Lightweight convolutional neural networks for real-time lane segmentation

Zhengxiao Yu

College of Letters and Science, University of California, Davis, CA, 95616, USA

yzxyu@ucdavis.edu

Abstract. Autonomous driving depends on reliable perception systems that involve various perception modules and advanced computer vision techniques. One crucial component of these systems is lane detection, which traditional methods often rely on basic features like color or edges that are sensitive to lighting and perspective changes. Recently, convolutional neural networks (CNNs), have revolutionized lane detection. Nevertheless, existing methods still have some limitations, such as the need for pixel-level labeling and computational inefficiency for real-time applications. To address these challenges, this work leverages PP-LiteSeg for real-time semantic segmentation. PP-LiteSeg's key elements are its Simple Pyramid Pooling Module (SPPM), Unified Attention Fusion Module (UAFM), and Flexible and Lightweight Decoder (FLD), which optimize lane detection efficiency. The FLD flexibly adjusts computational costs between the encoder and decoder, balancing efficiency and accuracy. The UAFM enhances feature representations using attention mechanisms, increasing segmentation accuracy. The SPPM efficiently aggregates contextual information while reducing computational complexity. The comprehensive method for lane segmentation achieves competitive results on popular lane detection datasets. The proposed model can adapt to different computational capabilities and significantly enhances lane detection efficiency for real-time applications.

Keywords: Convolutional neural network, Real-time segmentation, Lane segmentation.

1. Introduction

Fully achieving autonomous driving requires a comprehensive understanding of the surrounding environment, involving the integration of various perception modules and the application of numerous pattern recognition and computer vision techniques, with lane detection being a critical component [1].

Conventional traffic line detection methods typically rely on extracting elementary features such as color, or edges. These features are combined using techniques like the Hough transform, or Kalman filter to provide information about traffic line segments [2]. While these methods are adaptable to different environments with minimal adjustments, their performance is sensitive to factors like lighting, occlusion, and the image perspective, including high-angle aerial views or an in-car driver's perspective [3].

In recent times, image segmentation has been revolutionized Convolutional Neural Networks (CNNs). Semantic segmentation methods have shown outstanding performance in traffic line detection, allowing for inferences about shapes and locations [4,5]. However, existing methods have limitations. Semantic segmentation methods require pixel-level labeling or preprocessing, which can be cumbersome. They often predict many unnecessary points because the output size matches the input size, even when only a few points are needed for traffic line recognition. Moreover, these methods are not adaptable to varying computational capabilities, requiring significant architectural modifications for lightweight systems like embedded boards [6].

In order to overcome these constraints and improve the effectiveness of lane recognition, this study presents an innovative approach utilizing PaddleSeg's PP-LiteSeg for real-time semantic segmentation [7]. PaddleSeg is a comprehensive and efficient toolkit designed for picture segmentation tasks. It encompasses the whole development process, encompassing model construction, training, performance optimization, speed enhancement, and deployment [8]. The selected model, PP-LiteSeg, has an encoder-decoder architecture. It is made up of three components, the Simple Pyramid Pooling Module (SPPM), Unified Attention Fusion Module (UAFM), and Flexible and Lightweight Decoder (FLD) [9]. The encoder-decoder architecture is typical. An encoder usually extracts multi-level features and the decoder maps them to the lable space. Recent decoder designs maintain a consistent number of channels while increasing spatial size [10].

FLD, which progressively reduces channels while expanding spatial dimensions, with adaptable volume to align with the encoder. This flexible design balances computational complexity, enhancing overall model efficiency. Improving feature representations is pivotal for segmentation accuracy. The solution introduces the UAFM to efficiently strengthen feature representations. UAFM employs attention modules to generate weights, which are then used to fuse input features. UAFM includes spatial and channel attention modules to capture relationships within the features effectively. Contextual aggregation is another critical aspect of enhancing segmentation accuracy, but previous methods are computationally intensive for real-time applications. Leveraging the PPM framework, this work presents the SPPM which automatically learns multi-level embeddings from feature maps. Experimental results demonstrate that SPPM significantly contributes to segmentation accuracy with minimal additional computational cost.

2. Method

This section introduces three crucial components that collectively constitute the innovative approach of PP-LiteSeg for optimizing the encoder-decoder architecture in semantic segmentation. These components aim to resolve the issue of computational inefficiency in lightweight decoder models, which maintain a consistent number of feature channels across all levels, resulting in increased computational costs in shallower stages. These three key components are: FLD, UAFM, and SPPM. Subsequently, we provide an overview of the real-time semantic segmentation architecture known as PP-LiteSeg.

2.1. Flexible and Lightweight Decoder (FLD)

Benefits from the advancement of encoder-decoder architecture, the accuracy of image segmentation has been boosted. The encoder extracts hierarchical features by using a series of layers organized into stages. During the progression, when characteristics shift from lower to higher levels, there is an augmentation in the quantity of channels while the spatial dimensions diminish. Efficient encoding is achieved by properly distributing the computational load across multiple phases. On the contrary, contemporary decoder models that are designed to be lightweight exhibit a uniform distribution of feature channels across all levels. This design choice, however, results in noticeably increased computational expenses in shallower stages as compared to deeper stages, hence introducing redundancy.

For the purpose of optimizing the efficiency of the decoder, FLD module is leveraged. It gradually decreases the quantity of channels in features as they progress from higher to lower levels. This enables the adjustment of computational costs in a dynamic manner, promoting a more balanced collaboration among the encoder and decoder components.

2.2. Unified Attention Fusion Module (UAFM)

Furthermore, alongside the utilization of element-wise summation and concatenation techniques, scholars have introduced UAFM. It could complementarily leverage channel and spatial representations, for boosting performance, leveraging attention mechanism. In general, the UAFM model initially employs two attention modules to generate the weight α. Subsequently, the input features are combined by the multiplication (Mul) and addition (Add) operations. The following sections will introduce the other two modules. The utilization of an attention module by UAFM enables the generation of a weight α, which is subsequently used to combine the input features through the operations of Mul and Add. The input features are represented as F_{high} and F_{low}, with a comprehensive level of description. The output of the deeper module is referred to as F_{high}, whereas the equivalent from the encoder is denoted as Flow. It should be noted that these entities possess identical channels. The UAFM algorithm initially employs the bilinear interpolation operation to increase the resolution of F_{high}, resulting in an upsampled feature marked as Fup, which matches the size of F_{low}. Next, the attention module receives F_{up} and F_{low} as its input and generates the weight α. The attention module has the potential to function as a plugin, taking the form of other modules such as spatial attention or channel attention modules. Subsequently, attention-weighted features are obtained by applying the element-wise multiplication operation to F_{up} and F_{low}, respectively. Ultimately, the Unified Attention Fusion Module (UAFM) conducts element-wise summation of the attention-weighted features, resulting in the production of the fused feature. The aforementioned technique can be expressed as

$$F_{up} = Upsample(F_{high}) \tag{1}$$

$$\alpha = Attention(F_{up}, F_{low}) \tag{2}$$

$$F_{out} = F_{up} \cdot \alpha + F_{low} \cdot (1 - \alpha) \tag{3}$$

weight α is generated by UAFM, leveraging either two attention modules, followed by fusing the input features using multiplication (Mul) and addition (Add) operations.

2.3. Spatial Attention Module

The fundamental reason for incorporating a spatial attention module is to leverage spatial associations in order to compute a weight that signifies the importance of individual pixels within the input data. Based on the provided input features, represented as $F_{up} \in R^{C\times H\times W}$ and $F_{low} \in R^{C\times H\times W}$, this methodology first performs operations on channel dimension, including mean and max. As a consequence, four features are produced, each having dimensions $R^{1\times H\times W}$. Following this, the aforementioned four features are merged into a unified feature, denoted as $F_{cat} \in R^{4\times H\times W}$. The concatenated feature is obtained by using convolution and sigmoid processes, resulting in $\alpha \in R^{1\times H\times W}$, as described in Equations 4 and 5. Significantly, the spatial attention module provides a level of adaptability, including the ability to eliminate the max operation in order to enhance computing efficiency.

$$F_{cat} = Concat(Mean(F_{up}), Max(F_{up}), Mean(F_{low}), Max(F_{low})) \tag{4}$$

$$\alpha = Sigmoid(Conv(F_{cat})) \tag{5}$$

2.4. Channel Attention Module

The primary principle underlying this module is utilizing the inter-channel interaction to produce a weight that signifies the significance of each channel within the input features. The channel attention module presented in this study compresses feature maps leveraging average- and max-pooling. This process produces four features of $R^{C\times 1\times 1}$. Subsequently, the system combines these four characteristics by concatenating them along the channel axis. It then achieved a weight $\alpha \in R^{C\times 1\times 1}$, leveraging convolution and sigmoid. Finally, the processes can be expressed mathematically as equations 6 and 7.

$$F_{cat} = Concat(AvgPool(F_{up}), MaxPool(F_{up}), AvgPool(F_{low}), MaxPool(F_{low})) \tag{6}$$

$$\alpha = Sigmoid(Conv(F_{cat})) \tag{7}$$

2.5. Simple Pyramid Pooling Module (SPPM)

As depicted in Figure 1, the utilization of SPPM is employed. The initial step is utilizing this module to integrate the input functionality. It consists of three convolution processes, with bin sizes of 1 × 1, 2 × 2, and 4 × 4, respectively. Subsequently, the convolution and upsampling processes are applied to the resulting features. In the context of the convolution process, it is seen that the kernel size is 1×1, and furthermore, the output channel is comparatively smaller in magnitude than the input one. In conclusion, this study incorporates the upsampled features and applies a convolution operation to generate the refined feature. In contrast to the original Patch Permutation Model (PPM), the Simplified Patch Permutation Model (SPPM) decreases intermediate and output channel amounts, eliminates the use of a simpler connection, and substitutes the concatenate operation with an addition operation. As a result, it can be argued that this design is appropriate for real-time models.

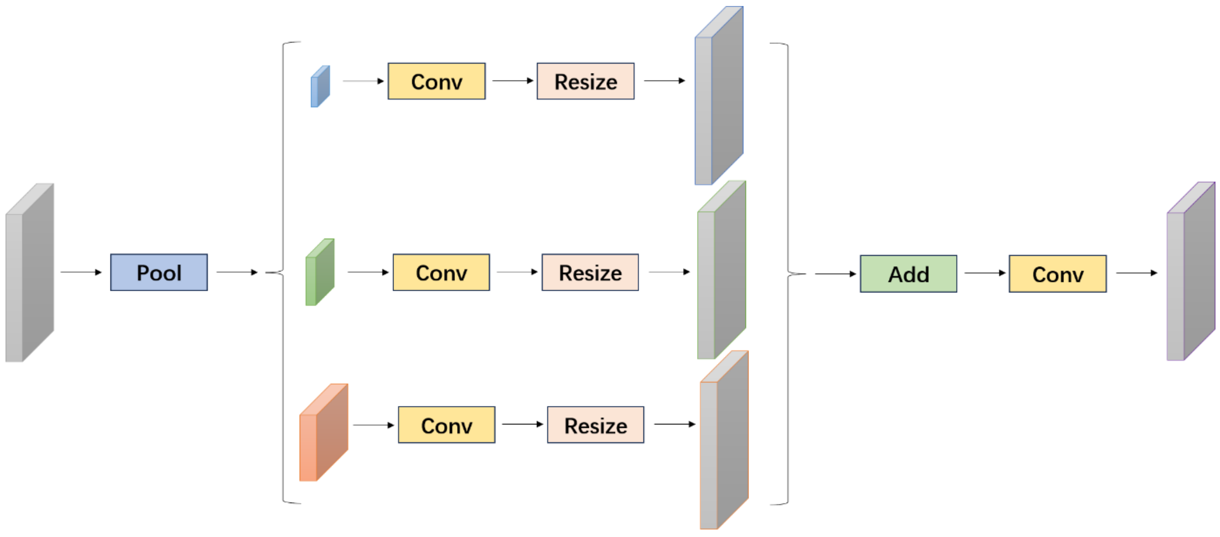

Figure 1. Architecture of SPPM module [9].

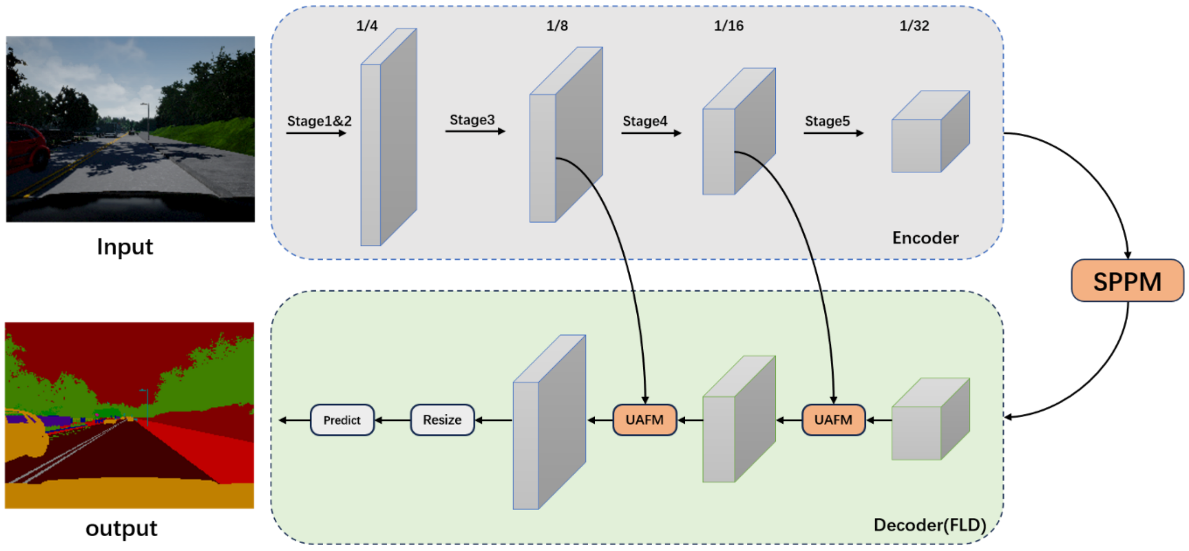

Figure 2. Architecture of PP-LiteSeg [7].

2.6. PP-LiteSeg

Figure 2 illustrates the architectural design of the envisaged PP-LiteSeg. The system is primarily composed of three distinct modules, namely the encoder, aggregation, and decoder. To begin with, when presented with an input image, PP-Lite employs a lightweight network as an encoder to extract hierarchical features. The encoder consists of five stages, with a stride of 2 for each stage. As a result, the final feature size is reduced to 1/32 of the input image. Additionally, PP-LiteSeg utilizes the Sequential Point Process Model (SPPM) to effectively capture and represent the long-range dependencies. The Spatial Pyramid Pooling Module (SPPM) incorporates global information by employing the output feature of the encoder. The suggested PP-LiteSeg algorithm effectively employs the FLD technique to progressively integrate multi-level information and generate the final image output. The FLD model is composed of two UAFM units and a segmentation head. The high-level characteristic of the envisaged PP-LiteSeg is generated either by the SPPM or the deeper fusion module. The Unsupervised Attention-based Feature Fusion Model (UAFM) generates fused features by employing a down-sample ratio at 1/8. The head reduces the channel count in the 1/8 down-sampled feature in order to align it with the total amount of classes. Following the process of upsampling the feature size to align with the input image size, the label for each individual pixel is determined through the utilization of an argmax.

3. Result and Discussion

3.1. Dataset

The dataset contains 4000 lane images with 15 distinct labels. This research focuses on the preprocessing and optimization of a dataset comprising. An initial observation reveals a common characteristic in the dataset: approximately one-third of the image's upper portion represents the sky, devoid of any lane markings. Leveraging this insight, a cropping process is implemented, resulting in substantial memory savings by removing the top 690 pixels.

Furthermore, to enhance the lane detection capabilities of the model, image sharpening and Canny edge detection techniques are applied. The Canny edge detection technique utilized here is also implemented based on the PaddlePaddle framework. After reading the color images, the lane images are converted to grayscale, preparing them for Canny edge detection. For a more refined Canny edge detection process using PaddlePaddle, the workflow involves several steps. Firstly, the image channels are split, and Gaussian filtering is applied to each channel to achieve blur reduction and noise mitigation. Subsequently, the Sobel operator is employed to compute the intensity gradients of the image, thereby determining edge gradients and directions and refining the edges. Finally, the batch dimension is removed, and additional image processing steps are performed, resulting in the post-edge detection image.

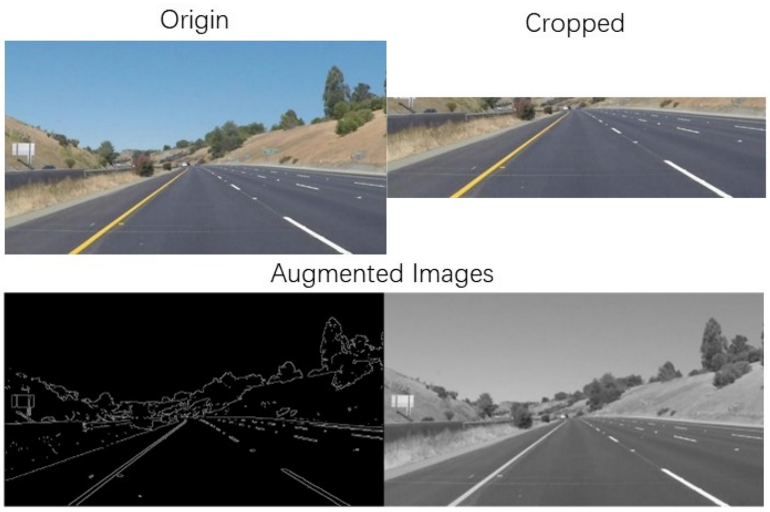

Figure 3. Original and augmented images (Figure Credits: Original)

This approach serves to enhance the accuracy and precision of the Canny edge detection process, contributing to the model's improved lane detection capabilities as shown in Figure 3.

3.2. Performance

In the initial phase of the workflow, this work systematically traverses through all mask image files and execute the following operations while reading them: utilizing 'cv2.imread' to load the mask images, representing them as grayscale (single-channel) images, computing the size of these mask images (denoted as 'size,' which signifies the total pixel count), and iteratively looping through 16 distinct categories (ranging from 0 to 15). For each category, the following sequence of actions is undertaken: the creation of a Boolean mask 'a' to identify the presence of pixels within the mask image having values corresponding to the current category, the calculation of the pixel ratio 'ratio_i' pertaining to the current category (indicating the proportion of pixels associated with that category relative to the total pixel count), and the accumulation of 'ratio_i' within 'count[i]' to maintain a comprehensive record of cumulative pixel ratios. After acquiring the collection of pixel ratios, this work proceed to compute the relative proportions for each category by dividing the pixel ratio of each category by the total pixel ratio. This computation results in the relative representation of each category within the overall pixel distribution.

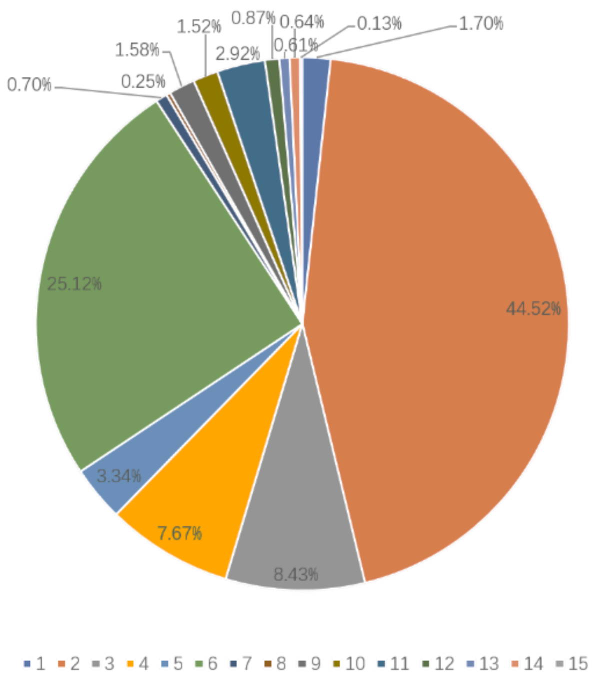

Figure 4. Class distribution (Figure Credits: Original)

The observation has been made that the dataset has an imbalance in the distribution of classes, as depicted in Figure 4. In the subsequent step of dataset optimization, this study undertakes a comprehensive investigation of the dataset with the objective of resolving the challenges posed by class imbalance. Image segmentation tasks often include encountering instances of non-uniform class distribution. This phenomenon is often found in several settings, including defect detection in industrial items, road extraction, and pathological area delineation.

Data augmentation is utilized as a strategy to mitigate the problem of data imbalance, hence achieving a more balanced dataset. Specifically, image processing functions are defined for the classes that require data augmentation. The first function involves introducing random noise into the input image, with a parameter set to control the noise intensity (threshold set at 32). The second function performs gamma correction on the input image to adjust its brightness. The gamma parameter is used to control the correction factor, and it is set to 0.7. Finally, the random_aug(img) function is applied to randomly augment the input image based on random probabilities. The augmentation operations include median blur, noise addition, and gamma correction with different gamma values. Random augmentation is employed to introduce variations in the images, which is particularly useful for enhancing the dataset, especially for tasks like image classification.

For each mask image file, they are read as a grayscale image (single-channel image) and calculate the size of the image (total number of pixels). Subsequently, for each class (with pixel values 2 and 6), this work calculates their pixel ratios within the image, denoted as ratio_a and ratio_b. If the sum of the pixel ratios for both classes is less than 0.003, indicating that these two classes have very few pixels in the image, this work takes the following steps:

Copy the original color image to the augmented directory. Apply data augmentation to the image using the random_aug function, generating four different augmented versions of the color image. Write these augmented images to the new target paths in the augmented directory. Results are shown in Figure 5. This process serves to balance the dataset by addressing the data imbalance issue. Finally, this work recomputes the pixel ratios for the labels and visualize the data. Compared to the previous state of the dataset, the data classes have been notably balanced.

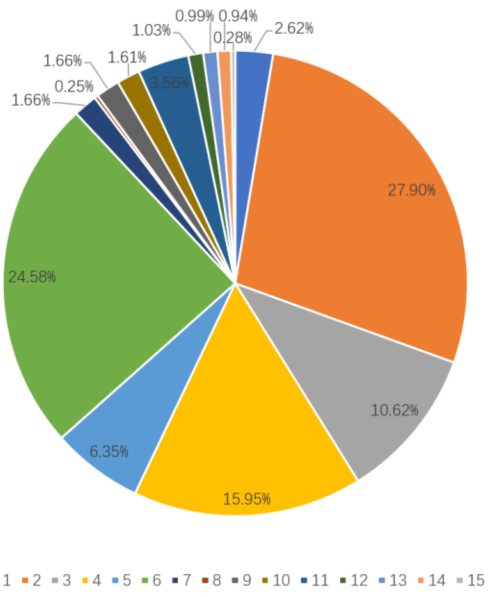

Figure 5. Class distribution after augmentation (Figure Credits: Original)

Upon completing all preparations, the model can be successfully trained by running the built-in 'train.py' script in the PaddleSeg framework.

the performance of a semantic segmentation model is applied to urban scene images. The dataset comprises 500 real urban road scene images, and this work employs a learning rate of 0.005 with a polynomial decay strategy across 40 training epochs, encompassing a total of 35,000 iterations, with each training batch containing 4 images. This comprehensive evaluation employs pivotal metrics such as mean Intersection over Union (mIoU), Accuracy (Acc), and Cohen's Kappa (Kappa), along with a granular examination of class-wise metrics, offering insights into the model's performance across distinct object categories.

Table 1. Result of segmentation.

Metrics	Values
mIoU	0.1979
Acc	0.9857
Kappa	0.5681

The findings are demonstrated in Table 1. Results indicate that the model attains an average mean Intersection over Union (mIoU) score of 0.1979, suggesting a moderate level of performance. This performance is mostly attributed to the limited availability of data for particular object classes. However, a noteworthy exception is the exceptional mIoU of 0.9872 for the first class, underscoring the model's proficiency in predicting this specific category. Pixel-level accuracy, represented as Acc, reaches 0.9857, signifying a high proportion of correct pixel-level classifications, affirming the model's proficiency at a fine-grained level. Cohen's Kappa attains a value of 0.5681, indicating substantial agreement between model predictions and ground truth labels.

Further, class-wise evaluation elucidates disparities in model performance across different object categories, with certain classes exhibiting notably higher IoU and Acc values than others, thus offering valuable insights into the model's performance variations.

4. Conclusion

This study presents a thorough performance evaluation of a semantic segmentation model when applied to urban scene images. While the model's overall performance is moderate, specific object categories exhibit exceptional performance. The model's high pixel-level accuracy and substantial Kappa coefficient underscore its potential for urban scene semantic segmentation tasks. Addressing class imbalance and augmenting dataset diversity represent avenues for further enhancing model performance.

References

[1] Yurtsever, E., Lambert, J., Carballo, A., & Takeda, K. (2020). A survey of autonomous driving: Common practices and emerging technologies. IEEE access, 8, 58443-58469.

[2] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, & Chen, T. (2018). Recent advances in convolutional neural networks. Pattern recognition, 77, 354-377.

[3] Li, J., Jiang, F., Yang, J., Kong, B., Gogate, M., Dashtipour, K., & Hussain, A. (2021). Lane-deeplab: Lane semantic segmentation in automatic driving scenarios for high-definition maps. Neurocomputing, 465, 15-25.

[4] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[5] Shrestha, A., & Mahmood, A. (2019). Review of deep learning algorithms and architectures. IEEE access, 7, 53040-53065.

[6] Guo, Y., Liu, Y., Georgiou, T., & Lew, M. S. (2018). A review of semantic segmentation using deep neural networks. International journal of multimedia information retrieval, 7, 87-93.

[7] Peng, J., Liu, Y., Tang, S., Hao, Y., Chu, L., Chen, G., et al. (2022). Pp-liteseg: A superior real-time semantic segmentation model. arXiv preprint arXiv:2204.02681.

[8] Liu, Y., Chu, L., Chen, G., Wu, Z., Chen, Z., Lai, B., & Hao, Y. (2021). Paddleseg: A high-efficient development toolkit for image segmentation. arXiv preprint arXiv:2101.06175.

[9] Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2881-2890.

[10] Chen, Guowei, et al. (2022). PP-Matting: High-Accuracy Natural Image Matting. arXiv preprint arXiv:2204.09433.