Generating Traffic Scene with Deep Convolutional Generative Adversarial Networks

Danchen Zhao
Institute of Artificial Intelligence and Robotics
Xi'an Jiaotong University
Xi'an, China
danchenzhao@stu.xjtu.edu.cn

Jingkun Weng
Institute of Artificial Intelligence and Robotics
Xi'an Jiaotong University
Xi'an, China
412377734@qq.com

Yuehu Liu
Institute of Artificial Intelligence and Robotics
Xi'an Jiaotong University
Xi'an, China
liuyh@mail.xjtu.edu.cn

***Abstract*—Training and testing unmanned vehicles need various real data. However, data of some special or dangerous testing environment may not be accessible, or may only be accessible at certain times. So, using generative adversarial networks to learn the real traffic scene and generate a new scene is an effective way of solving the problem. In this paper, a framework of deep convolutional generative adversarial networks (DCGAN) was used to generate new traffic scene images and videos. Firstly, 300 sets of videos and images of overtaking scenes were selected as training data. A generator with convolutional neural network was used to generate samples. Then the training data and the generated samples were trained in a two-class discriminator that provides the probabilities of the samples that come from the generated sample and the training data respectively. Then the generator was updated by back propagation algorithm. Afterward, a new sample was generated and trained in the discriminator again. Repeated several times, a serial of generated samples were generated, the probability distribution of which is basically the same as the training data. The experiments show that this method can effectively generate realistic traffic scene images and videos.**

Keywords—generative adversarial networks; traffic scene; generated image; unmanned vehicle

I. INTRODUCTION

In recent years, research in unmanned vehicles has made significant progress. However, unmanned vehicles still can't work properly in complex road conditions. The most authoritative way to verify unmanned vehicles is field test. However, field test is faced with difficulties such as limitation of test site, unrepeatable test condition, time-consuming and high-cost procedure.[1] Besides, training and testing unmanned vehicles in simulation environment need various real data. However, some special or dangerous testing environment may not be accessible, or may only be accessible at certain times. Thus, we searched for a method that can generate realistic traffic scene data.

The rapid development of artificial neural networks brings us new ideas. In 2014, Goodfellow et al. [2] proposed the Generative Adversarial Networks for generating data. Using generative adversarial networks to learn the real traffic scene and generate a new scene is an effective way of solving our problem. In this paper, a framework of deep convolutional generative adversarial networks (DCGAN) was used to generate new traffic scene images and videos. Firstly, 300 sets of videos and images of overtaking scenes were selected as training data. A generator with convolutional neural network was used to generate samples. Then the training data and the generated samples were trained in a two-class discriminator that provides the probabilities of the samples that come from the generated sample and the training data respectively. Then the generator was updated by back propagation algorithm. Afterward, a new sample was generated and trained it in the discriminator again. Repeated several times， a serial of generated samples were generated, the probability distribution of which is basically the same as the training data. What' s more, the discriminator can't distinguish the two samples. The traffic scene images are successfully generated.

II. RELATED WORKS

Generative image models are well studied and fall into two categories: parametric and nonparametric.

The non-parametric models often do matching from a database of existing images, often matching patches of images, and have been used in texture synthesis (Efros et al. [3], 1999), super-resolution (Freeman et al. [4], 2002) and in-painting (Hays et al. [5], 2007).

Parametric models for generating images has been explored extensively (for example on MNIST digits or for texture synthesis (Portilla et al. [6], 2000)). However, generating natural images of the real world have had not much success until recently. A variational sampling approach to generating images (Kingma et al. [7], 2013) has had some success, but the samples often suffer from being blurry. Another approach generates images using an iterative forward diffusion process (Sohl-Dickstein et al. [8], 2015). Generative Adversarial Networks (Goodfellow et al. [2], 2014) generated images suffering from being noisy and incomprehensible. A laplacian pyramid extension to this approach (Denton et al. [9], 2015) showed higher quality images, but they still suffered from the objects looking wobbly because of noise introduced in chaining

Authorized licensed use limited to: Florida Polytechnic University. Downloaded on October 20,2023 at 05:17:12 UTC from IEEE Xplore. Restrictions apply.

multiple models. A recurrent network approach (Gregor et al. [10], 2015) and a deconvolution network approach (Dosovitskiy et al. [11], 2014) have also recently had some success with generating natural images. However, they have not leveraged the generators for supervised tasks.

III. Generative Adversarial Networks (GANs)

The idea of GANs came from the zero-sum games in which one person's gains result in losses for the other participants. In GANs, the two participants are a generative model *G* that captures the data distribution, and a discriminative model *D* that estimates the probability that a sample came from the training data rather than *G*. The training procedure for *G* is to maximize the probability of *D* making a mistake.

D is trained to maximize probability of assigning the correct label to both training examples and samples from *G*. And *G* is trained to minimize log(1-*D*(*G*(*z*))).

In other words, *D* and *G* play the following two-player minimax game with value function *V*(*G*,*D*):

$$\min_G \max_D V(G,D) = E_{x\sim p_{data}(x)}[\log(D(x))] + E_{z\sim p_z(x)}[\log(1-D(G(z)))], \quad (1)$$

where x is the training data, z is generated sample, p_{data} is the probability distribution of training sample, p_z is the probability distribution of generated sample.

Compared with previous machine learning methods, GANs has the advantages showing below:

- According to the final results, it can generate better samples than other models. The image is more distinct and sharp.
- The framework of GANs can train any kind of generative networks.
- With no need for designing factorization model, GANs applies to any generative networks and discriminator.
- With no need for Markov chain model and inference in learning progress, GANs avoids troublesome probability problem of approximate calculation.

In spite of the advantages, GANs still has some problems:

- *Non-convergence problem*: Current theories indicate that GANs should perform excellently in Nash equilibrium. However, gradient descent achieves Nash equilibrium only with convex function.
- *Collapse problem*: GANs model is defined as min-max problem. It is difficult to distinguish whether training is making progress. When the learning progress comes up with collapse problem, the generator begins to degenerate, generating the same samples. At the same time, discriminator gives same direction for similar samples. The training progress will not go on.
- *The model is uncontrollable*: Compared with other generative model, GANs doesn't need to formulate *p*(*x*), which means it doesn't need modeling in advance. However, for larger images with more pixels, basic GANs will be uncontrollable. Thus, in each learning progress, *G* updates only one time after *D* updates *k* times.

IV. Generating Traffic Scene Images with DCGAN

Because of the disadvantages, GANs can't perform well in many tasks. Researchers have tried several methods to solve these problems. In 2015, Radford et al. [12] proposed deep convolutional generative adversarial networks (DCGAN) which combines supervised CNN with unsupervised GANs to solve the engineering problem. In 2017, Arjovsky et al. [13] proposed Wasserstein GAN, which caused great repercussions because of its ingenious theoretical analysis and simple algorithm. However, it's difficult to train the model and has low convergence rate. Then the authors provided new solution, gradient penalty which is to add gradient penalty item to penalty function. This new method is called WGAN-GP [14].

In practice, we found that the images generated by WGAN-GP are not better than DCGAN. And WGAN-GP has lower convergence rate than DCGAN. Therefore, we applied DCGAN to our problem.

A. Batch Normalization

Batch Normalization (BN) [15] is to solve the offset problem. In detail, the input of neural networks obeys normal distribution between 0 and 1. The parameters are updated according to chain rule to ensure the output has the same distribution as the input. In our experiment, if we don't use BN algorithm, we'll encounter collapse problem. In practice, BN is realized by function tensorflow.contrib.layers.batch_norm().

B. Building Generator

We define z as the input of generator. z is 100 dimensional uniform distribution valued [-1,1]. An [100,16384] array w_{h0} was generated randomly obeying normal distribution in (0,0.02). Then we got $h_0^{'}$ according to

$$h_0^{'} = w_{h_0} \cdot z + bias \quad (2)$$

where *bias* has initial value of zero. After we reshape $h_0^{'}$ to get h_0, applied BN algorithm to h_0.

In convolution layer, a series of four fractionally-strided convolutions then convert this high level representation into a 64 × 64 pixel image. In our experiments, the convolution kernel is 5. The operation in convolution layer of generator can be represented as follows:

$$h_j^l = \sum_{i\in M_j} f(dconv(w_{ij}, b_j, BN(h_j^{l-1}))) \quad (3)$$

where h_j^l is the *j*th feature map of *l*th image, f is transfer function, *dconv* is fractionally-strided convolution, w_{ij} and b_j are weight and bias. The initial value of w_{ij} is random normal distribution in (0,0.2). The initial value of b_j is zero. In practice, the transfer functions of convolution layer and last layer are function Relu and function tanh respectively.

Authorized licensed use limited to: Florida Polytechnic University. Downloaded on October 20,2023 at 05:17:12 UTC from IEEE Xplore. Restrictions apply.

C. Builging Discriminator

The discriminator of DCGAN is an improved convolutional neural networks. In practice, generator generates samples value in [-1,1]. However, the form of training data is unit8, so we should transfer the data form to float32. Besides, when saving the png images, generated samples were added 1 and then divided by 2. Thus, the value of generated samples is transferred to [0,1].

In the convolution layer, the operation can be represented as follows:

$$h_j^l = \sum_{i \in M_j} f(conv(w_{ij}, b_j, BN(h_j^{l-1}), 1)) \tag{4}$$

where conv means convolution with step length of 1, w_{ij} and b_j are weight and bias. The initial value of w_{ij} is random normal distribution in (0,0.2). The initial value of b_j is zero.

In the last layer, the probability output is:

$$h_l = f(w_{h_l} \cdot BN(h_{l-1}) + bias) \tag{5}$$

The transfer function of convolution layer is LRelu, with coefficient of 0.2. The transfer function of last layer is function sigmoid.

D. Adaptive Moment Estimation (ADAM)

The loss function of discriminator is

$$V(D) = -E_{x \sim p_{data(x)}}[\log(D(x))] - E_{x \sim p_{g(x)}}[\log(1 - D(x))] \tag{6}$$

The loss function of generator is

$$V(D) = E_{x \sim p_{g(x)}}[\log(1 - D(x))] \tag{7}$$

GANs trains the discriminator for *n* times and then trains generator. In practice, we make *n*=1. The result of loss function is used to firstly update discriminator with adaptive moment estimation (ADAM) [16] and then update the generator. The procedure is described in Algorithm 1.

Algorithm 1 ADAM, α=0.0002; β_1=0.5; β_2=0.9 ; ε =10^{-8}. m_0=0, v_0=0, *t*=0. *f* is the transfer function.

```
1: while $\theta_t$ has not converged, do
2:     $t = t + 1$
3:     $g_t = \nabla_\theta f_t(\theta_{t-1})$
4:     $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
5:     $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
6:     $\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$
7:     $\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$
8:     $\theta_t = \theta_{t-1} - \frac{\alpha \cdot \hat{m}_t}{\sqrt{v_t} + \varepsilon}$
9: end while
10: return $\theta_t$
```

E. DCGAN

The procedure of DCGAN is described in Algorithm 2.

Algorithm 2 DCGAN, α=0.0002; β_1=0.5; Whenever the generator is trained once, the discriminator is trained *k* times. Each time the generator generates *m* generated samples. Number of training data is *n*. *m* is minibatch. All the parameters value in [-0.02,0.02]. The number of training times is *iter*.

```
1: idx=n/m;
2: for iter do
3:   for idx to
4:     input m noises $\{z^{(1)} \cdots z^{(m)}\}$
5:     for k step do
6:       put m training samples $\{x^{(1)} \cdots x^{(m)}\}$ into discriminator
7:       put m noises into generator
8:       put the generated samples into discriminator
9:       $d_w = \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))]$
10:      $w_d = w_d + ADAM(d_w, \alpha, \beta_1)$
11:    end for
12:    $g_w = \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)})))$
13:    $w_g = w_g + ADAM(g_w, \alpha, \beta_1)$
14:  end for
15: end for
```

V. Experiments

Our experiment aims to train the DCGAN to generate realistic traffic scene images and videos. In the following, we will discuss the collection and usage of training data, generating method and the results.

A. Training Data

We use both open dataset online and dataset collected by ourselves. The open dataset online we choose KITTI dataset [17]. KITTI dataset is developed by Karlsruhe Institute of Technology and Toyota Technological Institute. At present, it's the largest dataset for evaluation of computer vision in autonomous driving environment. Besides of KITTI dataset, we use the dataset collected by ourselves. This dataset includes large amounts of traffic scene videos and GPS data from Xi'an to Changshu, including differents types of traffic scene such as city road, highway and country road. They were taken by 5 cameras installed on the vehicle. In our experiments, we use the videos taken by the camera in the middle.

We selected 300 videos of overtaking scenes. For each video, it shows one to nine seconds of scenes. The frame rate is 12 frames per second. Since we focused on the overtaking behavior of vehicles, we extracted more than 15000 images of real traffic scenes. Then we selected 993 images of traffic scenes that contain vehicles.

Authorized licensed use limited to: Florida Polytechnic University. Downloaded on October 20,2023 at 05:17:12 UTC from IEEE Xplore. Restrictions apply.

B. Generating Method

The number of iterations (*epoches*) was 500, and the batch size was 64. That is to say, each time we trained the discriminator, 64 images from training data were put into the nodes. Each time after the discriminator was trained, the generator was trained afterwards. During each iteration loop, the discriminator was trained 15 times.

64 random noises of 100 dimensions were put into another input node to generate 64 generated images with size of 64×64 pixels. Then the 64 generated samples and 64 training samples were trained in the discriminator. According to equation (6) and (7), we got loss of discriminator *dloss* and loss of generator *gloss*. After updating the discriminator, the generator was updated by ADAM. Here we defined α =0.0002; β_1=0.5; β_2=0.9. During each iteration loop, the discriminator was trained 15 times. Thus each image can be ensured discriminated by the discriminator.

C. Experiment Results

1) Results of KITTI dataset

We mainly used images in the folders of drive_0009, drive_0014, drive_0018, drive_0051, drive_0059, drive_0093, and drive_0096. Shown in Fig. 1, these images contain abundant and various items and details, such as pedestrians, vehicles, trees, buildings and shadows. The experiment results are shown in Fig. 2.

The generated images show distinct features of road, trees and some buildings. The generated images are 64×64 pixels. We can promote the resolution by adding iterations and enlarge the image size by changing the structure. We also found out an interesting phenomenon. Generated images using 64 noises of 100 dimensions are so similar to each other. This may caused by the structure of GANs itself.

2) Results of our dataset

The image generating results are shown in Fig. 3. The results show that the DCGAN well learned the features of highway road and the green belts. Besides, it learned some features of vehicles such as cars, and trucks. Compared with roads and green belts, vehicles have more various types and details. Therefore, the generated images can't well show the features of vehicles. Like the results trained by KITTI dataset, these results trained by our dataset are still faced with the problem of insufficient diversity.

The video generating results are shown in Fig. 4. We picked up 12 frames of the generated video. They showed a moving scene in which an object looks like a car appeared gradually. This result is completed after 239 iterations.

VI. Conclusion

In this paper, we discussed the generative adversarial networks (GANs) and deep convolutional generative adversarial networks (DCGAN). When we applied DCGAN to generate traffic scene images and videos, we achieved some efficient results. We are not the first one who generate traffic scene with GANs, but their results had limited difference with the original images, which isn't efficient for creating new realistic traffic scene. Our method achieved different results with original data and still performed realistic scenes with traffic features and details.

In order to generate better traffic scene images and videos, we may try some new methods. For example, using conditional generative adversarial nets (CGAN) [18] to generate abundant and various results. Or using Laplacian pyramid of adversarial networks (LP-GAN) [9] to enlarge the image size.

References

[1] D. Zhao, Y. Liu, C. Zhang, and Y. Li, "Autonomous driving simulation for unmanned vehicles," In IEEE Winter Conference on Applications of Computer Vision (WACV), 2015, pp. 185-190.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, "Generative Adversarial Nets," in Neural Information Processing Systems Conference, 2014, pp. 2672-2680.

[3] A. A. Efros, T. K. Leung, "Texture Synthesis by Non-parametric Sampling," In the Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 2, pp. 1033–1038.

[4] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based Super-resolution," In Journal of Computer Graphics and Applications, 2002, 22(2), pp. 56–65.

[5] J. Hays, and A. A. Efros, "Scene Completion Using Millions of Photographs," In ACM Transactions on Graphics (TOG), 2007, 26(3):4.

[6] J. Portilla, and E. P. Simoncelli, "A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients," In International Journal of Computer Vision, 2000, 40(1), pp. 49–70.

[7] D. P. Kingma and M. Welling, "Auto-encoding Variational Bayes," In arXiv preprint, arXiv:1312.6114, 2013.

[8] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning Using Nonequilibrium Thermodynamics," In arXiv preprint, arXiv:1503.03585, 2015.

[9] E. L. Denton, S. Chintala, and R. Fergus, "Deep Generative Image Models Using A Laplacian Pyramid of Adversarial Networks," In arXiv preprint, arXiv:1506.05751, 2015.

[10] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra, "Draw: A Recurrent Neural Network for Image Generation," In arXiv preprint, arXiv:1502.04623, 2015.

[11] A. Dosovitskiy, J. Springenberg, and T. Brox, "Learning to Generate Chairs with Convolutional Neural Networks," In arXiv preprint, arXiv:1411.5928, 2014.

[12] A. Radford, L. Metz, S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," In arXiv preprint, arXiv:1511.06434, 2015.

[13] M. Arjovsky, S. Chintala, L. Bottou, "Wasserstein GAN," In arXiv preprint, arXiv:1701.07875, 2017.

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," In arXiv preprint, arXiv:1704.00028, 2017.

[15] S. Ioffe, and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," In International Conference on Machine Learning, 2015, pp. 448-456.

[16] D. Kingma, and J. B. Adam, "A Method for Stochastic Optimization," In International Conference on Learning Representations (ICLR), 2015.

[17] http://www.cvlibs.net/datasets/kitti/

[18] M. Mirza, and S. Osindero, "Conditional Generative Adversarial Nets," In arXiv preprint, arXiv:1411.1784, 2014.

Authorized licensed use limited to: Florida Polytechnic University. Downloaded on October 20,2023 at 05:17:12 UTC from IEEE Xplore. Restrictions apply.

(a) sample image in folder drive_0009

(b) sample image in folder drive_0014

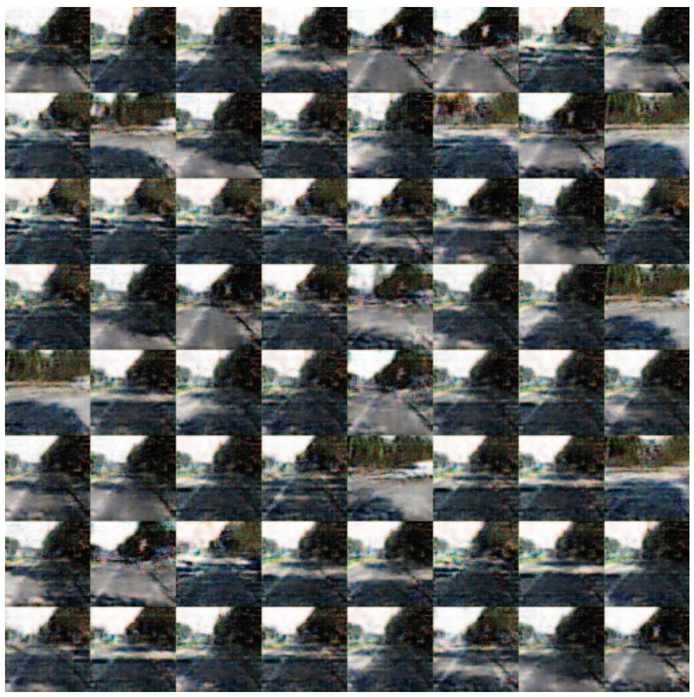
(c) sample image in folder drive_0051

(d) sample image in folder drive_0059

(e) sample image in folder drive_0093

(f) sample image in folder drive_0096

Fig. 1. Sample images from KITTI dataset.


(a) first group of results

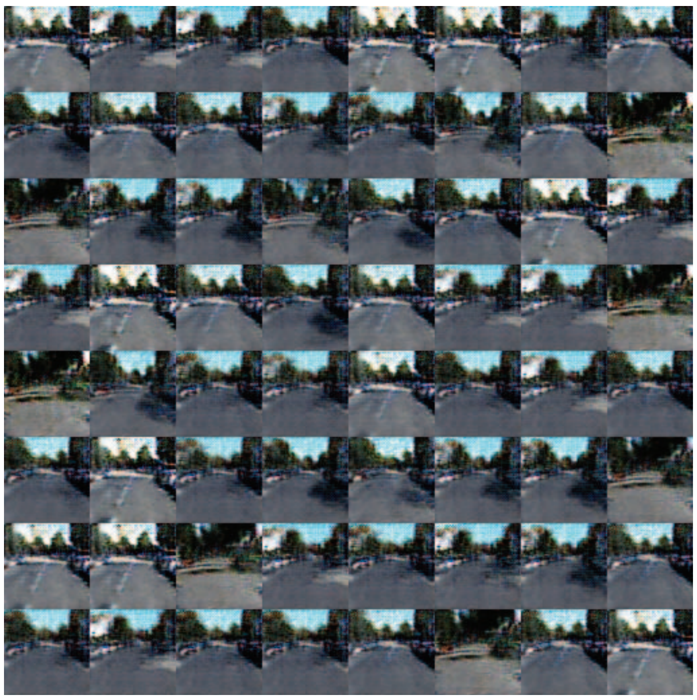
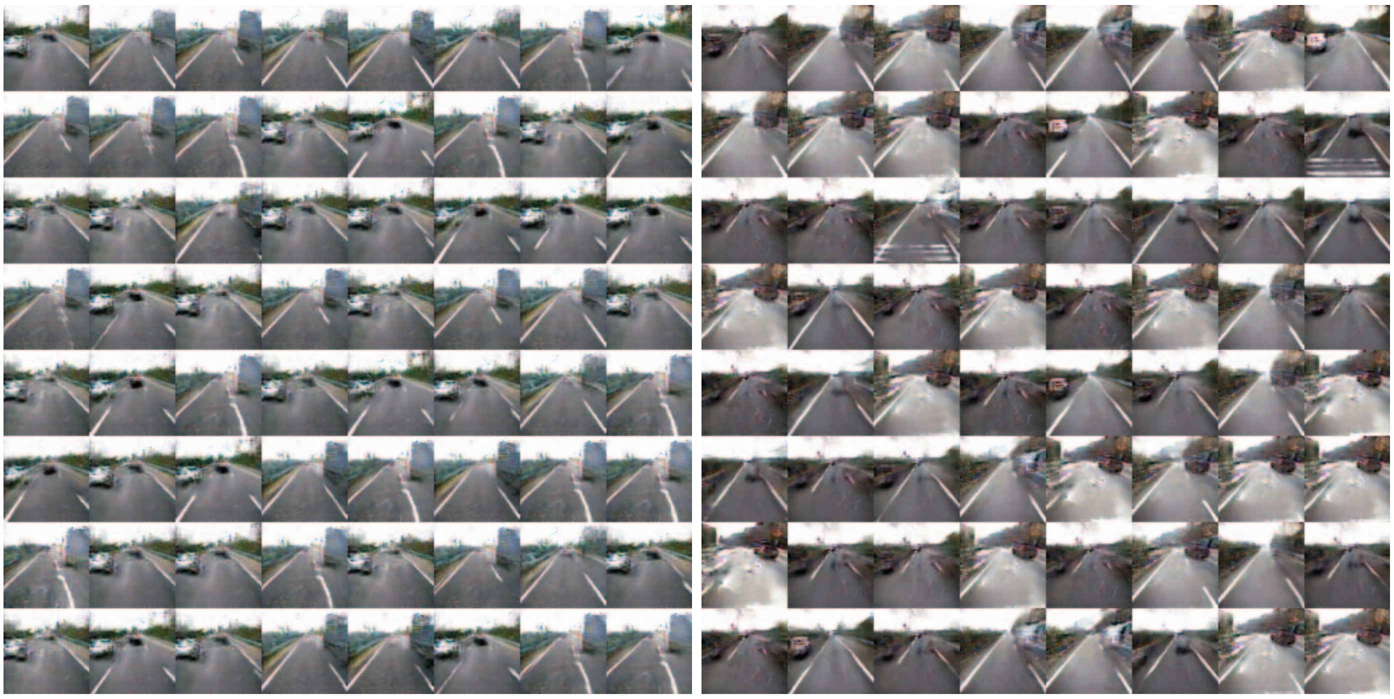

(b) second group of results

Fig. 2. Two groups of generating results trained with KITTI dataset.

Authorized licensed use limited to: Florida Polytechnic University. Downloaded on October 20,2023 at 05:17:12 UTC from IEEE Xplore. Restrictions apply.



(a) first group of results

(b) second group of result

Fig. 3. Two groups of generating results trained with our dataset

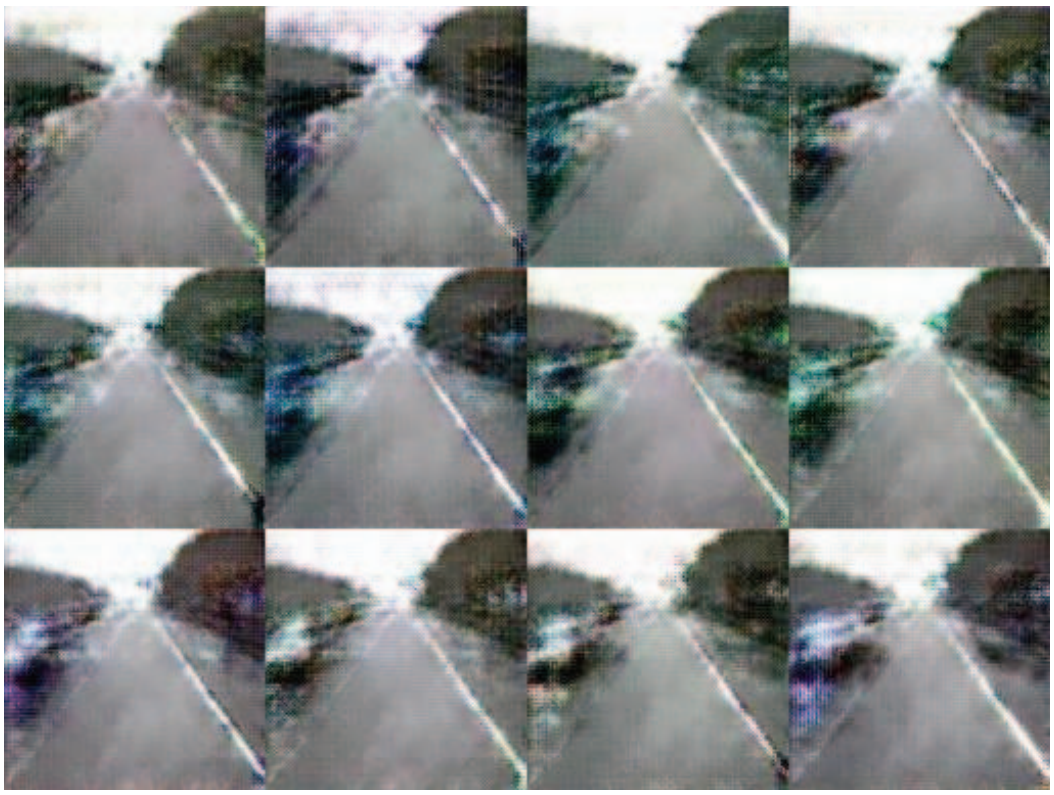


Fig. 4. Generated traffic scene video frames using DCGAN

Authorized licensed use limited to: Florida Polytechnic University. Downloaded on October 20,2023 at 05:17:12 UTC from IEEE Xplore. Restrictions apply.