Dynamic Weather-Aware Lane Detection

Justin Boverhof
Robotics
University of Michigan
Ann Arbor, United States
boverhof@umich.edu

Joseph Fedoronko
Robotics
University of Michigan
Ann Arbor, United States
fedoronj@umich.edu

Anay Moitra
Computer Science
University of Michigan
Ann Arbor, United States
amoitra@umich.edu

Andrew Rodriguez
Robotics
University of Michigan
Ann Arbor, United States
arodjr@umich.edu

***Abstract*—Lane detection is a critical task for autonomous driving, enabling safe navigation by identifying road boundaries and lane structures. In this project, we enhance the Ultra Fast Lane Detection (UFLD) framework to improve its robustness under challenging conditions, such as adverse weather and low visibility. Our contributions include the integration of a Feature Pyramid Network (FPN) for multi-scale feature extraction, a dynamic weather prompting mechanism to adapt the model based on current environmental conditions, and an unsupervised weather discovery system to eliminate the need for manual weather labeling. Together, these improvements enable real-time, weather-adaptive lane detection, demonstrating strong potential for practical deployment in autonomous systems operating in diverse environments.**

I. Introduction

Lane detection is a fundamental perception task in the domain of autonomous driving and mobile robotics, providing essential cues for localization, trajectory planning, and collision avoidance. Traditional methods based on hand-crafted features struggle with variations in lighting, road surface conditions, and adverse weather. Recently, deep learning-based approaches have significantly improved lane detection accuracy by leveraging convolutional neural networks (CNNs) and, more recently, attention-based models.

Despite these advances, many existing systems are limited in their ability to adapt to challenging weather conditions, such as rain, snow, and fog, which can obscure lane markings and degrade image quality. Still, real-time performance remains a critical requirement for deployment on resource-constrained robotic platforms.

In this project, we aim to address these challenges by building upon the Ultra Fast Lane Detection (UFLD) framework, known for its lightweight and efficient design. We propose three major enhancements: (1) the addition of a Feature Pyramid Network (FPN) to improve feature extraction across scales, (2) a dynamic prompting system that conditions the model on detected weather conditions during inference, and (3) an unsupervised clustering approach to automatically discover and label weather patterns in the dataset without manual annotation. These improvements collectively enable robust, real-time lane detection adaptable to diverse and dynamic environments.

II. Previous Work

Lane detection is a fundamental component in autonomous driving and robotics, providing crucial road structure information for safe navigation. Over recent years, deep learning has emerged as a powerful tool in this domain, with numerous methods developed to increase both accuracy and efficiency. One of the seminal works in this area is the Spatial CNN (SCNN), which introduced the concept of spatial message passing within convolutional layers to capture long-range dependencies across image rows and columns. This significantly improved lane detection performance in challenging scenes with occlusions or poor visibility conditions [4].

Building upon this, Lane2Seq presents a more structured and sequential approach to lane detection. Instead of treating lanes as dense segmentation masks or heatmaps, Lane2Seq formulates lane detection as a sequence prediction problem, similar to natural language processing tasks. By encoding lane points and predicting them in an ordered manner using transformer-based attention and a sequential decoding mechanism, the model achieves superior performance in representing curved and long-range lane geometries [3]. This approach not only improves accuracy but also aligns well with real-world lane topology, making it more interpretable and suited to planning modules in autonomous vehicles. The TuSimple dataset—used as one of its evaluation benchmarks—remains a cornerstone in assessing lane detection methods, although it focuses primarily on structured highway environments.

As the field evolved, researchers also explored more geometrically informed representations. Though not designed specifically for lane detection, PointNet demonstrated the power of directly processing 3D point clouds with neural networks, inspiring efforts to incorporate depth or LiDAR data into lane detection pipelines for enhanced robustness in 3D space [2].

In the pursuit of real-time performance, the Ultra Fast Lane Detection (UFLD) framework gained attention for its lightweight, anchor-based approach capable of running at high frame rates on edge devices [5]. Its balance between speed and accuracy makes it a compelling foundation for robotic applications. However, UFLD and similar methods often struggle under adverse weather conditions, where visibility is low and lane markings may be partially occluded. These limitations highlight the need for architectures that can better capture multi-scale features and dynamically adapt to environmental variability—an area that remains underexplored in current literature.

We also investigated transformer-based architectures such as O2SFormer, which offers a one-stage, one-shot solution through a powerful attention-based backbone [6]. While conceptually promising for capturing global context, practical limitations arose due to the reliance on the MMDetection framework and mmlab libraries, which presented significant installation and debugging challenges. These constraints illustrate the ongoing trade-offs between performance and usability in state-of-the-art lane detection models.

Through this literature review, we identified a clear gap in the robustness of lane detection systems under non-ideal conditions. This motivates the exploration of solutions that balance real-time capability with adaptability to challenging visual environments—goals we aim to address in our final implementation.

III. Methodology

We originally proposed implementing these 4 deep learning elements to improve the results of the paper. We initially considered adding local attention to the Lane2Seq model to enhance clarity in curved or occluded lanes. However, after switching to the Ultra Fast Lane Detection (UFLD) model for its real-time speed, we found its CNN-based design doesn't support attention without major architectural changes. The rest of the steps are explained in more the following sections.

A. Feature Pyramid Extraction

To address our main concerns about current lane detection solutions, our group aimed to improve the extraction of features to better identify small components of a lane line. To improve this, we integrated a Feature Pyramid Network (FPN) directly after the backbone. The FPN takes intermediate feature maps from multiple layers of the backbone and merges them using top-down and lateral connections (see Figure below). This allows the model to capture both fine-grained and high-level lane features across different spatial resolutions. Additionally, this FPN addition allows the network to keep small components of the original image that could be valuable, specifically in the context of adverse weather conditions with limited visibility.

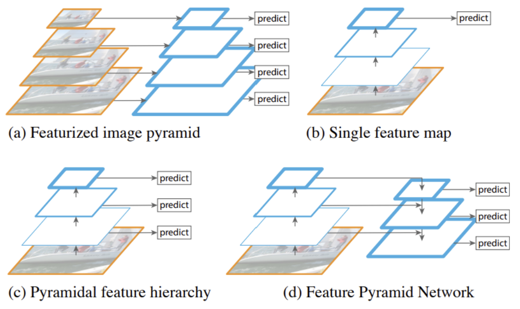

Fig. 1. FPN diagram. Adapted from [7]

Our custom FPN module processes these backbone outputs, performs up-sampling and fusion, and produces enhanced feature maps that are then passed to the lane prediction head. This architectural change improves lane detection performance, especially under occlusion and scale variation, while maintaining real-time inference.

With the addition of the FPN, the feature maps maintain higher spatial resolution compared to the original backbone output. This preservation of size leads to clearer, more detailed representations of lane structures in the feature space. As a result, the model can better localize and distinguish lane lines, improving detection accuracy especially in complex scenes. This is a result of the upsampling process which helps recover spatial details lost during downsampling in the backbone. Overall, this results in sharper and more informative feature maps for lane detection.

B. Dynamic Prompting for Weather Modes

To further better robustness under varying environmental conditions, we implemented a dynamic prompting mechanism that conditions the model on the current weather state. Lane detection models often struggle under adverse conditions due to reduced visibility, lighting variations, and road surface changes. Our approach enables the network to adapt its internal processing dynamically based on detected weather.

Specifically, we designed a weather condition module that encodes the current weather as a learnable embedding. Each weather type (clear, rain, snow, fog) is represented initially as a one-hot vector, which is projected into a dense embedding space through a small multi-layer perceptron (MLP). This weather embedding is then concatenated with the feature map output from the backbone network.

To combine these modalities in an efficient way, we introduced an additional fusion MLP that processes the combined feature representation. The parsingNet lane detection model was extended to accept an additional weather_condition input, ensuring that all downstream layers receive weather-aware features. The data loading pipeline was similarly updated to provide the correct weather prompt alongside each image during both training and inference.

This dynamic prompting strategy allows the model to learn weather-specific adaptations in its feature representations, improving detection accuracy across diverse conditions without requiring separate models for each weather type.

C. Unsupervised Learning for Weather Condition Discovery

While dynamic prompting requires weather labels, manual annotation for large datasets is impractical. To address this, we introduced an unsupervised, data-driven method for discovering and labeling weather conditions automatically.

We employ a pretrained ResNet-18 network as a feature extractor, removing the final classification layer to capture mid-level visual features from each image. The extracted features are then standardized and clustered using KMeans, with four clusters corresponding to hypothesized weather modes (clear, rain, snow, fog).

Each training image is assigned a cluster label based on its extracted features, serving as a proxy weather label. These cluster-derived labels are integrated into the training pipeline, providing weather prompts without requiring any manual supervision. To facilitate evaluation, we developed a visualization tool that samples and displays representative images from each cluster, enabling qualitative assessment of the discovered weather categories.

This unsupervised approach scales easily to large datasets, enables domain-adaptive lane detection, and can uncover latent environmental structure that impacts model performance.

D. Colab Demonstration of Dynamic Prompting and Unsupervised Learning

Due to significant environment and hardware challenges (detailed in Section V), we were unable to fully train and deploy our modified Ultra Fast Lane Detection (UFLD) model at scale. To validate the feasibility of our dynamic prompting and unsupervised weather clustering approach (mentioned in the subsections before), we developed a lightweight proof of concept demonstration using Google Colab.

First, we accessed a compressed version of the CULane dataset stored in the cloud. To avoid slow data extraction and reduce computational load, we limited extraction to 1000 images by selectively unpacking only the first encountered JPEG files from the .tar.gz archive. This sampling strategy gave us faster experimentation while preserving sufficient visual diversity.

Each extracted image was resized to 224x224 pixels and augmented with slight color jittering and random rotations to simulate minor environmental variability. We utilized a pre-trained ResNet-18 model as a fixed feature extractor, removing its final classification layer to obtain 512-dimensional feature vectors for each image. These feature vectors captured mid-level visual representations crucial for distinguishing environmental conditions. Next, an unsupervised KMeans clustering algorithm was applied to the extracted features, partitioning the images into four clusters. These clusters acted like proxy labels for weather conditions, without requiring manual annotations. After clustering, we distributed the dataset into training and testing splits to make sure of balanced representation of all discovered weather types.

To implement dynamic prompting, we constructed a small neural network that receives both image features and a learned embedding of the corresponding weather cluster. The weather embeddings were combined with the image features and passed through two fully connected layers to predict the correct weather cluster label. The model was trained for five epochs using cross-entropy loss, and we evaluated it after each epoch on the test set, reporting precision, recall, and F1 score. Across training, we saw rapid convergence to high performance, achieving a best F1 score of 0.9890 on the test set. This strong performance was created by the simplicity of the extracted features, the relatively small number of classes, and the pretraining of the feature extractor.

IV. Results

To validate the results of our methodology, we decided to train both two models, one without the addition of a simple weather-aware classifier and one with it. Both were trained for 2 epochs. We then utilized the built in evaluation command that came with the UFLD paper to see the precision, recall, and the F1 scores. For our tests, we were able to evaluate the model without any additions in all the different environment but did a basic score for the model with additions.

Table 1. Precision, Recall, and F1 score of Ultra Fast Lane Detection Trained Model After Two Epochs of Training.

Test Conditions	Precision	Recall	F1
Normal	0.846666	0.853769	0.850203
Crowd	0.656231	0.663965	0.660075
Highlight	0.538915	0.542433	0.540668
Shadow	0.634523	0.642907	0.638687
No-Line	0.409163	0.384709	0.396559
Arrow	0.791946	0.778755	0.785296
Curve	0.613008	0.574695	0.593234
Cross	0	-1	0
Night	0.591422	0.608464	0.599822

To better validate our dynamic prompting and unsupervised clustering pipeline, we conducted an additional lightweight experiment using a subset of 1000 extracted images from the CULane dataset, as described in Section D. The extracted features were clustered into four groups, and a simple weather-aware classifier was trained using the discovered cluster labels. This demonstration achieved very high classification performance and this confirms that our techniques can effectively model weather-driven variations even with limited supervision. The results are summarized below.

Table 2. Precision, Recall, and F1 Score for the Colab Dynamic Prompting Demonstration.

Experiment	Precision	Recall	F1
Colab Dynamic Prompting	0.9876	0.9853	0.9890

However, it is important to note that these results are somewhat inflated due to the simplicity of the task setup. The clustering and training are performed on a relatively clean and limited dataset with minimal environmental variability. As a result, the model benefits from reduced noise and more separable clusters, which would not necessarily generalize to real-world driving conditions. In deployment scenarios featuring more diverse and challenging weather patterns, the performance would likely decrease. Nonetheless, since we see higher scores for the model with these features, this serves as a proof of concept demonstrating the feasibility of dynamic prompting combined with unsupervised weather discovery.

V. Challenges

Throughout this project, our group faced many setbacks that forced us to reevaluate our approach. After our initial literature review, we wanted to recreate the "Lane2seq" paper and add additions to this model. However, this paper did not have open source code despite using a publicly available dataset. This lead us to pivoting to our backup option-O2SFormer.

But again we experienced issues related to dependencies with the MMDetection library that prevented us from running the provided code. After considering our options, we conducted further research to find a paper that would not have any of the issues that we previously experienced. This lead us to a paper called ”Ultra-Fast-Lane-Detection-V2”. However, after working with the code, there were many issues with getting the Nvidia drivers to work. When working with a Windows OS, Nvidia-Dali (a required library) was found to be incompatible with both a Windows environment and a WSL (Windows Subsystem for Linux). After our initial attempt to work on Windows failed we proceeded to try and run it on Linux instead. However, with the laptop we were using to run our code and solving our environment in, it was found that the Linux OS was unable to access the dedicated GPU likely due to Windows preventing Linux from accessing the dedicated GPU. After this our team met with our one of our classes' instructors and it was recommend to utilize a virtual environment utilizing Pop!_OS for the Linux OS instead of Mint which was previously being used for the Linux OS. This still didn't fully solve our problem since Windows OS was still preventing the virtual environment from accessing the GPU. The final solution found to solve the environment was to go completely reinstall WSL and disable NVML for the nvidia-dali library which, will it worked, significantly reduced the processing speed and power of the GPU leading to a single epoch of training taking about 10 hours and a evaluation taking about 30 minutes. Attempts were made to run the code on the great lakes cluster but issues arose with storage constraints of the cluster and the size of the dataset and we did not have the time to problem solve. Overall, most of the challenges with our project were in trying to solve the environment and getting the code to run.

Specifically pertaining to our methodology, we also faced challenges with dynamic prompting, specifically to make sure that the weather embeddings generalize well across diverse and noisy real-world conditions, rather than overfitting to clustered patterns in training data. Since we also used unsupervised learning, we were without reliable ground truth labels, so the model risked learning biased or incorrect associations between features and weather conditions.

To compensate for this, we developed a separate lightweight proof of concept in Google Colab, to demonstrate the core ideas of dynamic prompting and unsupervised weather clustering. While this allowed us to validate the feasibility of our proposed methods, it does not fully substitute for large-scale evaluation on the lane detection task.

VI. Conclusion

While utilization of dynamic prompting and unsupervised learning seems like a feasible methodology of increasing the precision, recall, and F1 score of a lane detection software, as of now there needs to be further testing to validate this methodology. Due to all the setbacks and challenges with working with the environment and getting the code to run we were unable to achieve the data we would want to fully validate this methodology. Even though we weren't able to find get all the data we wanted, we hope that in the future that someone else is able to pick up where we left off and give the model the proper time it needs to train and to validate the effectiveness of the solution.

References

[1] Next Move Strategy Consulting. Number of autonomous vehicles globally in 2022. 2023.

[2] C. R. Qi et al. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR*, 2017.

[3] J. Liu et al. Lane2seq: A sequential lane representation for end-to-end lane detection. *arXiv:2402.17172*, 2024.

[4] X. Pan et al. Spatial as deep: Spatial cnn for traffic lane detection. *arXiv:1612.00593*, 2018.

[5] Z. Qin et al. Ultra fast structure-aware deep lane detection. *ECCV*, 2020.

[6] Z. Zhang et al. O2sformer: One-stage one-shot lane detection via transformer. *arXiv:2303.11736*, 2023.

[7] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*, 2016.

Appendix

A. *Appendix A*

Here contains the modified code used for testing of this project. If you are unable to access this url for any reason, please contact arodjr@umich.edu so that access can be granted. Code base: https://github.com/arodjr604/ROB_498_Final_Project_Ultra

B. *Appendix B*

Here contains the pre-trained models used for the results portion of this project as well as AVI files to demonstrate how effective the model was: https://drive.google.com/drive/folders/1l-At4vEve8xUQUpaFw1MXGmSc6n7TRFk?usp=sharing. If you would like to run the evaluation for yourself on the UFLD without Additions, first download the model. After, navigate to your UFLD repository (fetch the code above if you don't have this repository already) and run the following commands:

- mkdir tmp
- python test.py configs/culane_res18.py –test_model /path/to/your/model.pth –test_work_dir ./tmp

If you would like to see the data for the UFLD with Additions, it's also located in the folder but instead of downloading the repository, there is a google colab folder inside that shows the data along with the pth file of the trained model.

C. *Appendix C*

It was noted in the Challenges section that there were many issues with getting the environment to work. So during the process a detailed list of commands and libraries were generated to track how to get the environment to work. If you are planning on trying to run our code and use/execute it, this link will provide a google sheet with details about the process of doing this: https://docs.google.com/document/d/1gr-oTzmCAW8lfZLU2bzW_uTJ5m-S-4A-96ePHXtP5xQ/edit?tab=t.0. It's important to note that most of this had been added to the INSTALL.md file of the project. This here is to emphasis the complexity and time that went into fixing and resolving the environment issues present.

D. *Statement of Contributions*

Justin Boverhof: I Justin Boverhof helped with the overall formatting of the poster and more specifically the motivation and citation sections. I also led the charge on trying to train our models on the Great Lakes cluster but was unsuccessful at first for the same reasons we were unsuccessful on local machines. And then later due to issues with great lakes and time constraints forcing me to stop. In general I did my best to assist the others when possible in their efforts. I also handled the construction and submission of the project website.

Joseph Fedoronko: I Joseph Fedoronko worked on the initial research to find multiple papers that we could implement and identify gaps to be addressed using concepts from class. I also was responsible for implementing the FPN level and writing code to add this to our original network. I attempted to run the code locally on a NVIDIA gpu computer but ran into issues due to the nature of the mmdetection library. I then worked on writing a large part of the poster and dealt with logistics for printing the poster as well as making sure we were prepared for the expo. For the final report, I created the template, worked on the literature review, and help transfer work from the poster over to our final report.

Anay Moitra: I Anay Moitra worked on complete implementation of dynamic prompting and unsupervised learning onto the Ultra Fast Lane Detection framework. I also helped by writing the dynamic prompting and unsupervised learning sections on the poster and the report. Since, our group was having trouble executing the code at scale, I decided to create a smaller version of the dynamic prompting and unsupervised learning frameworks on Colab to make sure that we have a proof of concept of our extensions working.

Andrew Rodriguez: I Andrew Rodriguez worked on debugging and getting the environment to run and function. Additionally I was responsible for the training and evaluation of the Ultra Fast Lane Detection w/o Annotations model found in Appendix B. This also included the creation of the demo AVI files found inside of Appendix B as well. In addition I was responsible for the creation of the ROB498_Environment_Debugging_Guide found in Appendix C and the implementation of these instructions into the INSTALL.md. Because of my involvement in the training and evaluation of the model in Appendix B, I was also responsible for getting and placing the data of the model without additions in the results section. Finally, I was also responsible for writing a majority of each challenges section for both the final report and the poster and the conclusion.