The Thirty-First International Florida
Artificial Intelligence Research Society Conference (FLAIRS-31)

GAN-Based Realistic Face Pose Synthesis with Continuous Latent Code

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

Abstract

We present a novel approach for face pose synthesis. We leverage the power of Generative Adversarial Networks to synthesize face poses in a realistic fashion. We apply a conditioning method to 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 Generative Adversarial Network. Our experiments show image synthesis with state-of-the-art quality, plus the absolute control of the pose of synthesized face images.

1 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 (Hassner et al. 2015; Zhu et al. 2015), but the results still look synthetic. Recently, a new generative method is helping to push forward the quality of face pose synthesis, this method is the Generative Adversarial Networks (GANs) (Goodfellow et al. 2014).

GANs are a recent class of methods able to learn generative models over complex data distributions. They have shown remarkable results, it is safe to say that currently they are one of the most popular topics in the Deep Learning field. 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. Furthermore, GANs are a framework relatively easy to extend. Several works apply GANs with different settings for a great variety of applications. An important aspect in GANs is that we can extend it to use supervision, allowing us to control some features of the latent space.

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 (Tran, Yin, and Liu 2017; Yin et al. 2017; Huang et al. 2017). Generally, GANs are trained to learn a disentangled representation of the face. In other words, it is desired to put meaningful information in some dimensions of the latent space, so we can control some features of synthesized images. We could, for example, choose to synthesize a face with or without sunglasses. Even though it is possible, in some cases, to learn disentangled representations in a complete unsupervised manner (Chen et al. 2016), most works achieve feature disentanglement using supervision.

In this work, we leverage 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 camera matrix used to capture the image, we then extract the rotation matrix and convert it to Euler angles (roll, pitch, yaw). We use the angles to train a conditioned version of a 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 (Gross et al. 2010), 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 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 to estimate the pose. We can perform pose estimation using a standard face landmark detector like MTCNN (Zhang et al. 2016). Finally, this 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 face matching, and even help in law enforcement.

2 Related Work

2.1 Generative Adversarial Network (GANs)

In recent developments in Deep Learning, Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) were introduced. Generative adversarial nets are composed of two differentiable functions, 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 prior $p_z(z)$, the generator maps a sample $G(z)$ to the data space aiming to learn its own distribution p_g over the real data x. The discriminator D takes an input data x and outputs a scalar, which is the probability that the input came from the real 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 each player aims to change the other player's cost function and they can change only their own parameters, this scenario is better described as a game rather than an optimization problem (Goodfellow 2016). Since its first appearance, several successful extensions have been proposed, including the Deep Convolutional GAN (DCGAN) (Radford, Metz, and Chintala 2015), InfoGAN (Chen et al. 2016), Wasserstein GAN (Arjovsky, Chintala, and Bottou 2017), and others.

2.2 Pose Synthesis

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, some works (Hassner et al. 2015; Zhu et al. 2015) 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 (Yin et al. 2017; Tran, Yin, and Liu 2017; Huang et al. 2017). 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 matching even further. In (Masi et al. 2016), 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 helped improving face matching.

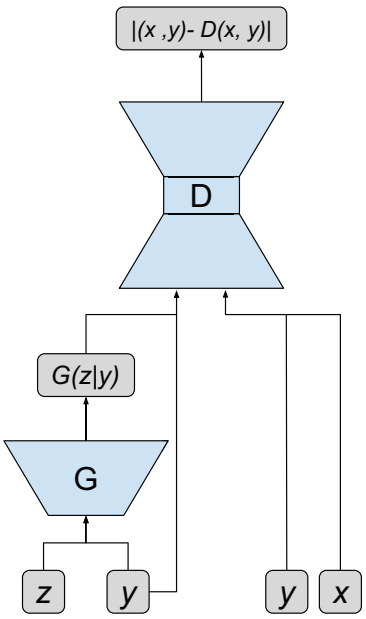

Figure 1: The Conditional BEGAN Model.

3 Proposed Method

In order to learn a disentangled representation and control face rotation along the three axes of space (roll, pitch, yaw), we first compute the extrinsic camera parameters using a generic 3D face model, then we use the camera rotation angles as labels to train a Conditional Generative Adversarial Network (Mirza and Osindero 2014).

3.1 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 images and the same landmarks detected on the surface of a generic 3D face model provided by (Hassner et al. 2015). 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 input to our Generative Adversarial Network.

3.2 Training Strategy

We build our method over the recent Boundary Equilibrium Generative Adversarial Network (BEGAN) (Berthelot, Schumm, and Metz 2017). The BEGAN framework introduces a successful equilibrium enforcing method that helps stabilizing training while allowing to control the trade-off between generated image quality and diversity. We extend the BEGAN framework to be conditioned using the Conditional Adversarial Network (CGAN) (Mirza and Osindero 2014) method. In this sense, we define our generator G as a regular Convolutional Neural Network (CNN) and the discriminator D as a Convolutional Auto-encoder (CAE)

(Masci et al. 2011). We extend the generator to be conditioned to a y vector. Therefore, from an input noise z sampled from a prior distribution, and a conditioning vector y, the generator maps a sample $G(z|y)$ to the data space x. The discriminator takes as input the real data x and the generated data $G(z|y)$, both along with a conditioning vector y, and outputs the autoencoder reconstruction for both the real data $D(x, y)$ and the generated data $D(G(z|y), y)$, respectively. The Conditional BEGAN (CBEGAN) model is shown in Fig 1. In order to define the cost function for the CBEGAN, first let's consider a simple pixel-wise reconstruction loss for a CAE:

$$\mathcal{L}(x) = |x - D(x)| \tag{2}$$

where x is training example and $D(x)$ is the CAE function that produces an output with same dimension as x. Adapting the CAE loss to accommodate the conditioning vector y we have:

$$\mathcal{L}(x, y) = |(x, y) - D(x, y)| \tag{3}$$

Therefore, we can define the CBEGAN objective function as follows:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x, y; \theta_D) - \mathcal{L}(G(z|y; \theta_G), y; \theta_D) & \text{for } \theta_D \\ \mathcal{L}_G = -\mathcal{L}_D & \text{for } \theta_G \end{cases} \tag{4}$$

where $\mathcal{L}_D$ is the loss for the discriminator, $\mathcal{L}_G$ is the loss for the generator, θ_D and θ_G are the discriminator and the generator parameters, respectively. For simplicity we abbreviate and omit the parameters θ_G and θ_D for G and D. In the BEGAN framework the equilibrium is relaxed through a new hyper-parameter $\gamma \in [0, 1]$ that controls the trade-off between image quality and diversity. For example, when γ is close to 1, quality is high and variety is low, when γ is close to 0, quality is low and variety is high. Thus, we consider our CBEGAN to be at equilibrium when:

$$\gamma = \frac{\mathbb{E}\left[\mathcal{L}(G(z|y), y)\right]}{\mathbb{E}\left[\mathcal{L}(x, y)\right]} \tag{5}$$

To ensure the equilibrium shown in Eq. 5, BEGAN applies Proportional Control Theory through a variable k_t. This variable dynamically adjusts the emphasis needed to put in the generator. Adding the equilibrium method above, we have complete CBEGAN objective function:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x, y) - k_t \mathcal{L}(G(z|y), y) \\ \mathcal{L}_G = \mathcal{L}(G(z|y), y) \\ k_{t+1} = k_t + \lambda_k(\gamma \mathcal{L}(x, y) - \mathcal{L}(G(z|y), y)) \end{cases} \tag{6}$$

where $k_t \in [0, 1]$ controls how much emphasis is put in the generator during gradient descent at a step t, and $\lambda_k \in [0, 1]$ controls the update size for k. At the initial step, k_0 is set to 0. Usually, in the beginning of GAN training the discriminator is not able to distinguish well between real samples and fake samples, which results in small gradients for the generator and, consequently, poor learning. The equilibrium method introduced by BEGAN ensures that more emphasis is put to the discriminator in the beginning, so balancing the generator and the discriminator losses becomes a less critical issue.

Table 1: Discriminator

Layer	Filter Size / Stride	Output Shape	# Params
Input (x, y)	-	$128 \times 128 \times 6$	0
Conv1	$3 \times 3/1$	$128 \times 128 \times 128$	7,040
Conv2	$3 \times 3/1$	$128 \times 128 \times 128$	147,584
Conv3	$3 \times 3/1$	$128 \times 128 \times 128$	147,584
Conv4	$3 \times 3/2$	$64 \times 64 \times 128$	147,584
Conv5	$3 \times 3/1$	$64 \times 64 \times 128$	147,584
Conv6	$3 \times 3/1$	$64 \times 64 \times 128$	147,584
Conv7	$3 \times 3/2$	$32 \times 32 \times 128$	147,584
Conv8	$3 \times 3/1$	$32 \times 32 \times 128$	147,584
Conv9	$3 \times 3/1$	$32 \times 32 \times 128$	147,584
Conv10	$3 \times 3/2$	$16 \times 16 \times 128$	147,584
Conv11	$3 \times 3/1$	$16 \times 16 \times 128$	147,584
Conv12	$3 \times 3/1$	$16 \times 16 \times 128$	147,584
Conv13	$3 \times 3/2$	$8 \times 8 \times 128$	147,584
Conv14	$3 \times 3/1$	$8 \times 8 \times 128$	147,584
Conv15	$3 \times 3/1$	$8 \times 8 \times 128$	147,584
Reshape	-	8192	0
FC1	-	128	1,048,704
FC2	-	8192	1,056,768
Reshape	-	$8 \times 8 \times 128$	0
Conv16	$3 \times 3/1$	$8 \times 8 \times 128$	147,584
Conv17	$3 \times 3/1$	$8 \times 8 \times 128$	147,584
NN Up.	-	$16 \times 16 \times 128$	0
Conv18	$3 \times 3/1$	$16 \times 16 \times 128$	147,584
Conv19	$3 \times 3/1$	$16 \times 16 \times 128$	147,584
NN Up.	-	$32 \times 32 \times 128$	0
Conv20	$3 \times 3/1$	$32 \times 16 \times 128$	147,584
Conv21	$3 \times 3/1$	$32 \times 16 \times 128$	147,584
NN Up.	-	$64 \times 64 \times 128$	0
Conv22	$3 \times 3/1$	$64 \times 16 \times 128$	147,584
Conv23	$3 \times 3/1$	$64 \times 16 \times 128$	147,584
NN Up.	-	$128 \times 128 \times 128$	0
Conv24	$3 \times 3/1$	$128 \times 128 \times 128$	147,584
Conv25	$3 \times 3/1$	$128 \times 128 \times 128$	147,584
Conv26	$3 \times 3/1$	$128 \times 128 \times 6$	6,918
Total			5,661,446

3.3 Model Architecture

We use architectures almost identical as BEGAN, except that we use a constant number of convolutional filters in the discriminator. The reason to use less filters in the discriminator is to reduce the number of parameters and, consequently, training time. In our experiments, this reduction did not seem to impact the overall results. We train two models, one that generates 64×64 pixel images and other that generates 128×128 pixel images. The architecture for both is identical, except that 64×64 pixel model has less convolutional layers, 2 in the generator and 6 in the discriminator. Our 128×128 model has around 8.3M parameters in total, while a similar BEGAN model has around 19M parameters.

Generator The generator input is a noise vector $z \in \mathbb{R}^{128}$ sampled uniformly from a prior distribution in the range $[-1, 1]$. The vector z is concatenated with a conditioning

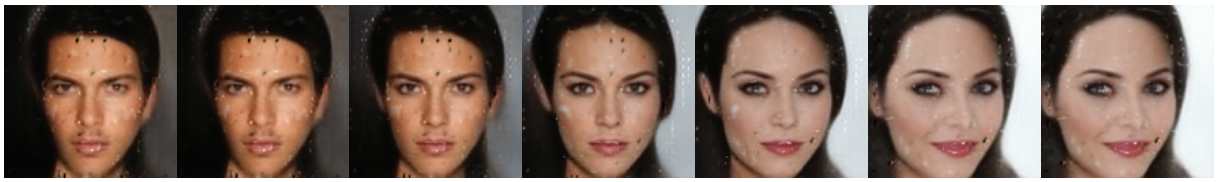

Figure 2: Interpolation in latent space between two randomly sampled faces at 128×128 pixels.

vector $y \in \mathbb{R}^3$ that represents the rotation $(r_x, r_y, r_z)^T$ of the face along the three axes of space. The generator input is then linearly projected to a higher dimensional space using a fully connected layer, which is then reshaped to form 3-dimensional convolutional volume. We start with many small feature maps, then a series of convolutions and upsampling operations convert this high level representation to a 128×128 pixel image. Upsampling is performed by Nearest Neighbour interpolation. Every convolutional layer is followed by an Exponential Linear Unit (ELU) (Clevert, Unterthiner, and Hochreiter 2015) activation, except the last one, which is not followed by any activation. The complete architecture for the generator is shown in Table 2.

Discriminator The discriminator is a Convolutional Auto-encoder (CAE), which is composed of an encoder and a decoder. The encoder has as inputs real images x and generated images $G(z|y)$, both concatenated with a conditioning vector $y \in \mathbb{R}^3$. We concatenate the conditioning vector in the feature map axis of the discriminator input, therefore, an RGB image with dimensions $128 \times 128 \times 3$ would result in a $128 \times 128 \times 6$ dimensional volume. This volume is followed by a series of convolutions. Downsampling is performed with strided convolutions. Each convolution with stride 2 reduces the spatial dimensions by half. Every convolutional layer is followed by an ELU activation. Finally the volume is reshaped and linearly projected to form 128-dimensional intermediate representation. The decoder part is identical to the generator architecture. It maps the 128-dimensional representation to a $128 \times 128 \times 6$ volume, the same dimensions as encoder input. The complete discriminator architecture is shown in Table 1.

Table 2: Generator

Layer	Filter Size / Stride	Output Shape	# Params
Input (z, y)	-	131	0
FC	-	8192	1,081,344
Reshape	-	$8 \times 8 \times 128$	0
Conv1	$3 \times 3/1$	$8 \times 8 \times 128$	147,584
Conv2	$3 \times 3/1$	$8 \times 8 \times 128$	147,584
NN Up.	-	$16 \times 16 \times 128$	0
Conv3	$3 \times 3/1$	$16 \times 16 \times 128$	147,584
Conv4	$3 \times 3/1$	$16 \times 16 \times 128$	147,584
NN Up.	-	$16 \times 16 \times 192$	0
Conv5	$3 \times 3/1$	$32 \times 32 \times 128$	147,584
Conv6	$3 \times 3/1$	$32 \times 32 \times 128$	147,584
NN Up.	-	$64 \times 64 \times 128$	0
Conv7	$3 \times 3/1$	$64 \times 64 \times 128$	147,584
Conv8	$3 \times 3/1$	$64 \times 64 \times 128$	147,584
NN Up.	-	$128 \times 128 \times 128$	0
Conv9	$3 \times 3/1$	$128 \times 128 \times 128$	147,584
Conv10	$3 \times 3/1$	$128 \times 128 \times 128$	147,584
Conv11	$3 \times 3/1$	$128 \times 128 \times 3$	3,459
Total			2,560,643

4 Experiments

4.1 Setup

Training Set We train our models using the aligned version of CelebA Dataset (Liu et al. 2015). 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. The rotation vector is then scaled to be in the range $[-1, 1]$. We perform a center crop in the images according to a fixed bounding box around the face. We then resize the image to match the size of the images we want to synthesize. Finally, we scale the pixel values of all images to be in the range $[-1, 1]$ as well.

4.2 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 (Kingma and Ba 2014) with a learning rate of 0.0008, $\beta_1 = 0.9$ and $\beta_2 = 0.99$ for both G and D. The weights are initialized using the Xavier method (Glorot and Bengio 2010). We set the quality/diversity ratio parameter $\gamma = 0.5$ and the learning rate of the equilibrium method parameter $\lambda_k = 0.001$. We carry out training for 20 epochs using minibatches of size of 64 for the network that produces 64×64 pixel images, and 11 epochs with minibatches of size 32 for the network that produces 128×128 pixel images. We generate images every 500 steps for visual inspection reasons. Training time was 2 days for the 64×64 pixel model and 6 days for 128×128 pixel model, both reported on a single NVIDIA Titan X (Pascal) GPU using Tensorflow framework.

4.3 Evaluation

A great challenge in working with generative models is that there is not a clear way of how to perform a quantitative evaluation. Instead, we evaluate qualitatively employing vi-

Figure 3: On each row: samples generated with a fixed z at 64×64 pixels varying only the code for pose with evenly spaced degrees from $-75°$ to $75°$.

Figure 4: On each row: samples generated with a fixed z at 128×128 pixels varying only the code for pose with evenly spaced degrees from $-75°$ to $75°$.

Figure 5: Results from (Masi et al. 2016). On each column group: pose generated at $0°$, $40°$ and $75°$, respectively.

sual inspection on generated samples. First we show that our learned latent representation is coherent. Fig. 2 shows interpolation in latent space between two randomly generated faces. Furthermore, in Fig. 3 and 4 we show our method can indeed learn a disentangled representation. We fix the input noise z and by varying the pose code 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,

and even so, our model can generalize and generate poses in more extreme angles. Although we apply rotation along three axes of space, we found that rotation along the x-axis and z-axis have a very low variation in the training data (it is rare for people in the training set to look up/down, for example) and therefore, changes along those axes do not yield the same level of quality. Compared to previous methods (Fig. 5), our method can synthesize better images, preserving semantics of the face. A downside of our method is that there is no direct way of mapping a face image to its respective z, meaning that we cannot generate poses of a given face directly. It is possible, however, to use gradient descent to approximate a noise z that would generate a given face, but this is not practical for real world applications.

5 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 to estimate the pose. Finally, this method can be applied in a variety of domains. It can be easily applied to perform data augmentation, aid face recognition, and even help in law enforcement.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Berthelot, D.; Schumm, T.; and Metz, L. 2017. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*.

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

Clevert, D.-A.; Unterthiner, T.; and Hochreiter, S. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256.

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

Goodfellow, I. 2016. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Gross, R.; Matthews, I.; Cohn, J.; Kanade, T.; and Baker, S. 2010. Multi-pie. *Image and Vision Computing* 28(5):807–813.

Hassner, T.; Harel, S.; Paz, E.; and Enbar, R. 2015. Effective face frontalization in unconstrained images. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

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

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

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

Masci, J.; Meier, U.; Cireşan, D.; and Schmidhuber, J. 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011* 52–59.

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

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

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

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

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

Zhang, K.; Zhang, Z.; Li, Z.; and Qiao, Y. 2016. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters* 23(10):1499–1503.

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