Towards High-Resolution Face Pose Synthesis

Douglas M. Souza
Pontifícia Universidade Católica
do Rio Grande do Sul
Av. Ipiranga, 6681
Porto Alegre, RS, Brazil
douglas.souza.002@acad.pucrs.br

Duncan D. Ruiz
Pontifícia Universidade Católica
do Rio Grande do Sul
Av. Ipiranga, 6681
Porto Alegre, RS, Brazil
duncan.ruiz@pucrs.br

***Abstract*—Synthesizing different views of a face image is a challenging task that can potentially help in several computer graphics and computer vision applications. In this work, we present a novel approach to address this task. We leverage the power of Generative Adversarial Networks (GANs) to synthesize face poses in a high-resolution and realistic fashion. We control the rotation of synthesized faces along the three axes of space (roll, pitch, yaw). We start by estimating the pose of each face in the training set and storing a vector containing the rotation angles. Then, we use the images along with the angles to train a conditioned version of a state-of-the-art GAN. Our experiments show image synthesis with a high-realistic finish, plus the absolute control of the pose of synthesized face images.**

I. Introduction

Processing face images are accompanied by a series of complexities, like variation of pose, light, face expression, and make up. Although all aspects are important, the one that impacts the most face-related computer vision applications is pose. In face recognition, for example, it has been long desired to have a method capable of bringing faces to the same pose, usually a frontal view, in order to ease recognition. Synthesizing different views of a face is still a great challenge, mostly because in non-frontal face images there are loss of information when one side of the face occludes the other (also known as self-occlusion). Several methods to address face pose synthesis were proposed [1]–[3], but the results still look artificial. Recently, a new generative method is helping to push forward the quality of face pose synthesis, this method is the Generative Adversarial Networks (GANs) [4].

GANs are a recent class of methods able to learn generative models over complex data distributions. They have shown remarkable results. Their capacity of learning very complex data distributions and their ability to generate high quality data samples attracted attention of both academia and industry. The power of GANs becomes evident when they are used to learn generative models over images, where they are able to synthesize sharper images when compared with other generative methods. The adversarial training introduced by GANs can be adapted to perform tasks beyond image synthesis. Some impressive results have been seen in tasks such as image-to-image translation [5]–[8], image inpainting [9], image editing [10], image super resolution [11], among others.

The idea behind using a GAN-based method for face pose synthesis is quite simple. Instead of taking care of all aspects related to the synthesis, like compensating for the information loss due to self-occlusion, we let the model learn a face representation and generalize to overcome issues like these. The most successful recent methods on pose synthesis are GAN-based [12]–[14]. In this case, GANs attempt to learn a *disentangled* representation of the face. Where we put some constraints on some dimensions of the learned representation, so we could control some features of synthesized images. We could, for example, choose to synthesize a face with or without sunglasses. There are different approaches to achieve feature disentanglement in GAN training. It is usually required the use of kind of supervision during training. In some cases this could be an issue, since labeled data may not be available. Ideally, we would like to have a way of synthesizing different views of a face having absolute control. Additionally, achieving such a solution having to use few or no labeled data at all, would be a leap of improvement.

In this work, we take advantage of the power of GANs to learn a disentangled representation of faces: we apply a conditioning method to control the rotation of synthesized faces along the three axes of space. First we estimate the rotation of the camera used to capture the image in Euler angles (roll, pitch, yaw). We use the angles to train a conditioned version of a high-resolution state-of-the-art GAN. To the best of our knowledge, all previous work on face pose synthesis treat the pose as discrete feature. Public datasets, like Multi-PIE [15], provide annotations of the angle of the pose of each face in the dataset as discrete feature, like $90°, 45°, 0°$. We, on the other hand, treat the pose as a continuous feature, with angles varying from $-75°$ to $75°$. The great advantage of having the pose as a continuous feature is that we have absolute control over the pose we would like to synthesize. Furthermore, we compute the pose using standard landmark locations, which can be extracted by face landmarks detectors, such as MTCNN [16], leaving behind the need of labeled data. Finally, our method can be applied in a variety of domains. It can be easily applied to perform data augmentation to improve training of face recognition algorithms, ease the job of face recognition systems in face matching, and even help in law enforcement.

978-1-5090-6014-6/18/$31.00 ©2018 IEEE

II. Background

A. Generative Adversarial Network (GANs)

Generative Adversarial Networks (GANs) [4] is a class of generative methods that learns generative models via an adversarial training process. In its traditional form, GANs are composed of two differentiable functions (e.g. neural networks), namely a Generator G and a Discriminator D. The generator and the discriminator are set to play a two-player minimax game. From an input noise z sampled from a simple, prior distribution $p_z(z)$ (e.g. uniform or Gaussian), the generator maps a sample $G(z)$ to the data space aiming to learn its own distribution p_g over the real data distribution $p_{data}(x)$. The z space is also known as the *latent* space. The discriminator D, on the other hand, takes an input data x and outputs a scalar, which is the probability that the input came from the real data $p_{data}(x)$ rather than from p_g. D is then trained to maximize the probability of assigning the correct class label for both the real data x and the fake data $G(z)$. The generator is trained simultaneously to make the discriminator mistakenly think that the data generated by the generator came from the real data distribution. In its classic form, the GAN objective is given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\left[log(D(x))\right] + \mathbb{E}_{z\sim p_z(z)}\left[log(1-D(G(z)))\right] \quad (1)$$

As showin in Eq. 1, each player aims to change the other player's cost function and they can change only their own parameters, therefore, this scenario is better described as a game rather than an optimization problem [17]. Because finding the solution for GAN training requires finding an *equilibrium* in high-dimensional space, GAN training is accompanied by a series of difficulties, such as non-convergence and mode collapse. Non-convergence is consequence of using gradient descent to finding the equilibrium of the game. If the capacity of G and D are not well balanced, it is possible that one of them wins easily, resulting in poor learning. Mode collapse, on the other hand, is one the most important issues in GAN research. During training, the generator may end up learning just one mode of the data, because it is more likely to fool the discriminator. In practice, complete mode collapse is rare, but partial mode collapse happens commonly. Since its first appearance, several successful improvements to GAN training have been proposed, including the Deep Convolutional GAN (DCGAN) [18], InfoGAN [19], LSGAN [20], Wasserstein GAN [21], among others.

B. Conditional GANs

In some cases, it may be desirable to have control over the data generated by the GAN generator (e.g. generate a face with a given face expression). This can be achieved by restricting some dimensions of the latent space to hold meaningful information. For a GAN trained in a dataset of faces, for example, some dimension of latent space could control hair color, while other could control face expression, and so on. In order to have this control, we need to learn a *disentangled* latent representation. Formally, we would like to provide a conditioning factor y for the generator alongside with the regular noise z and generate a sample $G(z|y)$ that correlates with the conditioning factor y.

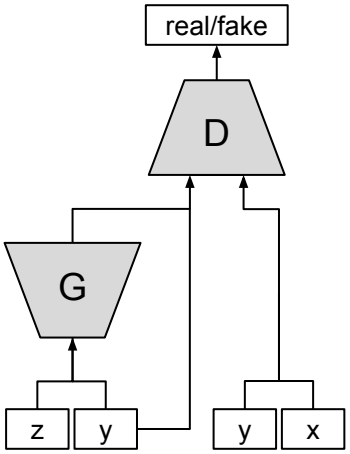


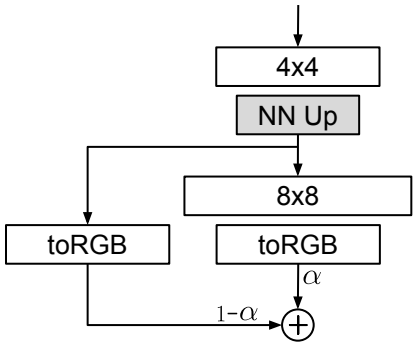
Fig. 1. Conditional GAN Scheme.

Currently, there are different approaches to train conditional GANs. One that is simple and proved to be very effective is the Conditional GAN (CGAN) [22] approach. In a CGAN, no additional loss term is required. The only difference to regular GAN training is that both generator and discriminator are provided side information during training. In Fig. 1, it is shown the CGAN scheme. The generator is fed with both a random noise z and a conditioning factor y and outputs a fake sample. The discriminator is trained to distinguish between the fake sample alongside with the conditioning factor y and the real sample alongside with the same conditioning factor y. In practice, the discriminator has more information to work with and, in order to the generator fool the discriminator, it has not only to generate a realistic sample, but also generate samples that correlate with the conditioning factor y.

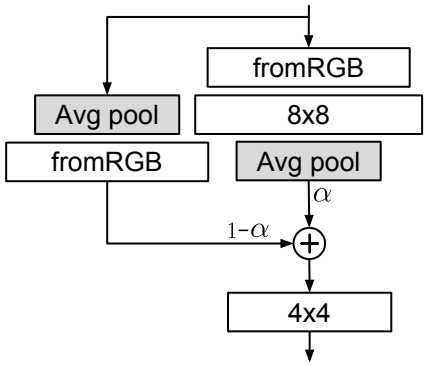
C. Progressive Growing of GANs

Issues like non-convergence and mode collapse present in GAN training become even more evident when training models at high-resolutions. Most previous works [17], [18], [21], [23] were able to reach resolutions up to 128×128 pixels. Recently, a new methodology for GAN training introduced by Karras et. al. [24] improved on these issues, allowing training of GANs of resolutions up to 1024×1024 pixels. The key idea is to progressively grow the generator and discriminator as training progresses. Training starts at a low resolution as 4×4 pixels. As training progresses, new layers are added on both discriminator and generator while all previous layers remain trainable. More layers are added until the target resolution for the model is reached. In a certain way, progressive growing, resembles layer-wise training of autoencoders [25].

Specifically, in progressive GAN, training alternates between two phases: *fade in* of new layers and *stabilization*



(a) Fading in a new layer in the generator.



(b) Fading in a new layer in the discriminator.

Fig. 2. Growing the progressive GAN networks.

of added layers. In order to preserve stability and not shock the networks, new layers are *faded in* smoothly. During a transition to a higher resolution, the networks operate at both the lower and higher resolution at the same time using a skip connection between layers. Fig. 2a and Fig. 2b show a transition from a 4×4 resolution to 8×8 resolution for the generator and discriminator, respectively. The weight α of the skip connection increases linearly until the transition is complete. After a transition is complete, the skip connection is discarded and the *stabilization* phase begins, where the networks are trained for more iterations before new layers could be added.

In the example of Fig. 2a, the *toRGB* layer projects the generator's output to 3 channels to form the RGB output image and *NN Up* is a layer that performs upsampling using nearest neighbor interpolation. In Fig. 2b, *fromRGB* is a layer that projects the RGB input image to the same number of channels as the next current convolutional layer and *Avg pool* is downsampling performed by average pooling. Both *toRGB* and *fromRGB* are usually composed by convolutions with filters of size 1×1.

III. Proposed Method

A. Pose Estimation

We start by approximating the camera matrix used to capture each image in the training set. We do so by seeking 2D-3D correspondences between face landmarks in the training

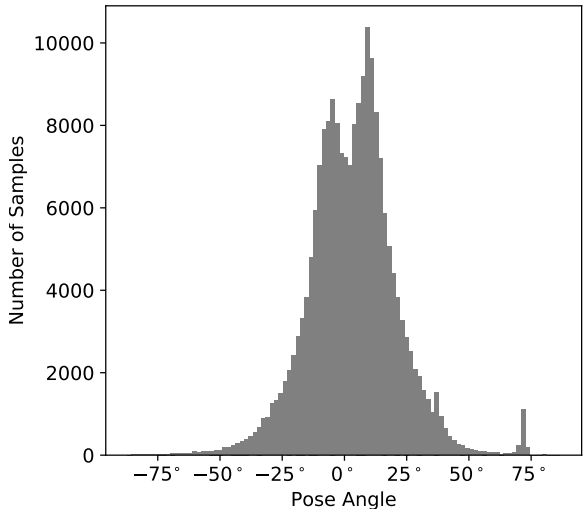


Fig. 3. Poses present in CelebA regarding y-axis (yaw). In this case, $0°$ means a complete frontal view of the face.

images and the same landmarks detected on the surface of a generic 3D face model provided by [1]. We approximate the camera matrix using standard camera calibration techniques. Once we compute the camera matrix, we extract the rotation matrix and convert it to Euler angles, yielding a vector containing the rotation along each axis of space $(r_x, r_y, r_z)^T$. Finally, we apply this vector as a conditioning factor to our Generative Adversarial Network.

B. Training Strategy

In order to synthesize high resolution images, we follow the steps of Karras et. al. [24] and extend a progressive growing GAN to be conditioned using the Conditional GAN method. We start training with images of size 4×4 pixel and carry out training doubling the resolution until we reach images of size 256×256 pixels. We apply the The Wasserstein GAN (WGAN) [21] training strategy. The WGAN brought a leap of improvement over previous training methodologies. The idea is to minimize the Earth-Mover distance (also known as Wasserstein-1) between the distribution of the real data $p_{data}(x)$ and generated data p_g. Another nice property is that we can train the WGAN discriminator (also known as *critic*, as it does not perform classification) to optimality under a Lipschitz continuity constraint. In the WGAN, this constraint was enforced by clipping the weights of the discriminator to fall in a compact space. Although it showed promising results, weight clipping leads to over-simplified functions. The Improved Wasserstein GAN (WGAN-GP) [23] improved on these issues. Instead of enforcing the Lipschitz constraint via weight clipping, the WGAN-GP achieves that using by adding a gradient penalty term to the discriminator loss function. Therefore, we chose the WGAN-GP loss function as it has been demonstrated to be stable and present very good results. The WGAN-GP loss function for the discriminator $\mathcal{L}_D$ is given by:

$$\mathcal{L}_D = \mathbb{E}[D(G(z))] - \mathbb{E}[D(x)] + \lambda_{GP}\mathbb{E}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] + \epsilon_{drift}\mathbb{E}[D(x)^2], \quad (2)$$

where $\hat{x}$ is gradient penalty input term, which is given by a point sampled along straight lines between real and generated data:

$$\hat{x} = \epsilon x + (1 - \epsilon)G(z). \quad (3)$$

Specifically, the value ϵ is randomly sampled from a uniform distribution $U[0, 1]$. An additional term $\mathbb{E}[D(x)^2]$ is also added to $\mathcal{L}_D$ with small weight ϵ_{drift} to avoid the loss drifting away from zero.

The loss for the generator $\mathcal{L}_G$ given by:

$$\mathcal{L}_G = -\mathbb{E}[D(G(z))] \quad (4)$$

It is important to note, however, that in Wasserstein GANs, $D(x)$ does not represent the probability that the input come from the real distribution as in traditional GAN loss (Eq. 1), $D(x)$ simple represents the raw output for the discriminator, where $\mathbb{E}[D(G(z))] - \mathbb{E}[D(x)]$ represent an estimate of the Wasserstein distance between the distribution of the real and generated data.

Rewriting the WGAN-GP loss to accommodate the conditioning factor we have the final objective for the discriminator as:

$$\mathcal{L}_D = \mathbb{E}[D(G(z|y), y)] - \mathbb{E}[D(x, y)] + \lambda\mathbb{E}[(\|\nabla_{\hat{x}} D(\hat{x}, y)\|_2 - 1)^2] + \epsilon_{drift}\mathbb{E}[D(x, y)^2] \quad (5)$$

And for the generator:

$$\mathcal{L}_G = -\mathbb{E}[D(G(z|y), y)] \quad (6)$$

Additionally, as in [24], we apply the following strategies to stabilize training and enforce diversity in the generator:

1) Equalized Learning Rate: In order to improve on poor weight initialization, weights are initialized using the method from He et. al. [26] and scaled by the c constant from the initialization method at runtime. This avoids the scenario where some weights have large dynamic range than others, which may cause the learning rate to be too small and to high at the same time.

2) Pixelwise Normalization in the Generator: In order to discourage unhealthy competition between the generator and discriminator, a pixelwise normalization is applied after every convolutional layer in the generator.

3) Minibatch of statistics: GANs naturally have difficulty to capture all the variance in the training data. Several methods have been proposed to improve on this and, therefore, improve the diversity of the samples from the generator. By default, the discriminator look at each sample individually. Some strategies try to enforce diversity by allowing the discriminator look at an entire minibatch of samples and encourage the minibatch to have a high variance, the Minibatch discrimination from Salimans et. al. [27] is an example. A simpler solution, however, is proposed in [24], where the standard deviation for each feature in each spatial location is computed and then averaged, yielding a single number, which is replicated to form an additional feature map that is inserted at end of the discriminator. This solution seems effective to enforce some diversity to the generator. Without this trick, however, we found partial mode collapse to happen during training.

TABLE I
GENERATOR

Layer	Filter Size / Stride	Padding	Output Shape	# Params
Input (z, y)	-	-	$1 \times 1 \times 515$	0
Conv1	$4 \times 4/1$	3	$4 \times 4 \times 512$	4,219,392
Conv2	$3 \times 3/1$	1	$4 \times 4 \times 512$	2,359,808
NN Up.	-	-	$8 \times 8 \times 512$	0
Conv3	$3 \times 3/1$	1	$8 \times 8 \times 512$	2,359,808
Conv4	$3 \times 3/1$	1	$8 \times 8 \times 512$	2,359,808
NN Up.	-	-	$16 \times 16 \times 512$	0
Conv5	$3 \times 3/1$	1	$16 \times 16 \times 512$	2,359,808
Conv6	$3 \times 3/1$	1	$16 \times 16 \times 512$	2,359,808
NN Up.	-	-	$32 \times 32 \times 512$	0
Conv7	$3 \times 3/1$	1	$32 \times 32 \times 512$	2,359,808
Conv8	$3 \times 3/1$	1	$32 \times 32 \times 512$	2,359,808
NN Up.	-	-	$64 \times 64 \times 512$	0
Conv9	$3 \times 3/1$	1	$64 \times 64 \times 512$	2,359,808
Conv10	$3 \times 3/1$	1	$64 \times 64 \times 512$	2,359,808
NN Up.	-	-	$128 \times 128 \times 512$	0
Conv11	$3 \times 3/1$	1	$128 \times 128 \times 256$	1,179,904
Conv12	$3 \times 3/1$	1	$128 \times 128 \times 256$	590,080
NN Up.	-	-	$256 \times 256 \times 256$	0
Conv13	$3 \times 3/1$	1	$256 \times 256 \times 128$	295,040
Conv14	$3 \times 3/1$	1	$256 \times 256 \times 128$	147,584
Conv15	$1 \times 1/1$	0	$256 \times 256 \times 3$	387
Total				27,670,659

TABLE II
DISCRIMINATOR

Layer	Filter Size / Stride	Padding	Output Shape	# Params
Input (x, y)	-	-	$256 \times 256 \times 6$	0
Conv1	$1 \times 1/1$	0	$256 \times 256 \times 64$	448
Conv2	$3 \times 3/1$	1	$256 \times 256 \times 64$	36,928
Avg. Pool	-	-	$128 \times 128 \times 64$	0
Conv3	$3 \times 3/1$	1	$128 \times 128 \times 128$	73,856
Conv4	$3 \times 3/1$	1	$128 \times 128 \times 128$	147,584
Avg. Pool	-	-	$64 \times 64 \times 128$	0
Conv5	$3 \times 3/1$	1	$64 \times 64 \times 256$	295,168
Conv6	$3 \times 3/1$	1	$64 \times 64 \times 256$	590,080
Avg. Pool	-	-	$32 \times 32 \times 256$	0
Conv7	$3 \times 3/1$	1	$32 \times 32 \times 512$	1,180,160
Conv8	$3 \times 3/1$	1	$32 \times 32 \times 512$	2,359,808
Avg. Poo	-	-	$16 \times 16 \times 512$	0
Conv9	$3 \times 3/1$	1	$16 \times 16 \times 512$	2,359,808
Conv10	$3 \times 3/1$	1	$16 \times 16 \times 512$	2,359,808
Avg. Pool	-	-	$8 \times 8 \times 512$	0
Conv11	$3 \times 3/1$	1	$8 \times 8 \times 512$	2,359,808
Conv12	$3 \times 3/1$	1	$8 \times 8 \times 512$	2,359,808
Avg. Pool	-	-	$8 \times 8 \times 512$	0
MB stats	-	-	$4 \times 4 \times 513$	0
Conv13	$3 \times 3/1$	1	$4 \times 4 \times 512$	2,359,808
Conv14	$4 \times 4/1$	0	$1 \times 1 \times 512$	4,203,008
Flatten	-	-	512	0
FC	-	-	1	513
Total				20,686,681

Fig. 4. Random samples from the generator.

C. Model Architecture

We use architectures with a decaying number of convolutional filters in the discriminator and generator. The architecture for both is identical, except that the generator has more convolutional filters. The generator has about 27M parameters while the discriminator have about 20M.

1) Generator: The generator input is a noise tensor $z \in \mathbb{R}^{1\times1\times512}$ sampled from a prior normal distribution $\mathcal{N}(0, 1)$. The tensor z is concatenated with a conditioning tensor $y \in \mathbb{R}^{1\times1\times3}$ that represents the rotation $(r_x, r_y, r_z)^T$ of the face along the three axes of space, yielding a tensor of dimension $1 \times 1 \times 515$. Unlike a common setup in GANs where a fully connected layer is used to project z to higher dimensional space and form 3-dimensional convolutional volume, the generator starts by applying convolutions direct to z using padding to compensate to the small spatial dimensions of the z tensor. We start with many small feature maps, then a series of convolutions and upsampling operations convert this high level representation to a 256×256 pixel image. Upsampling is performed by Nearest Neighbour interpolation. Every convolutional layer is followed by a Leaky Rectified Linear Unit (Leaky Relu) activation with a leaky factor of 0.2, except the last one, which is not followed by any activation. The complete architecture for the generator is shown in Table I.

2) Discriminator: The discriminator has as inputs real images x and generated images $G(z|y)$, both concatenated with a conditioning vector $y \in \mathbb{R}^{1\times1\times3}$. We concatenate the conditioning vector in the feature map axis of the discriminator input, therefore, an RGB image with dimensions $4 \times 4 \times 3$ would result in a $4 \times 4 \times 6$ dimensional volume. This volume is followed by a series of convolutions. Downsampling is performed by average pooling. Each average pooling layer reduces the spatial dimensions by half. Every convolutional layer is followed by a Leaky Relu activation. Finally, in the last block, we concatenate the minibatch of statics in the feature map axis of the input for the first convolutional layer. After the last convolution, the volume is reshaped and linearly projected to form the discriminator output, which is 512-dimensional representation. The complete discriminator architecture is shown in Table II.

IV. Experiments

A. Dataset

We use the aligned version of CelebA Dataset [28] for training. CelebA is composed of 202,055 images of 10,177 identities. Along with the images, annotations for 40 binary attributes and 5 landmark location for each face are provided. We use the full dataset as training set. As a form of compensating for biases in pose, we add to the dataset a horizontally flipped version of each image. Along with the images, for each one, we compute the rotation vector, $(r_x, r_y, r_z)^T$ using landmark annotations provided in the dataset. Fig. 3 shows the distributions of poses along the y-axis of space (yaw). The rotation vector is then scaled to be in the range $[0, 1]$. In order to bring the images to a square format and maintain the aspect ratio, we perform a center crop in the images of the size of its smallest dimension. We then resize the images to match the current size being used in training. Finally, we scale the pixel values of all images to be in the range $[-1, 1]$.

B. Implementation Details

We train the generator and discriminator using simultaneous gradient descent. In other words, at each training step we update the weights of D and G, respectively. We use Adam optimizer [29] with a learning rate of 0.001, $\beta_1 = 0$ and $\beta_2 = 0.99$ for both G and D. The weight for the gradient penalty term λ_{GP} is set to 10 and drift penalty ϵ_{drift} is set to 0.001. We show the discriminator 800k real images for the *stabilizing* phase and 800k real images for the *fade in* phase. We grow the networks until we reach the target resolution of 256×256 pixels. During training, we generate images every

(a) Rotating a face by fixing the noise and evenly changing the conditioning factor from -75° to 75°.

(b) Results from Masi et. al. [3] at 0°, 40°, 75°

Fig. 5. Results.

Fig. 6. Interpolation between two random faces in latent space.

500 steps for visual inspection reasons. Training time was 6 days, reported on a single NVIDIA 1080 Ti GPU using PyTorch framework.

C. Inverse Mapping of the Generator

GANs are generative models that learn an implicit distribution p_g over the real data distribution $p_{data}(x)$, meaning that there is no direct mapping from an input x to latent space z. In this sense, it is not possible to directly map a face from a given person to latent space, modify the pose and reconstruct. A natural approach would be to train an encoder network to learn the mapping $f_\theta : x \mapsto z$. The ICGAN [30] approach at temps to learn that mapping for conditional GANs, specifically, with two encoders, one to map $f_\theta : x \mapsto z$ and another to map $f_\theta : x \mapsto y$, where y is the conditioning factor. In our experiments, such approach did not work. We believe that there are two main reasons why it did not succeeded. First, prior GAN training used to employ latent space of lower dimensional size (e.g. 128-dimensional or smaller) as opposed as ours (512-dimensional). Learning a mapping to small latent space is a lot easier. Secondly, a consequence of progressive growing training is that generator becomes a highly non-linear function, meaning that is non-trivial to train another network to approximate its inverse. Finally, it is possible to use gradient descent to approximate the latent tensor z that would generate a given image x by minimizing the L1 reconstruction error of the generator ($|G(z) - x|$). This is an approach that is not practical for real world applications, but should give a fair reconstruction of the given image x. In our experiments, we used Adam to approximate the z tensor for test images, but surprisingly, it had difficulty to do so, taking a long time to approximate and not yielding good results. Our experience showed to be difficult to perform the inverse mapping of the progressive GAN generator. To the best of our knowledge, there are no previous work that tackles this problem. We believe this is an issue that requires an in-depth investigation.

D. Evaluation

A great challenge in working with generative models is that there is not a clear way of how to perform a quantitative evaluation. Usually, generative models are used as part of another application, where we can measure the overall impact on the final application only. In our case, we perform a qualitatively evaluation employing visual inspection on generated samples. First we show that our learned latent representation is coherent. Fig. 6 shows interpolation in latent space between two randomly generated faces. Furthermore, in Fig. 5a we show our method can indeed learn a disentangled representation. We fix the input noise z and by varying the conditioning factor we can rotate the face preserving the identity of the person. A key aspect is that the CelebA dataset is composed mostly of near frontal faces (as shown in Fig. 3), and even so, our model can generalize and generate poses in more extreme angles. Altough we capture the rotation of the face along the three axes of space, we found rotation along other axes to have very low variety in the training set (it is rare for people to be looking up or down, for example). Therefore, rotation along other axes do not present sharp results. Compared to previous methods (Fig. 5b), our method can synthesize better looking images, preserving semantics of the face. As noted previously, a drawback is that there is not a way to perform the inverse mapping of the generator, especially in the case of progressive GANs.

V. Related Work

Synthesizing face poses has been long desired in face-related computer vision tasks. Most of prior work focus on synthesizing a frontal view of the face, also known as *face frontalization*, aiming to aid face recognition. In this sense, previous works [1], [2] make use of classic computer graphics algorithms to bring faces to a frontal view. The challenging part of such methods is that they need to take care of compensating for information loss due to self-occlusion, which

may compromise the quality of final results. More recently, other methods that employ Deep Learning have been proposed [12]–[14]. These methods brought a leap of improvement but still lack the ability to generate realistic images. Generating a frontal view of the face may aid face recognition. However, we argue that generating faces in different poses may help face recognition even further. In [3], for example, it is proposed a method to synthesize faces in different poses. Although the goal was to perform data augmentation for training face recognition algorithms, the synthesis at test time showed promising results in helping improving face matching.

VI. Conclusion

We present a novel method for face pose synthesis: we apply a conditioning method to control the rotation of synthesized faces along the three axes of space (roll, pitch, yaw). We compute the angles of the face in space and use the angles to train a conditioned version of a state-of-the-art GAN. We treat the pose as a continuous feature, with angles varying from -75° to 75°. The great advantage of having the pose as a continuous feature in latent space is that we have absolute control over the pose we would like to synthesize. Furthermore, our method does not require training data to have annotations of any kind, we can estimate the pose using standard face landmark detectors. Finally, this method can be easily applied to perform data augmentation to improve training of face recognition algorithms, ease the job of face recognition systems in face matching, and even help in law enforcement.

References

[1] T. Hassner, S. Harel, E. Paz, and R. Enbar, "Effective face frontalization in unconstrained images," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2015. [Online]. Available: http://www.openu.ac.il/home/hassner/projects/frontalize

[2] X. Zhu, Z. Lei, J. Yan, D. Yi, and S. Z. Li, "High-fidelity pose and expression normalization for face recognition in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 787–796.

[3] I. Masi, A. T. Trn, T. Hassner, J. T. Leksut, and G. Medioni, "Do we really need to collect millions of faces for effective face recognition?" in *European Conference on Computer Vision*. Springer, 2016, pp. 579–596.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint*, 2017.

[6] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," *arXiv preprint arXiv:1711.09020*, 2017.

[7] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," *arXiv preprint arXiv:1711.11585*, 2017.

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.

[9] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[10] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *European Conference on Computer Vision*. Springer, 2016, pp. 597–613.

[11] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv:1609.04802*, 2016.

[12] L. Tran, X. Yin, and X. Liu, "Representation learning by rotating your faces," *arXiv preprint arXiv:1705.11136*, 2017.

[13] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker, "Towards large-pose face frontalization in the wild," *arXiv preprint arXiv:1704.06244*, 2017.

[14] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis," *arXiv preprint arXiv:1704.04086*, 2017.

[15] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "Multi-pie," *Image and Vision Computing*, vol. 28, no. 5, pp. 807–813, 2010.

[16] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, Oct 2016.

[17] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[18] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[19] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2016, pp. 2172–2180.

[20] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 2813–2821.

[21] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[22] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[23] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[24] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[25] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[27] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[28] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[30] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, "Invertible conditional gans for image editing," *arXiv preprint arXiv:1611.06355*, 2016.