Final Project Report

AI20BTECH11007: Digjoy Nandi

AI20BTECH11017: Omkaradithya Pujari

AI20BTECH11018: Perambuduri Srikaran

AI20BTECH11025: Vaishnavi W

AI20BTECH11028: Ayush Kumar Singh

Paper ID

Abstract

GANs are generative models where we use adversarial process and simultaneously train two models, a generative model and a discriminative model. The generative model G captures the data distribution, and the discriminative model D estimates the probability that a sample came from the training data rather than G. It uses implicit density to generate new samples, unlike traditional methods such as AutoEncoders which minimize explicit density of training samples and produces unrealistic blurry images.
In this project, we aim to use GANs for face aging also known as age synthesis, where we render a face image with natural aging and rejuvenating effects on the individual face. We further discuss various versions of GANs such as Conditional GAN, where we feed the input with a conditioning variable to condition on to both generator and discriminator, Age-cGANs where we focus on identity preserving face aging.

1. Introduction

The applications of face aging include, finding lost children and cross-age face recognition. Traditional face aging are roughly of two types: prototyping and modeling. Prototyping approach of face aging are simple and fast - they estimate average faces within predefined age groups. Since they are based on general rules, personalised information is discarded, producing unrealistic images. On the other hand, modelling approaches using parametric models to simulate aging process. To train a model, we require various age sequences of the same person which is expensive.

To solve this problem, we will prefer generative models which generates new samples by estimating the probability density of given samples. Generative models are of two kinds - explicit and implicit density models. Explicit density models try to maximize the likelihood estimation of training data to generate new samples but the images produced are of low quality. Implicit density models, such as GANs, use game theoretic approach to generate realistic samples.

There are a different types of GANs, for the purpose of face aging we are going to discuss about Conditional GANs where we use target age group as conditional input.

2. Literature Review

2.1. Generative Models

Deep generative models (DGM) are neural networks with many hidden layers trained to approximate complicated, high-dimensional probability distributions using a large number of samples. When trained successfully, we can use the DGMs to estimate the likelihood of each observation and create new samples from the underlying distribution.

As of today, the three most popular generative models are Fully Visible Belief Networks (FVBN), Variational AutoEncoder and GAN.

2.2. Generative Adversarial Network

Generative Adversarial models uses a game-theoretic approach, where we train two networks - generator network and discriminator network.

The generator network tries to fool the discriminator by generating real looking images. The discriminator network tries to distinguish between real and fake images.
The generator network models a distribution of real looking images from a random noise. And, the discriminator network outputs the likelihood in $(0, 1)$ of real image. We train both the networks jointly using a minimax objective function.

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))] \quad (1)$$

where, $D_{\theta_d}(x)$ is the discriminator output for the real data x and $D_{\theta_d}(G_{\theta_g}(z))$ is the discriminator output for the generated fake data $G(z)$.

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

Figure 1. Training pipeline for model

2.3. Conditional GAN

Generative Adversarial Networks can be extended to a conditional probabilistic model if, a conditional variable y is fed into both generator and discriminator as an input layer. In the generator, the prior input noise $p_z(z)$ and y are combined in joint hidden representation. In the discriminator, x and y are presented as input to a discriminative function.

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x|y) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z|y)))] \quad (2)$$

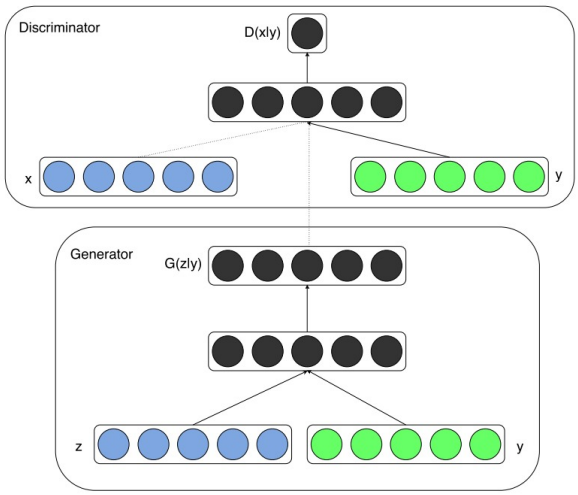

Figure 2. Conditional adversarial network

2.4. Age-cGAN

In Age-cGAN, we use a novel latent vector optimization approach which allows Age-cGAN to reconstruct an input face image preserving the original person's identity. The Age-cGAN has multiple stages of training. It has 4 networks which get trained in 3 stages.

1. Conditional GAN training: Training of generator and discriminator networks
2. Initial latent vector approximation: Training of encoder network
3. Latent vector optimization: Training of facial recognition network

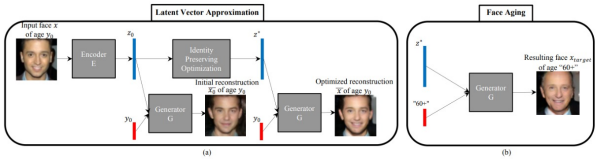

Figure 3. Face aging method using Age-cGAN

2.4.1 Latent Vector Optimization

The ultimate goal of the face aging task is to change the age of the person without changing the identity. Use of initial latent approximations results in loss of identity of the original person in about 50% of the cases. Therefore, initial latent approximations must be improved.
Pixel-wise latent vector optimization can be used for the above purpose. But, it has 2 clear downsides. It increases the blurriness of reconstruction and it focuses on unnecessary details of input face images, which have nothing to do

with the person's identity. Instead, identity preserving latent vector approach is used where the difference between the identities in the original and the reconstructed images is expressed as the Euclidean distance between the corresponding embeddings. The objective is to minimize this distance.

$$z_{IP}^{*} = \arg\min_{z} \|FR(x) - FR(\bar{x})\|_{L_2} \tag{3}$$

This face aging method can be used for synthetic augmentation of face dataset and for improving the robustness of face recognition solution in cross-age scenarios.

2.5. Progressive Face Aging

Traditional methods of modeling face age progression can be roughly divided into two categories: physical model and prototype-based methods. Physical models mechanically stimulate the changes in facial appearance over time using varying parameters. However, these models don't generalize well and are computationally expensive. Prototyping models, on the other hand, compute the average face of an age group as protoype. The final output is then calculated by adding the difference between given image and prototype. These models are computationally efficient but it cannot map personalized features well.

GAN based models such as cGAN use different age groups as conditioning variables while training the generator. A single model is trained to learn the aging effects between different age groups. Therefore, these models cannot perform following three functions - image quality, identity preservation and aging accuracy.

In PFA-GANs, multiple sub-networks are trained to learn the aging effects between two adjacent age groups. Since, aging is a gradual process, these models perform better than traditional cGANs. In these models, novel encoding scheme is introduced by adding binary gates to control aging flow.
Therefore, the progressive aging framework from the source age group s to target one t can be formulated as follows:

$$X_t = G_{t-1} \circ G_{t-2} \circ .....G_s(X_s) \tag{4}$$

Residual skip connections are also introduced in these sub-networks to prevent memorizing the exact copy of input faces. Target age group image is then found by controlling the sequence of binary gates. The adjacent age group output can be re-written as:

$$X_{i+1} = G_i(X_i) = X_i + \lambda_i G_i(X_i) \tag{5}$$

Here, $\lambda \in \{0, 1\}$ is the binary gate controlling whether the sub-interval G_i is involved in the path to target age group.

Finally, a given face X_s of age group s, the aging from s to group t can be written as

$$X_t = G(X_s, \lambda_{s:t}) \tag{6}$$

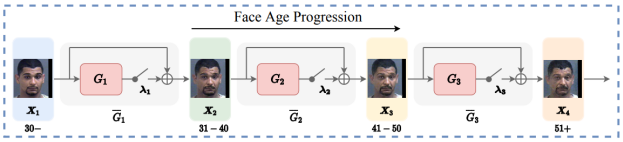

Figure 4. sub-networks in PFA

Along with these sub-networks, PFA-GAN also contains pre-age estimation network. These networks help to characterize the face-age distribution for an improved aging accuracy. These networks use both age-classification term and age regression term, to check whether the generated face belongs to the target age group or not.

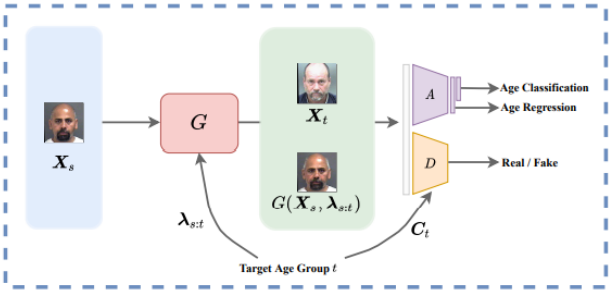

Figure 5. PFA-GAN Framework

2.5.1 Loss function

The overall loss function of PFA-GAN contains the following three components:

- **Adversarial Loss:** To produce high quality images.
- **Age Estimation Loss:** To improve aging accuracy.
- **Identity Consistency Loss:** To preserve identity.

1. **Adversarial Loss:** Unlike normal-GANs, least squared loss function is used as an adversarial loss. It improves the quality of generated images and stabilize the training processes. Given a face X_s from age group s, the output of G from s to a age group t is $G(X_s, \lambda_{s:t})$. In this context, the adversarial loss is given as:

$$L_{adv} = \frac{1}{2} E_{X_s} \left[D([G(X_s, \lambda_{s:t}); C_t]) - 1 \right]^2 \tag{7}$$

2. **Age Estimation Loss:** Age estimation network was included in PFA-network to regularize the face-age distribution by minimizing age-estimation loss which includes both regression and classification loss. The age-estimation loss between estimated age $\hat{y}$ and target age y for the generator age G is defined as:

$$L_{age} = E_{X_s}\left[\|y - \hat{y}\|^2 + l(A(X)W, c_t)\right] \quad (8)$$

3. **Identity Consistency Loss:** To preserve the identity related information of the face, the identity consistency loss has three components. These components are pixel-wise loss, structure similarity loss and feature level loss. The three losses are defined as:

$$L_{pix} = E_{X_s}\left[G(X_s, \lambda_{s:t} - X_s\right] \quad (9)$$

$$L_{ssim} = E_{X_s}\left[1 - SSIM(G(X_s, \lambda_{s:t})), X_s\right] \quad (10)$$

$$L_{fea} = E_{X_s}\left\|\phi(G(X_s, \lambda_{s:t})) - \phi(X_S)\right\|_F^2 \quad (11)$$

3. Preliminary Results

3.1. Dataset

We have used "Cats faces 64*64" dataset available on Kaggle.

3.2. Architecture

The architecture which produced the best results as follows:

Discriminator				
Layer	Filter	Stride	Activation	Output Shape
Conv	4 * 4	2	Leaky ReLU	16 x 32 x 32
Conv	4 * 4	2	Leaky ReLU	32 x 16 x 16
Conv	4 * 4	2	Leaky ReLU	64 x 8 x 8
Conv	4 * 4	2	Leaky ReLU	128 x 4 x 4
Conv	4 * 4	1	-	1 * 1 * 1

Table 1. Architecture of the discriminator network

Generator				
Layer	Filter	Activation	Padding	Output Shape
Deconv	4 * 4	ReLU	0	128 * 4 * 4
Deconv	4 * 4	ReLU	1	64 * 8 * 8
Deconv	4 * 4	ReLU	1	32 * 16 * 16
Deconv	4 * 4	ReLU	1	16 * 32 * 32
Deconv	4 * 4	Tanh	1	3 * 64 * 64

Table 2. Architecture of the generator network

3.3. Experiments and Results

3.3.1 Trial 1

We used the learning rate as 0.002, the number of epochs as 60, the parameters β of Adam optimizer are $\{0.5, 0.999\}$. We started with an architecture as follows:

Discriminator				
Layer	Filter	Stride	Activation	Output Shape
Conv	4 * 4	2	Leaky ReLU	6 * 32 * 32
Conv	4 * 4	2	Leaky ReLU	10 * 16 * 16
Conv	4 * 4	2	Leaky ReLU	20 * 8 * 8
Conv	4 * 4	2	Leaky ReLU	40 * 4 * 4
Conv	4 * 4	1	-	1 * 1 * 1

Table 3. Architecture of the discriminator network

Generator				
Layer	Filter	Activation	Padding	Output Shape
Deconv	4 * 4	ReLU	0	40 * 4 * 4
Deconv	4 * 4	ReLU	1	20 * 8 * 8
Deconv	4 * 4	ReLU	1	10 * 16 * 16
Deconv	4 * 4	ReLU	1	6 * 32 * 32
Deconv	4 * 4	Tanh	1	3 * 64 * 64

Table 4. Architecture of the generator network

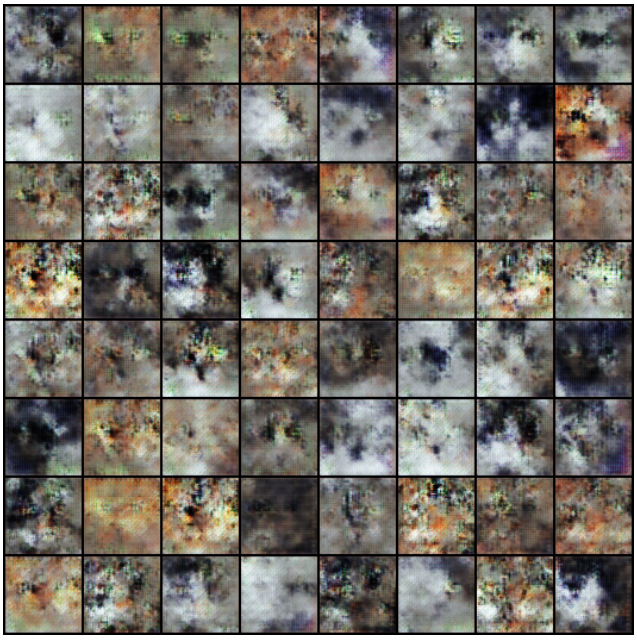

Figure 6. Generated cat images

3.3.2 Trial 2

We used the learning rate as 0.004, the number of epochs as 100, the parameters β of Adam optimizer are $\{0.65, 0.999\}$. We started with an architecture as follows:

Discriminator				
Layer	Filter	Stride	Activation	Output Shape
Conv	4 * 4	2	Leaky ReLU	6 * 32 * 32
Conv	4 * 4	2	Leaky ReLU	10 * 16 * 16
Conv	4 * 4	2	Leaky ReLU	15 * 8 * 8
Conv	4 * 4	2	Leaky ReLU	25 * 4 * 4
Conv	4 * 4	1	-	1 * 1 * 1

Table 5. Architecture of the discriminator network

Generator				
Layer	Filter	Activation	Padding	Output Shape
Deconv	4 * 4	ReLU	0	25 * 4 * 4
Deconv	4 * 4	ReLU	1	15 * 8 * 8
Deconv	4 * 4	ReLU	1	10 * 16 * 16
Deconv	4 * 4	ReLU	1	6 * 32 * 32
Deconv	4 * 4	Tanh	1	3 * 64 * 64

Table 6. Architecture of the generator network

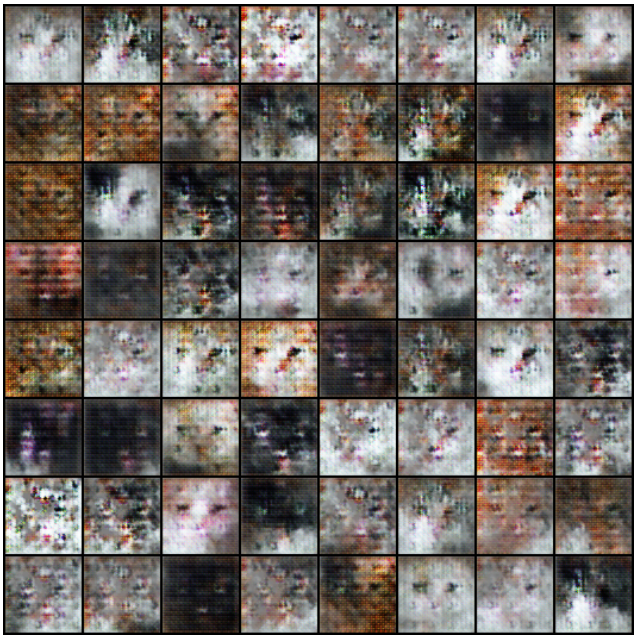

Figure 7. Generated cat images

When we used lesser number of kernels in the architecture and lesser number of epochs, the images are blurry. On increasing the number of kernels and epochs, the images are clear.

Figure 8. Final generated cat images

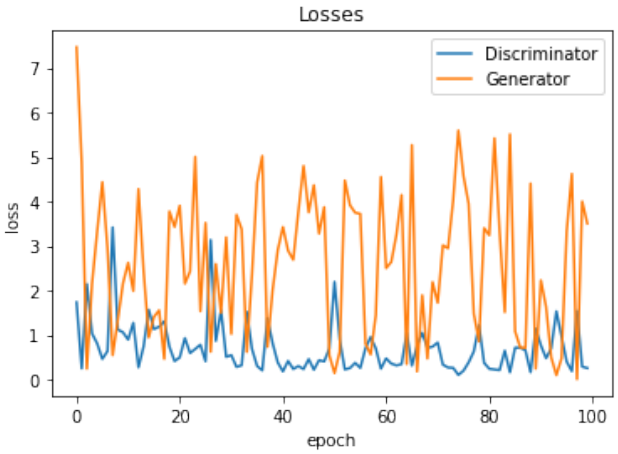

Figure 9. Plot of losses v/s epochs

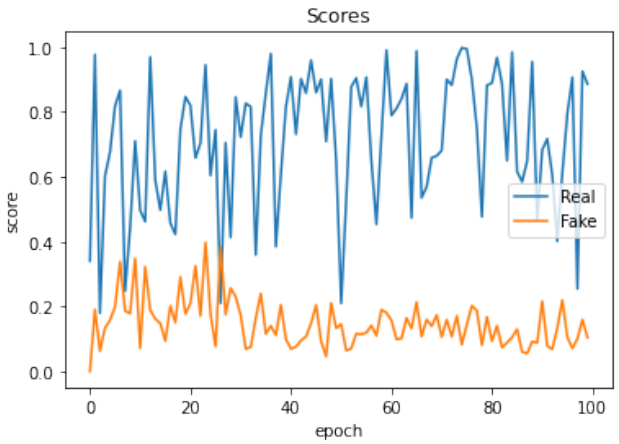

Figure 10. Plot of scores v/s epochs

4. Final work

Building on this idea, we implemented age synthesis using cGAN.

4.1. Age synthesis using cGAN

4.1.1 Dataset

cGAN model was trained on the IMDB-WIKI dataset with about 60k face images with age labels.

Figure 11. Real Images from dataset

4.1.2 Data Pre-processing

We created a *calc_age* function to calculate the age of each image using the birth year. We assumed that the photo was taken in the middle of the year. We scaled pixel values of all the images to $[-1, 1]$. We finally created 6 categories of age namely: 0-18, 19-29, 30-39, 40-49, 50-59 and 60+. We used one-hot encoded vector for each age category.

4.1.3 Data Loader

For creating our data loader, we resized our images to $3 * 64 * 64$ and used a batch size of 128 images.

4.1.4 Architecture

The following architecture of the model gave the best results. In discriminator, we took an input image of size $3 * 64 * 64$ and concatenated it with labels of dimension $6 * 64 * 64$ by extending our one-hot encoded label vector.

Discriminator				
Layer	Filter	Stride	Activation	Output Shape
Conv	4 * 4	2	ReLU	64 x 32 x 32
Conv	4 * 4	2	ReLU	70 x 16 x 16
Conv	4 * 4	2	ReLU	128 x 8 x 8
Conv	4 * 4	2	ReLU	256 x 4 x 4
Conv	4 * 4	2	ReLU	512 x 2 x 2
Conv	4 * 4	2	-	1 * 1 * 1

Table 7. Architecture of the discriminator network

We used 6 convolution layers with kernel size in each of the layers as $4 * 4$ and stride of 2.

Generator				
Layer	Filter	Stride	Activation	Output Shape
DeConv	4 * 4	2	ReLU	512 x 4 x 4
DeConv	4 * 4	2	ReLU	256 x 8 x 8
DeConv	4 * 4	2	ReLU	128 x 16 x 16
DeConv	4 * 4	2	ReLU	64 x 32 x 32
DeConv	4 * 4	2	ReLU	3 x 64 x 64

Table 8. Architecture of the generator network

4.1.5 Hyperparameters

For the purpose of training the model, we used Adam optimiser for both generator and discriminator. The chosen parameters are learning rate $= 0.0002$, $\beta_1 = 0.5$, $\beta_2 = 0.999$.

4.1.6 Training

We used binary cross entropy loss for both the generator and the discriminator. We defined total loss for the discriminator as the sum of loss of real images from the data set and fake images generated by the generator. Here, the discriminator acts as a binary classifier and we choose label 1 for real images and 0 for fake images. We also used smooth age labels for training the discriminator to avoid **mode collapse**. The total loss for the discriminator is given by

$$\begin{aligned} D_{total} &= \frac{D_{real} + D_{fake}}{2} \\ &= \frac{-y \log D(x) + (1-y) \log D(G(z))}{2} \end{aligned} \tag{12}$$

Then, we train our generator by choosing a noise vector from uniform distribution and passing it to the discriminator trained earlier. We chose labels for the generated images as 1. The loss for the generator is given by,

$$-y \log D(G(z)) \tag{13}$$

4.1.7 Final Results

We trained our model on 50 epochs and obtained the following results.

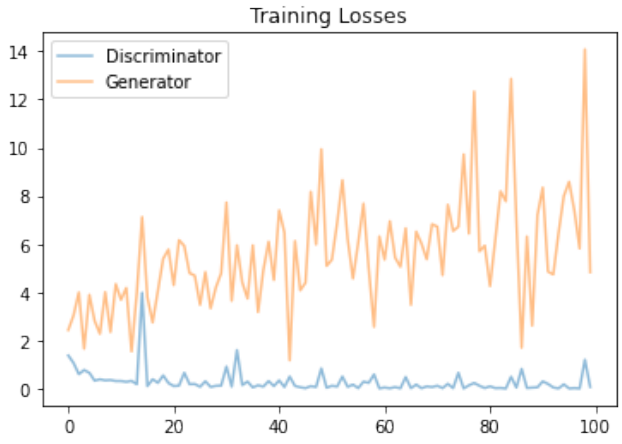

Figure 12. Plot of training losses of discriminator and generator

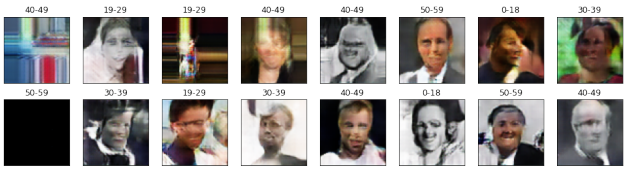

Figure 13. Output images generated by cGAN

5. Conclusion

We used cGANs to generate face images of a given age group. The results shown in this paper are preliminary, yet they show the potential of cGANs. The idea of using conditional vector in GANs can be extended to improve age synthesis models. These synthetic images can be further used to train other models. Latent vector approximations and use of reconstructed face for training generator are some other ways to improve age synthesis models. We are using a single model to learn age mapping among all age categories which can be improved by adding a residual generator.

6. Challenges

- One of the main challenges in doing the project was the limited computational power. It constrained the architectural and hyperparameter tuning related experiments. Low computational capacity hindered further improvement of the model thus, producing blurred and unrealistic images.
- Vanishing Gradients: In some cases, the discriminator became too strong, which resulted in vanishing gradients while training of generators. This may be due to poor initialisations or architectural imbalance between generators and discriminators. We tried to tackle this issue by giving a head start to generator and reducing the complexity of the discriminator. We mitigate this problem by using one-sided label smoothing in discriminator training.

7. Literature Contributions

- Digjoy Nandi: GANs - Goodfellow [4], VAE [3]
- Omkaradithya Pujari: Face Aging with Conditional Generative Adversarial Networks [6], PixelRNN [1]
- Perambuduri Srikaran: Conditional GAN [5], PixelCNN [2]
- Vaishnavi W: FaceNet [9], DCGAN [7]
- Ayush Kumar Singh: PFA-GAN: Progressive Face Aging with Generative Adversarial Network [8], CNN Visualisation [10]

8. Model Implementation Contributions

- Digjoy Nandi: Architecture experimentation, EDA
- Omkaradithya Pujari: Data collection and cleaning
- Perambuduri Srikaran: Architecture experimentation
- Vaishnavi W: Network training and debugging
- Ayush Kumar Singh: Hyperparameter tuning and optimisation

References

[1] Aäron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu (2016) "Pixel Recurrent Neural Networks" 7

[2] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, Koray Kavukcuoglu (2016) "Conditional Image Generation with PixelCNN Decoders" 7

[3] Diederik P. Kingma, Max Welling (2014) "Auto-Encoding Variational Bayes" 7

[4] Ian Goodfellow et al. (2014) "Generative Adversarial Nets", NIPS 2014 7

[5] Mehdi Mirza, Simon Osindero (2014) "Conditional Generative Adversarial Nets" 7

[6] Grigory Antipov, Moez Baccouche, Jean-Luc Dugelay (2017) "Face Aging with Conditional Generative Adversarial Networks" 7

[7] Xinhua Liu, Yao Zou, Chengjuan Xie, Hailan Kuang, Xiaolin Ma (2019) "Bidirectional Face Aging Synthesis Based on Improved Deep Convolutional Generative Adversarial Networks" 7

[8] Zhizhong Huang, Shouzhen Chen, Junping Zhang, Hongming Shan (2020) "PFA-GAN: Progressive Face Aging with Generative Adversarial Network" 7

[9] Florian Schroff, Dmitry Kalenichenko, James Philbin (2015) "FaceNet: A Unified Embedding for Face Recognition and Clustering" 7

[10] Matthew D. Zeiler, Rob Fergus (2013) "Visualizing and Understanding Convolutional Networks" 7

[11] https://towardsdatascience.com/getting-started-with-gans-using-pytorch-78e7c22a14a5

Github link: https://github.com/srikaran-p/AI2100—Project