Synthetic Data Generation of Cars.

Click for Github code: Synthetic Data Generation

Georgios Sartzetakis
s151005@student.dtu.dk
Technical University of Denmark
Copenhagen, Denmark

Georgios Kostantinos Zefkilis
s151074@student.dtu.dk
Technical University of Denmark
Copenhagen, Denmark

Daniel Alejandro Campos
s180014@student.dtu.dk
Technical University of Denmark
Copenhagen, Denmark

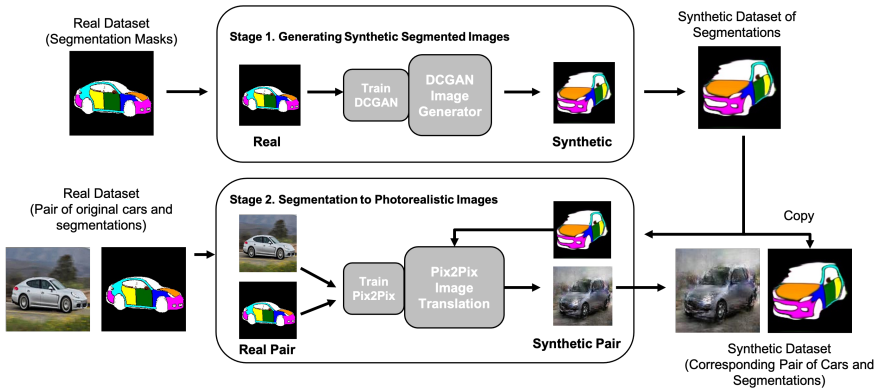

Figure 1: Project workflow.

ABSTRACT

This project aims to create photo-realistic synthetic car images using generative adversarial networks(GANs). We first train a deep convolutional GAN (DCGAN) to develop new car geometries and then use an image translation model (Pix2PixHD) to convert the geometries we created from the first step to photo-realistic car images. The first step yielded car geometry images with FID score of 160, where the benchmark is 109, and the second stage created photo-realistic car images with FID score of 227 where the benchmark is 77. Those results refer to cars' orientation A (see figure 2) due to limitations in using the whole dataset.

KEYWORDS

Synthetic data, DCGAN, image translation, Pix2PixHD, FID

1 INTRODUCTION

Generative modelling is an unsupervised learning task in machine learning that automatic discovers and learns the regularities or patterns in input data so that the model can be used to generate new examples that plausibly could have been drawn from the original dataset. For this use case, we need to increase the size of a car dataset that could potentially be used in the insurance industry to identify damages in segmented car parts.

The project's scope is to build a generative adversarial network (GAN) [2] that synthesizes photorealistic car images. We aim to increase the current dataset's size and move onwards, balance the dataset in terms of the number of doors and car orientations. The GAN model captures the training data's distribution to generate new data from that same distribution. The benefits of using GANs models are broad such as fulfilling requirements or privacy policies, ensuring security and confidentiality.

The project's idea was to divide the car dataset into four big classes corresponding to four orientations as illustrated in figure 2 to increase the representation accuracy of the synthetic images.

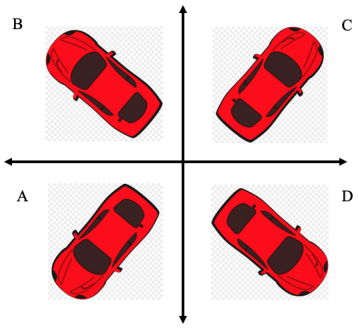

Figure 2: Car orientations

Each orientation covers the entire angle range for its class (e.g. orientation C: 0-90 degrees). For the present study, we used orientation A for training. Once the final model architectures were defined, we tested the data set for all four orientations.The results that are presented in the study refers to orientation A and a discussion on the entire dataset is made in section 5.

2 DATA

The original data set comprises around 3000 car images and their respective segmented car parts images, as shown in figure 3. The dataset contains augmented (flip, rotation, zoom) car images and cars with three and five doors.

Figure 3: Data set Sample

To build the initial model, we used only un-augmented car images with five doors. The photos' original size was 256x256, but we resized them to 128x128 to assist in the modelling procedure. We split the project into two stages. In the first stage, we need to create new car geometries. Therefore, it was necessary to perform image manipulation on the current data and create a new data set to use for building the GAN model. Figure 4 outlines the steps for making the final dataset. Initially, we used an instance segmentation model [12] to extract the whole car's white mask(second image). Then we thresholded the original segmented image to isolate only the segmented car parts (third image). Finally, we overlayed the white masks with the segmented car parts to create the final dataset (fourth image). The final dataset consists of 208 images.

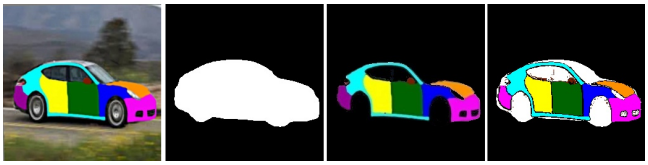

Figure 4: Data set Analysis

3 METHODS

To generate synthetic data, we split the problem statement in two parts [3] as is shown in Figure 1.

- Stage-I GAN: Generating synthetic segmented images by using a DCGAN model.
- Stage-II GAN: Translate the masks produced in stage-I from segmentation to photo realistic images by using a Pix2Pix model.

The stage-I GAN takes as an input the segmentation masks, passes them through a DCGAN model and generates synthetic segmentation masks. The stage-II GAN, trains on the pairs of actual cars and segmented parts. Then we generate the synthetic cars by testing the stage-II GAN on the synthetic segmented images from Stage-I.

3.1 Stage-I DCGAN

The DCGAN (Figure 5) architecrure use deep convolutional neural networks for generator and discriminator. Generally, DCGAN outperforms other GAN models due to application of convolutional layers instead of pooling layers. Pooling layers reduce the spatial size of the representation, and although they improve computational efficiency, they result in the loss of important features [3].

The discriminator (D) is a binary classifier built by standard convolutional neural network that takes an input image and returns 0 or 1 that represents whether the image is unrealistic or realistic respectively. The generator (G) is initialized with a random noise vector (z) while D is trained with a small set of real data. Generally speaking, the generator is a deeper neural network, having more convolutional layers and nonlinearities. The noise vector is upsampled and the weights of G are learned through backpropagation, eventually producing data that is classified as real by the discriminator [3].

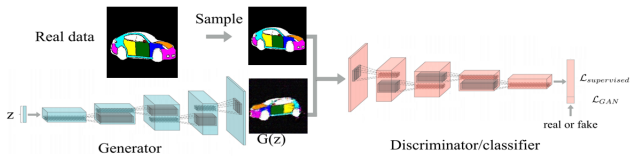

Figure 5: DCGAN Generator and Discriminator

The Stage-I model involves a minimazation-maximization game between the generator and discriminator. The discriminator seeks to maximize the average of the log probability of real images and the log of the inverse probability for fake images. While, at the same time the generator seeks to minimize the log of the inverse probability predicted by the discriminator for fake images. This has the effect of encouraging the generator to generate samples that have a low probability of being fake [1].

The objective minmax loss function used in Stage-I is given by equation 2:

$$\min_G \max_D V(D,G) \tag{1}$$

$$V(D,G) = \mathbb{E}_{x \sim pdata(x)}[logD(x)] + \mathbb{E}_{z \sim P_z(Z)}[log(1 - D(G(z))] \tag{2}$$

At this stage two architectures were used in order to identify which one generates better synthetic data. The first architecture uses transposed deconvolutional layers for upsampling and it was built according to DCGAN pytorch documentation [5] and modified to fit our own data. While in the second architecture the upsampling method changes to nearest neighbor instead in order to obtain synthetic images without artifacts [6]. Further explanation on the architectures is provided in section 4.1.

3.2 Stage-II Pix2PixHD

The purpose of Pix2Pix is to translate the segmented geometries from the first stage to corresponding photorealistic images. The model from [11] proposes a method for synthesizing high- resolution photorealistic images from semantic label maps with conditional GANs. The pix2pixHD framework uses a coarse-to-fine generator, a multi-scale discriminator architecture, and a robust adversarial learning objective function. The **coarse-to-fine** generator in Figure 6 trains the network G1 on lower resolution images. Then, network G2 is appended to G1, and the two networks are trained jointly on high-resolution images. The input to the residual

blocks in G2 is the element-wise sum of the feature map from G2 and the last feature map from G1.

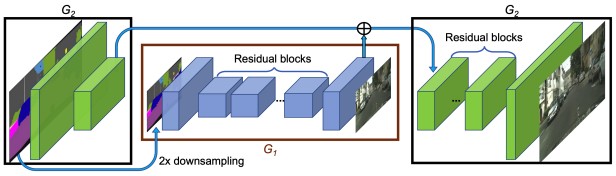

Figure 6: Network architecture of the generator

The **multi scale discriminators** consists of three identical discriminators that operate in three different scales; D1, D2 and D3. D1 is trained on the whole image size, and then D2 and D3 are downsampled by a factor of 2 and 4 respectively in regards to D1. The discriminator that operates at the coarsest scale has the largest receptive field and gives an overall view to the generator to synthesize globally consistent images. The discriminator at the finest scale guides the generator to produce images with finer details. The full objective function of the pix2pixHD is given by:

$$\min_G \left(\left(\max_{D1,D2,D3} \sum_{k=1,2,3} L_{GAN}(G, D_k) \right) + \lambda \sum_{k=1,2,3} L_{FM}(G, D_k) \right) \tag{3}$$

where

$$L_{FM}(G, D_k) = \mathbb{E}_{(s,x)} \sum_{i=1}^{T} \frac{1}{N_i} [||D_k^i(s, x) - D_k^i(s, G(s))||_1] \tag{4}$$

Eq 4 is called the feature matching loss. This loss extracts features from multiple layers of the discriminator and learns to match these intermediate representations from the real and the synthesized image. $\mathbf{s}$ represents the semantic label, $\mathbf{x}$ is the ground truth and $\mathbf{G(s)}$ is the synthesized image. k denotes the discriminator number, T is the total number of layers and N_i denotes the number of elements in each layer. Therefore, Eq 3 is a combination of the GAN loss, which we outlined in the previous section and the feature matching loss over the three discriminators. Parameter λ controls the importance of the two terms.

3.3 Frechet Inception Distance(FID)

The Frechet Inception Distance [4] is a measure that calculates the distance between feature vectors for real and generated images.

$$FID(x, g) = ||\mu_x - \mu_g||_2^2 + Tr\left(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}}\right) \tag{5}$$

The μ_x and μ_g refer to the feature-wise mean of the real and generated images. The Σ_x and Σ_g are the covariance matrix for the real and generated feature vectors. The left part of the equation measures the distance between the means and the right part measures variability between the feature vectors. We need the inceptionV3 model, which is a pre-trained model used for image classification. Firstly, we remove the model's output layer, and then the output is taken as the activations from the last pooling layer. The output layer has 2,048 activations; therefore, each image is predicted as 2,048 activation features. This is the feature vector for the image. For the DCGAN dataset, a good FID score is 106, and for the pix2pixHD gan, a good FID score is 77. To evaluate those numbers, we shuffled the real dataset and then split it into two parts. Then we assessed the FID score between the two parts. We repeated the process ten times, and the final score is the average from the 10.

4 RESULTS

4.1 Stage 1

The purpose of the Stage-I GAN is to generate varied segmentation masks. We built the generator of the DCGAN with two different methods. In the first one, we used deconvolutions-Pytorch ConvTranspose2d- and for the second, we used upsampling -Pytorch Upsample- to increase the size from a latent vector to $128x128x3$.

Deconvolution

In figure 7 the DCGAN with the deconvolution architecture is illustrated.

Generator

Layer	Layer type	Output Size
Input	Latent Vector	100 x 1 x 1
1	Deconvolution+BN+ReLU	1024 x 4 x 4
2	Deconvolution+BN+ReLU	512 x 8 x 8
3	Deconvolution+BN+ReLU	256 x 16 x 16
4	Deconvolution+BN+ReLU	128 x 32 x 32
5	Deconvolution+BN+ReLU	64 x 64 x 64
6	Deconvolution+Tanh	3 x 128 x 128

Discriminator

Layer	Layer type	Output size
Input		3 x 128 x 128
1	Convolution+LeakyReLU	64 x 64 x 64
2	Convolution+BN+LeakyReLU	128 x 32 x 32
3	Convolution+BN+LeakyReLU	256 x 16 x 16
4	Convolution+BN+LeakyReLU	512 x 8 x 8
5	Convolution+BN+LeakyReLU	1024 x 4 x 4
6	Convolution + Sigmoid	1

Figure 7: DCGAN architecture with deconvolution

Figure 8 illustrates the results of the DCGAN with deconvolution. The FID score between the synthesized images and the real dataset is 290. Even though the images resemble the segmentation masks from the real dataset, the downside is that the model creates checkerboard artifacts [8].

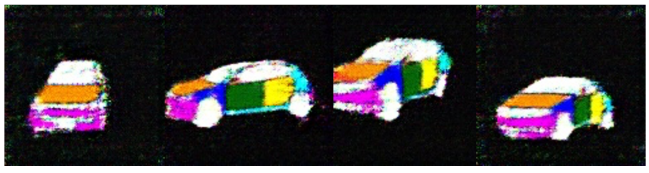

Figure 8: DCGAN results with deconvolution

Figure 9 shows the losses and the FID score during training. The model reaches the lowest score at 550 epochs and then it starts to gradually increase. After 600 epochs the the discriminator loss goes close to zero whilst the generator loss starts to increase. This is a sign that the model is not learning anymore whilst is something that we can see from the FID curve as well.

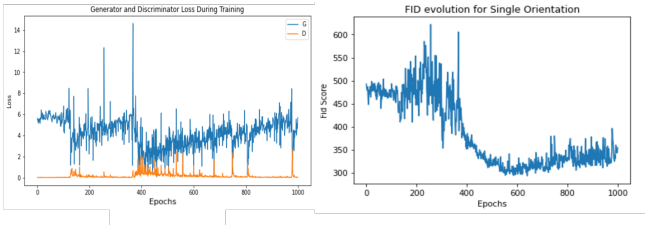

Figure 9: Losses and FID evolution with Deconvolution

Upsampling

Generator

Layer	Layer type	Output Size
Input	Latent Vector+Linear	100 -> 512 x 4 x 4
1	Conv+BN+LR+Conv+BN+LR	512 x 4 x 4
2	Upsample+Conv+BN+LR+Conv+BN+LR	512 x 8 x 8
3	Upsample+Conv+BN+LR+Conv+BN+LR	512 x 16 x 16
4	Upsample+Conv+BN+LR+Conv+BN+LR	256 x 32 x 32
5	Upsample+Conv+BN+LR+Conv+BN+LR	128 x 64 x 64
6	Upsample+Conv+BN+LR+Conv+BN+LR	64 x 128 x 128
7	2x Conv+LR	64 x 128 x 128
8	Conv+Tanh	3 x 128 x 128

Discriminator

Layer	Layer type	Output size
Input		3 x 128 x 128
1	2x Conv + LR	16 x 128 x 128
2	Conv+BN+LR+Conv+BN+LR+AvgPool	32 x 64 x 64
3	Conv+BN+LR+Conv+BN+LR+AvgPool	64 x 32 x 32
4	Conv+BN+LR+Conv+BN+LR+AvgPool	128 x 16 x 16
5	Conv+BN+LR+Conv+BN+LR+AvgPool	256 x 8 x 8
6	Conv+BN+LR+Conv+BN+LR+AvgPool	512 x 4 x 4
7	Conv+BN+LR+Conv+BN+LR+AvgPool	512 x 1 x 1
8	Linear+Sigmoid	1

Figure 10: DCGAN architecture with Upsampling

To aleviate the problem of checkerboard artifacts in the images, we used Upsampling (Nearest Neighbor) in the generator's layers. Figure 10 shows the networks architectures for the generator and discriminator. The architectures are inspired from [6]. We can see from Figure 11 that the synthesized images are smoother and have better quality. The FID score for those images is 160.

Figure 11: DCGAN results with Upsampling

Figure 12 displays the losses and the FID score on the left and right side respectively, over 1400 epochs. We notice whenever the generator loss is spiking, the FID metric is spiking as well.

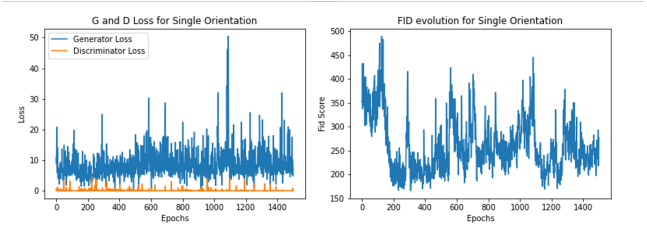

Figure 12: Losses and FID evolution with Upsampling

We can see from Figures 9, 12 that longer training times do not correspond to better image quality. The FID score serves two purposes. Firstly, it references how good the synthesized images are, and secondly, it works as a model selection. That means, that the photos in Figures 11, 8 comes from the model where the weights are optimized for the epoch where the FID curves reach the lowest level.

4.2 Stage 2

We tried two different approaches to train the pix2pix model to derive photo-realistic images on the synthetic data. In the first one, we trained the pix2pix model solely on train data provided by Deloitte(see figure 3 and figure 4. In the second approach, we pre-trained the model using the Stanford car dataset [7]. We then carried on the training on the Deloitte dataset (the same one used in the first approach). The idea behind was to get pre-trained weights to train our data to obtain high-quality images with greater background variety and car shapes.

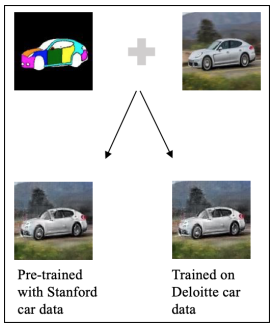

Figure 13: Training results from pix2pixHD

Figure 13, illustrates the outcome of the training process. It seems that the pre-trained model yields the same results with the model without any pre-training.

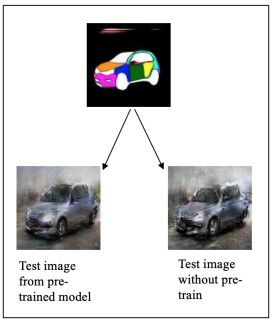

Figure 14: Test results from pix2pixHD

To test the two pi2pix approaches, we used the synthetic data derived from the interpolation architecture because it recorded the lowest FID score. Figure 14 shows a representative example of how the two pix2pix approaches performed on the synthetic data. The outcome looks very much alike in both cases. However, we observe a slightly better representation without the pre-trained model. To get a better idea of which of the two results is better, we calculated both models' FID scores. For the pix2pix FID calculation, we used the official implementation of Fréchet Inception Distance by PyTorch [10]. We depict the results in table 1.

Table 1: FID scores of pix2pix models

Model	*FID*
Test w/o pre-training	227
Test w pre-training	240
Train	77

The FID score for the train operates as a benchmark and indicates the test results' desired/ideal FID score. The two models abstain significantly from the benchmark. It is noteworthy that the without pre-train model scored lower FID score compared to the one with pre-training. This is an unusual case that might be an indicator of overfitting the model trained on Stanford car data. A possible reason that could cause over-fitting is that the Stanford car pairs had different masks than Deloitte's ones. In figure 15 is illustrated a training example from Stanford data during the pre-training. The masks capture the car's overall geometry without specified each car part separately as in Deloitte data.

Figure 15: Train result from Stanford data set

5 DISCUSSION/LIMITATIONS

To achieve the best results in generating synthetic data (Stage-I) and photo-realistic images on those data (Stage-II), we used several approaches to prepare the data, build the model and execute the project workflow.

In Stage-I, the GAN model with the interpolation architecture performed better with FID score 160 against 240 of the deconvolutional architecture. In Stage-II the pix2pix model without pre-training generated better images with FID score 227 against 240 of the pre-trained model. Then the models with the lowest FID scores were chosen as the preferred ones for the present study. Therefore, data with all four orientations were fed into the entire project flow(see figure 1) to see how well the model generalizes in orientations that haven't been exposed before. We observed several limitations in different stages of the project during the training of 4 orientations, which affected the study's outcome.

5.1 Data set

In Deloitte's data set, there was an unequal amount of training examples for four orientations. The imbalanced data could lead the model to confusion, meaning that it will not distinguish accurately among the orientations. Additionally, we observed that there were dissimilar sizes of semantic labels in the data for four orientations. For example, for an image of size 128x128, the semantic label occupied an area of 70x70 while in other cases 100x100. This dissimilarity led to poor representation of synthetic data for four orientations because the model was trying to create a bigger image in a smaller area.

5.2 Stage-I

While training the model with the interpolation architecture in 4 orientations, we observed that the model failed to distinguish the car's front and backside successfully. Figure 16 shows that the model assigns green colour in front of the car instead of orange. Apart from the car's back and front, the model fails to distinguish the 3-door car from the 5-door cars. From figure 16 it is not clear how many door-car the synthetic data are corresponding to.

Figure 16: Example of synthetic data from interpolation architecture on 4 orientations

The above limitations are directly related to the data limitations described in section 5.1.

The Stage-I GAN model revealed instability in the training process when we used all four orientations data. The model was failing to converge with the discriminator reaching zero losses very early in the training process. We also observed mode collapse. The generated synthetic data lacked variability, and therefore the images

were quite similar and in most of the cases identical to each other.

5.3 Stage-II

As described in section 4.2, the pix2pix model on the pre-trained data shows signs of overfitting. However, we cloned the particular model from NVIDIA GitHub repository[11]. Therefore no limitations are expected to be found linked to the model itself. Our case seems to be related to the differences to the car-pairs between the Stanford and Deloitte dataset.

5.4 FID

Although the FID score is the most used metric when comparing data distribution of features extracted between real and synthetic images, it can be deceiving. For example, an FID score of 100 between two different datasets, does not coincide with the same image quality. The FID score is a better measurement when similar data sets are comparing to each other.

6 CONCLUSION

The project's overall purpose was to generate synthetic images and create photo-realistic results from those to enhance the already existing car-data sets. Throughout the project, we successfully generated semantic label images using two different GAN architectures. Additionally, we synthesized photo-realistic images from the semantic labels in Stage-II GAN. However, the model came with a few limitations which affect the outcome of the models. Therefore, it is necessary to take actions to improve the performance of the project. Initially, the data fed into the model should be balanced and exclude data with dissimilar semantic sizes.

To overcome the limitations related to Stage-I process, we can use an Auxiliary Classifier GAN [9]. The ACGAN only has the condition input into the generator, and the discriminator tries to predict the condition using a separate output layer for classification. The classification loss(categorical loss) is also combined in both the discriminator and the generator. For example, the generator wants to make data points that are classified correctly as well, and the discriminator wants to be able to classify the conditions correctly. Thus, we expect by applying the ACGAN the model will generate a better representation of the original images. For a better generation of photo-realistic images, we should give more data to pix2pix model during training. However, this is time consuming and expensive process. Thus, we first need to investigate if the test data will generate better images using the same masks as Stanford data on the pre-trained model. Once, we apply the above recommendations, the next step of the project will be to build a semantic segmentation model, to capture car parts on the synthetic images.

REFERENCES

[1] Jason Brownlee. 2019. *A Gentle Introduction to Generative Adversarial Network Loss Functions.* https://machinelearningmastery.com/generative-adversarial-network-loss-functions/
[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. arXiv:1406.2661 [stat.ML]
[3] John T. Guibas, Tejpal S. Virdi, and Peter S. Li. 2018. Synthetic Medical Images from Dual Generative Adversarial Networks. arXiv:1709.01872 [cs.CV]
[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium. *CoRR* abs/1706.08500 (2017). arXiv:1706.08500 http://arxiv.org/abs/1706.08500
[5] Nathan Inkawhich. 2017. *DCGAN TUTORIAL.* https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
[6] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION. (2018). https://arxiv.org/pdf/1710.10196.pdf
[7] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3D Object Representations for Fine-Grained Categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13).* Sydney, Australia.
[8] Augustus Odena, Vincent Dumoulin, and Chris Olah. 2016. Deconvolution and Checkerboard Artifacts. *Distill* (2016). https://doi.org/10.23915/distill.00003
[9] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional Image Synthesis With Auxiliary Classifier GANs. arXiv:1610.09585 [stat.ML]
[10] Maximilian Seitzer. 2020. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid. Version 0.1.1.
[11] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*
[12] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. https://github.com/facebookresearch/detectron2.