Sketch-to-Car: Synthesizing Stylized Car Images From Sketches

Shujia Liu
shujia@stanford.edu

Abstract

We investigate the inception module as a building block of conditional generative adversarial network (GAN) in training a sketch-to-car model. We demonstrate that this approach is effective at synthesizing photos from edge maps. Experimental results show that, the use of inception modules significantly accelerates the training process of conditional GAN compared to those using convolutional layers, while maintaining a similar frechet inception distance (FID) score.

1. Introduction

Many problems in image processing, computer graphics, and computer vision can be posed as translating an input image into a corresponding output image. It's about translating one possible representation of a scene into another, given sufficient training data. Our goal in this paper is to train a GAN to synthesize car images from sketches. As a big fan of cars, it would be great if I could take ideas from a few sketches and turn them into compelling car images. Fortunately, GAN allows us to create a smart paintbrush so that if we want to create new car images, we can draw the shapes of cars, and the neural network can then fill in all the texture and shadows, and the colors based on things that is learned from a large database of real world car images. The real advance is that we would be able to synthesize car images with a lot more diversity and more fidelity than we were able to in the past.

In this paper, we explore conditional GANs with inception module. Convolutional neural network (CNN) is the common choice behind a wide variety of image prediction problems, though it has limitations. Due to the huge variation in the location of information, choosing the right kernel size for CNN becomes tough. A larger kernel is preferred for information that is distributed more globally, and a smaller kernel is preferred for information that is distributed more locally. Very deep networks are prone to overfitting. It also hard to pass gradient updates through the entire network, suffering from the vanishing gradient problem. Stacking large convolution operations is computationally expensive and would result in a long training process. To overcome those limitations of CNN, the idea of inception module is to have filters with multiple sizes operate on the same level [13]. This provides a systematic way to tune kernel sizes. Also, the network would get a bit wider rather than deeper, thus mitigates overfitting. And it's relatively less computationally expensive, as its training speed would get benefit from computing matrix operations in parallel.

The input to our GAN discriminator contains two parts: the first part is a 256 * 256 * 3 sketch image, the second part is a $256 * 256 * 3$ image that is either from the dataset, or generated from our GAN generator. The output of the discriminator is a probability value in $[0, 1]$ saying how likely the input image is a real image. The input to our GAN generator is a 256 * 256 * 3 sketch images. And the output of our generator is a $256 * 256 * 3$ synthesized car image. In this paper we use edge maps as sketches of cars.

2. Related work

The U-Net structure is a popular choice in GAN generator [6] [7] [8] . The U-Net is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks. The skip connection provides an alternative path for the gradient with backpropagation. This additional path is often beneficial for the model convergence in training very deep neural networks. However, existing research mainly focuses on using convolutional layers as the building block in U-Net, thus the convolutional layers could be the bottleneck of the GAN in terms of training speed and quality of generated images. The Inception module is a good alternative to the convolutional layer, which is used as the building block in our model.

Conditional GANs are widely used in sketch to image synthesize models [3] [6] [7] [8] [9]. Unlike unconditional GANs, both the generator and discriminator in Conditional GANs observe the input edge map. Several other papers have also used GANs for image-to-image mappings, but only applied the GAN unconditionally, relying on other terms to force the output to be conditioned on the input [11] [17] [18]. We lean to the conditional GANs approach as it results in a simpler setup in terms of loss functions and training procedure.

The type of objects in the dataset is critical in training GANs. Some papers choose to use datasets with landscape images or landscape art images because those tend to yield great results [7] [10]. Landscape images usually have less restrictions in terms of their components: mountains and rivers tend to have flexible shapes. And landscape arts usually include blur effects without clear boundaries between objects in images, which tend to be easily generated by GANs. Several other papers use photographic faces as datasets [16] [5] [3]. The strengths of these models are their abilities to capture the structure of human faces (two eyebrows, two eyes, two ears, one nose, one mouth). Though these models tend to focus on generating faces looking from the front, it would be challenging for these models to synthesize high quality images of faces looking from sides.

There are several metrics to use in terms of evaluating synthesized images from GANs. Inception score is a common metric to evaluate the quality of generated images [12]. It uses a pre-trained neural network to capture the desirable properties of generated images: highly classifiable and diverse with respect to class labels. It measures the average KL divergence between the conditional label distribution of samples and the marginal distribution obtained from all the samples. Mode score addresses an important drawback of the Inception score which is ignoring the prior distribution of the ground truth labels [2]. Frechet Inception Distance embeds a set of generated samples into a feature space given by a specific layer of Inception Net [5]. Viewing the embedding layer as a continuous multivariate Gaussian, the mean and covariance are estimated for both the generated data and the real data. The Frechet distance between these two Gaussians is then used to quantify the quality of generated images. FID is used in our paper because it captures the similarity of generated images to real ones better than the Inception Score, and it's more sensitive to distortions compared to Mode Score, which is what we want in terms of measuring synthesized images [1].

3. Methods

A conditional GAN with inception module is trained to accomplish this task of translating sketches into synthesized car images. The generator G is trained to produce outputs that cannot be distinguished from real images by discriminator D, which is trained to detect the generator's fake images.

3.1. Objective

The objective of the conditional GAN in our paper can be expressed as

$$L_{cGAN}(G,D) = E_{x,y}logD(x,y)+E_{x,z}log(1-D(x,G(x,z)))$$

where generator G tries to minimize this objective against generator D that tries to maximize it.

$$G^* = arg\min_G \max_D L_{cGAN}(G,D)$$

3.2. Network architectures

The proposed GAN includes two parts:

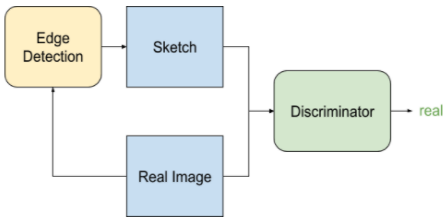

Figure 1: Illustration of the proposed GAN discriminator.

The discriminator learns to distinguish the generator's fake data from real data. The input of discriminator contains two parts: the first part is a $256*256*3$ sketch image, the second part is a $256*256*3$ image that is either from the dataset, or synthesized from our GAN generator. The output of the discriminator is a probability value in $[0,1]$ saying how likely the input image is a real image. When training the discriminator with real data, we put the real image into our training pipeline, generates a $256*256*3$ sketch image with the edge detection component, then combine the sketch image with the real image as input and put that into the discriminator.

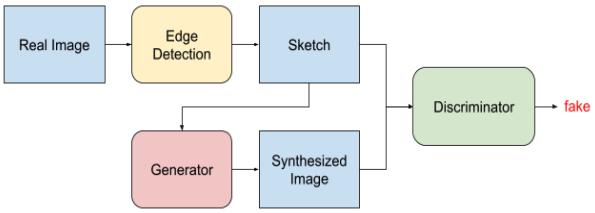

Figure 2: Illustration of the proposed GAN generator.

The generator learns to generate plausible data. The generated instances become negative training examples for the discriminator. The input of the generator is a $256*256*3$ image containing sketches. The output of the generator is a $256*256*3$ synthesized car image. When training the discriminator and generator with synthesized image, we put a real image into the data pipeline, generate a $256*256*3$ sketch image with the edge detection component, then pass the sketch image to the generator to generate a synthesized image, then combine the sketch and the synthesized image as input to the discriminator.

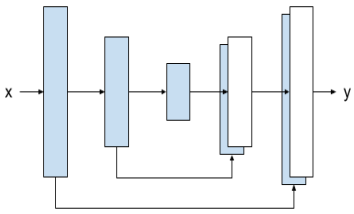

Figure 3: Illustration of U-Net encoder-decoder.

Inside the generator, we use the U-Net architecture, which is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks, as shown in Figure 3. The choice of encoder-decoder architecture is based on the observation that the input and output of our generator differ in surface appearance, but both are rendering the same underlying structure. Therefore, structure in the input is roughly aligned with structure in the output. The skip connections are added to enable the generator to shuttle the shared low-level information, such as the location of prominent edges, directly across the model.

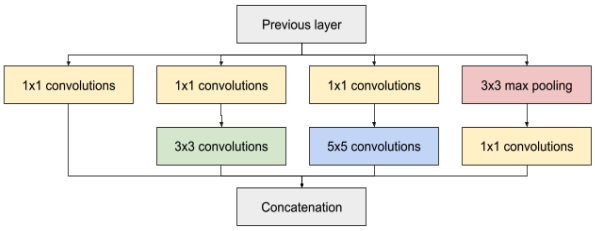

Figure 4: Illustration of the inception module.

Inside the U-Net, we use the inception module as the fundamental building blocks for each layer. An inception module has filters with multiple sizes operating on the same level. With the inception module, the network essentially gets a bit wider. It performs convolution on an input, with 3 different sizes of filters (1x1, 3x3, 5x5). Additionally, max pooling is also performed. The outputs are concatenated and sent to the next layer. On top of that, we can limit the number of input channels by adding an extra 1x1 convolution before the 3x3 and 5x5 convolutions. The Inception module is used to allow for more efficient computation and deeper networks through a dimensionality reduction with stacked convolutions. It's designed to solve the problem of computational expense, as well as overfitting.

4. Dataset and features

We use two existing datasets of car images to train our model. One is the CompCars Dataset [15], including images from web-nature and surveillance-nature. The web-nature data contains 163 car makers with 1,716 car models, for a total of 136,726 images capturing the entire cars and 27,618 images capturing the car parts. The surveillance-nature data contains 50,000 car images captured in the front view. The second dataset comes from Kaggle, including 64,467 images. The combination of these two dataset gives us a total of 278,811 images.

We split the dataset into three parts: a training dataset with 276,811 images, a validation dataset with 1,000 images, and a test dataset with 1,000 images.

Figure 5: Illustration of data samples.

4.1. Preprocessing

To accelerate the training process, we compute the edge maps of cars separately and store them in a local directory. Edge maps are generated by finding discontinuities in brightness of images.

One data augmentation technique we used before training our model is to do random jitter. That is to resize input images (both the sketch image and the observed image) to $320 * 320 * 3$ with zero paddings, that is to fill black pixels into images. Then apply random cropping in the same position of the sketch image and the observed image to reduce the image size to $256 * 256 * 3$. After that, we do a random horizontal flipping on those images.

We also do data normalization on images. That is to transform the input into a distribution with a mean of 0, and a standard deviation of 1.

In the testing pipeline, we resize images to $256 * 256 * 3$ with zero paddings, then apply data normalization to transform the input into a standard normal distribution, before feeding images into the model.

5. Experiments

We use pix2pix, a popular image-to-image translation model [6] as the baseline to compare results from our model.

5.1. Hyperparameters

To optimize our model, we alternate between one gradient descent step on D, then one step on G. As suggested in the original GAN paper [4], rather than training G to minimize $log(1 - D(x, G(x, z)))$, we instead train to maximize $logD(x, G(x, z)$. We use minibatch stochastic gradient descent with a learning rate of 2^{-4} and batch size of 32, and apply the Adam optimizer with momentum parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$.

5.2. Evaluation metrics

The primary metrics in evaluation is FID. FID embeds a set of generated samples into a feature space given by a specific layer of inception net. Viewing the embedding layer as a continuous multivariate Gaussian, the mean and covariance are estimated for both the generated data and the real data. The Frechet distance between these two Gaussians is then used to quantify the quality of generated samples.

$$FID(r, g) = ||\mu_r - \mu_g||_2^2 + Tr(\sum_r + \sum_g - 2(\sum_r \sum_g)^{\frac{1}{2}})$$

where $(\mu_r, \sum_r)$ and $(\mu_g, \sum_g)$ are the mean and covariance of the real data and model distributions, respectively. Lower FID means smaller distances between synthetic and real data distributions.

5.3. Analysis of visual results

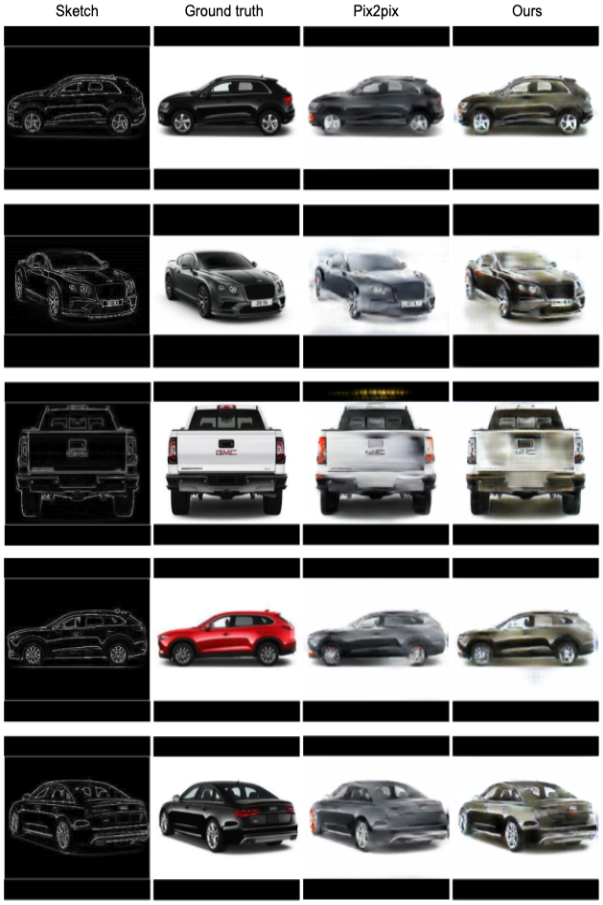

Figure 6: Illustration of synthesized images without background.

Figure 6 shows representative testing results given the same sketches without background as input. Compared to the pix2pix model, our model generates car images with similar quality. One interesting observation is that, the pix2pix model tends to fill the body of a car with gray and blue, while our model tends to fill the body of a car with brown and green. This is due to the difference of initial weights that we randomly assigned to networks.

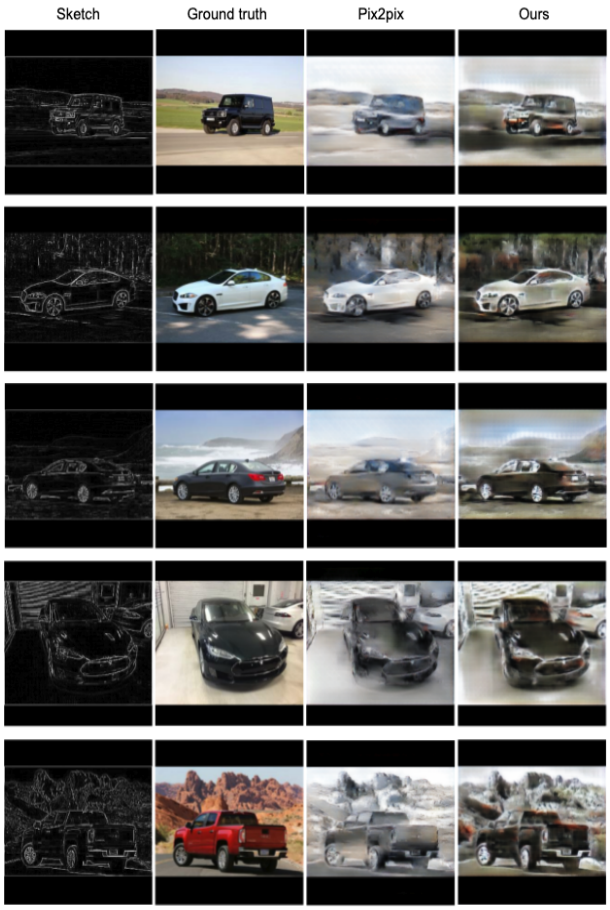

Figure 7: Illustration of synthesized images with background.

As shown in Figure 7, we also train models on car images with background. Generated images from the pix2pix model tend to be blurry between the car and the surroundings, while generated images from our model tend to have clear boundaries between cars and surroundings. Both models could pick up the background information and render that in synthesized images. Also, both models could render car images along with background, while our model produces slightly better quality images with clear boundaries between cars and surroundings.

5.4. Analysis of train processes

We store the discriminator loss and generator loss every 200 batches. For quick reference, one epoch in our training dataset contains 8651 batches. We then plot the losses to compare our model and the pix2pix model.

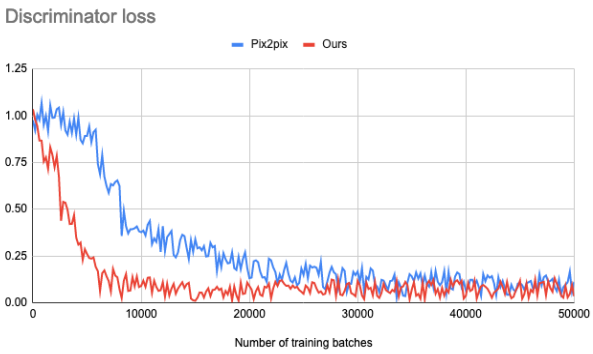

Figure 8: Illustration of discriminator loss in training process.

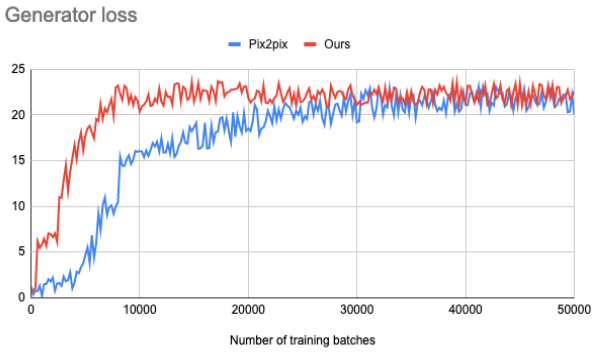

Figure 9: Illustration of generator loss in training process.

From Figure 8 and Figure 9, we can see that our model converges faster in the training process. At the end of the first training epoch (around batch 8,600), our model reaches a stable point for both the discriminator and the generator. The pix2pix model reaches a stable point around 4 epochs (around batch 34,600). The experimental result shows that the inception module outperforms CNN in the training speed of conditional GAN. We expect the discriminator loss would go up a little bit at the very end of training, as the generator learns to synthesize images to fool the discriminator. However, we couldn't observe that from the figures above, as the discriminator loss fluctuates in a certain range and tends to be stable at the very end of the training process.

Dataset	Validation						Test
Training batches	0	10,000	20,000	30,000	40,000	50,000	50,000
Pix2pix	504.52	355.28	254.15	197.68	162.21	160.54	161.53
Ours	505.12	177.25	168.34	163.45	158.91	158.77	158.91

Table 1: FID scores along training models.

From Table 1 we can see that our model converges faster in the training process, which matches the observation we get from Figure 8 and Figure 9. We initialize weights with random values from a distribution with a mean of 0 and a deviation of 0.02, which explains the initial FID scores for pix2pix and our model when the training starts. The FID scores in the test set is close to the FID scores in the validation set after training with 50,000 batches for the pix2pix model and our model, which indicates that the pix2pix model and our model don't get overfitting in this experiment.

5.5. Analysis of challenging test cases

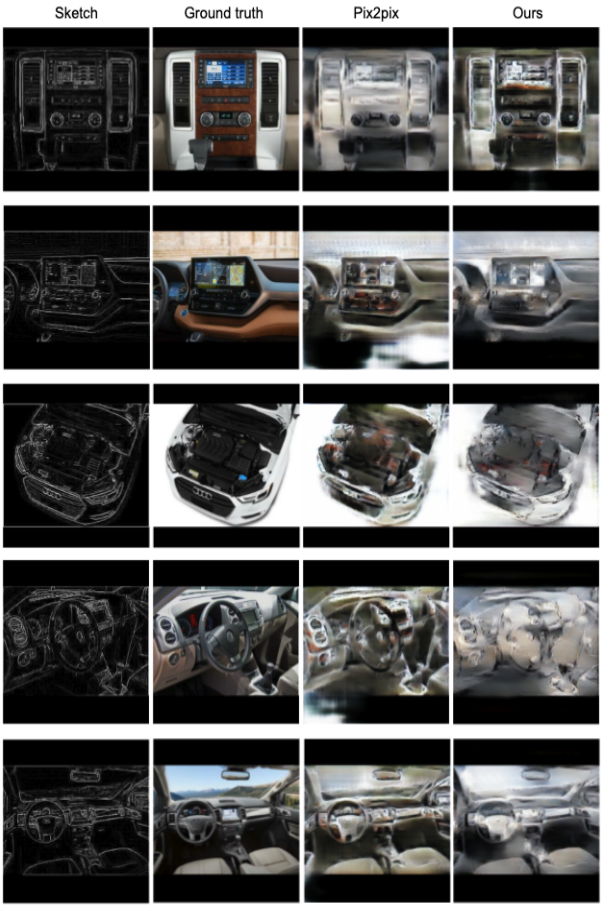

Figure 10: Illustration of challenging test cases.

Some interesting cases in our test set show that the pix2pix model and our model don't generalize well in some cases. We have sketches that draw car stereos from the interior, we also have test cases that draw the things under the hood of cars, and test cases that draw the steering wheel of cars, neither the pix2pix model nor our model performs well on those cases. One reason is that the training set has more car images taken from exterior, while having only a small amount of images taken from interior, thus the models tend to learn very well on drawing cars from exterior, but not very good at drawing cars from interior. The second reason is that, the models tend to fill details in paint brush style, so when the input sketch image contains many drawing lines, the models tend to align large areas of pixels with each drawing line, thus generating blurry images.

6. Conclusion and future work

The results in this paper suggest that using the inception module is a promising approach in terms of accelerating the training process of conditional GAN.

Future work would be to improve the quality of synthesized car images. One idea we could try is to apply Fourier transformation on input images [14]. In terms of building applications, future work could be to develop a web service to integrate the sketch-to-car model, so that it could take user sketches from web browsers as input, and return synthesized images to users in real time. Also, we could extend the dataset from 2D to 3D, and those generated outputs could be printed out with 3D printers as stunning car models.

7. Contributions and acknowledgements

Shujia Liu designed, trained, evaluated the sketch-to-car model (conditional GAN with inception module), and wrote this paper. All code in this paper could be found in https://github.com/sugia/sketch-to-car.

References

[1] A. Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[2] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

[3] S.-Y. Chen, W. Su, L. Gao, S. Xia, and H. Fu. Deepfacedrawing: Deep generation of face images from sketches. *ACM Transactions on Graphics (TOG)*, 39(4):72–1, 2020.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[5] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

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

[7] B. Liu, K. Song, Y. Zhu, and A. Elgammal. Sketch-to-art: Synthesizing stylized art images from sketches. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[8] F. Liu, X. Deng, Y.-K. Lai, Y.-J. Liu, C. Ma, and H. Wang. Sketchgan: Joint sketch completion and recognition with generative adversarial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5830–5839, 2019.

[9] Y. Lu, S. Wu, Y.-W. Tai, and C.-K. Tang. Image generation from sketch constraint using contextual gan. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 205–220, 2018.

[10] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[11] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[12] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[14] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

[15] L. Yang, P. Luo, C. Change Loy, and X. Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3973–3981, 2015.

[16] J. Zhao, X. Xie, L. Wang, M. Cao, and M. Zhang. Generating photographic faces from the sketch guided by attribute using gan. *IEEE Access*, 7:23844–23851, 2019.

[17] Y. Zhou and T. L. Berg. Learning temporal transformations from time-lapse videos. In *European conference on computer vision*, pages 262–277. Springer, 2016.

[18] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European conference on computer vision*, pages 597–613. Springer, 2016.