Conditional Generative Adversarial Network: generate new face images based on attributes

Moritz Feuerpfeil
Data Engineering Master, HPI
mfeuerpfeil@uni-potsdam.de

Jiahao Hu
Digital Health Master, HPI
jiahao.hu@student.hpi.de

Jennifer Daniel Onwuchekwa
Digital Health Master, HPI
Jennifer.DanielOnwuchekwa@student.hpi.de

Hazem Abou Hamdan
Master of Data Science
Potsdam University
abouhamdan@uni-potsdam.de

Mohammad Wasil Saleem
Master of Data Science
Potsdam University
saleem1@uni-potsdam.de

Project Summary

Conditioned face generation, an extension of the generative adversarial network (GAN), is a complex model with several applications reported in several domains such as security, styling and entertainment. Conditional GAN (cGAN) is being applied in generating examples of image datasets, human faces or cartoon characters, image-to-image translation, text-to-image translation, face aging and video prediction. In cGAN, two models are trained simultaneously by an adversarial process - a generator ("the artist") learns to create images that look real, while a discriminator ("the art critic") learns to discriminate real images from the fake and synthetically generated images. [1].
In this project, we aimed to combine certain attributes or conditions to generating facial images using deep convolutional generative adversarial network on a well-known CelebFaces Attributes Dataset (CelebA) containing celebrity face images. Therefore, we called this architecture a Conditional Deep Convolutional Generative Adversarial Network (cDCGAN). We implemented the conditional network by exploring the state of the art of generative adversarial learning field. Despite the limited computational capacity, we selected a subset of the original dataset and successfully generated novel face images based on three selected attributes (eyeglasses, rosy-cheeks and goatee classes) from the same model. The generator loss was 2.242, discriminator losses in detecting real and fake images were 0.691 and 0.323 respectively. Finally, we validated our model using the inception score and the Frèchet inception distance evaluation techniques. The inception score yielded a final score of 2.47 and standard deviation of 0.32, while the Frèchet inception distances for the eyeglasses, rosy-cheeks and goatee classes were 166.637, 157.183 and 165.206 respectively.

1 Introduction

Today, Artificial Intelligence (AI) is a thriving field with many practical applications and active research topics. It can tackle and solve problems that are intellectually difficult for human beings but

[1] https://www.tensorflow.org/tutorials/generative/dcgan

relatively easy for computers - problems that can be described by a list of formal, mathematical rules. In a data driven era, the world can be represented as a large 'unlabeled' dataset, consisting of relatively easily obtainable natural or human-created artifacts such as photos, audio recordings, videos, news articles or even x-rays (as in the medical world). Labels can be attributed to such data often by asking humans to tag or make judgements about a given piece of unlabeled data. For instance, "*this photo contains a cat lying on the bed*". It is quite expensive to obtain these labelled data, hence, requires machine learning models. Computers can learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined in terms of its relation to simpler concepts. This hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. A graph showing how these concepts are built on the top of each other can be drawn, and is described as being deep with many layers. For this reason, this approach to AI is called Deep Learning (DL) [6].

So far, DL has shown great promise in discriminative models, as they have shown high accuracy in distinguishing the differences amongst a group of objects. On this note, classifiers and regressors are both examples of discriminative models. Regression models predict a number such as the weight of the animal, while classification models predict a state or type of animal such as a cat or dog. The striking success of these models have been based on the backpropagation and dropout algorithms. They have found their applications in many well-known fields, such as image recognition, natural language processing, and speech recognition.

Nevertheless, there is more to deep learning than just discriminative models. Infact, we have been witnessing the booming of generative models. They represent an unsupervised learning task that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset. Deep generative models have had less of an impact, due to the difficulty of approximating many intractable probabilistic computations that arise in maximum likelihood estimation and related strategies, and due to difficulty of leveraging the benefits of piecewise linear units in the generative context[12]. Generative adversarial networks are a recently introduced method for training generative models with neural networks. This approach sidesteps some of the common problems among generative models and adopts a simple SGD training regime. The generative model yielded by a learned GAN can easily serve as a density model of the training data. Sampling is simple and efficient: the network accepts some noise as input and outputs new samples of data in line with the observed training data [10]. More specifically, the GAN architecture consists of two sub-models: a generator model for generating new examples and a discriminator model for classifying whether generated examples are real (from the domain) or fake (generated by the generator model).

An extension to the GANs is the Conditional Generative Adversarial Network. In this case, the generative model is trained using the input data provided with some additional information, such as a class label (female or male), to generate conditionally an output. In this report, we will borrow the convolutional architecture that has been proven successful for discriminative computer vision problems and show via cGAN, they can be leveraged to generate photorealistic images. The report consists of 6 sections. In section 2, we present different approaches and techniques proposed by scientists working in computer vision field. In section 3, we describe the main charateristics of the data set we used and how we preprocessed the images before they could be fed into the algorithm. The architecture and the parameters we used to train our model are presented in the section 4. This is followed, therefore, by the evaluation section where we document the method we used to evaluate the quality of the images generated by the model. In the second last section, we provide details regarding the architecture that we implemented to generate face images using all the 40 attributes, the various experiments we carried out, and the results it yielded. This section is, then, followed by the discussion section where we review the results obtained and present further improvements that might help us to better the model in future.

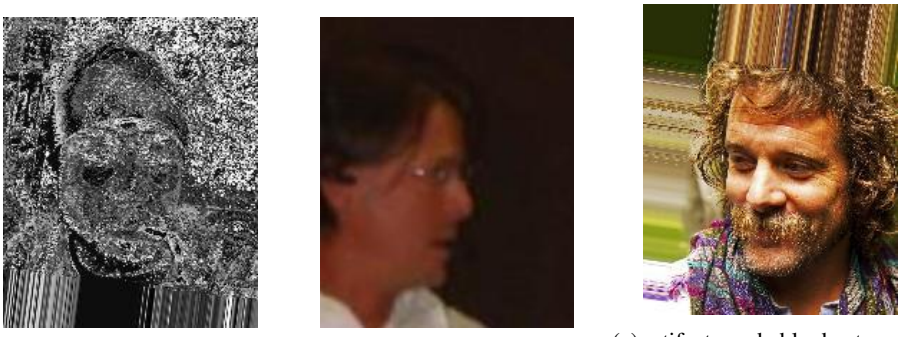
2 Related Work

Owing to the several successes obtained by discriminative models, generative adversarial networks (GANs) have become an interesting and practical tool in deep learning since its introduction in 2014 by Ian Goodfellow [12]. Improving the training of GANs has been an active research area in the past couple of years, where several scientific papers have focused on improving the efficiency and performance of the GAN [19] [18] [12], including the application of batch normalization, the

normalization of input, and different activation functions that can be deployed. For instance, Wasserstein distance has been introduced as a new objective, which has non-zero gradients anywhere in the Wasserstein GAN (WGAN) work[18]. Its implementation was as simple as just removing the sigmoid function in the objective and adding weight clipping to the discriminator network. WGAN is demonstrated to be free of the many problems in the original GAN, such as mode collapse and unstable training process. A related work to the WGAN is the Loss-Sensitive GAN, whose goal is to minimize the loss for real data and maximize it for the fake ones. The common property of Least Square GAN, WGAN, Loss-Sensitive GAN and this work is the usage of objective functions with non-vanishing gradients.
The progressive growing of Neural network performance can be improved by utilizing some training methodology for GAN. For example, starting with low-resolution image and then increasing the resolution of the image by adding new layers to the network. However, when new layers are added to the networks, it is faded in smoothly. This avoids sudden shocks to the already well-trained but smaller-resolution layers. This technique is shown to be greatly useful in order to enhance the resolution of the generated image, especially if we have thousands of instances (images); however, it requires a few days to train the generator [12].
Furthermore, the variation of the generated data can be increased using mini-batch standard deviation. Therefore, by computing feature statistics across the mini-batches, it is possible to enhance the generated data, that would have similar statistic features to the training data. This technique could be helpful if we have enough data but lack of features; thus, creating new features could make the generated images more arbitrary. Moreover, another article states that virtual batch normalization (VBN) could help to improve Neural Network performance, and it is validated that it can improve the efficiency of DCGAN as well [18]. The VBN is done by normalizing each example based on the statistic of collected on a reference batch of examples that are chosen once and fixed at the beginning of the training process. The reference batch is, thus, normalized using only its own statistics. The VBN is implemented only in the generator network and it is computationally expensive because we have to run two mini-batch in forward propagation simultaneously[18].

3 Dataset

The dataset that we used for this project was the CelebFaces Attributes Dataset (CelebA) [13]. It contains about 202.6 thousand colored pictures of celebrities which are all 178x218 pixels in dimensions. The over 200 thousand pictures are distributed among 10,177 individuals. Even though the pictures seem to have an overall good quality, there are also outliers as shown in Figure 1.
Each picture has a configuration of 40 face attributes (-1 or 1 for every attribute) attached to it



(a) unrecognizable face (b) blurry or bad angle (c) artifacts probably due to cropping

Figure 1: Examples of low quality pictures of the CelebA dataset.

that were determined by a 'professional labeling company' [13]. Figure 3 shows the distribution of these attributes per person and Figure 4 depicts the total distribution of the attributes. As one can see very clearly, the attributes are heavily imbalanced in numbers due to the nature of the attributes themselves. While 83.5% of the pictures depict faces without a beard, only 2.24% of the pictures

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

Figure 2: Examples for eight of the forty attributes from the CelebA dataset. The image was taken from the official CelebA website[2].

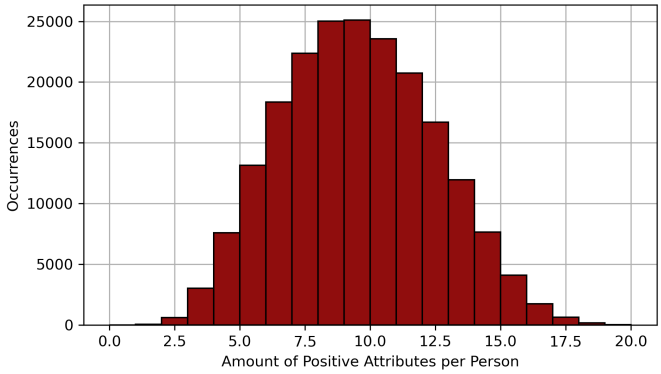


Figure 3: Distribution of positive attributes per person. The attributes per person seem to be normal distributed with a mean of 9.03 and a standard deviation of 2.94. The max value is 20 and the min value is 1.

show celebrities who are bald. This imbalance must be kept in mind when training the model and evaluating the performance on the different conditions.

Even though the pictures are not too big in file size, we decided to rescale them to make the model training computationally more feasible. In the end we decided that 128 by 128 pixels would be the perfect fit for us in regards to the trade off between quality and computational difficulty.We also scaled the range of the input data to [-1,1] so that it matches the output of the tanh activation function which is used at the output layer of our generator [14].
Since we came across some major hardware limitation issues (discussed in Section 7) there are two different datasets used for two different experiments. The first dataset consists of images with the attributes "Eyeglasses", "Rosy_Cheeks" and "Goatee", where each class contributes equally to the whole dataset of 36,000 images which were drawn randomly. We chose those classes because there were not many more instances of them beyond the 12,000 mark which means that we are not losing training data for this experiment.

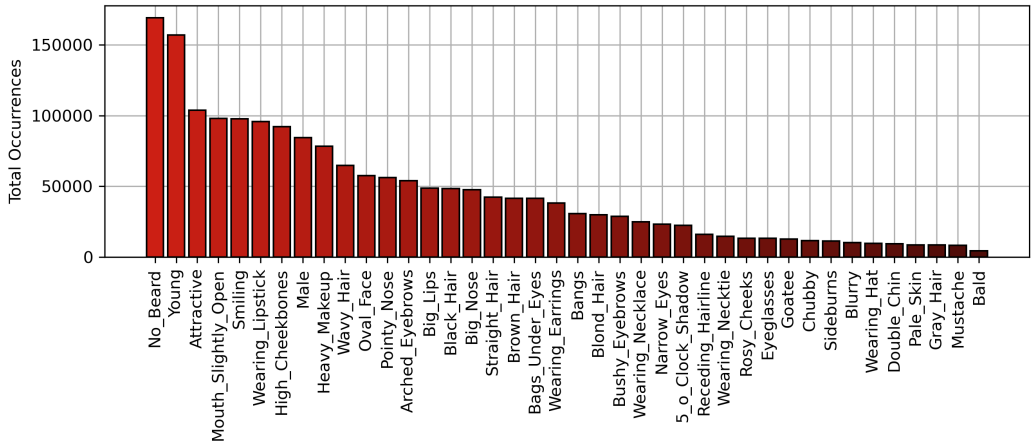


Figure 4: Total occurrences of attributes in the whole dataset.

The second dataset consists of all 40 attributes with only 750 images per attribute in order to deal with the hardware limitations.

4 Architecture and Training

Our main architecture is a slightly different version of the DCGAN [14] that we adjusted to also take labels as inputs, making it a conditional DCGAN (cDCGAN). The hyperparameters are mainly inspired from the DCGAN paper and the used frameworks are Tensorflow [1] and Keras [7]. The inputs to the generator G are a latent vector of size 100 (sampled from standard normal distribution) and a label. The label goes through an embedding and is then added to the low resolution picture as an additional channel.We used four transpose convolution layers (4x4 kernel size, stride of 2, amount of filters halving every layer starting at 1024) to upsample the image and finally a convolutional layer with tanh activation to reduce the amount of channels to three (RGB). Between the transpose convolutions we applied the ReLU (not leaky ReLU) activation functions as suggested in [14]. G has a total of 11,119,129 trainable parameters.
The discriminator D also has two inputs: the class label and an image of size 128 by 128 with three color channels. The label went through an embedding to be subsequently added to the picture as an additional channel. Our D made use of four convolutional layers (3x3 kernel size, stride of 2, doubling filters starting with 64) and a single dense layer with sigmoid activation for the output. After every convolution a LeakyReLU was applied with a slope value of 0.2 (see [14]). No pooling layers were used in any model. We tried using batch normalization but it did not yield better results in our case, so we did not include it in the final architecture. D has a total of 2,420,055 trainable parameters. The whole model is shown in Figure 9b.
We made use of two regularization methods, the first being a simple dropout layer as the second to last layer of the discriminator. The second regularization method is called one-sided label smoothing [17]. Both models were compiled using the Adam optimizer with a learning rate of 0.0002 and a β_1 of 0.5. Even though there are many different loss functions for GANs, we stuck with a simple binary cross-entropy. We trained the models using stochastic gradient descent (SGD) with a batch size of 128, where the batch for training the discriminator is split in half for the real and fake images. The summary below describes how we trained the model in a single batch. We repeated that process for the whole dataset and then for multiple epochs. It is to be noted that we trained this model on the first dataset (see Section 3) with only three classes due to its complexity and hardware requirements.

1. Train D
   (a) Sample 64 real images and their class labels
   (b) Train D on the real data and smoothed labels (0.9 instead of 1) as ground truth
   (c) Generate 64 fake images with G and their class labels
   (d) Train D on the fake data and half a batch of zeroes as ground truth

2. Train G
   (a) Sample 128 latent vectors and labels randomly
   (b) Train the joint model (with D not in training mode) with the input of G (latent vectors and labels) and a batch size of ones as the ground truth

We saved the model's weights every five epochs and saved visual progress of the same latent vector every epoch. Some of the code was inspired by a tutorial of Jason Brownlee[3] and a class of Jeff Heaton[4].

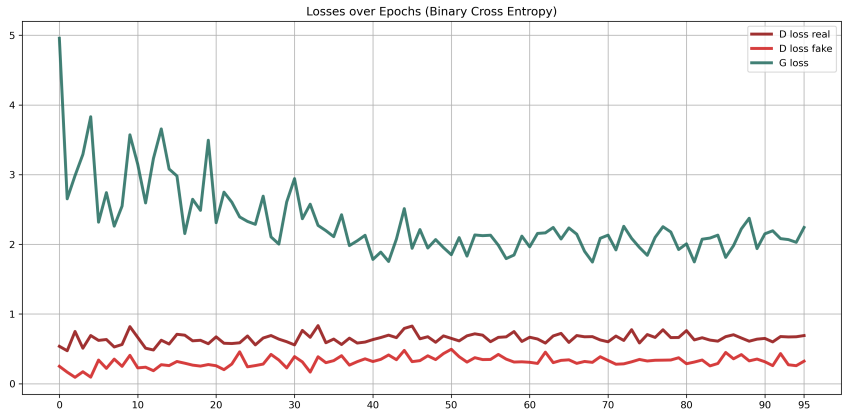


Figure 5: Training losses of the generator and discriminator over the epochs. While the discriminator losses are rather constant, the generator loss converges slightly towards two. We interpret these losses that both models are still learning and that they could improve upon further training.

As you observe in the above figure (Figure5), after the whole training, the model reached 0.691 as the discriminator real loss, 0.323 as the discriminator false loss, and 2.242 as the generator loss.

5 Evaluation

In this section, we evaluate the quality of our cDCGAN model using qualitative and quantitative evaluation strategies. GANs have proved to be remarkable effective in generating high-quality synthetic images, as described in section 2. There are a number of research papers that have benchmarked and evaluated the accuracy and quality of face recognition outputs. This quality metric evaluation is based on the results and how realistic the synthetic generated images are. As you can see in the Figure6, the model is able to generate distinct facial attributes that are clear to human eyes. Firstly, we report the manual inspection based on the pre-trained faces and attribute detection models. Secondly, we implement and interpret the inception score (IS) and Frechèt Inception Distance (FID), two widely used image quality evaluation techniques for GANs.

When deciding on what final GAN model architectures and configurations to use during training, manual inspection of the images generated by our model was done. This was done by subjectively examining the generated images by the generator, as it demonstrates the capability of the GAN generator model. This included assessing the capability of the generator to generate and differentiate between the different conditional images of the different classes, observing facial features extracted from the images, e.g. expression, contrast, brightness, lens distortion, background, and how realistic the images look with respect to the real input images. This visual examination of samples have been described to be one of the most common and most intuitive ways to evaluate GANs [5]. In adjusting for a better model, for instance, the activation function was tweaked using ReLU, Wasserstein loss,

[3]https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/

[4]https://github.com/jeffheaton/t81$_5$58$_d$eep$_l$earning

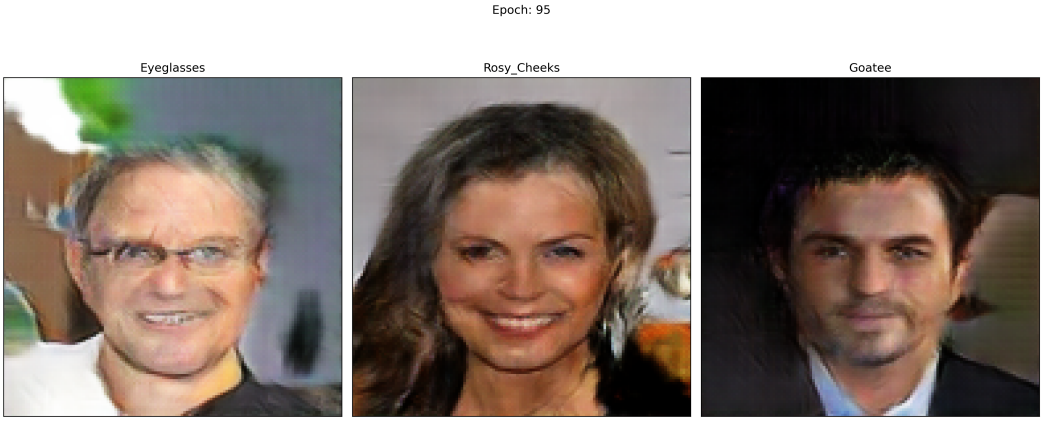


Figure 6: Figure containing the three classes we included in the complex model.

Leaky ReLU etc, different number of classes to train in one model at a time was tried - all 40 classes, just two unique classes: sunglasses vs no sunglasses, sunglasses vs moustache, male vs females, and finally we got better output when we trained on three classes: sunglasses, goatee and rosy-cheeks. The initial examples of architecture and configurations used provided hints for a better model as described in section 6.

The concept of inception score that was introduced by T. Salimans in 2016 was an attempt to remove the subjective human evaluation in images, as described above. Notably, the IS involves using a pre-trained deep learning neural network model for image classification to classify the generated images. In particular, the inception V3 pre-trained on ImageNet with 1000 object classes is used[8]. The paper states that the calculation of the inception score on a group of images involves first using the inception v3 model to calculate the conditional probability for each image.

$$(p(y|x))$$

The marginal probability is, then, calculated as the average of the conditional probabilities for the images in the group.

$$(p(y))$$

The Kullback-Leibler (KL) divergence is, consequently, calculated for each image as the conditional probability multiplied by the log of the conditional probability minus the log of the marginal probability.

$$KL \;\; divergence = p(y|x) \times (log(p(y|x)) - log(p(y)))$$

Finally, the KL divergence is summed over all images and averaged over all classes and the exponent of the result is calculated to give the final score. Two important features are captured in the inception score. They are the image quality, that is to say, how good our generator model is at creating distinct objects with well-defined contours, and the image diversity, which explains how varied the generated images are. In our specific case, we generated 300 images including all 3 classes: eyeglasses, rosy_cheeks, and goatee. They are, firstly, converted into a four dimensional array and shuffled. Secondly, they were resized accordingly to the required input size of the inception model before fed to it. It yielded 2.47 and 0.32 as final score and standard deviation respectively. Since we have 3 classes, we would expect to have a final score very close to 3. Thus, the result we obtained from the IS model indicates that our generator model is able to produce distinct objects based on the selected attributes.

To further investigate the performance of our generative adversarial network, we implemented a second quantitative metric, which is the aforementioned FID. It is a metric that computes the distance between feature vectors calculated for real and generated images[11]. The score summarizes how similar the two groups are in terms of statistics on computer vision features of the raw images calculated using again the inception v3 model used for image classification. Lower scores indicate the two groups of images are more similar, or have more similar statistics, with a perfect score being 0.0 indicating that the two groups of images are identical[6]. Our intent is to evaluate how realistic

the generated images are compared to the ground truth images used to train the generator model itself. Similarly to the IS, the FID method takes the last pooling layer of the inception V3 model to compute the feature vector and the following formula is used to calculate the distance:

$$d^2 = ||\mu_1 - \mu_2||^2 + Tr(C_1 + C_2 - 2 \times \sqrt{C_1 \times C_2})$$

The μ_1 and μ_2 indicate the extracted feature vectors from the real and generated images; meanwhile Tr is the trace linear algebra operation applied to the covariance matrices for the real and generated feature vectors (C_1 and C_2).
We computed 3 FID scores, each per class. In order to accomplish it, we sampled our training dataset chosing 300 images, $\sim$100 per each of the 3 attributes introduced earlier in this section. Similarly to what we did to calculate the IS scores; here, we also transformed the real and fake images into numpy arrays. They were, then, reshaped and pre-processed before fed to the model. The results are shown in the below table:

	Eyeglasses	Rosy_cheek	Goatee
FID	166.637	157.183	165.206

Table 1: Table reporting the FID scores computed for each class

The scores are pretty high, which shows that the synthetic images are not close to the real picture. Therefore, we can conclude that our generator model can generate distinct classes but they are still far from the ground truth so they can fool human eyes. This was also noted during our training process, an example is provided below in Figure7 where the probabilities of being real are reported.

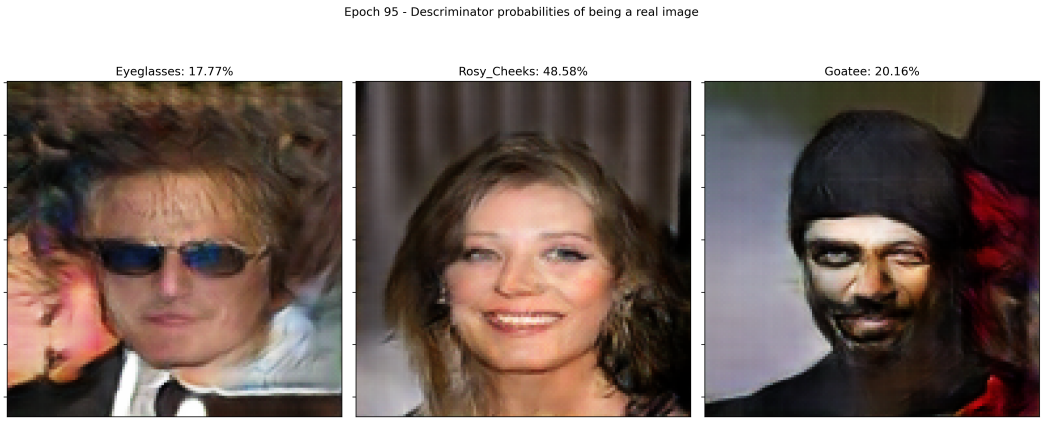


Figure 7: Figure containing the three classes and the probabilities of them being real computed by the discriminator after 95 epochs.

Nevertheless, we achieved pretty good results as the mode collapse, one of the most common failures in generative networks, did not happen. This means that our generator model can generate a wide variety of images conditioned by only one attribute. Some example figures are reported in the appendix, where the latent vectors are constant across the three classes but only the condition label changes (Figure10, Figure11 and Figure12).

6 Experiments

As soon as we encountered the problems with the amount of data, we experimented with different solutions to bypass that bottleneck. A short selection of two experiments are presented here as they did not yield good results.

6.1 Less training data

One of the architectures that we trained was on the second data set that contains all 40 attributes, with only 750 images per attribute. With this experiment we wanted to see whether we could produce acceptable images for all 40 classes with data that would fit in main memory.

The inputs to the generator were the same as defined in the previous sections. However, we made several changes to the architecture of the model. The Generator used 5 transpose Convolutional layers with decreasing numbers of filters from 1024 to 64, with the foundation image of size 4 by 4 and it upsampled the image to shape 128 by 128 by three color channels. In addition, for every transpose Convolutional layer, we used LeakyReLU activation functions.

The inputs to the discriminator were also identical to what has been described previously in this report4. The model consisted of 3 Convolutional layers, with constant 128 numbers of filters for each layer, the shape of the filters was 3 by 3. Apart from this, no other changes were applied to the model. For every layer except the final output one, we used LeakyReLU activation function[3].

Most of the time, the generator loss was oscillating between 0.9 and 1, and the total loss of discriminator for the real and fake image was around 1.38. The generator was able to generated images but not all 40 attributes were distinguishable, one example figure is documented in the appendix (Figure13). Due to the relative small dataset we used, it did not return satisfactory results.

6.2 Less complex model

Earlier work included training less complex models was done to observe the trade-off between model complexity and training duration. One of our simpler training models had just two convolutional layers in the discriminator, three transposed convolutional layers in the generator, and was trained on 1000 images per attribute. Some results after 190 epochs can be found in Figure 8. The model did not perform better visually even after tuning the hyper-parameters, for instance: amount of filters or dropout rate, etc. It seemed that the low model complexity and the small amount of training data were not compensated for by training for more epochs. Since there could have been many issues that could make the training of GANs unstable, we did not look further into building smaller models from scratch.

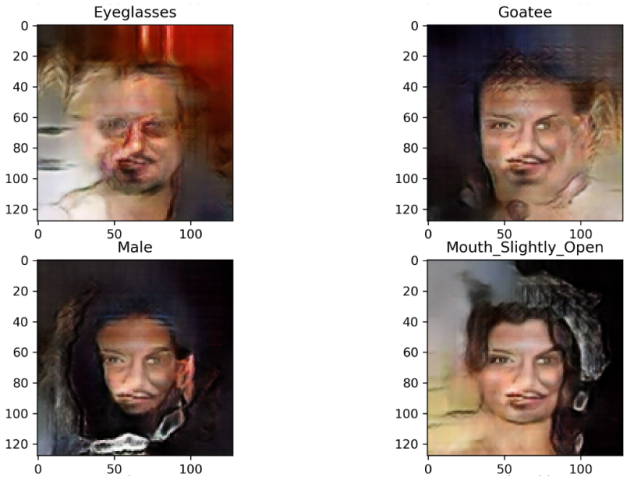


Figure 8: The results of a less complex model after training for 190 Epochs on 1000 images per attribute.

7 Discussion

The biggest problem we faced was the required computational power of the deep learning models. At first we tried using more data but the cloud platform "Google Colab" (which we used in the free version) quickly ran out of memory and epochs were taking too long for it to be feasible. When using a data loader to bypass the RAM restrictions, there was a heavy drop in performance due to the disk I/O. With the restriction to three classes we could show the power of the model while keeping training time reasonable (one epoch took about 15 minutes). We tried many things to reduce the amount of trainable parameters or the amount of training data in order to use all 40 classes provided but the results were not as impressive (see Section 6). There was also an attempt to reduce the dimensionality by using gray-scale images instead of the RGB colored format for training, but this also yielded rather disappointing results.
There are many tricks for training better adversarial nets [17]. While we incorporated e.g. the one-sided label smoothing, there are many more techniques like Wasserstein Loss, historical averaging or even just tuning parameters more carefully. But since we had very limited time we could unfortunately not experiment with these techniques. Additionally the visual results were good enough for us to be content with this model.
Future work would definitely include the usage of more training data and all forty classes. When time is not of essence then using a data loader would allow for the models to train on the whole dataset of 202 thousand pictures. In regards to our model that was limited to three classes, future work would include incorporating higher resolution support and maybe trying out other classes with more complex features like hats or earrings.
One of the main difficulties for quantitative assessment lies in the fact that the distribution is only specified implicitly – one can learn to sample from a predefined distribution, but cannot evaluate the likelihood efficiently [16]. In our specific scenario, we chose inception score and Frèchet inception distance in accordance with our goal, which is, merely, conditionally generating face images. Therefore, the primary assessment was based on whether the chosen attributed is produced or not. The IS has shown, in section 5, the generator model is able to produce distinguishable faces with the chosen attributes. Besides, human visual assessment could also confirm it. Nevertheless, there are limitations that need to be addressed. For instance, the sensitivity to weights as using different version of inception models could lead to different results[2]. Moreover, it is very sensitive to resolution[15]. Additionally, the FID requires that features are Gaussian distributed which is not always guaranteed[4]. Thus, we could in future train the inception model using the celebA dataset in order to guarantee a more consistent method for validation. Other options such as Frèchet joint distance could be also implemented as it is able to capture even intra-class variation if we want use all the 40 attributes in addition to generate more images [9].

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Shane Barratt and Rishi Sharma. A note on the inception score. 01 2018.

[3] Sukarna Barua, Sarah Monazam Erfani, and James Bailey. Fcc-gan: A fully connected and convolutional net architecture for gans. *arXiv preprint arXiv:1905.02417*, 2019.

[4] Ali Borji. Pros and cons of gan evaluation measures. 06 2018.

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

[6] Jason Brownlee. *Generative Adversarial Networks with Python*. 2019.

[7] Francois Chollet et al. Keras, 2015.

[8] Sergey Ioffe Jon Shlens Zbigniew Wojna Christian Szegedy, Vincent Vanhoucke. Rethinking the inception architecture for computer vision. 2015.

[9] Terrance DeVries, Adriana Romero, Luis Pineda, Graham W. Taylor, and Michal Drozdzal. On the evaluation of conditional {gan}s, 2020.

[10] Jon Gauthier. Conditional generative adversarial nets for convolutional face generation.

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

[12] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial nets. 2014.

[13] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[14] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2016.

[15] M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In *Workshop on Theoretical Foundations and Applications of Deep Generative Models (TADGM) at the 35th International Conference on Machine Learning (ICML)*, July 2018.

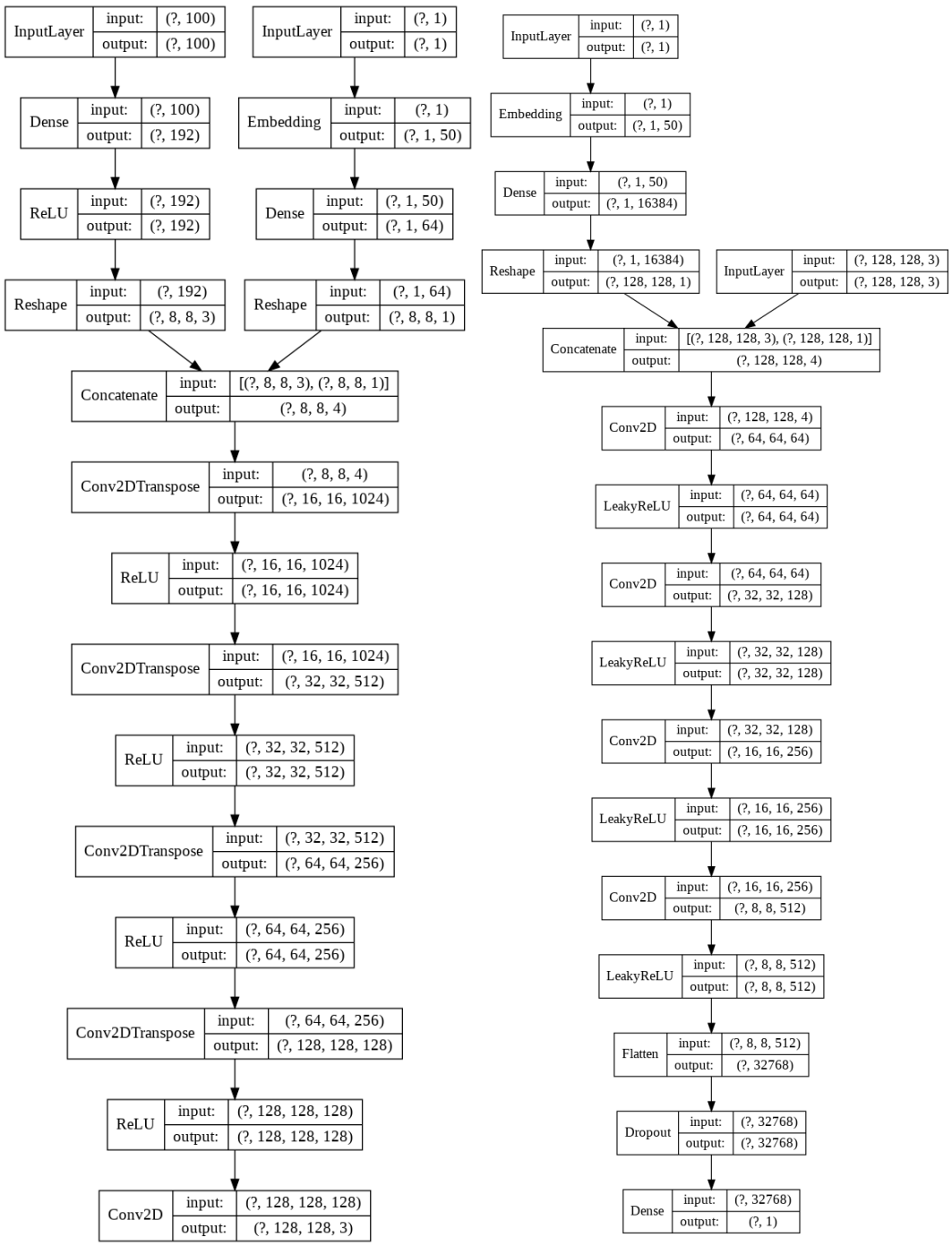
[16] Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5228–5237. Curran Associates, Inc., 2018.

[17] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[18] Samuli Laine Jaakko Lehtinen Terro Karras, Timo Aila. Progressive growing of gans for improved quality, stability , and variation. 2018.

[19] Wojciech Zaremba Vicki Cheung Alec Radford Xi Chen Tim Salimans, Ian Goodfellow. Improved techniques for training gans.

Appendix



(a) Architecture of the generator.

(b) Architecture of the discriminator.

Figure 9: Architecture of our two models. Mostly inspired by [14].

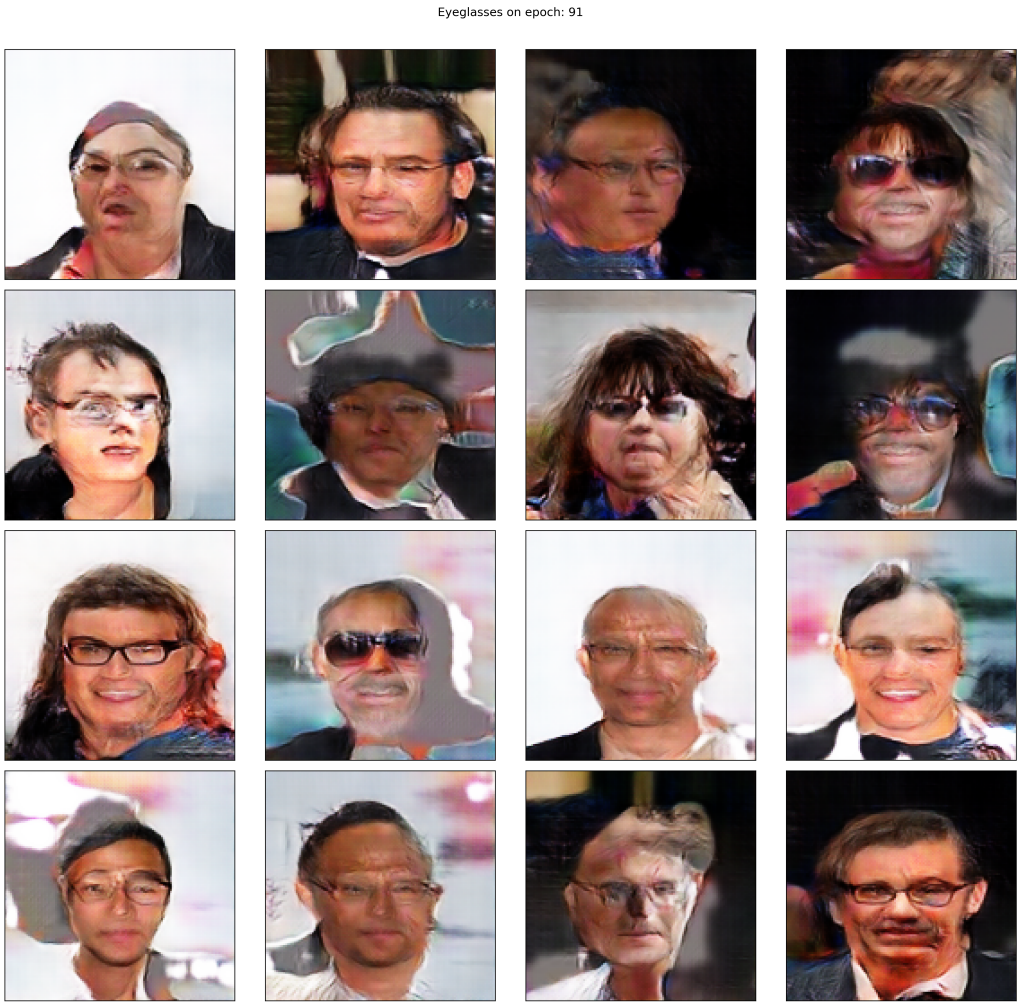


Figure 10: Figure containing generated faces with eyeglasses.

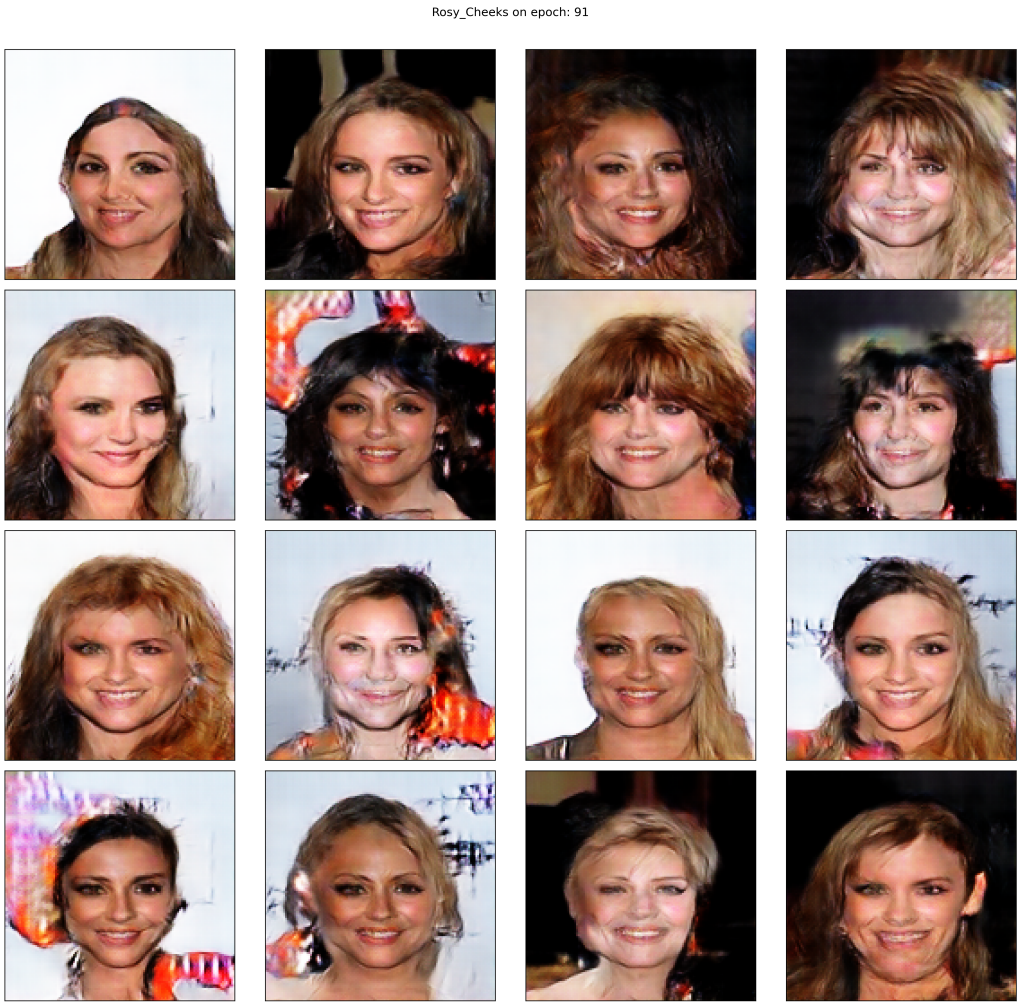


Figure 11: Figure containing generated faces with rosy cheeks.

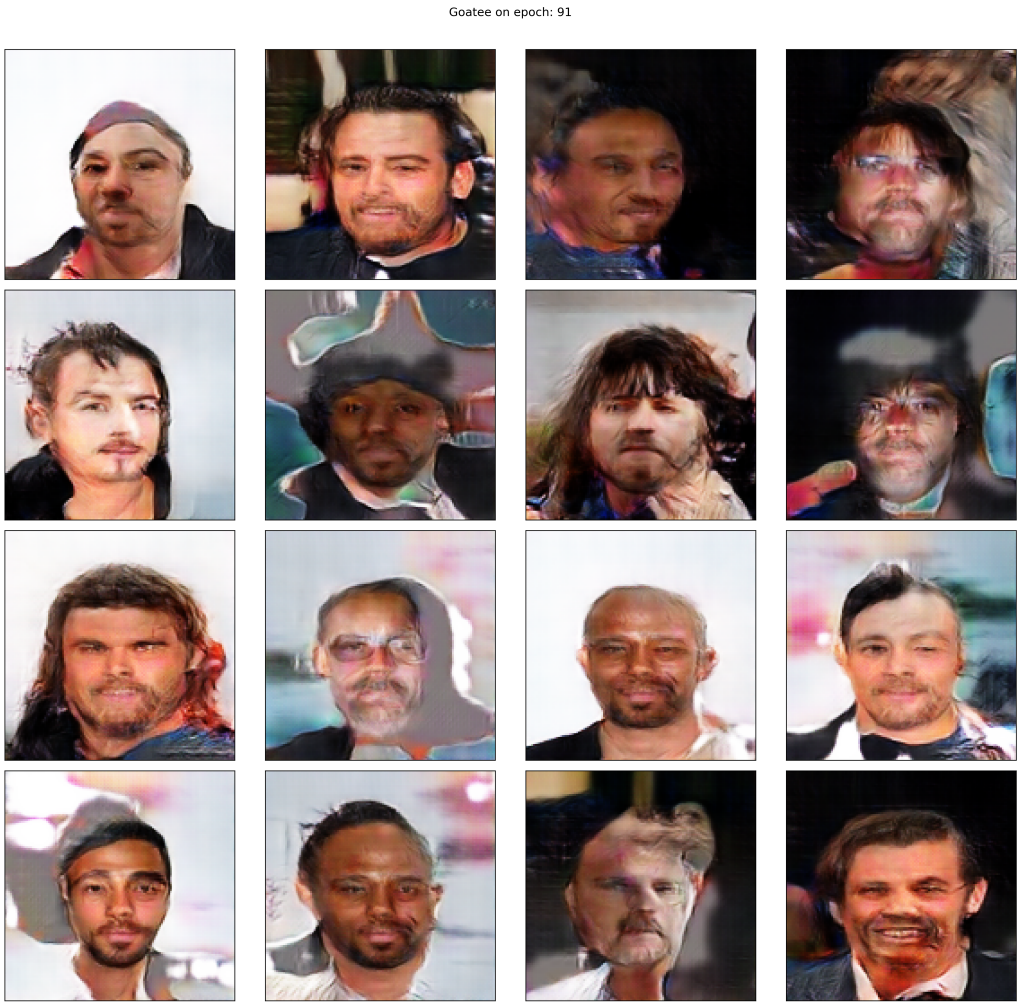


Figure 12: Figure containing generated faces with a goatee.

Figure 13: Figure containing generated faces conditioned by all 40 classes.