crossing resampling technique simplifies representation, enhancing the convergence performance of the instance attention-based decoder. These innovations led LGMap to achieve an $OLUS$ score of 0.66.
One of the backbones we have attempted to run is Swin and Vit-t. BoschXCASW / Supertrainer proves that it is possible to use Vit-t for up-scaling and Swin as a backbone to generate BEV features, which leads to a considerable increase in their scores. Lastly, MapVision improved by utilizing YOLOX as its element detection backbone and incorporating both Standard-Definition (SD) maps and multi-perspective camera images into the input of LaneSegNet. These inputs enhanced scene understanding, particularly in occluded images and the far end of roads. This approach significantly boosted their model's performance. Using the workstation to train this network for 24 epochs took around 104 hours on an RTX 3090 GPU. Locally, to train the model for five epochs, it took around 37 hours on an RTX 3080 GPU. Utilizing the online tool available at https://mlco2.github.io/impact/#compute, we estimated the carbon emissions to amount to approximately 18.42 kg CO2 equivalent through 141 hours of GPU computing, assuming a carbon efficiency of 0.382 kg/kWh, according to https://www.nowtricity.com/country/netherlands/ for May 2024 in the Netherlands. In total, this approximates driving around 74 km with an average internal combustion engine car[7]. The carbon emission figures are estimated because we were unable to obtain precise measurements of emissions for the workstation we utilized from the UvA campus, the amount of power used for both training should be slightly higher due to CPU usage.

[6]As the challenge just ended and submissions of technical reports just finished, we suppose that they are still being reviewed. As a result, we were unable to have access to the actual reports and provide this link to the abstracts of the different submissions mentioned in Table 2.

5. Conclusion

In this paper, we reproduced the LaneSegNet paper for topology prediction without using HD maps. We attempted to improve the model by changing out the ResNet-50 backbone of the BEV encoder module to several different models. We encountered difficulties with several backbones having larger memory requirements, despite having fewer parameters than ResNet-50. This issue is particularly significant for our research area, as the models we develop are intended for deployment in self-driving vehicles, which have limited hardware resources available. The larger version of ResNet that we tried did run successfully but performed worse than the baseline. We hypothesize that this may be caused by issues during the training procedure. It could also have been caused by ResNet-101 overfitting due to its greater expressivity.

6. Limitations & Future research

6.1. Limitations

During our research, we encountered various issues. The first of which was the size of the OpenLane-V2 dataset [20]. With a size of roughly 142.5 GB, downloading and preprocessing the dataset locally, on Snellius, and on the workstation was a timely process. The most glaring issue we encountered during our research was issues we faced with the various environments provided by TopoNet, TopoMLP, and LaneSegNet. Due to outdated packages or version mismatches in the environments provided by TopoNet and TopoMLP, training these models would take too much time,

[7]https://www.epa.gov/energy/greenhouse-gases-equivalencies-calculator-calculations-and-references

thus forcing us to focus our efforts on LaneSegNet. The LaneSegNet environment also contained multiple version mismatches that would produce errors based on the hardware architecture it was run on. On our older Nvidia RTX 2070 Super GPU, the environment could be run with minimal adjustments to the versions of the cuDNN, Shapely, and Yapf packages. However, to run LaneSegNet with a ResNet-50 or larger backbone, we would have to train on a machine that has access to a considerable amount of VRAM, such as Snellius. Tracing the steps we had done locally, the LaneSegNet environment would produce CUDA or MMCV errors on Snellius. To solve these issues, we have attempted to change package versions individually and use an alternative environment provided by the authors through communication with them that did not solve our issues. In none of our attempts have we succeeded in getting LaneSegNet to run on Snellius, which cost us a significant amount of time. Another limitation in our research was the long training times of LaneSegNet. Training LaneSegNet for 24 epochs to match the original paper would take up to five days and had to be done multiple times to verify the LaneSegNet results and improve upon them. The amount of time lost through training and dependency issues limited us in the changes we could make to LaneSegNet in the remaining weeks. For this reason, we decided to stick to implementing higher-level adjustments that could yield a relatively high increase in both the DET_l and DET_a scores.

6.2. Future Research

One promising area for future research is the development of more advanced backbone networks, particularly those tailored specifically for traffic-related tasks. By designing a backbone that can effectively capture the unique features of traffic environments, such as varying lighting conditions and occlusions, the overall performance of lane detection and segmentation models could be improved.

Exploring variations in the LaneSegNetTransformer layers is another valuable research direction. This includes experimenting with different configurations of DETR layers, such as altering the number of attention heads or adjusting the dropout rates. This can help determine the optimal balance between model complexity and performance, leading to more efficient and effective lane detection systems.

Additionally, we attempted to perform data augmentation on OpenLane-V2. Incorporating a wider range of data augmentation techniques is critical for enhancing model generalization and robustness. We made a horizontally flipped version of images and corresponding annotated coordinates. This was however not tested due to a lack of resources and time. We also anticipated that this data augmentation would not be beneficial to the model as it was suggested that this would introduce noise in the dataset and confuse the model. Future research should investigate augmentation strategies that can simulate various traffic scenarios and conditions more effectively. Such as geometric transformations and colour/brightness augmentations to enrich the training dataset. This will help create models that can better adapt to real-world traffic environments.

Additionally, we could employ upsampling. Upsampling plays a crucial role in restoring high-resolution details in segmentation tasks. Future work could focus on evaluating various upsampling techniques, such as transposed convolutions or advanced interpolation methods. This may result in increased spatial accuracy and detail preservation in lane segmentation outputs.

At last, the most recent model, Topo2D [7], claims to have achieved an OLS of 44.5% on the OpenLane-V2 test set. Topo2D's novel approach centres on leveraging 2D lane instances to initialize 3D queries and 3D positional embeddings using a Transformer-based framework. By explicitly incorporating 2D lane features into the detection of topology relationships among lane centerlines and between lane centerlines and traffic elements, Topo2D outperformed existing methods. It achieved 44.5% OLS on the OpenLane-V2 benchmark for multi-view topology reasoning and a 62.6% F-score on the OpenLane benchmark for single-view 3D lane detection. Thus, incorporating 2D lane priors into 3D lane segment detection has great potential to increase performance.

References

[1] Yigit Baran Can, Alexander Liniger, Danda Pani Paudel, and Luc Van Gool. Structured bird's-eye-view traffic scene understanding from onboard images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15661–15670, 2021. 1

[2] Yigit Baran Can, Alexander Liniger, Danda Pani Paudel, and Luc Van Gool. Topology preserving local road network estimation from single onboard camera image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17263–17272, 2022. 1

[3] Wenjie Ding, Limeng Qiao, Xi Qiu, and Chi Zhang. Pivotnet: Vectorized pivot learning for end-to-end hd map construction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3672–3682, 2023. 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[5] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019. 4

[6] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. 4

[7] Han Li, Zehao Huang, Zitian Wang, Wenge Rong, Naiyan Wang, and Si Liu. Enhancing 3d lane detection and topology reasoning with 2d lane priors. *arXiv preprint arXiv:2406.03105*, 2024. 7

[8] Tianyu Li, Li Chen, Xiangwei Geng, Huijie Wang, Yang Li, Zhenbo Liu, Shengyin Jiang, Yuting Wang, Hang Xu, Chunjing Xu, et al. Topology reasoning for driving scenes. *arXiv preprint arXiv:2304.05277*, 2023. 2

[9] Tianyu Li, Li Chen, Huijie Wang, Yang Li, Jiazhi Yang, Xiangwei Geng, Shengyin Jiang, Yuting Wang, Hang Xu, Chunjing Xu, Junchi Yan, Ping Luo, and Hongyang Li. Graph-based topology reasoning for driving scenes, 2023. 2

[10] Tianyu Li, Peijin Jia, Bangjun Wang, Li Chen, Kun Jiang, Junchi Yan, and Hongyang Li. Lanesegnet: Map learning with lane segment perception for autonomous driving. *arXiv preprint arXiv:2312.16108*, 2023. 2, 3

[11] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European conference on computer vision*, pages 1–18. Springer, 2022. 3

[12] Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. Maptr: Structured modeling and learning for online vectorized hd map construction. *arXiv preprint arXiv:2208.14437*, 2022. 2, 4

[13] Bencheng Liao, Shaoyu Chen, Bo Jiang, Tianheng Cheng, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Lane graph as path: Continuity-preserving path-wise modeling for online lane graph construction. *arXiv preprint arXiv:2303.08815*, 2023. 2

[14] Yicheng Liu, Tianyuan Yuan, Yue Wang, Yilun Wang, and Hang Zhao. Vectormapnet: End-to-end vectorized hd map learning. In *International Conference on Machine Learning*, pages 22352–22369. PMLR, 2023. 2

[15] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 4

[16] Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. *CoRR*, abs/2003.13678, 2020. 4

[17] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. 4

[18] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. 4, 6

[19] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *European conference on computer vision*, pages 459–479. Springer, 2022. 4

[20] Huijie Wang, Tianyu Li, Yang Li, Li Chen, Chonghao Sima, Zhenbo Liu, Bangjun Wang, Peijin Jia, Yuting Wang, Shengyin Jiang, et al. Openlane-v2: A topology reasoning benchmark for unified 3d hd mapping. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 3, 4, 6

[21] Dongming Wu, Jiahao Chang, Fan Jia, Yingfei Liu, Tiancai Wang, and Jianbing Shen. Topomlp: A simple yet strong pipeline for driving topology reasoning, 2023. 1, 2

[22] Zhenhua Xu, Yuxuan Liu, Yuxiang Sun, Ming Liu, and Lujia Wang. Centerlinedet: Road lane centerline graph detection with vehicle-mounted sensors by transformer for high-definition map creation. *arXiv preprint arXiv:2209.07734*, 1 (2):7, 2022. 1

[23] Tianyuan Yuan, Yicheng Liu, Yue Wang, Yilun Wang, and Hang Zhao. Streammapnet: Streaming mapping network for vectorized online hd map construction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 7356–7365, 2024. 2

[24] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 4

[25] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. 4