

## Face Aging Using GANs

Digjoy Nandi: ai20btech11007@iith.ac.in

Omkaraditya Pujari: ai20btech11017@iith.ac.in

Perambuduri Srikanth: ai20btech11018@iith.ac.in

Vaishnavi W: ai20btech11025@iith.ac.in

Ayush Kumar Singh: ai20btech11028@iith.ac.in

Indian Institute of Technology, Hyderabad

May 6, 2022

# Generative Models

Generative models are a class of unsupervised learning models which generates data similar to given training data. We do this by modeling a probability distribution similar to training data.

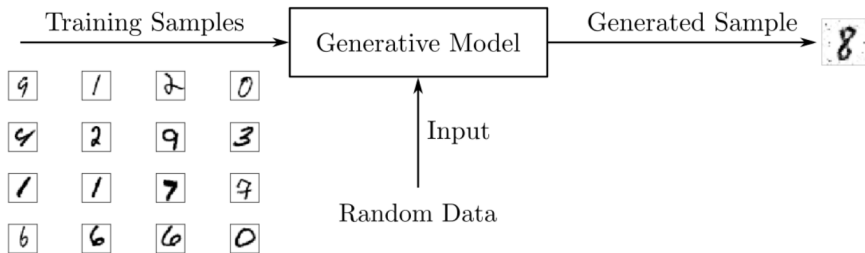


Figure: Overview of generative models

# Types of Generative Models

There are mainly two types of generative models

- Explicit density models : Models which explicitly define density function and calculate  $p_{model}$
- Implicit density models : Models which calculates  $p_{model}$  without explicitly defining it.

# Taxonomy of Generative Models

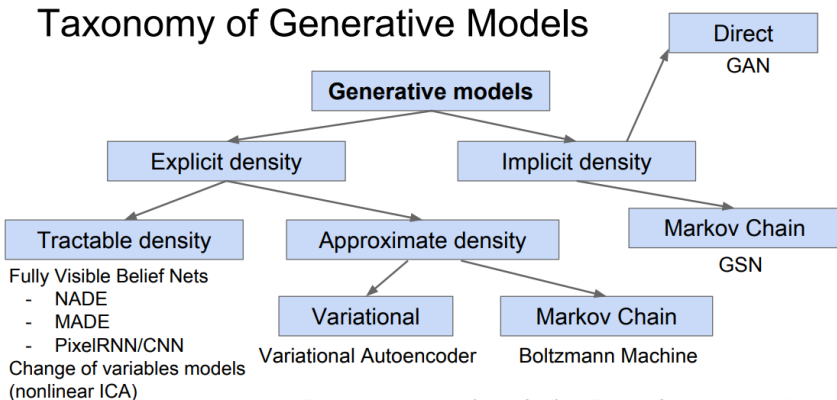


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

# Generative Adversarial Networks (GANs)

GANs are adversarial networks containing two neural networks,

- ① Generator (G) : Generates samples by passing random noise
- ② Discriminator (D) : Classifies if the image is real or fake

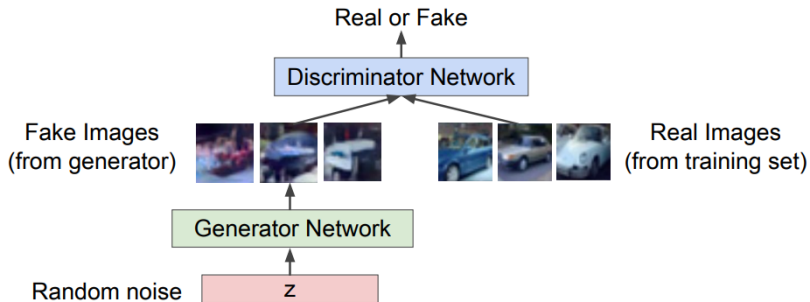


Figure: GANs

Both the networks are jointly trained using a minimax objective function.

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

where,

- $p_{data}$  is probability distribution of training data
- $p(z)$  is probability distribution of noise variable
- $D_{\theta_d}(x)$  is the discriminator output for the real data  $x$
- $D_{\theta_d}(G_{\theta_g}(z))$  is the discriminator output for the generated fake data  $G(z)$

While training, we alternate between,

- 1 **Gradient ascent** on discriminator:

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

- 2 **Gradient descent** on generator:

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

# GANs

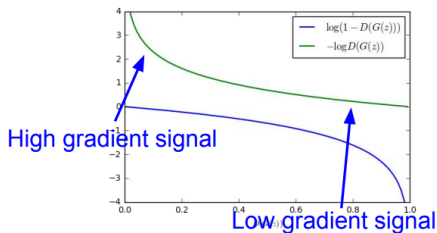


Figure: GANs

In the initial iterations, when the sample is likely fake, gradient in the loss of the generator is flat. Generator doesn't train well.

Improved objective of the generator  
Maximize the likelihood of discriminator being wrong



While training, we now alternate between,

① **Gradient ascent** on discriminator:

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

② **Gradient ascent** on generator:

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

**for** number of training iterations **do**

**for**  $k$  steps **do**

- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Sample minibatch of  $m$  examples  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  from data generating distribution  $p_{\text{data}}(\mathbf{x})$ .
- 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  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- 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: Training pipeline for model

# GANs Implementation

Using learning rate as 0.004, the number of epochs as 100 and the parameters  $\beta$  of Adam optimizer are  $\{0.65, 0.999\}$

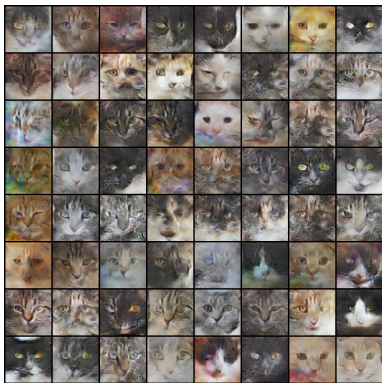


Figure: Generated Cat images

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

# Conditional GANs (cGANs)

- GANs can be extended to a conditional probabilistic model
- A conditional variable  $y$  is fed into both the generator and the 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.

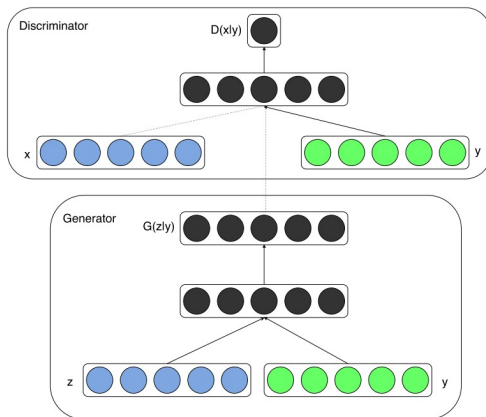


Figure: Structure of a simple conditional adversarial net

The minimax objective function,

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

We then train the model using the similar algorithm discussed above.

# Age-cGAN

Age-cGANs are a class of conditional GANs, where we use a novel age-preserving latent vector optimisation approach for age synthesis. It has 4 networks which get trained in 3 stages.

- ➊ Conditional GAN training: Training of generator and discriminator networks
- ➋ Initial latent vector approximation: Training of encoder network
- ➌ Identity preserving latent vector: Training of facial recognition network

Once Age-cGAN is trained, face aging is done in 2 steps. Given input face image  $x$  of age  $y_0$ ,

- 1 Image Reconstruction: The image is passed to the encoder and identity preserving optimiser, generating an optimized latent vector. It is passed through the generator producing a reconstructed image.
- 2 Age Synthesis: The target image is generated by passing the reconstructed image along with the target age condition.



# Age-cGAN

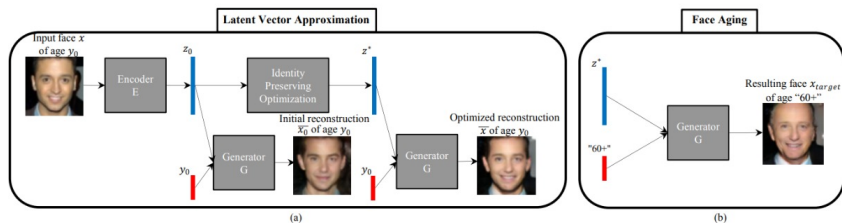


Figure: Face aging method using Age-cGAN

## Latent Vector Optimization

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.

$$z_{IP}^* = \arg \min_z \|FR(x) - FR(\bar{x})\|_{L_2}$$

This is an internal representation of FaceNet CNNs.

- Train a unified system for face verification, recognition and clustering.
- Map an input image to a lower dimensional Euclidean space where, the squared  $L_2$  distance between the mapping corresponds to image similarity.
- Triplet loss is the novel loss function employed in the model.

# FaceNet - Triplet Loss

The triplet loss function is given by,

$$\sum_i^N \left[ \|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha - \|f(x_i^a) - f(x_i^n)\|_2^2 \right]$$

- Anchor image  $x_i^a$ : Current image for which the loss function is calculated
- Positive image  $x_i^p$ : Image belonging to the same class as anchor image.
- Negative image  $x_i^n$ : Image not belonging to the same class as anchor image.
- $\alpha$ : Margin between positive and negative pairs

# Face Synthesis Implementation

In our implementation, we generate images given an age group as a conditioning variable.

## Dataset

IMDB-WIKI dataset with about 60k face images with age labels

## Data pre-processing

- Created a function to calculate the age of each image using the birth year. Assuming that the photo was taken in the middle of the year.
- Scaled the pixel values of all the images to  $[-1,1]$
- Created 6 categories: 0-18, 19-29, 30-39, 40-49, 50-59, 60+
- Used one-hot encoding to represent each category

# Face Synthesis Implementation

## Data Loader

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

## Architecture

In discriminator, we took an input image of size  $3 \times 64 \times 64$  and concatenated it with labels of dimension  $6 \times 64 \times 64$ .

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

# Face Synthesis Implementation

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: Architecture of the generator network

## Hyperparameters

We used Adam optimiser for both the generator and the discriminator. The chosen parameters are  $\eta = 0.0002$ ,  $\beta_1 = 0.5$ ,  $\beta_2 = 0.999$ .

# Face Synthesis Implementation

## Training

- We used binary cross entropy loss for both generator and discriminator.
- Total loss of the discriminator is the sum of loss of real images from the data set and fake images generated by the generator.
- The discriminator chooses 1 for real images and 0 for fake images.
- We also used smooth age labels for training the discriminator to avoid vanishing gradients.
- Total loss of the discriminator

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



# Face Synthesis Implementation

## Training

- 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.
- Loss of the generator

$$-y \log D(G(z))$$

# Face Synthesis Implementation

## Results

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

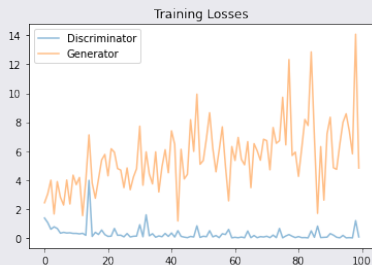


Figure: Plot of training loss of discriminator and generator

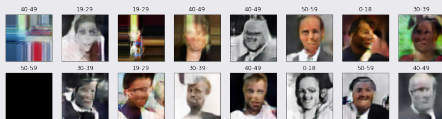


Figure: Output images generated

GAN based models such as cGAN use a single model to learn the aging effects between different age groups. Since, aging is a gradual process, the quality of generated image reduces as the age gap increases.

Therefore, in PFA GANs multiple subnetworks are trained to learn the aging effect between two adjacent age groups. These models perform the following three functions better than traditional models

- Better image quality
- Aging accuracy
- Identity preservation

In this model, a novel encoding scheme is introduced by adding binary gates. Residual skip connections are also used in these subnetworks, to prevent memorizing the exact copy of the input face. Output image for a subnetwork can be written as

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

Here, the binary gates  $\lambda \in [0, 1]$  control whether the subinterval  $G_i$  is involved in the path to target age group.

# PFA GAN

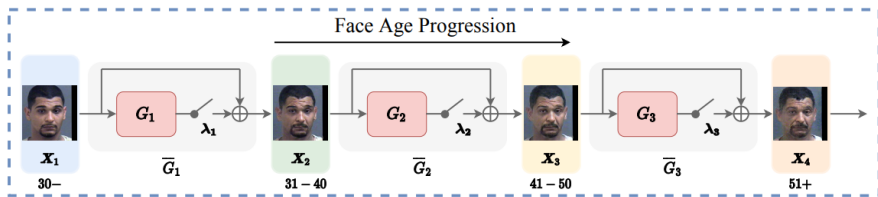


Figure: Sub-networks in PFA

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

## Adversarial Loss

To produce high quality images. Unlike normal GANs, least square loss function is used as an adversarial loss in PFA GAN.

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

## Age Estimation Loss

To improve aging accuracy. In our progressive face aging framework, age estimation network is introduced to regularize the face-age distribution by minimizing the age estimation loss which is given by

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

## Identity Consistency Loss

It has three components: Pixelwise loss, structural similarity loss and feature level loss

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

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

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



# Face Synthesis Implementation

## Challenges

- Limited computational power: Constrained the architectural and hyperparameter tuning related experiments. It hindered the improvement of the model, resulting in blurred images.
- Vanishing Gradients: The discriminator became too strong, resulting in vanishing gradients in the generator.

## Methods used to tackle the problems

- Decreasing learning rate of discriminator
- Generator given headstart (Initializing discriminator every certain number of epochs)
- Halt the training of the discriminator for few epochs
- Used one-sided label smoothing in discriminator training

# 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.







# Literature Contributions

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

# Model Implementation Contributions

- Digjoy Nandi: Experimented with different generator architectures and wrote the generator code
- Omkaradithya Pujari: Data pre-processing and data loading
- Perambuduri Srikan and Vaishnavi: Hyperparameter tuning, Network training and loss optimization code
- Ayush Kumar Singh: Wrote the discriminator architecture code
- L<sup>A</sup>T<sub>E</sub>Xtypesetting (Report): Digjoy Nandi and Vaishnavi
- L<sup>A</sup>T<sub>E</sub>Xtypesetting (Slides): Perambuduri Srikan

Github link: <https://github.com/srikan-p/AI2100—Project>

-  Aäron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu (2016) "Pixel Recurrent Neural Networks"
-  Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, Koray Kavukcuoglu (2016) "Conditional Image Generation with PixelCNN Decoders"
-  Diederik P. Kingma, Max Welling (2014) "Auto-Encoding Variational Bayes"
-  Ian Goodfellow et al. (2014) "Generative Adversarial Nets", NIPS 2014
-  Mehdi Mirza, Simon Osindero (2014) "Conditional Generative Adversarial Nets"
-  Grigory Antipov, Moez Baccouche, Jean-Luc Dugelay (2017) "Face Aging with Conditional Generative Adversarial Networks"

-  Xinhua Liu, Yao Zou, Chengjuan Xie, Hailan Kuang, Xiaolin Ma (2019) "Bidirectional Face Aging Synthesis Based on Improved Deep Convolutional Generative Adversarial Networks"
-  Zhizhong Huang, Shouzhen Chen, Junping Zhang, Hongming Shan (2020) "PFA-GAN: Progressive Face Aging with Generative Adversarial Network"
-  Florian Schroff, Dmitry Kalenichenko, James Philbin (2015) "FaceNet: A Unified Embedding for Face Recognition and Clustering"
-  Matthew D. Zeiler, Rob Fergus (2013) "Visualizing and Understanding Convolutional Networks"
-  <https://towardsdatascience.com/getting-started-with-gans-using-pytorch-78e7c22a14a5>