GANs to hallucinate cars from Cityscapes

Arnau Coiduras
Willem Van de Mierop

CITY UNIVERSITY
LONDON

Contents

1 Problem Domain, Motivation and Goals 2

2 Data 2

3 Models 3
- 3.1 Mask R-CNN . . . 3
- 3.2 Generative-Adversarial Network . . . 4
- 3.3 Bigger DCGAN . . . 4
- 3.4 BigGAN . . . 4
- 3.5 Latent Optimisation for Generative Adversarial Networks . . . 5
- 3.6 Evaluation . . . 6
  - 3.6.1 Inception Score . . . 6
  - 3.6.2 Fréchet Inception Distance . . . 6

4 Results 7
- 4.1 DCGAN . . . 7
  - 4.1.1 Baseline DCGAN . . . 7
  - 4.1.2 Bigger DCGAN . . . 7
- 4.2 BigGAN . . . 8
- 4.3 ResNet GANs . . . 10
  - 4.3.1 ResNet18 GAN . . . 10
  - 4.3.2 ResNet34 GAN . . . 11
- 4.4 Latent Optimization for GAN . . . 12
  - 4.4.1 LOGAN . . . 12
  - 4.4.2 Bidirectional LOGAN . . . 12
  - 4.4.3 ResNet18 GAN with Latent Optimisation . . . 13
  - 4.4.4 ResNet18 GAN with Latent Optimisation and Self-Attention . . . 13

5 Conclusions and future work 15

1 Problem Domain, Motivation and Goals

Computer Vision is a multidisciplinary scientific field aimed to mimic the human visual system, that is, to endow machines with the ability to process graphical stimuli. Although it has been an active area of research for more than 50 years, until recently there was no technology available in Computer Graphics to uphold the application in a real-life context of many of the field's most promising findings. Today, different Computer Vision applications are becoming evermore relevant and are expected to disrupt the status-quo of economic sectors of different nature; from self-driving cars in the transport industry to facial recognition software in the security sector.
In this essay we are going to leverage instance segmentation network Mask R-CNN (He, Gkioxari, et al. 2017), pre-trained on Microsoft's COCO (Common Objects in Context) (Lin et al. 2014) data set, to train a Generative-Adversarial Network to "hallucinate" realistic graphic representations of cars. Cityscapes (Cordts et al. 2016) data set will be used as a baseline for us to create a body of data appropriate for our purposes: one of images with cars only.

2 Data

The Cityscapes data set is formed of 5000 street-view images *in the wild* in different European cities, providing pixel-level annotations for up to 30 classes for each of them. Using all these images (train, test and val sets) and their corresponding ground truth masks, we iterate through all car mask instances per image in order to create individual samples (.png files) for each one of them. All pixels outside the corresponding ground truth mask is set to a bright green color in the new car images (RGB[51, 255, 51]). Subsequently, in order to prevent low quality or obscured instances (see figure 2) to be included in our data set, we implement the pre-trained Mask-R-CNN (He, Gkioxari, et al. 2017) network to filter out samples not predicted with a "confidence" of at least 90% by the model. Although this proves to filter out a non-negligible amount of *bad* car image instances, we still find samples in which represented cars are partly blocked or fragmented in the resulting body of data. We decide however not to manually cleanse the data set. Finally, the bounding box predicted by the model is used to crop out all content outside its boundaries. The outcome is a new data set of close images of cars only, of about 6,700 samples.

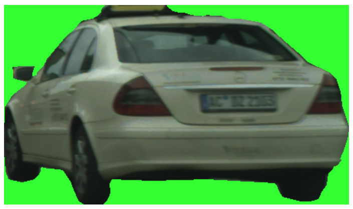


Figure 1: Final data set sample

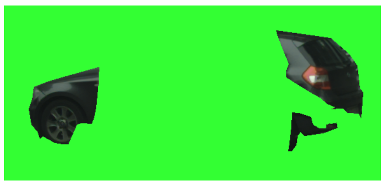


Figure 2: Obscured car

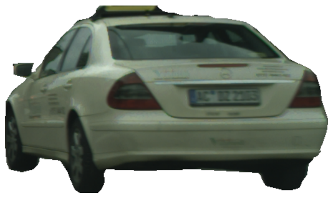


Figure 3: RGBA data set sample

Before producing the above defined data set, several experiments have been carried out with different pre-processing steps that have yielded bodies of data of varying characteristics. As our initial plan was to keep the original positioning of the car within each image so the generative model would also encode spatial relationships, no cropping was implemented at first.

Images were converted to RGBA (red green blue alpha) and all pixels but those corresponding to the car were set to transparent (see Fig. 3). However, due to the meaningless images generated by the model trained on that data set we decided to proceed only with RGB data. Results after training experiments with data sets of uncropped images where the background was set to a specific color were also poor, while keeping content from the original images beyond each sample car would be counterproductive for our purposes. For all those reasons, the final data set is formed of cropped images with colored background (see Fig. 1).

3 Models

As noted in the introduction, different models have been implemented for different purposes. On the one hand, an instance segmentation network, Mask R-CNN, is used to identify on a pixel-level all instances of cars from the Cityscapes data set and filter out low quality samples from the newly created data set. On the other hand, different GAN architectures have been implemented and trained on those images, so the Generator model develops a fine-grained understanding of how cars look, allowing us to create instances of novel yet realistic-looking vehicles. Both models, with their corresponding variations when applicable, will be respectively discussed in detail in the following subsections.

3.1 Mask R-CNN

Mask R-CNN, or Mask Regional-CNN, is a Deep Convolutional Neural Network enabling both efficient object detection and pixel-level instance segmentation in images. As an extension to Faster R-CNN, Mask R-CNN builds on top of its Feature Pyramid Network (FPN) for feature extraction, RoI (region of interest) proposal network (RPN) and classification layers with a fully convolutional network (FCN) that outputs a binary mask indicating the pixels where the object in a bounding box is.

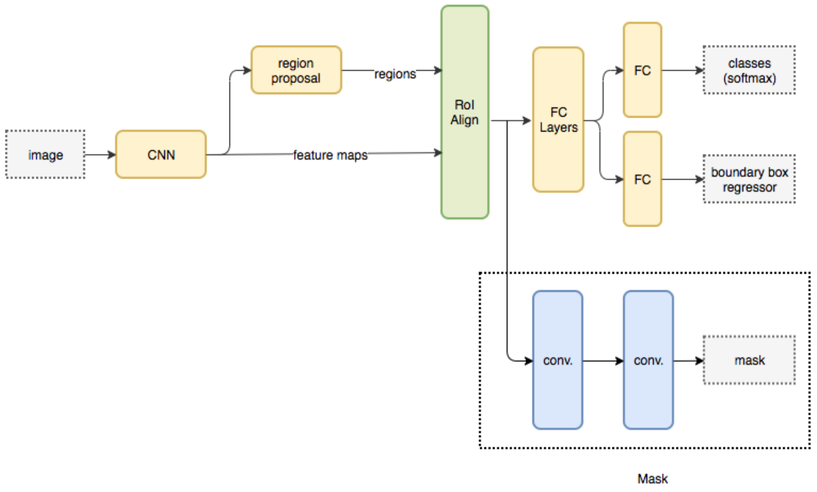


Figure 4: Mask R-CNN architecture. Hui (2018)

3.2 Generative-Adversarial Network

Generative Adversarial Networks consist of two neural networks, a generator (G) and a discriminator (D), competing with each other. Given a training set, the generator will "strive" to produce samples with a statistical distribution similar to those of the data set so that the discriminator will not be able to distinguish real samples from generated ones (Goodfellow et al. 2014). The generator is therefore trained to maximize the probability of D misclassifying synthesised samples, while the discriminator is trained to correctly discriminate real from generated samples. This presents a minimax decision rule between the discriminator and the generator with value function $V(G, D)$ shown in equation 1.

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x \sim p_z(z)}[\log(1 - D(G(z))] \tag{1}$$

DCGANs (Deep Convolutional GAN) are an extension of the original GAN introduced in ibid., where the generator and the discriminator use transposed convolutional and convolutional layers respectively (Radford, Metz, and Chintala 2016). Following a similar structure, both Generator and Discriminator have batch normalization layers after each convolution for improved stability (Ioffe and Szegedy 2015), but G is implemented with ReLU activation function, while D uses Leaky ReLU function. The generator receives an input vector, also called latent vector (z), of arbitrary length, initialized with a standard normal distribution. G then subsequently performs the transposed convolution operations and outputs a tensor of shape [batch_size, n_channels, height, width]. D receives batches of images as input, of the same size as outputted by G, and performs a sequence of feature extraction operations to finally classify each sample as *real* (1) or *generated* (0). Our original baseline DCGAN is based on the PyTorch tutorial on DCGAN's (Inkawhich 2018).

3.3 Bigger DCGAN

This bigger DCGAN is a more complex DCGAN because it has 6 convolutional layers for the Discriminator and 7 transposed convolutional layers. The addition of more convolutional layers might allow the GAN to produce better images. This results in an increase in parameters for the Discriminator from around 2.8 million to 9.1 million and an increase in parameters for the Generator from around 3.5 million to 16 million.

3.4 BigGAN

BigGAN is a generative adversarial framework resulting from an effort to combine a suite of recent best practices in training GANs (Brock, Donahue, and Simonyan 2018). We shortly present below some of its features that we have implemented in our work:

- **Larger batch size**: in Brock, Donahue, and Simonyan (ibid.), the authors argue that GANs benefit dramatically from scaling up training batch size.
- **Orthogonal weight initialization**: weights are initialized as a "random orthogonal matrix" Saxe, McClelland, and Ganguli (2013) to prevent exploding or vanishing gradients and improve training stability.
- **Orthogonal regularization**: weight regularization term implemented so parameters maintain their original (when hence initialized) orthogonal property (Brock, Lim, et al. 2016).

- **Hinge loss**: often used as objective function for classification in Supporting Vector Machine, it was first introduced in H. Zhang et al. (2018) as criterion for generative models and subsequently used in Brock, Donahue, and Simonyan 2018.
- **Self-Attention**: based on H. Zhang et al. (2018), introduces attention maps and combines them with the convolution feature maps, allowing both the Discriminator and Generator to "focus" on different parts of an image. Figure 5 depicts the structure of the Self-Attention module.
- **Spectral normalization**: previously applied only to the Discriminator (Miyato et al. 2018), the authors of H. Zhang et al. (2018) conclude that applying spectral normalization to both Generator and Discriminator Self-Attention modules improves training dynamics.
- **ResNet architecture**: deeper convolutional networks, with skip connections that allow connecting input latent points to specific layers deep in the network (He, X. Zhang, et al. 2015).

Three different experiments with a permutation of the above presented features are performed, in order to analytically decide which ones proof beneficial for our efforts. Their respective implementation is discussed in section 4.

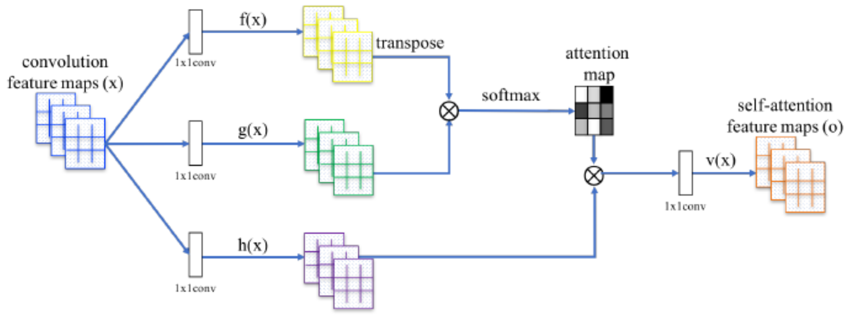


Figure 5: Self-Attention module diagram. H. Zhang et al. (2018)

3.5 Latent Optimisation for Generative Adversarial Networks

GAN training requires a delicate balance between updates to the discriminator and generator. Wu et al. (2019) introduced Latent Optimisation for Generative Adversarial Networks (LOGAN) which applies natural gradient descent to optimise the latent space z during training, following the scheme shown in Figure 6 (a). Figure 6 (b) shows that LOGAN with Natural Gradient Descent performs the best for this reason we are only going to implement Natural Gradient Descent. Latent vector updating is computed using equation 2.

$$\Delta z = \frac{\alpha}{\beta + ||g||^2} * g \tag{2}$$

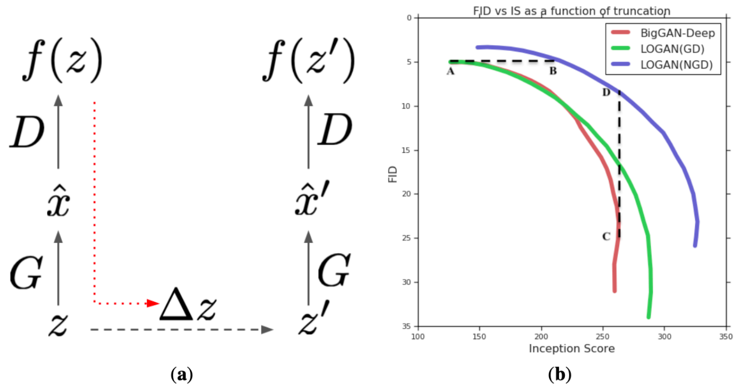


Figure 6: LOGAN (Wu et al. 2019)

Ramkumar (2019) proposes a Bidirectional LOGAN, here the train images are passed through an encoder network that is trained jointly with the generator and the discriminator, the paper shows that this leads to improved ability to capture the space of the true distribution.

3.6 Evaluation

Quantitatively evaluating GANs is not an easy task. Although several methods have been proposed in the last years, there is no consensus as to which measure best assesses the strengths (and weaknesses) of generative adversarial models. In this work, we implement two methods that are currently widely accepted (Borji 2018).

3.6.1 Inception Score

The Inception score (IS) (Salimans et al. 2016), uses a pre-trained neural network (Inception V3 model trained on ImageNet (Deng et al. 2009)) to evaluate both quality and diversity of generated images. To do so, the average (Kullback–Leibler) divergence between class conditional distribution (i.e. model confidence in image instance classification; scores quality) of all generated images and the marginal probability from all samples (scores diversity) classify the generated images. This model calculates the probability of the image belonging to each class is predicted class and these predictions are summarized in the inception score. The inception score has a lowest value of 1.0 and a highest value of 1000 (1000 is the number of classes the inception v3 model can classify).

3.6.2 Fréchet Inception Distance

The Fréchet Inception Distance (FID) (Heusel et al. 2017) also uses the Inception V3 model. However, In this case both image samples are embedded into a specific layer of the model to capture computer-vision-specific features of an input image. These features are calculated for both real and generated images, estimating their mean and covariance. Finally the distance between the two distributions is computed to quantify the quality of generated images. The lower the distance, the better the quality of generated images.

4 Results

4.1 DCGAN

4.1.1 Baseline DCGAN

The initial GAN structure is trained for 3500 epochs with a learning rate of 0.0001 with an Adam optimizer for both the Generator and the Discriminator with a weight decay of 0.001 and a batch size of 256. The FID and IS scores can be observed in Figure 7. This figure shows that our DCGAN is performing the best around 1400 epochs with a FID score of 371.40 and an IS score of 1.050. We can see on figure 8a that some parts of a car are visible on the hallucinated image but there is still room for improvement.

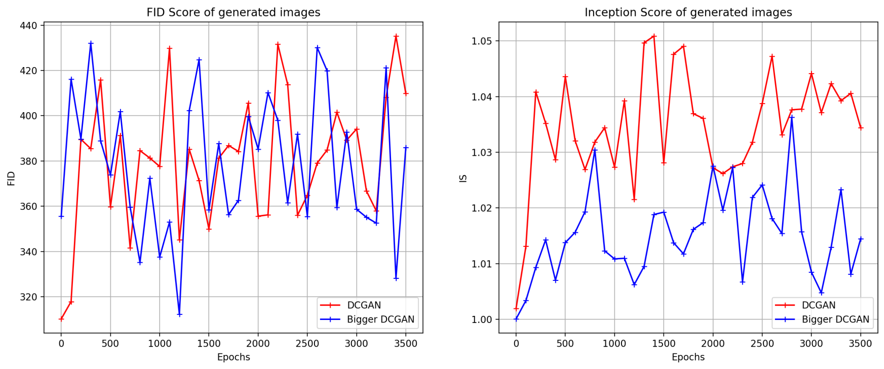
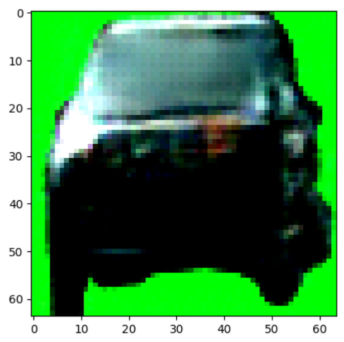


Figure 7: DCGANs

4.1.2 Bigger DCGAN

Bigger DCGAN was trained with the same parameters as the original DCGAN. Figure 7 shows the respective FID and IS scores of both the Bigger DCGAN. Bigger DCGAN reaches an IS score of 1.036 after 2800 epochs and this results in an FID score of 359.41. We can see that the addition of more convolutional layers results in a better FID score but a lower IS score, meaning that the generated images contains less car features (Fig. 8b). This shows the necessity for the implementation of the skip connections to our GAN.


(a) DCGAN 1400 epochs

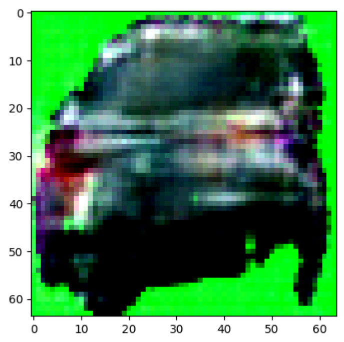

(b) Bigger DCGAN 2800 epochs

Figure 8: Hallucinated images DCGANs

4.2 BigGAN

As previously noted, several experiments with a permutation of the different elements presented in section 3.4 have been performed in order to better understand their respective impact on the model's performance. First, we replace the model's criterion and train the previously described DCGAN with Hinge Loss. All hyper-parameters but batch size, which is increase to 512, are kept the same as in 4.1. After 3000 epochs the model produces only noisy, indistinguishable images (Fig. 9): due to limited time resources we decided not to proceed with its training and focus on other experiments.

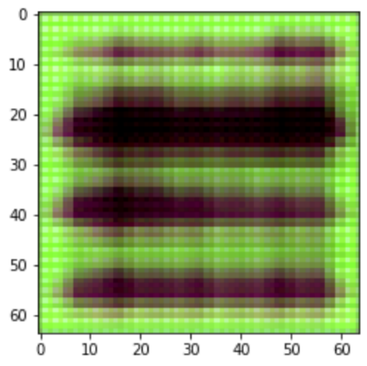
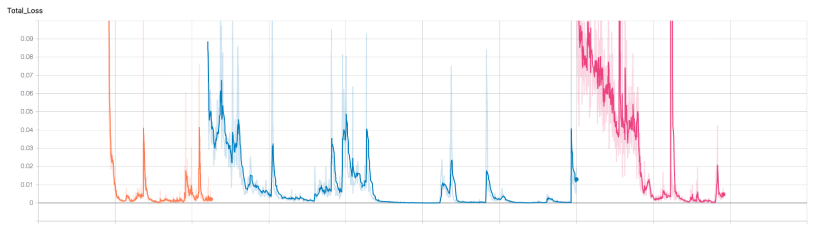
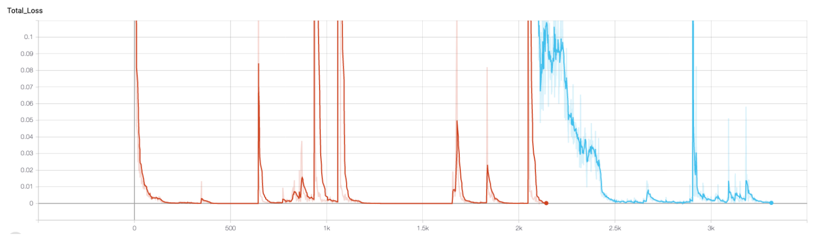


Figure 9: DCGAN with Hinge Loss

Subsequently, we add Self-Attention to the DCGAN architecture, at both G and D networks. Experiments both with and without orthogonal initialization and regularization to D and G parameters are performed. Same set of hyper-parameters used for basic DCGAN training are used, but with batch size of 512. Figure 10 shows the training dynamics of both Self-Attention GAN with (10b) and without (10a) orthogonal parameter initialization and regularization. Different line colors in the graphs indicate different training runs. Although the loss curves here portrayed seem to indicate a slightly more stable behaviour when parameters are contrived to have orthogonal properties, the orthogonally-parametrized model produces noisy images with indistinguishable content for the 3000 epochs it has been trained.



(a) DC Self-Attention GAN



(b) DC Self-Attention GAN with orthogonal parameter initialization and regularization

Figure 10: Images generated with Self-Attention DCGAN

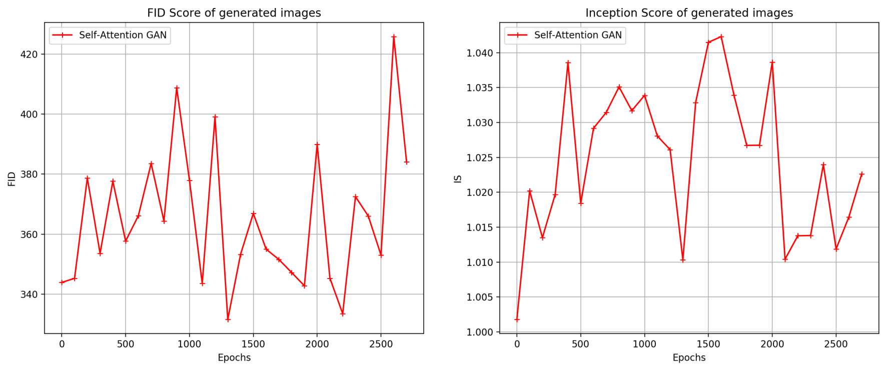


Figure 11: Metrics: Self-Attention GAN

As we can observe in Figure 11, neither FID nor IS scores follow a clear trend throughout training. From the metrics, we could assume that images generated at around epoch 1.6k would better adhere to the data set images' distribution. Although it is hard to tell at bare sight, images generated at later epochs with the Self-Attention model incorporated do not show a substantial improvement with respect to those generated with DCGAN and Bigger DCGAN, as shown in Figure 12 below.

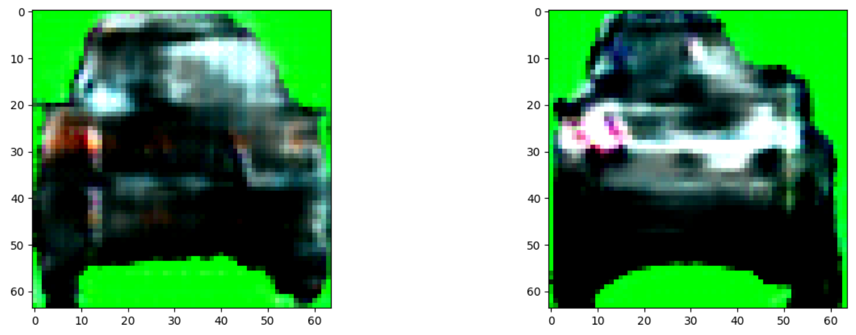

(a) 1600 epochs (b) 2300 epochs

Figure 12: Images generated with Self-Attention DCGAN

Finally, we implement Self-Attention in a generative adversarial model with ResNet architecture, stacking up almost all elements discussed in section 3.4 together. Results are thoroughly discussed in section 4.3. First, we present our experiments with ResNet architectures.

4.3 ResNet GANs

4.3.1 ResNet18 GAN

The ResNet architectures were trained with a batch size of 256 so that better comparison to the original DCGAN was possible. The weights of the layers were initialized with Xavier uniform and the ResNet architecturess were trained both with and without gradient clipping, with a maximum gradient of 1.0.

Figure 13 shows that in this case gradient clipping results in better FID and IS scores of the ResNets. The graphs show that between 2700 and 2950 epochs we should find the best images created by the ResNet GAN. Figure 15a shows that the images produced after 2950 epochs contain more car elements than after 2750 epochs. The FID score after 2950 epochs is 401.72 and the IS score is 1.036. Although the scores of the DCGAN are slightly better, we find that more car features, such as four wheels or license plate, can be more clearly distinguished in the generated images of the ResNet architecture (Fig. 15a).

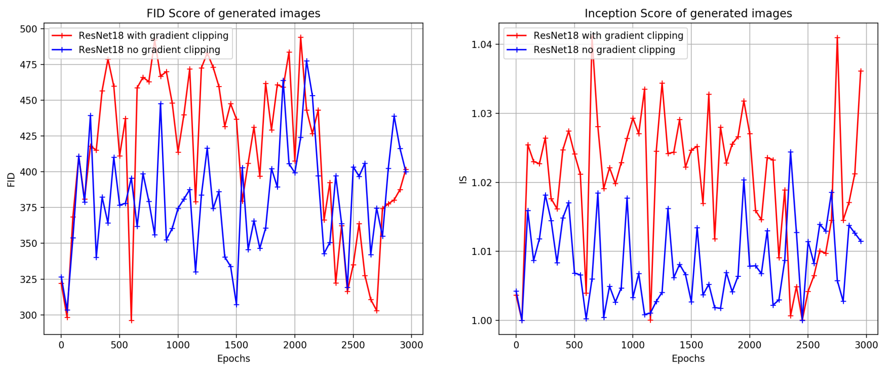


Figure 13: ResNet18 with and without gradient clipping

4.3.2 ResNet34 GAN

This ResNet discriminator has the exact same structure as a ResNet34, meaning layer1 has 3 blocks, layer2 has 4 blocks, layer3 has 6 blocks and layer4 has 4 blocks, resulting in 21.298.918 parameters for the discriminator. The ResNet Generator has a structure similar but not identical, the last layer only has one convolutional transpose layer, this results in 22.958.884 parameters for the generator.

Figure 14 shows that for the ResNet34 the IS score remains low and that the IS peaks after 3400 epochs resulting in a FID score of 378.69 and an IS score of 1.06. However, figure 15b shows that the generated image is meaningless. We believe this can be because of an error with the inception model where it thinks that it is recognizing a class but it is not. This shows that the ResNet34 does not generalize well to our data set.

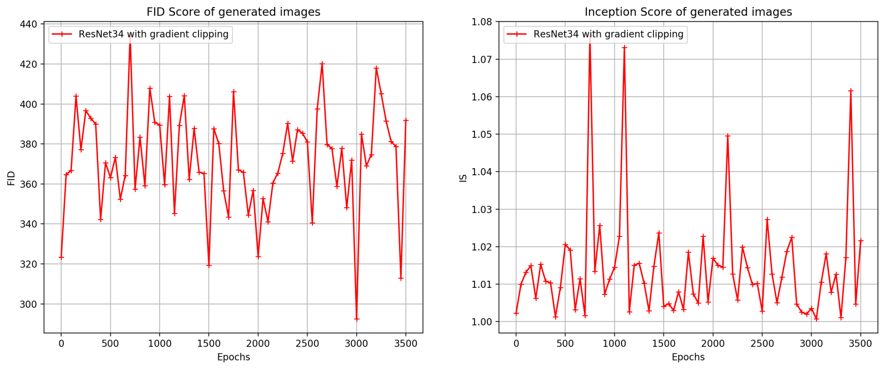
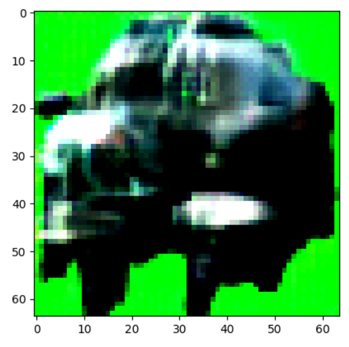
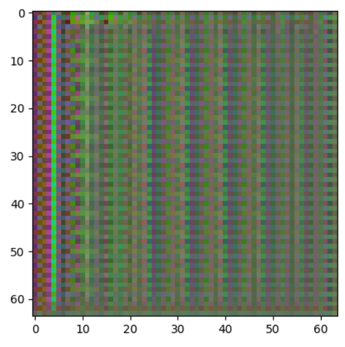


Figure 14: ResNet34



(a) ResNet18 2950 epochs



(b) ResNet34 3400 epochs

Figure 15: Hallucinated images ResNet

4.4 Latent Optimization for GAN

4.4.1 LOGAN

Figure 16 shows the performance of a latent space optimised DCGAN and was trained with the same paramters as our DCGAN and a batch size of 256 to facilitate comparison in performance. Figure 17a shows the hallucinated images of the lowest FID score and the highest IS score respectively. The LOGAN produces an IS score of 1.061 after 2200 epochs and a FID of 397.59. We can see that although the implementation results in a worse FID score we can observe an increase in IS score and a more meaningful hallucinated image (Fig. 17a). By looking at the generated images we observe that the LOGAN starts to overfit after 2500 epochs.

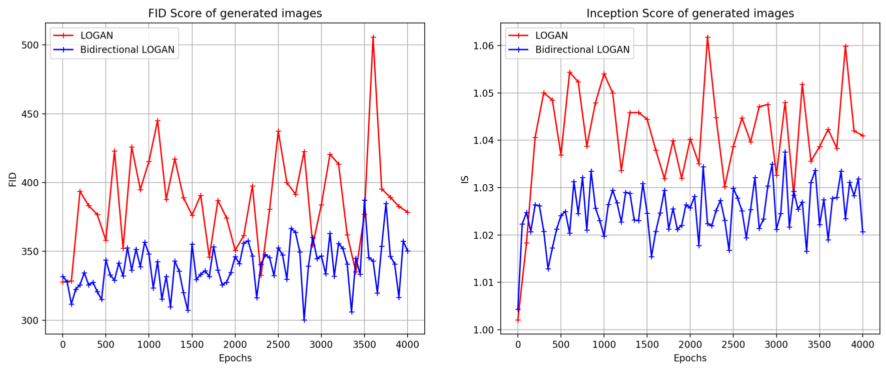
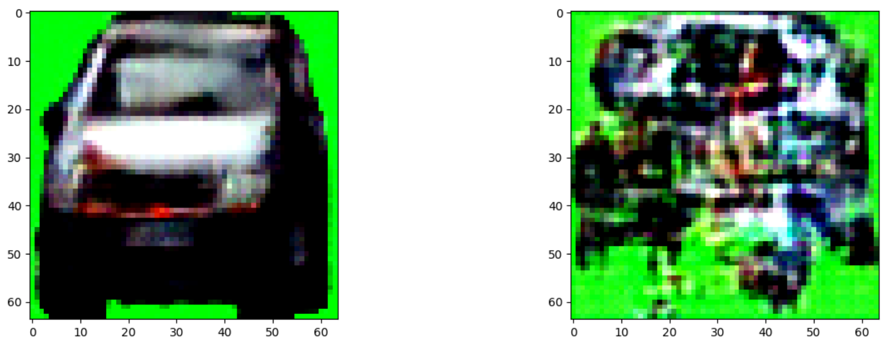


Figure 16: Logan metrics

4.4.2 Bidirectional LOGAN

Figure 16 shows that the Bidirectional LOGAN produces an FID score of 361.14 and an IS score of 1.037 after 3100 epochs. The Bidirectional LOGAN produces a better FID score but as figure 20 shows the Bidirectional LOGAN is not able to produce meaningful images, showing that for our data set LOGAN is more powerful. For this reason we are going to implement LOGAN and not Bidirectional LOGAN in our next models.



(a) LOGAN 2200 epochs

(b) Bidirectional LOGAN 3100 epochs

Figure 17: Hallucinated images LOGAN and LOGAN-B

4.4.3 ResNet18 GAN with Latent Optimisation

This ResNet18 is trained with latent optimisation. Figure 18 and the generated images show that the GAN is performing best around 1650 epochs with a FID score of 375.12 and an IS score of 1.055. This is an FID score improvement of 26.6 and an IS score improvement of 0.019 to our original ResNet18 GAN, showing again that the latent optimisation results in improved performance.

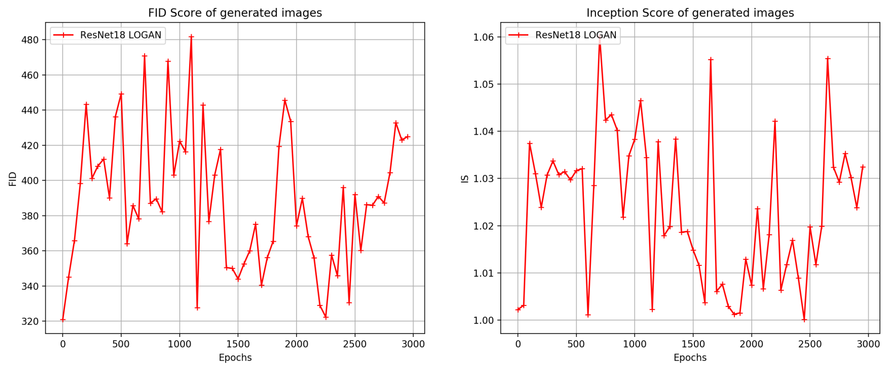


Figure 18: ResNet18 LOGAN

4.4.4 ResNet18 GAN with Latent Optimisation and Self-Attention

This model combines our best performing methods to one powerful model. Due to memory limits this model had a batch size of 350. Figure 19 shows that this model is able to produce an IS score of 1.055 with a FID score of 361.61. These are the best scores we have produces thus far. This can be clearly seen if we compare the hallucinated image of the DCGAN (Fig. 8a) with the hallucinated images of this model (Fig. 19). This model was trained with images of resolution 64*64, according to the BigGan (Salimans et al. 2016) paper a higher resolution will lead to increased performance. Our code has an implementation to train the model with images of a higher resolution. However, due to computational limits and time limits we were unable to train our model with the same batch size and for the same amount of epochs.

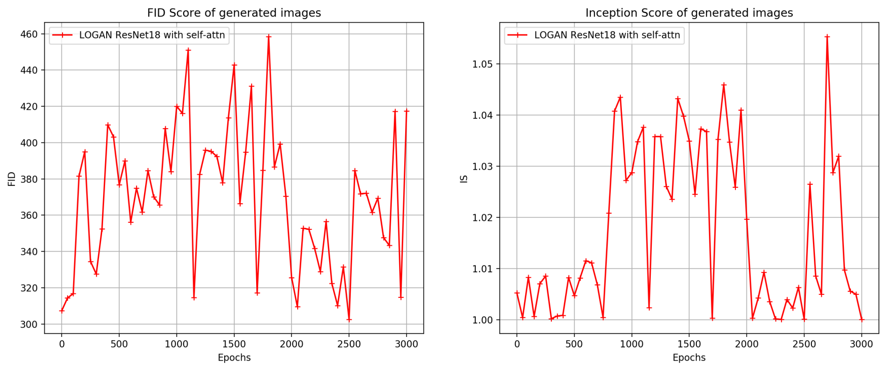


Figure 19: ResNet18 LOGAN with Self-Attention

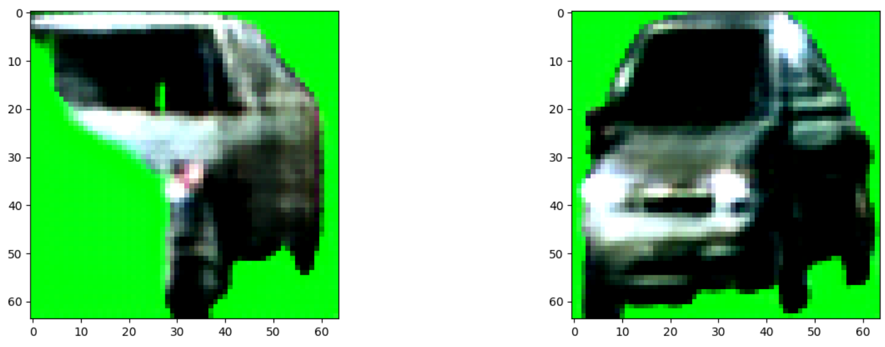


Figure 20: Hallucinated images ResNet18 LOGAN with Self-Attention

5 Conclusions and future work

Model	FID	IS
DCGAN	371.40	1.050
Bigger DCGAN	359.41	1.036
SA-GAN	354.96	1.042
ResNet18	401.72	1.036
ResNet34	378.69	1.060
LOGAN	397.59	1.061
LOGAN-B	363.14	1.037
ResNet18 LOGAN	375.12	1.055
ResNet18 SA-LOGAN	361.61	1.055

In this work we experiment with state of the art generative adversarial models in the field of computer vision, leveraging some of the most advanced network architectures and training techniques. To artificially generate novel, realistic and diverse image representations of an object is not a straightforward task. Cars, the image class we aim to produce, prove to be a challenging object to model due to its fixed structural patterns (i.e. four tires, two driving mirrors, license plates, etc.). While GANs excel at synthesising image classes with few structural constraints (e.g. landscapes), it often fails to produce coherent representations of more complex classes (H. Zhang et al. (2018)).

We have structured our experiments scaling up network complexity, from a basic Deep-Convolutional GAN to BigGAN and Bi-directional LOGAN. We find however that model complexity does not always correlate with better results. Cars can be clearly appreciated in the images generated with most of the models, although it is also obvious to the viewer that they are not real. We argue that the data set *quality* and time constraints are two important factors that have hindered better results. Although image resolution from Cityscapes data set is high, we use images of cars that are not always fully represented, as discussed in section 2. Therefore, we train our models with car instances that are not completely coherent with what a full representation of a car should be. In terms of time invested in the experiments, we acknowledge that state of the art results are often achieved after a number of training iterations of a higher order of magnitude than our average experiments training length. We have renounced to lengthier experiments with single-model grid search in order to be able to try more different approaches. Furthermore, we find that not only training but also quantitatively evaluating GANs is a difficult task. Although both FID and IS do show a consistent relationship with the quality of generated images, best scored images do not always coincide with our qualitative appreciation of best produced samples.

Aside from all of this we were able to increase the quality of our hallucinated images because of our ResNet architecture with self-attention and latent optimisation. As a future work, we would like to train the same models with Stanford's Cars Dataset (Krause et al. (2013)), which contains more than 15.000 different curated images of cars, and experiment with lengthier training experiments and hyper-parameter grid search.

References

Borji, Ali (2018). "Pros and Cons of GAN Evaluation Measures". In: *CoRR* abs/1802.03446. arXiv: 1802.03446. URL: http://arxiv.org/abs/1802.03446.

Brock, Andrew, Jeff Donahue, and Karen Simonyan (2018). "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *CoRR* abs/1809.11096. arXiv: 1809.11096. URL: http://arxiv.org/abs/1809.11096.

Brock, Andrew, Theodore Lim, et al. (2016). "Neural Photo Editing with Introspective Adversarial Networks". In: *CoRR* abs/1609.07093. arXiv: 1609.07093. URL: http://arxiv.org/abs/1609.07093.

Cordts, Marius et al. (2016). "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Deng, J. et al. (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*.

Goodfellow, Ian J. et al. (2014). "Generative Adversarial Networks". In: URL: https://arxiv.org/abs/1406.2661?context=cs.

He, Kaiming, Georgia Gkioxari, et al. (2017). "Mask R-CNN". In: *CoRR* abs/1703.06870. arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

He, Kaiming, Xiangyu Zhang, et al. (2015). "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

Heusel, Martin et al. (2017). "GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium". In: *CoRR* abs/1706.08500. arXiv: 1706.08500. URL: http://arxiv.org/abs/1706.08500.

Hui, Jonathan (2018). *Image Segmentation with Mask R-CNN*. https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272. [Online; accessed 06-May-2020].

Inkawhich, Nathan (2018). *DCGAN Tutorial*. URL: https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html. (accessed: 04.03.2020).

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167. arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

Krause, Jonathan et al. (2013). "3D Object Representations for Fine-Grained Categorization". In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia.

Lin, Tsung-Yi et al. (2014). "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312. arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

Miyato, Takeru et al. (2018). *Spectral Normalization for Generative Adversarial Networks*. arXiv: 1802.05957 [cs.LG].

Radford, Alec, Luke Metz, and Soumith Chintala (2016). "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1511.06434.

Ramkumar, Sharath (2019). *LOGAN-B : Bidirectional Latent Optimized Generative Adversarial Networks*.

Salimans, Tim et al. (2016). "Improved Techniques for Training GANs". In: *CoRR* abs/1606.03498. arXiv: 1606.03498. URL: http://arxiv.org/abs/1606.03498.

Saxe, Andrew M, James L McClelland, and Surya Ganguli (2013). “Exact solutions to the non-linear dynamics of learning in deep linear neural networks”. In: *arXiv preprint arXiv:1312.6120*.

Wu, Yan et al. (2019). *LOGAN: Latent Optimisation for Generative Adversarial Networks*. arXiv: 1912.00953 [cs.LG].

Zhang, Han et al. (2018). *Self-Attention Generative Adversarial Networks*. arXiv: 1805.08318 [stat.ML].