Learning to Generate Chairs with Generative Adversarial Nets

Evgeny Zamyatin

Department of Computer Science ITMO University Saint-Petersburg, PA 199034 ezamyatin3@gmail.com

Andrey Filchenkov

Department of Computer Science ITMO University Saint-Petersburg, PA 199034 aaafil@mail.ru

Abstract

Generative adversarial networks (GANs) has gained tremendous popularity lately due to an ability to reinforce quality of its predictive model with generated objects and the quality of the generative model with and supervised feedback. GANs allow to synthesize images with a high degree of realism. However, the learning process of such models is a very complicated optimization problem and certain limitation for such models were found. It affects the choice of certain layers and nonlinearities when designing architectures. In particular, it does not allow to train convolutional GAN models with fully-connected hidden layers. In our work, we propose a modification of the previously described set of rules, as well as new approaches to designing architectures that will allow us to train more powerful GAN models. We show the effectiveness of our methods on the problem of synthesizing projections of 3D objects with the possibility of interpolation by class and view point.

1 Introduction

Shortly after the great success achieved in tasks of image recognition and object detection by deep convolutional networks [1–3], the problem of image synthesis has been recognized as an important one. Image synthesis finds application in the entertainment industry [4, 5], 3D modeling [6], medicine [7] and many other areas [8, 9].

In 2014, Alexey Dosovitsky et al. [10] suggested a powerful generative model that was able to synthesize accurate images of 3D objects projections given a view point and its class, as well as suggest new images for previously unseen points of view and even interpolate between any two image classes, creating a line of morphed images, changing, for instance, from an armchair to a stool.

In the same year, Ian Goodfellow has introduced the revolutionary work Generative Adversarial Nets [11], in which he proposed a new concept of creating synthesizing networks. Then, in 2015, Alec Radford [12] showed that it is possible to build deep convolutional models using GAN, called DCGAN, which are able to synthesize images with a high degree of realism. He also showed that the model doesn't just remember how certain images should look, but it is trained in general, and is capable of generating realistic images that were not represented in the dataset. But, unfortunately, it is very difficult to train convolutional GAN models, so the authors introduced a list of restrictions on the discriminator and generator architectures, following the rules of which it is possible to successfully train the models.

Following the rules of the list of restrictions from DCGAN, the concept of Generative Adversarial Nets can not be applied to the Dosovikiy generator: it is indicated in the restrictions that the fully-connected hidden layers should be removed wherever possible. But such layers are necessary, so that the model be able to build a powerful internal representation.

In our work, we propose the way to build more powerful GANs that can process images by adopting the best practices from generative and discriminative models. We show that this model is capable to synthesize more accurate images than any other model.

The rest of the paper is organized as follows. Section 2 contains a brief description of neural networks that allow to generate complex objects, especially images. We present the approach in Section 3. In Section 4, we describe experiment setup and datasets we use. Results of these experiments are presented in Section 5. Section 6 concludes the paper.

You can find implementation here¹.

2 Related Work

2.1 Generative models

In paper [10], the authors addressed the problem of synthesizing the projections of 3D objects from a given view point and the class of the object. To solve it, a model of the neural network was developed, the architecture of which can be divided into two blocks:

- 1. Folding the class and view point arguments to the internal feature vector using a block of the fully-connected hidden layers.
- Unfolding internal feature vector to the resulting image using a block of the deconvolutional layers.

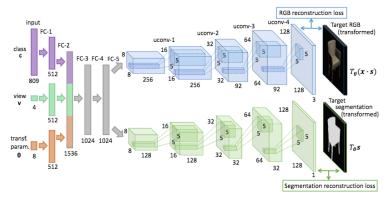


Figure 1: Dosovitskiy's generative model.

The model was trained with the l_2 -loss function. This model solves show good results in the task of generating images of 3D objects and show an ability to interpolate objects by class and angle. But the resulting images from this model are rather blurry.

In this work, the authors has used optical flow [13] algorithms after synthesizing for obtaining more sharp images.

In the papers [14–16], the authors presented models that were effective in the task of transferring the style of different pictures on the photo. Those models differed in that the image was generated not according to a parameter vector, but according to another photo containing required style.

2.2 Normalization for GANs

The training of GANs is a very complicated optimization problem, the result of which depends heavily on the initialization of parameters, the learning rate and other structural parameters. The use of normalization in layers greatly helps in this situation.

Batch normalization [17] is a layer of deep neural network that normalizes the input data based on the statistics computed on the current batch. Thus, if we have a set of random variables distributed

¹https://github.com/EvgenyZamyatin/chair-gan-code

in accordance with the normal distribution as the input, then at the output we get a set of random variables distributed around zero with unit variance. Then we can shift and multiply the obtained distribution by the necessary quantities.

Instance normalization [16] is a normalization layer that is almost similar to the batch normalization, except that the statistic for normalization is evaluated in each instance of batch independently. This normalization is usually used when working with images. It showed high efficiency on the problem of style transfer.

Weight normalization [18] is a normalization layer, which, as well as the previous one, was inspired by the batch normalization. Unlike the two previous types of normalization, this one does not use statistics counting at all. Instead, normalization occurs through the theoretical calculation of the mean and variance.

$$\bar{x} \sim N(\mu_x, \sigma_x^2)$$

$$\bar{y} = \bar{x} \cdot W^T$$

$$\bar{y} \sim N(\mu_y, \sigma_y^2)$$

$$\mu_{y,i} = \sum_j W_{i,j} \cdot \mu_x, \sigma_{y,i}^2 = \sum_j W_{i,j}^2 \cdot \sigma_x^2$$

$$\frac{\bar{y}}{\sum_j W_{i,j}} \sim N(\mu_x, \sigma_y^2)$$

$$\frac{\bar{y}}{\sqrt{\sum_j W_{i,j}^2}} \sim N(\mu_y, \sigma_x^2)$$

Thus, we can normalize the data without using the statistics of the batch.

2.3 Generative Adversarial Nets

Generative Adversarial Networks is a new concept for training synthesizing models proposed by Ian Goodfellow et al. [11]. This concept involve two functions: $G:Z\to X'$ representing a generator and $D:(X\cup X')\to S$ representing a discriminator, where X denotes a set of elements from real distribution (that is dataset), X' denotes a set of elements produced by the generator, Z denotes some generation seed, usually random noise, and S is a real number from S to 1 corresponding to whether an argument is received from S or from S or from S or from S or from the noise S one network, which corresponds to S or from to emulate the distribution of S from the noise S. The second network, which corresponds to S or from the noise S or from the

Formally, we have two functions: $G(z,\theta_g)$ that maps the random noise z to a data distribution and which is parametrized by θ_g , and $D(x,\theta_d)$ that maps a data distribution in the interval [0..1] and which is parameterized by θ_d . G is trained to minimize $\log(1-D(G(z)))$, and D is trained to maximize $\log(D(X))$. Thus, they play a min-max game with a value-function:

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

Deep Convolutional GANs (DCGANs) are a family of architectures, described in paper [12]. Its authors introduced a list of limitations on the architecture of the discriminator and the generator and showed that such models were effective in the task of image synthesis. DCGANs are capable to produce highly realistic images. The restrictions imposed on the model consist of the following items:

- Replace each pooling layer with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- 2. Use batch normalization in both the generator and the discriminator.
- 3. Remove fully-connected hidden layers for deeper architectures.
- 4. Use ReLU activation in generator for all layers except for the output, which uses tanh.
- 5. Use LeakyReLU activation in the discriminator for all layers.

The authors of paper [19] proposed a method, by which the output of the generating network can be parameterized. Recall that previous models were trained in the unsupervised style from random noise z. In contrast, the authors propose to use a generator and a discriminator of the following form: D = D(x|y) and G = G(z|y). Thus, the discriminator says whether x and y are consistent, and the generator is trying to issue the appropriate x for z and y.

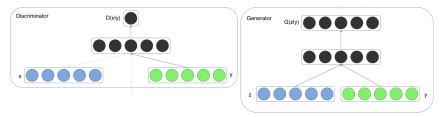


Figure 2: Conditional generative adversarial nets.

Various other models exist based on DCGAN architecture [20–25]. In the [26–28] authors offered new methods for more stable learning of GAN models. In some works the authors use optical flow algorithms [13] to obtain more realistic images.

3 Approach and Model Architecture

In this section, we describe the model we suggest, namely the generator and discriminator, as well as the learning methods. There are distinguished cases for image generation: when only a single objects corresponds to a label (as in the well-known chairs dataset) and when there are lots of objects with a single label (as in ImageNet). The models used in these two tasks are quite different, but two our discriminators are very similar.

3.1 Generator for absolutely conditional synthesis

In the first case, there is bijection between set of images and classes. The generator we use is based on the Dosovitsky's generator [10]. As we previously described, the original model was designed to generate 3D objects, and coped quite well with this task. It was able to qualitatively interpolate objects by angle and class. The architecture of the original generator is presented in Figure 1.

However, it is impossible to simply use this architecture in GANs. Without changing the basics of this architecture, we can only add normalization layers and change the nonlinearities. We tried to follow the DCGAN pipeline where it was possible, so we decided to replace the nonlinearity at the network output with tanh and sigmoid. The use of batch normalization did not allow the successful training of the network, therefore, various normalizations were tried. We found that using instance normalization before the deconvolutional block and weight normalization after each deconvolution operation allows the network to learn much more efficiently and produce images of better quality. So, summing up, we change this model in the following way:

- 1. Before applying the image deconvolution, we use instance normalization [16]
- 2. After each deconvolutional layer, we use weight normalization [18]
- 3. In the end, we apply tanh for the rgb image and sigmoid to the mask.

The architecture of the generator is present in figure 3.

3.2 Generator for partially conditional synthesis

In the second case, several images has the same label, we slightly change the generator. We added random variable **z**, which is responsible for the variety of images for a particular label as the input of the generator. We found that in the case of presence of **z**, it is much more efficient to concatenate internal representations after reshaping. The generator architecture is presented in figure 4.

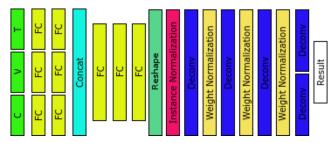


Figure 3: Generator architecture for absolutely conditional synthesis.



Figure 4: Generator architecture for partially conditional synthesis.

3.3 Discriminator

As the architecture of the discriminator, we use the model of "prover" that is the binary classifier solving task of determination if input arguments correspond to the input object. Its goal is to check whether it is true that "given object x corresponds to class x". A similar approach is used by the authors of paper [19]. A schematic description of the model is presented in Figure 2.

In the discriminator, we use weight normalization after all convolutional layers instead of batch normalization, as in section 3.1. In all other aspects, we followed the DCGAN [12] recommendations, except removing fully-connected hidden layers.

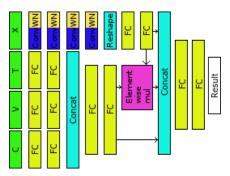


Figure 5: Discriminator architecture.

The architecture of the discriminator is present in figure 5. This architecture can be divided into three blocks:

- 1. Information Hidden Vector Retrieval Block that is used for folding information vectors into the internal representation of the network.
- 2. Image Hidden Vector Retrieval Block that is used for folding the image to an internal representation of the network.
- 3. Hidden Vector Combination Block that is used for combining the resulting vectors with subsequent processing in order to obtain classification result.

We describe these blocks in a more detailed way.

Information Hidden Vector Retrieval Block. This part of the discriminator architecture is absolutely identical to the analogous block in the generative model. Each of information arguments is

considered as an input for fully-connected layers. Output vectors are combined using concatenation with another subsequent application of the fully-connected layer. As a result, at the output of this block we get vector \mathbf{x}_{info} , which corresponds to the internal network representation of the information arguments.

Image Hidden Vector Retrieval Block. This block includes a sequence of convolutional layers and application of the weight normalization after each convolution. After folding, we obtain a feature-vector of image. Then we process it with a fully-connected layer in order to obtain the resulting internal representation \mathbf{x}_{imq} of this image.

Hidden Vector Combination Block. This block is designed to combine vectors that are provided by the blocks described above, namely internal representation of the information arguments \mathbf{x}_{info} and internal representation of the image \mathbf{x}_{img} . The most straightforward way to combine these vectors is to use simple concatenation, as we did in Information Hidden Vector Retrieval Block. However, the experiments showed that the use of such method does not allow the generator and discriminator to correctly learn and produce good images with correct correspondences to the information arguments.

We found the following approach to be effective in combining vectors \mathbf{x}_{info} and \mathbf{x}_{img} : the vector of the internal representation of the arguments validity should be computed as

$$\mathbf{x}_{corr} = \langle \mathbf{x}_{info} \odot \mathbf{x}_{img}, \mathbf{x}_{info}, \mathbf{x}_{img} \rangle,$$

where \odot denotes the element-wise product, and <> is the concatenation of its arguments. Vectors \mathbf{x}_{info} and \mathbf{x}_{imo} must be of the same dimension.

Finally, a sequence of fully-connected layers with the sigmoid nonlinearity at the end is applied to the obtained vector \mathbf{x}_{corr} , the result of that is the degree of consistency of the given image and the given information arguments.

In the partially conditioned case the architecture of the discriminator remains the same, except for the difference in the number of conditional arguments. Also we use dropout in convolutional layers and l_2 regularization on the discriminator.

3.4 Learning Algorithms

In the previous subsection, we described the generator that receives information vectors as the input and generates a suitable image and the discriminator the is being learned to distinguish if an information vector and an image are consistent. The way how to train the generator is clear — it will be trained to "fool" the discriminator. That is, we will train the generator to produce such images for given arguments, that the discriminator will accept it. This leads to equation 1. In order to create algorithms to learn the discriminator, we need to formalize its goal. Its output should be:

- 1. 0, if an unrealistic image is given (that is, obtained from the generator)
- 2. 0, if a realistic image is given (that is, an image from a dataset), but it does not correspond to a given set of information vectors
- 3. 1, if a realistic image is given and it corresponds to a given set of information vectors

This description can be now used to suggest a loss functions that will be used for training.

In our task, the three values $(\mathbf{c}, \mathbf{v}, \mathbf{t})$ are the information vectors, which are the class of the object, the view point and the transformation. The standard set of loss functions for the generator and discriminator of a GAN model is as follows:

$$V_G(\mathbf{c}, \mathbf{v}, \mathbf{t}) = BinaryCrossentropy(D(\mathbf{c}, \mathbf{v}, \mathbf{t}, G(\mathbf{c}, \mathbf{v}, \mathbf{t})), 1)$$
(1)

$$V_D^{gen}(\mathbf{c}, \mathbf{v}, \mathbf{t}) = BinaryCrossentropy(D(\mathbf{c}, \mathbf{v}, \mathbf{t}, G(\mathbf{c}, \mathbf{v}, \mathbf{t})), 0)$$
(2)

$$V_D^{real}(\mathbf{c}, \mathbf{v}, \mathbf{t}, x) = BinaryCrossentropy(D(\mathbf{c}, \mathbf{v}, \mathbf{t}, x), 1)$$

$$BinaryCrossentropy(p, y) = -(y \cdot \log(p) + (1 - y) \cdot \log(1 - p)),$$
(3)

where x is an image from the dataset. So, equations 2 and 3 corresponds to the first and the third item from discriminator behaviour description.

However, these functions are not enough, because we do not probed any information to the discriminator on non-matching images and their descriptions, which should be rejected. Therefore, the generator has the freedom to return images that do not correspond to the input arguments. This leads to the poor quality of the resulting images.

We handle this problem using *negative sampling*. In addition to training the network on the correct data, we will specially add the data with incorrect correspondences. There are three vectors describing an image: c is the class label, v is the view point and t is the transformation vector. The simplest case is with c: for each data set, we will generate one more exactly the same, replacing the class labels with random ones. We will get one more loss function for the discriminator:

$$V_D^{neg_c}(\mathbf{c}, \mathbf{c}', \mathbf{v}, \mathbf{t}, x) = BinaryCrossentropy(D(\mathbf{c}', \mathbf{v}, \mathbf{t}, x), 0), \mathbf{c}' \neq \mathbf{c}$$

Since the class labels are a discrete value, they are not particularly problematic. \mathbf{v} and \mathbf{t} are continuous values, so we cannot simply replace them with others and claim that the correspondence in the data has become incorrect. Intuitively, the degree of correctness of the correspondence must be inversely proportional to the distance between the original (correct) vector and the random generated (wrong) one. So the loss function for neg_v is as follows:

$$V_D^{neg_v}(\mathbf{c}, \mathbf{v}, \mathbf{v}', \mathbf{t}, x) = \|\mathbf{v} - \mathbf{v}'\|^2 \cdot BinaryCrossentropy(D(\mathbf{c}, \mathbf{v}', \mathbf{t}, x), 0).$$

For neg_t , the situation is the same.

Now we have the resulting loss function for the discriminator, which will look like:

$$V_D = \alpha \cdot V_D^{real} + \beta \cdot V_D^{gen} + \gamma_c \cdot V_D^{neg_c} + \gamma_v \cdot V_D^{neg_v} + \gamma_t \cdot V_D^{neg_t}$$

4 Experiments

In our work, we use three different dataset. The first of them is the well-known dataset that was used by Dosovitskiy et al. [10]. This dataset contains a set of images of 3D chairs in various projections, as shown in figure 6. For each image, a label of the class c corresponding to the type of the chair presented in the image is known. Description of view point v is a set of four numbers corresponding to sinus and cosine of azimuth and altitude angles. Results on this dataset are presented in the body of the paper. We also use augmentation for model training, identical to augmentation from the work



Figure 6: Dataset samples example.

of Dosovitsky et al. For each image, the transformation vector t was added.

The two other datasets are CelebFaces² and Cifar10³. CelebFaces dataset is a set of images with faces and labels with it attributes. Cifar10 is a set of images where each image is one from ten classes, such as airplane, car, horse, dog and so on. In these datasets, the labels do not uniquely specify the images. Results on these datasets are presented in the appendix.

We tested Dosovitskiy's generator learned with l_2 -loss and bunch of different DCGAN models, including ACGAN [21], WGAN [27], improved GAN models [26] besides our model. We train generator for 500 epochs. Composition of generator and discriminator we train 1000 epochs, but generator was trained every third time relative to discriminator. We use batches of size 16.

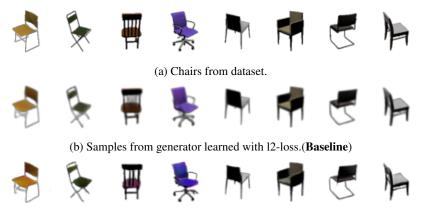
5 Results and Discussion

None of different DCGAN based models we tried coped with the task of accurately interpolating object viewing position. In paper [20], the authors evaluated their model on 3D chairs dataset, but

²http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

³https://www.cs.toronto.edu/~kriz/cifar.html

their model produce images of less accuracy. Results of the chair generation, view point and class interpolation are presented in Figures 7–9.



(c) Samples from generator learned with gan-loss. (\pmb{Ours})

Figure 7: Random chairs.

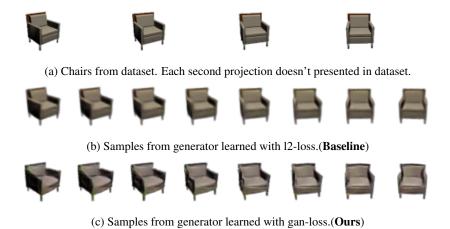
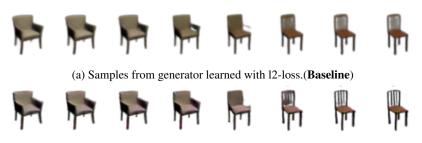


Figure 8: Rotation. Each second projection is not represented in the dataset.



(b) Samples from generator learned with gan-loss.(Ours)

Figure 9: Class interpolation.

According to the results of experiments on all three datasets, it is clear that the images obtained by the proposed generator trained with the discriminator, are more sharp. The generator also copes successfully with the problems of interpolation by angle and class. More examples of network operation can be seen in the appendix.

It is worth paying attention to how well the generator that have never seen exact real images is able to be learned by the expense of a discriminator reproducing so accurately the smallest details of objects, correctly preserving view points.

6 Conclusion

In our work, we combined the Generative Adversarial Nets capabilities with the power and flexibility of complex models, such as the Dosovitsky et al.'s generator [10]. We presented new approaches for the design of generators and discriminators, which allow these models to be well trained in composition. We also showed the effectiveness of such models on the task of generating 3D object projections. The suggested architecture allows simple switching between who different cases: absolutely and partially conditional ones.

Based on this, we can formulate a modification of the rule set described in DCGAN [12], which allows to use fully-connected layers and getting more powerful GAN models:

- 1. Use instance normalization [16] in the generator before starting the deconvolution of image
- 2. In all convolutional and deconvolutional layers, use weight normalization [18]
- 3. Use element-wise multiplication and negative sampling in conditional discriminator

In addition to applying the GAN concept to create synthesis models capable of generating realistic images, GAN can also be used to improve quality and robustness of classifiers. Using the approaches described in our work, it is possible to build strong classifiers and train them using generators. In our work, we did not conduct such studies, but we plan to do it in the future.

Acknowledgments

We'd like to thank Servers.com for donating a GPU server used in this work.

References

- 1. *Krizhevsky A., Sutskever I., Hinton G. E.* Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. 2012. Pp. 1097–1105.
- Rich feature hierarchies for accurate object detection and semantic segmentation / R. Girshick [et al.] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2014. — Pp. 580–587.
- 3. Overfeat: Integrated recognition, localization and detection using convolutional networks / P. Sermanet [et al.] // arXiv preprint arXiv:1312.6229. 2013.
- Generative adversarial text to image synthesis / S. Reed [et al.] // Proceedings of The 33rd International Conference on Machine Learning. Vol. 3. — 2016.
- 5. Stylebank: An explicit representation for neural image style transfer / D. Chen [et al.] // arXiv preprint arXiv:1703.09210. 2017.
- 6. *Girod B.*, *Greiner G.*, *Niemann H.* Principles of 3D image analysis and synthesis. Vol. 556. Springer Science & Business Media, 2013.
- MRI-TRUS Image Synthesis with Application to Image-Guided Prostate Intervention / J. A. Onofrey [et al.] // International Workshop on Simulation and Synthesis in Medical Imaging. — Springer. 2016. — Pp. 157–166.
- 8. *Miyamoto Y. S. I.*, *Takeuchi K.* Application of Sign Language Image Synthesis Technology for Free Mobility Project. —.
- Using image synthesis for multi-channel registration of different image modalities / M. Chen [et al.] // SPIE Medical Imaging. — International Society for Optics, Photonics. 2015. — 94131Q–94131Q.
- 10. *Dosovitskiy A.*, *Springenberg J. T.*, *Brox T.* Learning to Generate Chairs with Convolutional Neural Networks // CoRR. 2014. Vol. abs/1411.5928. URL: http://arxiv.org/abs/1411.5928.
- Generative Adversarial Networks / I. J. Goodfellow [et al.] // ArXiv e-prints. 2014. June. — arXiv: 1406.2661 [stat.ML].

- 12. Radford A., Metz L., Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks // CoRR. 2015. Vol. abs/1511.06434. URL: http://arxiv.org/abs/1511.06434.
- 13. High accuracy optical flow estimation based on a theory for warping / T. Brox [et al.] // European Conference on Computer Vision (ECCV). Vol. 3024. Springer, 05/2004. Pp. 25—36. (Lecture Notes in Computer Science). URL: http://lmb.informatik.uni-freiburg.de/Publications/2004/Bro04a.
- 14. *Johnson J.*, *Alahi A.*, *Li F.* Perceptual Losses for Real-Time Style Transfer and Super-Resolution // CoRR. 2016. Vol. abs/1603.08155. URL: http://arxiv.org/abs/1603.08155.
- 15. Dumoulin V., Shlens J., Kudlur M. A Learned Representation For Artistic Style // CoRR. 2016. Vol. abs/1610.07629. URL: http://arxiv.org/abs/1610.07629.
- 16. Ulyanov D., Vedaldi A., Lempitsky V. S. Instance Normalization: The Missing Ingredient for Fast Stylization // CoRR. 2016. Vol. abs/1607.08022. URL: http://arxiv.org/abs/1607.08022.
- 17. *Ioffe S.*, *Szegedy C*. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // CoRR. 2015. Vol. abs/1502.03167. URL: http://arxiv.org/abs/1502.03167.
- 18. Salimans T., Kingma D. P. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks // CoRR. 2016. Vol. abs/1602.07868. URL: http://arxiv.org/abs/1602.07868.
- 19. Mirza M., Osindero S. Conditional Generative Adversarial Nets // CoRR. 2014. Vol. abs/1411.1784. URL: http://arxiv.org/abs/1411.1784.
- InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets / X. Chen [et al.] // CoRR. 2016. Vol. abs/1606.03657. URL: http://arxiv.org/abs/1606.03657.
- 21. Odena A., Olah C., Shlens J. Conditional Image Synthesis With Auxiliary Classifier GANs // ArXiv e-prints. 2016. Oct. arXiv: 1610.09585 [stat.ML].
- Generative Visual Manipulation on the Natural Image Manifold / J. Zhu [et al.] // CoRR. 2016. Vol. abs/1609.03552. URL: http://arxiv.org/abs/1609.03552.
- 23. *Dosovitskiy A., Brox T.* Generating Images with Perceptual Similarity Metrics based on Deep Networks // CoRR. 2016. Vol. abs/1602.02644. URL: http://arxiv.org/abs/1602.02644.
- 24. White T. Sampling Generative Networks: Notes on a Few Effective Techniques // CoRR. 2016. Vol. abs/1609.04468. URL: http://arxiv.org/abs/1609.04468.
- Adversarially Learned Inference / V. Dumoulin [et al.] // ArXiv e-prints. 2016. June. arXiv: 1606.00704 [stat.ML].
- 26. Improved Techniques for Training GANs / T. Salimans [et al.] // CoRR. 2016. Vol. abs/1606.03498. URL: http://arxiv.org/abs/1606.03498.
- 27. Arjovsky M., Chintala S., Bottou L. Wasserstein GAN // ArXiv e-prints. 2017. Jan. arXiv: 1701.07875 [stat.ML].
- AdaGAN: Boosting Generative Models / I. Tolstikhin [et al.] // ArXiv e-prints. 2017. Jan. — arXiv: 1701.02386 [stat.ML].

A Additional images



Figure 10: Set of random chairs in different projections.



Figure 11: Set of random view point/class interpolations of chairs.



Figure 12: Generated images from Cifar10 dataset. The same class horizontally.

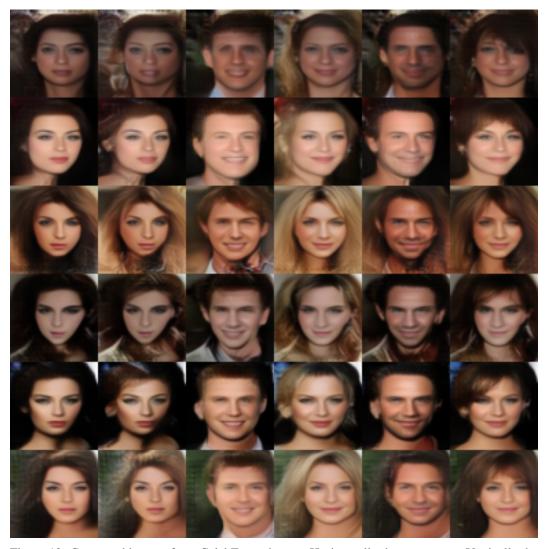


Figure 13: Generated images from CelebFaces dataset. Horizontally the same z var. Vertically the same c var.

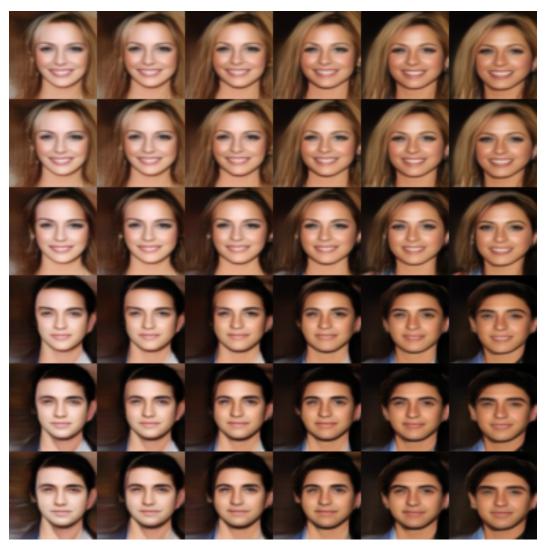


Figure 14: CelebFaces interpolation between two z vars(horizontally) and two c vars(vertically).