Lane Line Detection Dataset Expansion and nighttime lane line detection algorithm based on Cycle GAN

ECE543: Computer Vision final project

Hongqing Liu*
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, USA
hl85@illinois.edu

Weichen Liu*
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, USA
wl45@illinois.edu

Abstract—With the development of the autonomous driving industry, the problem of data imbalance in different scenarios plagues most autonomous driving datasets and models. Data generation based on existing data can help improve the existing data imbalance problem and improve the robustness of related algorithms. Through the existing large-scale automatic driving data set training cycle generative adversarial network(Cycle GAN), this paper realizes the expansion of the related data set, and based on the expanded data, the performance of the lane line detection algorithm and the basic lane line detection algorithm based on pixel processing are improved .

1. Introduction

Today's autonomous driving development is data-driven and oriented. In order to obtain autonomous driving algorithms that can perform well in various environments, large-scale data sets are often required to cover many long-tail problems, such as rainy days, dense fog, snow, night and other scenes. However, the cost of data collection is high, and compared with conventional road scenes, it is often impossible to predict and collect special road scenes when they occur. The lack of data in this specific scenario has caused the problem of unbalanced data distribution in the data set. Specifically, the dataset used by the algorithm has severe sample imbalances. The unbalanced samples of training data will reduce the accuracy and robustness of automatic driving algorithms such as lane line detection in special environments, which is unacceptable for automatic driving algorithms with extremely high safety requirements. In order to solve the long-tail problem of autonomous driving algorithms, it is a low-cost and high-efficiency solution to expand the data set in unconventional scenarios by training and generating data in special scenarios based on a large amount of existing conventional scenario data. This project uses Cycle GAN[1] for style migration between different scenarios to achieve data expansion. Specifically, the day and night lane images in BDD100k[2] are converted to each other through Cycle GAN, and CULane[3] is used as the test data set to qualitatively evaluate the data generation effect. After the data expansion is completed, the generated data samples are input into the lane detection algorithm to realize lane line detection.

1.1. Lane Line Detection

The purpose of lane line detection is to detect lane lines through on-board cameras or lidars. The lane line detection task can be analogized to a segmentation task, which not only requires obtaining the direction and shape of the lane lines, but also distinguishes different types of lane lines. The construction of the lane line detection dataset requires the data balance of each scene category, such as highways, urban roads, construction roads, night, rainy, snowy and other data to simulate the real driving environment. At present, there are four main types of detection methods based on deep learning: segmentation-based

*These authors contributed to the work equally and should be regarded as co-first authors.

methods, detection-based methods, parametric curve-based methods, and key point-based methods.

The classic works of segmentation-based methods include SCNN[4], RESA[5], and LaneNet[6]. The performance is poor when the lane lines are occluded; the related works of detection-based methods include: LineCNN[7], LaneATT[8], CondLaneNet[9], UFAST[10], etc. By preset Anchor, in the case of severe occlusion, etc. It is still possible to obtain continuous instances of lane line detection, but the shape of the anchor will affect the flexibility of detection. Works related to key point-based methods include FOLOLane[11], GANet[12], and parametric curve-based methods include PolyLaneNet[13] and BezierLaneNet[14].

In general, when the lane lines are visible , the lane detection is simple, in the case that the lane line is occluded and covered, the lane line of the occluded part can be predicted by the unconcluded lane line. However, existing datasets lack data in weather environments such as heavy snow, making data inference work only possible on mild weather data. In recent years, with the rise of GAN[15] in the field of style transfer, it has become feasible to generate extreme weather road data from existing datasets, providing data for lane detection algorithms to face long-tail problems and improving the robustness of the algorithm.

1.2. Data Expansion

Image generation refers to the process of training a model based on an existing dataset and using it to generate new images that are highly similar to the original dataset. Image generation can be especially important when we need to acquire a large number of images or perform image recognition. The most widely used algorithms in the field of image generation are various neural networks, among which GAN and its derivative networks are the most prominent. According to different requirements for images in various fields, image generation has been widely used in fields such as image style conversion[16], image super-resolution restoration[17], target detection and tracking[18], image fusion[19] and so on. In the field of image generation, we usually use variational inference VAE[20] and GAN[15] as the training algorithms. However, since images generated by variational inference is sometimes too fuzzy[21], GAN has become one of the most promising methods to in recent years. The GAN consists of two networks, a generator and a discriminator. The generator is responsible for reconstructing the input image; the reconstruction results and the real data set are sent to the discriminant network for judgment, and the discriminant network is responsible for distinguishing whether the output of the generator is the real image from the real data set or the generated image from itself.

The generator will try to make the generated images fool the discriminator. by generating more and more real images; the discriminator will try to distinguish whether the input is real or generated, and its feedback will guide the generator to generate more realistic pictures. By implementing this close loop, the output images will be more and more similar in content and style to the real dataset, like shown in Fig 1.2.1.

Fig 1.2.1 Training result sample by GAN

However, since the GAN does not have constraints in the image's content and it always models collapse during training, we are inclined to use Cycle GAN to make constraints in the image's content.

Cycle GAN uses two different styles of datasets. It achieves the purpose of style constraints by optimizing the adversarial loss. This process requires two networks, the generator and the

discriminator; Cycle GAN achieves the purpose of content constraints through a reverse change. Similarly, this process also requires the generator and the discriminator. Therefore, the network components of Cycle GAN are four parts: two generators and two discriminators. The loss component is composed of two parts: the confrontation loss and the content distance loss of the original input and the inversely transformed output.

$$L_{GAN}(G, D_Y, X, Y) = E_y \sim pdata(y)[log\ log\ D_Y(y)] + E_x \sim pdata(x)[log(1 - D_Y(G(x))],$$

where G tries to generate images G(x) that look similar to images from domain Y , while DY aims to distinguish be- tween translated samples G(x) and real samples y. G aims to minimize this objective against an adversary D that tries to maximize it, i.e.,
$minG\ maxDY\ LGAN(G, DY, X, Y)$. We introduce a similar adversarial loss for the mapping function F : Y → X and its discriminator DX as well:
i.e.,$minF\ maxDX\ LGAN(F, DX, Y, X)$. [1]

2. Detail of the Approach

This part will introduce the data used in the project, image generation method and lane line detection steps.

2.1. Data

Fig 2.1.1 Example figure of the BDD100K[2]

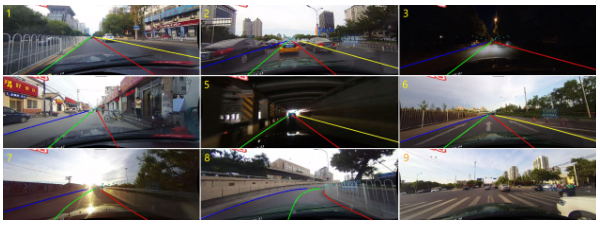

Fig 2.1.2 Example figure of the CULane[3]

The work in this paper is based on two open source datasets, where BDD100K is a diverse autonomous driving dataset for heterogeneous multi-task learning, and CULane[3] is a large-scale challenging dataset for lane detection . BDD100K-oriented tasks include image labeling, object detection and instance segmentation. Due to the limitation of training scale and training time, all the data used in the training of this project are static image data in BDD100K instead of video data. After distinguishing the day and night data according to the data label, the training data used by the model was obtained, including 36,728 daytime data and 27,971 nighttime data. Because CULane[3]'s night and day data have serious data imbalance, and CULane[3] only marks the lane pixels of each frame, so it is only used as a test data set.

2.2. Day2Ninght&Night2Day

In this part, we implemented data expansion based on the Cycle GAN. Thanks to the structure of the Cycle GAN, we can train both day2night and night2day models at the same time. Cycle GAN can use unpaired data to establish a mapping between source domain X and target domain Y: $G: X \rightarrow Y$, so that the image of the source domain X is converted to the target domain Y distribution of similar images. That is, the image $G(X)$ cannot be distinguished whether it was sampled from Y or generated by G. But if there is only $G: X \rightarrow Y$, it is obviously impossible to complete this task. Because this mapping can only ensure that $G(X)$ is similar to the samples in the target domain Y, it cannot

ensure that it corresponds to the image before generation. To solve this problem, a cycle consistency loss is introduced in Cycle GAN. It is not enough to have only $G: X \rightarrow Y$, it is necessary to introduce another mapping $F: Y \rightarrow X$, so that $G(X)$ remaps back to the source domain X, and measures the gap between $F(G(X))$ and X, hoping minimize the gap as small as possible.

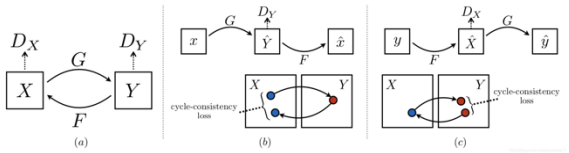

Fig 2.2.1 Cycle GAN workflow diagram[1]

The architecture for our generative networks are from Johnson et al. [21] This network contains three convolutions, several residual blocks [22], two fractionally-stride convolutions with stride 12, and one convolution that maps features to RGB. We use 6 blocks for 128 × 128 images and 9 blocks for 256 × 256 and higher-resolution training images. Similar to Johnson et al. [21], we use instance normalization [23]. For the discriminator networks, we use 70 × 70 Patch GANs [24, 25, 26], which aim to classify whether 70 × 70 overlapping image patches are real or fake. Such a patch-level discriminator architecture has fewer parameters than a full-image discriminator and can work on arbitrarily-sized images in a fully convolutional fashion [24].

We set the model to save parameters every five rounds of training as a reference for subsequent evaluation of the training process.

2.3. Lane line detection

The pipeline of this part is shown in Fig 2.3.1.

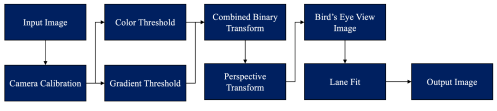

Fig 2.3.1 Pipeline of landline detection

For the image processing, we first convert the RGB image into the HSL color space. The threshold for deciding what pixels should be set as black or white is applied to the saturation value of the image. The filtered output of the image is a binary image. We also applied a gradient filter to the image in both the x and y direction. The gradient and color filtered images are combined together and we removed some noise from the image.

We then apply a perspective transform based on the approach from this web[27] to the combined image where the four source points for the transformation were measured through pixel locations in Microsoft Paint. We picked four source points that were close to the left and right lanes while the size of the transformed output image was determined by the maximum width and height of the source points trapezoid.

After the image is processed, we pass it into our lane fit function that outputs two polynomials for the left and right lanes. Our approach is to first calculate the histogram of the image based on the x-direction white pixels and set the two highest values x, one for the left lane and one for the right lane, as the starting points of our detection boxes. The y-direction of each lane in the image is broken down into 9 bounding boxes with set margin width, and the mean of the white points in each bounding box is recorded into a vector. If the number of white pixels detected within a bounding box is less than our tuned threshold while the noise is minimal, we use the previous two bounding boxes' locations to decide the current bounding box's location by assuming that the three boxes are collinear. If there is a lot of noise in the image, we only use the previous bounding box's location to decide the current bounding box's location. We then apply polynomial fit on the vectors for the left and right lanes, subsequently returning the output image.

3. Result

This chapter will be divided into two parts to explain the results of data generation and the details of lane detection.

3.1. Day2ninght&Night2Day

Based on the characteristics of the Cycle GAN model, the network architecture after training can convert daytime images into nighttime styles, and can also convert nighttime images into daytime styles. In order to prevent the divergence of GAN and its variants during the training process, a learning rate of 0.002 is selected for the training parameters, which gradually decreases to 0 at the end as the number of training rounds increases. The project is trained for a total of 100 epochs.

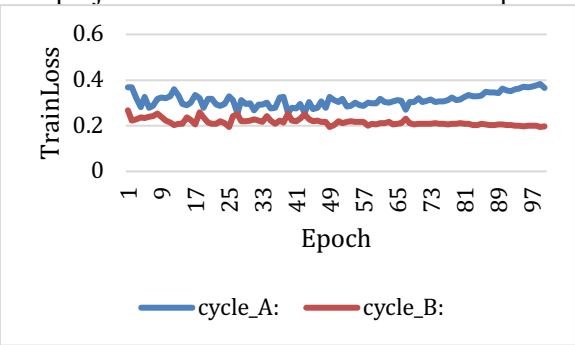

Fig 3.1.1 Generator and discriminator loss vs. number of training epochs

From the change trend of the loss function, it can be seen that the training loss of the night image generated from the daytime tends to increase in the later stage of training, while the network training loss of the daytime image generated from the night image tends to be stable.

Fig 3.1.2a Original dark night pictures

Fig 3.1.2b Night2Day,10 Epochs

Fig 3.1.2c Night2Day,30 Epochs

Fig 3.1.2d Night2Day,50 Epochs

Fig 3.1.2e Night2Day,80 Epochs

Fig 3.1.2f Night2Day,100 Epochs

Using the night lane picture in CULane[3] as the original image, the night-to-day generation effect is shown in Fig 3.1.2a-f. It can be seen that when the number of training rounds is small, there are still obvious bright spots in the generated image, and the illumination distribution is not uniform. As the training progresses, the brightness of the generated image is gradually evenly distributed.

Fig 3.1.3a Original Day night picture

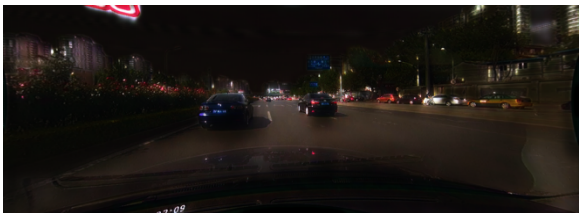

Fig 3.1.3b Day2Night,10 Epochs

Fig 3.1.3c Day2Night,30 Epochs

Fig 3.1.3d Day2Night,50 Epochs

Fig 3.1.3e Day2Night,80 Epochs

Fig 3.1.3f Day2Night,100 Epochs

It can be seen that due to the photosensitive element of the camera, Day2Night photos have reduced a lot of noise compared to Night2Day images, making the image generation effect better. Also note that at lower training epochs, the conversion effect on image details is still not satisfactory.

3.2. Lane line detection

The original picture of the nighttime lane line is shown in fig 3.2.1.

Fig 3.2.1 Original nighttime lane line picture

The day image transferred by Cycle GAN is shown in fig 3.2.2. It is clear that the image has a lot of noise.

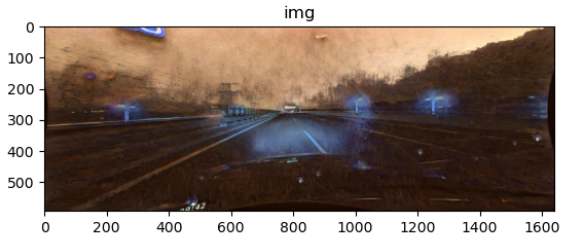

Fig 3.2.2 Daytime lane line picture

The color filter in HSL color space and gradient filter is shown in fig 3.2.3 and fig 3.2.4.

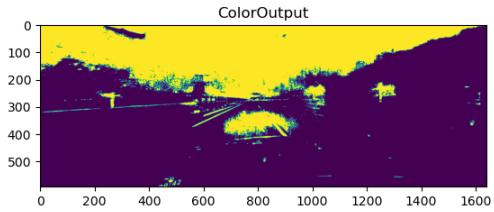

Fig 3.2.3 Color filter in HSL color space

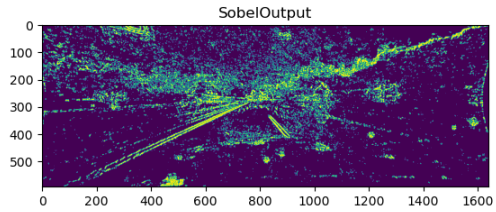

Fig 3.2.4 Gradient filter image

The parameters of gradient filter and color filters we applied to lane line images were shown in table1. The gradient values for the lane lines are usually the highest of the whole picture, so we need to make both minimum and maximum thresholds high to filter out irrelevant objects in

the scene. Lowering the minimum threshold further would result in significantly less filtered noise.

For the color threshold, the real-world footage has a relatively higher saturation value which means its minimum and maximum thresholds have to be relatively high as well. By tuning the thresholds for different inputs, we can find the best filters to remove more noise from the images and consequently generate better lane detection down the line.

Table I. Threshold minimum and maximum for gradient and color filters

	Threshold minimum	Threshold maximum
Gradient threshold	200	255
Color threshold	100	255

The combined image of color filter image and gradient image is shown in Fig 3.2.5.

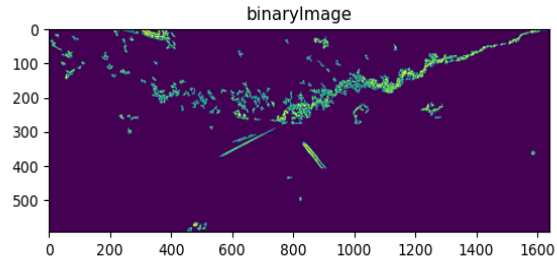

Fig 3.2.5 Combined image

By using methods on the web[26], we can pick up the region we are interested in and transfer it into a bird-eye view. The bird-eye view image is shown in Fig 3.2.6.

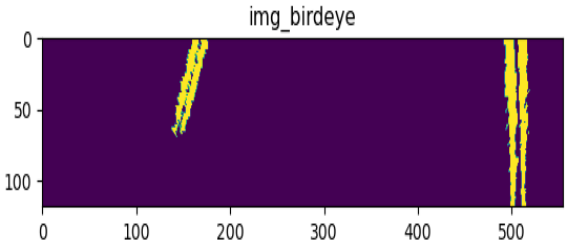

Fig 3.2.6 Bird-eye view image

Four points coordinate we used to pick the place we interested in are shown in Table II.

Table II. Source point locations for image transformation

Point A	Point B	Point C	Point D
[492, 350]	[384, 398]	[939, 398]	[884, 350]

The lane line detection model is shown below, the box is the place where the algorithm detects the lane line.

The bird-eye detection and fit images are shown in Fig 3.2.7 and Fig 3.2.8.

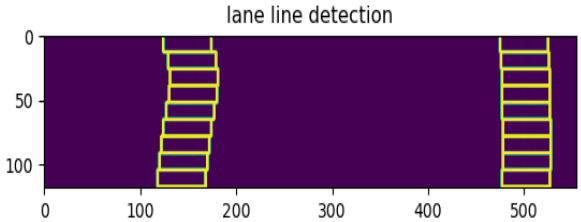

Fig 3.2.7 Bird-eye lane line detection image

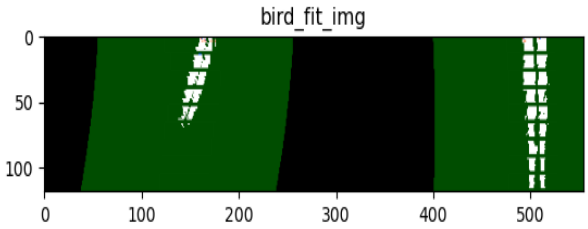

Fig 3.2.8 Bird-eye fit image

We apply polynomial fit on the vectors for the left and right lanes, subsequently returning the output image.

The annotated image for transferred image (daytime lane line image) and original image (nighttime lane line image) is shown in Fig 3.2.9 and Fig 3.2.10.

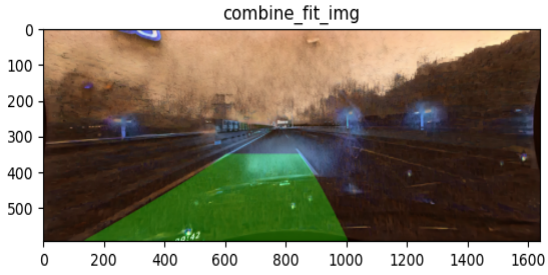

Fig 3.2.9 Annotated image in transferred image

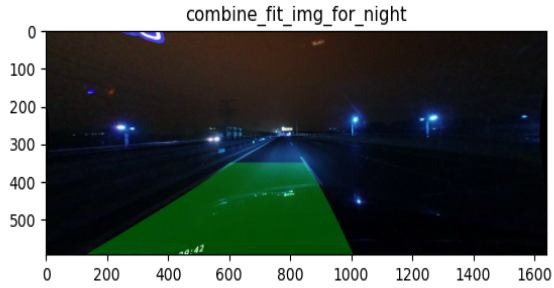

Fig 3.2.10 Annotated image in nighttime image

We can find out that the algorithm can successfully detect lane line in night time by adding Cycle GAN as a preprocessing section.

4. Discussion and Conclusion

This part will introduce the project's attempts in other directions, potential optimization directions and the final summary.

4.1. Data Expansion

In terms of data expansion, we believe that there are parts worth optimizing in terms of network training parameters and generation effects.

4.1.1. Training epochs

From the loss-number of rounds trend image, it can be seen that after about 70 epochs of training, one side of the generation confrontation network has a tendency to diverge. Since the loss function of Cycle GAN[3] is different from traditional GAN[15], we did not modify the hyperparameters in the momentum descent function. In fact, we believe that excessively large hyperparameters will cause the model to diverge when the number of training rounds is too large, and eventually diverge and cause model training to fail. Although there was no extreme case of model divergence in this project, according to the research of Liu[28] et al., the selection of hyperparameters plays an important role in the convergence speed of the model and whether it will eventually diverge.

4.1.2. Night2Day

There is a lot of noise in the final output of Night2Day, which is caused by the high sensitivity of the camera when shooting in a dark environment. This paper tried mean filtering, Gaussian filtering, and median filtering, but did not achieve better results. . We believe that the light distribution of the original image in the image leads to poor denoising effect: uneven lighting causes uneven distribution of noise on the sensor of the camera, so simple filtering and denoising cannot remove such noise .

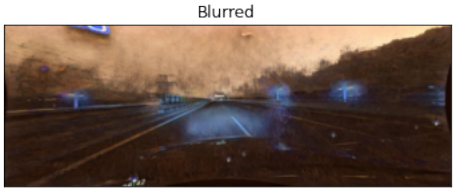

Fig 4.2.1 Mean filter

Fig 4.2.2 Gaussian filter

Fig 4.2.3 Median filter

4.2. Lane Line Detection

From the annotated image in nighttime, it can be seen that the basic pixel-based algorithm can successfully detect lane lines for nighttime images after transferring them by cycleGan. By tuning the parameters in color threshold and gradient threshold, the algorithm can successfully detect the lane line and fit it into a function.

4.3. Conclusion

This paper implements the expansion of related data sets, and based on the expanded data, improves the performance of the lane detection algorithm and the basic lane detection algorithm

based on pixel processing. We trained the CycleGan network using daytime lane marking pictures and night lane marking pictures, so that the lane marking pictures can be converted from night to day and from day to night to achieve the purpose of expanding the data set. On the basis of the generated data, we applied a filtering-based lane line detection algorithm to improve the robustness of the algorithm. Future research topics include, but are not limited to, research on methods and parameter configurations for faster and more stable training of generative adversarial networks, and research on noise reduction algorithms for data expansion under low-light conditions.

5. Statement of Individual Contribution

Hongqing Liu*: Proposed the project, lead the group and implement the basic lane line detection algorithm. Wrote the abstract, introduction and the part of lane line detection in the report. I also participated in the discussion of Cycle GAN.

Weichen Liu*: Responsible for the work of the Cycle GAN part and filtering the generated images. Revised the abstract, the introduction of the extended data section, the writing of the results and discussion, and the polishing and layout of the full report.

*Two authors contributed to the work equally.

6. REFERENCES

[1] Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE international conference on computer vision (pp. 2223-2232).

[2] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., ... & Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2636-2645).

[3] Pan, X., Shi, J., Luo, P., Wang, X., & Tang, X. (2018, April). Spatial as deep: Spatial cnn for traffic scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).

[4] Parashar, A., Rhu, M., Mukkara, A., Puglielli, A., Venkatesan, R., Khailany, B., ... & Dally, W. J. (2017). SCNN: An accelerator for compressed-sparse convolutional neural networks. *ACM SIGARCH computer architecture news*, *45*(2), 27-40.

[5] Zheng, T., Fang, H., Zhang, Y., Tang, W., Yang, Z., Liu, H., & Cai, D. (2021, May). Resa: Recurrent feature-shift aggregator for lane detection. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 4, pp. 3547-3554).

[6] Wang, Z., Ren, W., & Qiu, Q. (2018). Lanenet: Real-time lane detection networks for autonomous driving. *arXiv preprint arXiv:1807.01726*.

[7] Li, X., Li, J., Hu, X., & Yang, J. (2019). Line-cnn: End-to-end traffic line detection with line proposal unit. *IEEE Transactions on Intelligent Transportation Systems*, *21*(1), 248-258.

[8] Tabelini, L., Berriel, R., Paixao, T. M., Badue, C., De Souza, A. F., & Oliveira-Santos, T. (2021). Keep your eyes on the lane: Real-time attention-guided lane detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 294-302).

[9] Liu, L., Chen, X., Zhu, S., & Tan, P. (2021). Condlanenet: a top-to-down lane detection framework based on conditional convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 3773-3782).

[10] Shyam, P., Yoon, K. J., & Kim, K. S. (2021). Weakly supervised approach for joint object and lane marking detection. In *Proceedings of*

the IEEE/CVF International Conference on Computer Vision (pp. 2885-2895).

[11] Qu, Z., Jin, H., Zhou, Y., Yang, Z., & Zhang, W. (2021). Focus on local: Detecting lane marker from bottom up via key point. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 14122-14130).

[12] Morley, M. S., Atkinson, R. M., Savić, D. A., & Walters, G. A. (2001). GAnet: genetic algorithm platform for pipe network optimisation. *Advances in engineering software*, *32*(6), 467-475.

[13] Tabelini, L., Berriel, R., Paixao, T. M., Badue, C., De Souza, A. F., & Oliveira-Santos, T. (2021, January). Polylanenet: Lane estimation via deep polynomial regression. In *2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 6150-6156). IEEE.

[14] Cheng, Z., Zhang, G., Wang, C., & Zhou, W. (2022). DILane: Dynamic Instance-Aware Network for Lane Detection. In *Proceedings of the Asian Conference on Computer Vision* (pp. 2075-2091).

[15] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139-144.

[16] J. Zhang and Y. Hou, "Image-to-image Translation Based on Improved Cycle-consistent Generative Adversarial Network," *Journal of Electronics and Information Technology,* vol. 42, no. 5, pp. 1216-1222, 2020.

[17] X. Sun, X. G. Li, J. F. Li, and L. Zhuo, "Review on Deep Learning Based Image Super-resolution Restoration Algorithms," *Zidonghua Xuebao/Acta Automatica Sinica,* vol. 43, no. 5, pp. 697-709, 2017.

[18] X. Wang, A. Shrivastava, and A. Gupta, "A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] Z. Jiang, H. Tao, Y. Chen, L. Zhang, D. Wu, and W. Li, "Two Optical Image Fusion Model of Transmission Line Based on DCGAN," in *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)*: IEEE, pp. 2999-3003.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114,* 2013

[21] Johnson, J., Alahi, A., & Fei-Fei, L. (2016, October). Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision* (pp. 694-711). Springer, Cham.

[22] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[23] Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

[24] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125-1134).

[25] Li, C., & Wand, M. (2016, October). Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European conference on computer vision* (pp. 702-716). Springer, Cham.

[26] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., ... & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4681-4690).

[27] https://theailearner.com/2020/11/06/perspective-transformation/

[28] Liu, W., Gu, Y., & Zhang, K. (2021, September). Face Generation using DCGAN for Low Computing Resources. In *2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)* (pp. 377-382). IEEE.