Journal of Engineering and Applied Sciences 14 (Special Issue 5): 9073-9085, 2019
ISSN: 1816-949X

Using Generative Adversarial Networks (GANs) to Generate Facial Attributes

Mustafa Radif
School of Computer Science and IT, University of Al-Qadisiyah, Al-Qadisiyah, Iraq
mustafa.radif@qu.edu.iq

Abstract: The study presents a new training method for Generative Adversarial Networks (GANs). The new method is about adding low-resolution layers to images in the networks and gradually increasing the resolution until high resolution with good and realistic images are developed. The method helps to create stable images of good quality and the speed of training also, increases by a factor of 2. A score of 8.8 was achieved for unsupervised CIFAR 10. The study presents details of the implementation where rivalry and competition between the two networks are reduced.

Key words: Generative adversarial networks, CelebA, CIFAR10, image generation, GAN, training

INTRODUCTION

Generation of rendered faces and objects and physical attributes finds use in a number of areas such as movies, advertisements, gaming, crime and terrorism management, crowd handling and identification and in other areas. Artificial Intelligence (AI) is exploring new methodologies and technologies to create realistic models of people through datasets. The requirement is for these models to show emotion and to allow modification of their attributes (Chen *et al*., 2013). The status of this technology is far from realistic and the images that are generated appear artificial, lifeless, clunky, they cannot be modified easily and they are not satisfactory (Khosla *et al*., 2013). This study examines the technology, trends and development for generation of faces and modification of their attributes. Several datasets were used in the networks to generate images of acceptable, higher quality.

Generative Adversarial Networks (GANs) include different classes of AI algorithms that aid unsupervised machine learning. Two neural networks, generative and discriminative, oppose each other in a zero-sum game process. The method is used to generate pictures that appear realistic with many attributes that appear like photographs. In GAN, one network is used to generate data while another test and evaluates it. The generative network creates a map of a latent space in a specific data distribution and the discriminative networks finds differences in the instances from the distribution of data and image. The training goal is to fool the discriminative network by creating synthesised instances that are projected to come from the true data distribution (Ghosh *et al*., 2018).

As indicated in the previous paragraph, generative and discriminative algorithms are used in GANs. Discriminatory algorithms use input data, examine them and suggest a label or category for the data. As an example, a discriminative algorithm to evaluate, if a message is spam or not spam, examines the collection of words from the email. When the match for the words with words from the algorithm are high, then there is a probability y/x that the formulation has a probability of y as spam, given x as the word it contains. The algorithm then maps to features to labels and they are concerned with the correlation and flag a mail as spam. Therefore, features are mapped to labels. Generative algorithms try to predict certain features for a given label (Heusel *et al*., 2017). Figure 1 gives an example of training algorithm with step size • and minibatch.

The method used by GANs is explained as follows. The generator creates new data instances while the discriminator checks it for authenticity and if each instance of the reviewed information is from the training data. In effect, the generator wants to pass data that the discriminator must consider as authentic. Data, whether of images or text is in the form of encoded numbers. The generator takes random numbers and generates an image and passes it to the discriminator along with other images from the dataset. The discriminator reads the data for both fake and real images and sends probabilities in the form of 0 indicating fake and 1 indicating authentic. Thus, a double feedback loop is created and the process is called training where both networks train themselves to send authentic images and detect fake images (Ba *et al*., 2016). Figure 2 gives a block diagram of GAN principle.

The discriminator network is in the form of a convolutional network that segregates the images it is given with a binomial classifier label. The generator

Input: minibatch images x, matching text t, mismatching $\hat{t}$, number of training batch steps S
for $n = 1$ **to** S **do**
  $h \leftarrow \varphi(t)$ {Encode matching text description}
  $\hat{h} \leftarrow \varphi(\hat{t})$ {Encode mis-matching text description}
  $z \sim \mathcal{N}(0,1)^Z$ {Draw sample of random noise}
  $\hat{x} \leftarrow G(z,h)$ {Forward through generator}
  $s_r \leftarrow D(x,h)$ {real image, right text}
  $s_w \leftarrow D(x,\hat{h})$ {real image, wrong text}
  $s_f \leftarrow D(\hat{x},h)$ {fake image, right text}
  $\mathcal{L}_D \leftarrow \log(s_r) + (\log(1-s_w) + \log(1-s_f))/2$
  $D \leftarrow D - \alpha\partial\mathcal{L}_D/\partial D$ {Update discriminator}
  $\mathcal{L}_G \leftarrow \log(s_f)$
  $G \leftarrow G - \alpha\partial\mathcal{L}_G/\partial G$ {Update generator}
end for

Fig. 1: Sample training algorithm for GANs (Heusel *et al*., 2017)

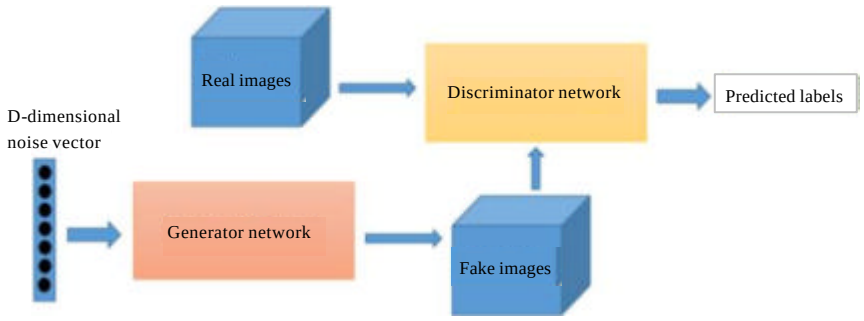

Fig. 2: Block diagram of GANs principle (Ba *et al*., 2016)

on the other hand is an inverse convolutional network. The convolutional classifier uses an image, runs a down sampling and produces a probability. The generator tries to take a vector image of the noise and uses upsampling to give an image. The discriminator discards the data by downsampling while the generator uses the downsampled data and turns it into a new image. Thus, an optimising exercise is run with opposite objectives, creating a zero-sum game or an actor-critic model. With training, the rejections reduce and the acceptance rate increases (Arjovsky *et al*., 2017). The process is illustrated in Fig. 3. The following objectives are proposed for this study:

- Develop a deep understanding and background of GANs
- Develop a new training method for GANs using CelebA images to generate images of high resolution from multiple layers of low-resolutions
- Test the method with experiments

Literature review: Generative image methods are of two types, parametric and non-parametric. Non-parametric models carry out matching from a dataset of pictures, match the patches and they are used to produce texture synthesis, painting and for super-resolution. GAN is a parametric method of image generation. Until recently, the process of generating realistic and natural images has not been very successful, especially, at higher resolution. A type of variation sampling method was used by Kingma and Welling (2013) but the samples were blurred and suffered from grains. Sohl-Dickstein *et al*. (2015) used the iterative forward diffusion process to generate images.

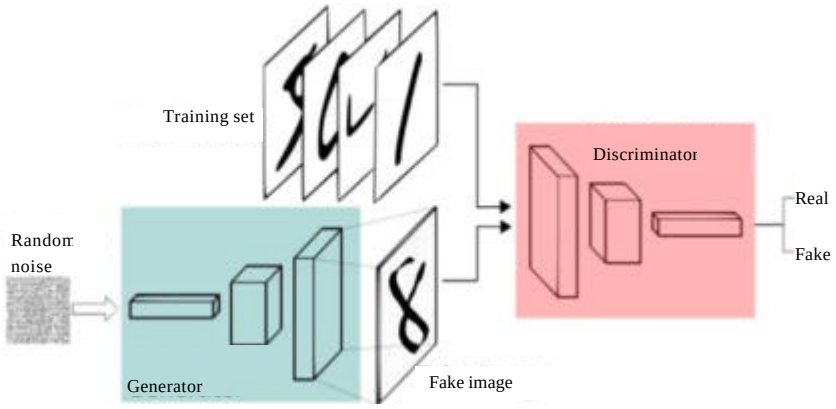

Fig. 3: Working principles of GANs (Arjovsky *et al*., 2017)

Goodfellow *et al*. (2014) developed the Generative Adversarial Networks and produced images that had less noise and were clearer. Denton *et al*. (2015) used the Laplacian pyramid extension to GAN and produced higher quality images. These images had problems of noise, blur and they suffered from loss of clarity. Gregor *et al*. (2015) used a deconvolutional network method to produce generative images. These methods did not use the generator and discriminator and other methods to increase image clarity.

Zeiler and Fergus (2014) critics neural networks and argue that they function as black-box methods and do not provide an understanding of human-consumable algorithm. They showed that with deconvolutional and filtering the activations, it is possible to develop a convolutional filter for the network. If a gradient descent is used with the input data, then the ideal image that uses filter subsets can be known. Attempts were made to upscale GANs with CNNs for image modelling. Dai *et al*. (2017) and Denton *et al*. (2015) used an alternative method to use iterations to create low-resolution upscale images with higher reliability. Simonyan and Zisserman (2014) developed a convolutional net that was used in the place of maxpooling along with strided convolutional process. This method helped the networks to develop learning from spatial downsampling. This approach is used in this study.

Another method was to remove all the connected layers of convolutional features such as the global average pooling used in the place of the art image classification (Salimans and Kingma, 2016). It is observed that global average pooling helped the stability to increase but the convergence speed reduced. An alternative approach was to link the topmost convolutional attributes to the input of the generator and output of the discriminator. It was seen that the base layer of GAN that uses noise distribution as input can be linked, However, this created a four-dimensional tensor that was used at the beginning of the convolutional stack (He *et al*., 2015).

Ioffe and Szegedy (2015) suggested using batch normalisation that normalises input for all datasets to have a mean and unit variance of zero. The approach is useful for training that occurs due to errors in initialisation and allows gradients to form. It helps deep generators to start the learning process without merging all samples to a single point, a known problem with previous failure modes of GANs. When the batchnorm is directly applied on all data layers, the output showed model instability and oscillation. The problem was overcome by refraining the batchnorm to the output and input layers of the generator and discriminator. Nair and Hinton (2010) used ReLU activation in the generator network and a Tanh function for the discriminator network. A bounded activation was used that increased the speed of learning and further covered and saturated the training distribution. The discriminator had a leaky rectified activation that increased the higher resolution modelling.

Reed *et al*. (2016) conducted an interesting and related study on generative adversarial text to image synthesis. The research involved automatic synthesis of realistic pictures from text. The researchers used deep convolutional generative adversarial networks to generate high-resolution images for various categories such as faces, birds, room interiors, animals, etc. As per this method, it is possible to write a text such as this small bird has a pink breast and crown and black primaries and secondary's, to generate an image of a thrush sitting on a branch. The main feature of the research was to use conditional GANs and make the model conditions on text descriptions rather than class labels. They used a manifold interpolation to regularize for the GAN generator and this helped to increase the quality of generated samples. They trained a deep convolutional GAN that was

conditioned on text features that are encoded with hybrid character level convolutional neural network. The generator network and discriminator used feed-forward inference of conditioned text feature.

Szegedy *et al.* (2013) argue that generative adversarial networks are falsely considered with adversarial examples. These are seen with gradient-based optimisation of input datasets in a classification network. However, adversarial networks are not applications to train generative models. Rather, they are used to show neural networks that behave in certain ways and classify pictures differently with a greater confidence level, even though the variations between these images are not clear to humans. Such examples would indicate that GANs could be inefficient, since, the implication is that discriminative networks tend to recognise various classes of objects without copying any human-understandable features of the class.

Tu (2007) used discriminative criteria in training a generative model. These methods use features that are not acceptable for deep generative model. The methods are complex and difficult to develop, since, they use probability ratios that cannot be calculated with variational approximations of probability. In some instances, noise-contrastive estimation was used to train generative model with weights learning and help to separate discriminate data with fixed noise distribution. A trained model with noise distribution helps in training other models to increase the quality. It serves as an informal competition system that is the same as a formal competition model, applied on adversarial networks. The discriminator in this case is taken as the ratio of the densities probabilities of the model distribution. It needs to have the capability to run evaluations from other densities.

Background: Generative methods that create samples from images and other data distributions see application in a number of areas such as image generation, text and speech analysis, developing images from sentences in image-to-image translations, image in painting, etc. A number of approaches are available for generation and these include autoregressive models, variational autoencoders or VAEs and generative adversarial networks. Some examples of autoregressive models are PixelCNN and it creates well-defined images, they are slow and latent representation is not possible. Another tool is VAEs and this can be trained easily but they generate blurred images. GAN is selected for this research, since, it generates sharp images but with small resolutions and the training is somewhat unstable. Some hybrid models are also available that use the features of the three models but the image quality is low (Ngiam *et al.*, 2011).

The generator in GANs generates an image using a latent code and the best condition is when the image distribution is the same as training distribution. A discriminator network is used to evaluate the images. A gradient is created that can guide the networks to develop the best image. Generator is persistent and used through the exercise while the discriminator serves as an adaptive loss and it is removed after the training of the generator is completed. Some problems can emerge in this process. When the distance between the two distributions is measured, the gradients can show random directions. This happens when differences are evident. The Jensen-Shannon divergence was used originally to measure the distance and other methods such as least squares, Wasserstein distance and absolute deviation with margin. In this research, the Wasserstein loss and the least-square loss methods are used (Theis *et al.*, 2015).

One obvious problem is that when high-resolution images are generated, these can be easily identified from the training images and this increases the gradient problem. In other instances, large resolutions mean that small batches are used, since, there are system memory problems. However, the generator and discriminator can be developed in phases by initially using low-resolution images and then adding new layers to increase the resolution. This method increases the training speed. An important point is that GANs does not need the full training data distribution to be indicated in the model. In the initial years, some adjustments had to be made between trade-off and image quality variation. The extent of variation is understudy through methods such as inception score, multi-scale structural similarity, explicit test and birthday paradox (Gulrajani *et al.*, 2017). A new method for measuring is developed in this research and discussed.

MATERIALS AND METHODS

Method of progressive growing with GANs: The main method used is to begin with low-resolution pictures and then add layers to increase the resolution. Generator and discriminator networks expand simultaneously, since, they are mirror images. The layers that are added in both networks can be trained during the process. When new layers are added, these are faded and blended in smoothly and this process helps to remove shocks to the well-defined and trained smaller-resolution layers (Odena *et al.*, 2017). Table 1 gives the network architecture of the two generators used for the CelebA dataset.

The two networks have three layers that were introduced individually during the training. The table give

Table 1: Generator and discriminator applied with CelebA-HQ to produce 1024×1024 images

Generator	Act.	Output shape	Params
Latent vector	-	512×1×1	-
Conv 4×4	I.RcI.U	512×4×4	4.2M
Conv 3×3	I.RcI.U	512×4×4	2.4M
Upsample	-	512×8×8	-
Conv 3×3	I.RcI.U	512×8×8	2.4M
Conv 3×3	I.RcI.U	512×8×8	2.4M
Upsample	-	512×16×16	-
Conv 3×3	I.RcI.U	512×16×16	2.4M
Conv 3×3	I.RcI.U	512×16×16	2.4M
Upsample	-	512×32×32	-
Conv 3×3	I.RcI.U	512×32×32	2.4M
Conv 3×3	I.RcI.U	512×32×32	2.4M
Upsample	-	512×64×64	-
Conv 3×3	I.RcI.U	256×64×64	1.2M
Conv 3×3	I.RcI.U	226×64×64	590k
Upsample	-	256×128×128	-
Conv 3×3	I.RcI.U	128×128×128	295k
Conv 3×3	I.RcI.U	128×128×128	148k
Upsample	-	128×256×256	-
Conv 3×3	I.RcI.U	64×256×256	74k
Conv 3×3	I.RcI.U	64×256×256	37k
Upsample	-	64×512×512	-
Conv 3×3	I.RcI.U	32×512×512	1.8k
Conv 3×3	I.RcI.U	32×512×512	9.2k
Upsample	-	32×1024×1024	-
Conv 3×3	I.RcI.U	16×1024×1024	4.6k
Conv 3×3	I.RcI.U	16×1024×1024	2.3k
Conv 1×1	linear	3×1024×1024	51
Total trainable parameters			23.1M
Diseriminator			
Input image	-	3×1024×1024	-
Conv 1×1	I.RcI.U	16×1024×1024	64
Conv 3×3	I.RcI.U	16×1024×1024	2.3k
Conv 3×3	linear	32×1024×1024	4.6k
Downsapmple	-	32×512×512	-
Conv 3×3	I.RcI.U	32×512×512	9.2k
Conv 3×3	linear	64×512×512	18k
Downsapmple	-	64×256×256	-
Conv 3×3	I.RcI.U	64×256×256	37k
Conv 3×3	linear	128×256×256	74k
Downsapmple	-	128×128×128	-
Conv 3×3	I.RcI.U	128×128×128	148k
Conv 3×3	linear	256×128×128	295k
Downsapmple	-	256×64×64	-
Conv 3×3	I.RcI.U	256×64×64	590k
Conv 3×3	linear	512×64×64	1.2M
Downsapmple	-	512×32×32	-
Conv 3×3	I.RcI.U	512×32×32	2.4M
Conv 3×3	linear	512×32×32	2.4M
Downsapmple	-	512×16×16	-
Conv 3×3	I.RcI.U	512×16×16	2.4M
Conv 3×3	linear	512×16×16	2.4M
Downsapmple	-	512×8×8	-
Conv 3×3	I.RcI.U	512×8×8	2.4M
Conv 3×3	linear	512×8×8	2.4M
Downsapmple	-	512×4×4	-
Minibatch stddcv	-	513×4×4	-
Conv 3×3	I.RcI.U	512×4×4	2.4M
Conv 1×1	linear	512×1×1	4.2M
Fully-connected	linear	1×1×1	513
Total trainable parameters			23.1M

the values for generator and discriminator. In the table, the last row, Conv. 1×1 from the generator table, links with to RGB. The exercise started with a 4×4 resolution and the networks were trained until the discriminator showed images of 800 kb. The two phases of fade and stabilise where used for the first of the three layers for the next set of 800 kb pics, then they were stabilised and then again faded until the training was completed. Latent vectors [-1, 1] from the study on a hyperspace of 512 dimensions were represented by training and generated pictures (Yu *et al*., 2015).

A leaky ReLU that had leakiness of 0.2 for all layers in the two networks was used for linear activation. Pixelwise normalisation was used after each conv 3×3 layer. All bias parameters were initialised to zero and the weights were standardised to the normal distribution. The across-minibatch was injected for the extra feature map at 4×4 resolutions when the end of the discriminator was reached. In Table 1, the operations of upsampling and downsampling are linked to 2×2 element replication with average pooling. Networks were trained with Adam with values of • = 0.001, $•_1$ = 0, $•_2$ = 0.99 and • = 108. Exponential running average was used for the generator weight with decay of 0.999 for visualising the generator output during training. Size of the minibatch was 16 for resolutions of $4^2 = 128^2$ and then the size was decreased as per $256^2 = 6$, $512^2 = 6$, $1024^2 = 3$, so that, the memory was not exhausted. The WGAN-GP was used and the process was shifted between the generator and the discriminator with ncritic = 1 for the per-minibatch. A fourth term was used for the discriminator loss with a small weight of 12 to prevent drifting of discriminator output from zero.

System requirements: For this research, several datasets, images, networks and other assets were used. Datasets that were used are CIFAR 10, LSUN and CelebA. The CelebA dataset provides output resolution of 1024×1024 pixels. Online repositories such as GitHub provide assets such as the TensorFlow source code and Theano. For this research, TensorFlow was used, since, it provides high performance, provides dataset in TFR Records and the code quality is good. Pre-trained networks are available in data repositories while others were developed through training and were stored as Python PKL files. The following system specifications were used in the research (Karras *et al*., 2017):

- Windows 10 operating system was used while Linux can be used if available
- Anaconda3 with 64-bit Python installation and numpy 1.13.3
- A NVIDIA Pascal. Volta GPU, 16GB DRAM
- NVIDIA 391.25 driver, cuDNN 7.12 and CUDA toolkit 9.0

Importing pre-trained networks: Pre-trained networks and the networks generated by the training script can be imported with the pickle system. Importing is allowed when the GAN code repository and directory are included in PYTHONPATH and the tf. Session () assets are created and used as default. Three instances of PKL file are available. The following codes can be used Yang *et al*. (2017):

```
# Import official CelebA-HQ networks.
with open('karras2018iclr-celebahq-1024x1024.pkl', 'rb') as
file:
    G, D, Gs = pickle.load(file)
    # G = Instantaneous snapshot of the generator, mainly
useful for resuming a previous training run.
    # D = Instantaneous snapshot of the discriminator, mainly
useful for resuming a previous training run.
    # Gs = Long-term average of the generator, yielding
higher-quality results than the instantaneous snapshot.
```

After the networks are imported, the Gs.run () can be called and images generated for specific latent vectors and the generator network can be included in the TensorFlow code. The following steps are suggested.

- Place the GAN repository to the PYTHONPATH
- Use pip install-r requirements-pip.txt and install the Python packages
- The file import_example.py can be downloaded from the networks/tensorflow-version/example_import _script
- The file karras2018iclr-celebahq-1024x1024.pkl can be downloaded from networks/tensorflow-version into the same folder
- Run the script with the Python file
- The script generates 10 png images

The GAN repository has a command line to create replicas of the data used. It also has many utilities to process the data. A methodology and experiments were used for the equalised learning rate. An initialisation of N (0, 1) was used and the weights were scaled at runtime. Parameters were set at $w^{\wedge}_{i} = w_i/c$ where, w_i is the weight and c is the constant of each layer. When this process is used during initialisation, the scale-invariance of adaptive stochastic gradient process such as RMSProp is brought into play. This step helps the gradient to normalise by the amount of stddev, so that, the update does not rely on the parameter scale. Therefore, when parameters have higher values, they take more time to normalise and adjust increasing the learning time. The process used helps to keep the learning speed the same for all weights. A possibility exists that the discriminator and generator magnitudes would go beyond the normal range due to competition between them. To overcome this problem, the feature vector is normalised for every pixel with reference to the length obtained in the generator for every convolutional layer (Grinblat *et al*., 2017). A local response normalisation is used for this purpose. This is given as.

$$b_{x,y} = a_{x,y} / \sqrt{\frac{1}{N}\sum_{j=0}^{N-1}\left(a_{x,y}^{j}\right)^2 + \epsilon}$$

Where:

- ϵ = 10^8

N = Feature maps

$a_{x,y}$ and $b_{x,y}$ = Normalised feature vector for pixel (x, y)

While this variant does not reduce the generator, the increase of signal magnitudes does not happen. The actual process of comparing GANs takes a lot of time and efforts. Therefore, automated methods are used with proper metrics. Available methods such as MS-SSIM are not helpful to detect small problems such as loss of colours, variations and textures and they are not helpful in evaluating image quality. Considering that an optimum generator would create samples with local image structure that is the same for all training, another approach was used. This is the multi-scale similarity between local images obtained from Laplacian pyramid for target images that begin with 16×16 pixels. The pyramid increases by two times until the optimum resolution is obtained. Every layer encodes the difference and posts to the up-sampled image of the previous layer. It is seen that one Laplacian pyramid layer links to a band of a specific spatial frequency. A few images were randomly sampled and a number of descriptors were obtained for each level of the pyramid (Grinblat *et al*., 2017). The descriptor was a 7×7 pixel having three colour channels indicated by $x \cdot R^{7\cdot7\cdot3} = R^{147}$.

The patches were indicated from 11level of the training set and they generated sets of $\{x_i\}_{i=1}^{2^8}$ and $\{y_i\}_{i=1}^{2^8}$. The x set was normalised first and y set was normalised with

reference to the standard deviation and mean of the colour channel. The statistical similarity was calculated by considering the Wasserstein Distance (SWD). Lesser distance shows that the patches distribution is the same indicating that training pictures and samples of the generator are the same in appearance. The patch distance is taken from the 16×16 pictures and it shows the similarity in larger images. Smallest level patches tend to encode data about the picture noise and edge sharpness.

RESULTS AND DISCUSSION

The CelebA data set was used to create 500 pictures in 1024×1024 resolutions. A set of randomly selected images were collected and placed in the dataset. These images showed differences in visual quality and resolution. Some of these had faces of people in different positions while others were only of the face. A number, if image-processing steps were applied to maintain quality. The method of creating images is shown in Fig. 4. The image quality was improved by submitting each image through a convolutional autoencoder that filtered JPEG features from the dataset and then through GANs trained 4x high resolution network. In some instances, the face was beyond the image and this problem was solved by padding and filtering that increased the picture sizes. An oriented crop angle was selected using facial landmark symbols from the CelebA datasets. The image is illustrated in Fig. 4. The following commands and methods were used for CelebA dataset:

$$x' = e_1 - e_0$$

$$y' = \frac{1}{2}(e_0 + e_1) - \frac{1}{2}(m_0 + m_1)$$

$$c = \max\left(4.0 \cdot |x'|, 3.6 \cdot |y'|\right)$$

$$x = \text{Normalize}\left(x' - \text{Rotate90}\left(y'\right)\right)$$

$$y = \text{Rotate90}(x)$$

In the above computation:

e_0, e_1, m_0 and m_1 = The 2D pixel locations of the eyes that were taken as salient features and landmarks and the mouth features

c and s = The centre and size of the required crop-rectangle

x and y = Represent the orientation

The formulas were used to empirically obtain and ensure that the crop rectangle remains constant where the picture is seen from different views. After the crop angle was calculated, the rectangle was transformed to

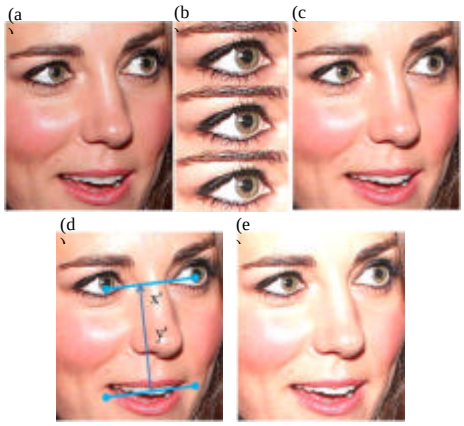

Fig. 4: Generating the CelebA dataset. The first image: a) Is a raw image of the subject. The visual quality of the image was enhanced in; b) By using JPEG artefact filter. A 4x super-resolution was then applied. The image was extended by mirror padding; c) A and with Gaussian filters. These methods produced an acceptable picture with depth-of-field effect; d) Facial features and landmarkswere located to develop a crop region. High-quality resampling was used to develop the final image and e) with a 1024×1024 resolution. This was the final output of the exercise

Table 2: Values for unsupervised runs

Methods	References	Inception score
ALI	Dumoulin *et al.* (2016)	5:29±0:03
GMAN	Durugkar *et al.* (2016)	6:01±0:17
Improved GAN	Salimans and Kingma (2016	6:83±0:07
CEGAN-Ent-VI	Dai *et al.* (2017)	7:06±0:08
LR-AGN	Yang *et al.* (2017)	7:16±0:16
DFM		7:73±0:14
WGAN-GP 7	Gulrajani *et al.* (2017)	7:87±0:00
Splitting GAN	Grinblat *et al.* (2017)	7:89±0:09
The best run		8:81±0:05
Computed from 10 runs		8:56±0:06

4096×3096 pixels with a bilinear filter. A box filter was used to reduce the image to a 1024×1024 resolution. This processing was used in a large number of images from the dataset and images with acceptable results were included and the rest were removed. A frequency-based quality metric was used that uses images that had a power spectrum with a larger number of frequencies and which was radially symmetric. It is possible that blurry images and the ones with directional features developed problems with halftone patterns.

Results of CIFAR 10 are explained as follows. Several images of non-curated variety were generated without supervision and after the training were completed. These are illustrated in Fig 5 and 6 and the values of previous images with reference to inception scores are explained in Table 2.

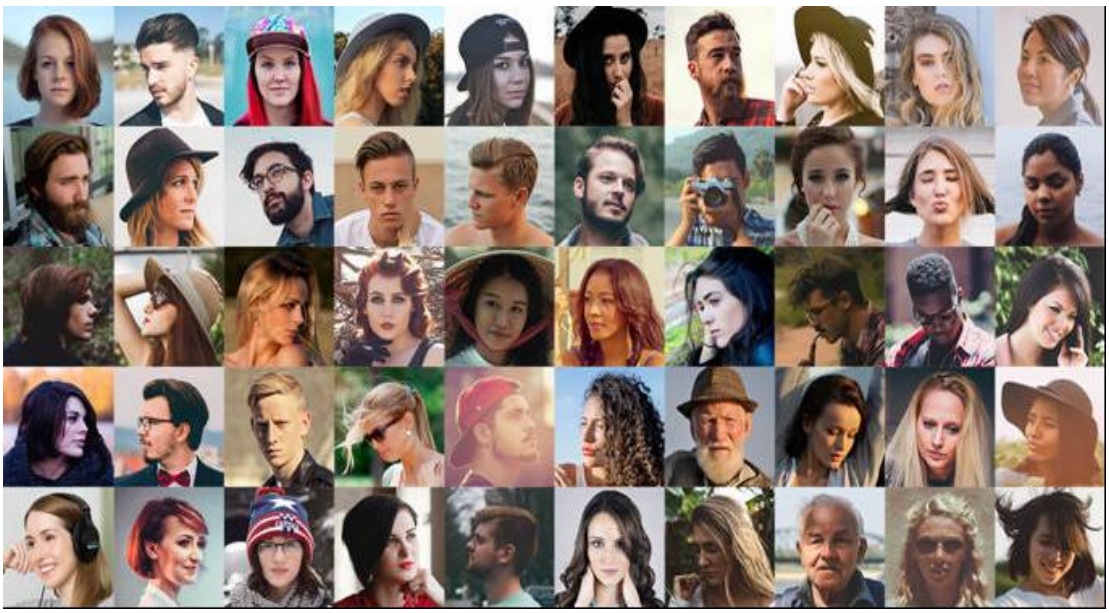

Fig. 5: First samples of non-curated images. CIFAR10 images, obtained through a network generated using a network that did not have label conditioning and was trained. Inception scores of 8.83 were obtained

Table 3: Conditioned label inception test scores

Methods	Reference	Inception score
DCGAN	Radford *et al.* (2015)	6:59
Improved GAN	Salimans and Kingma (2016)	8:08±0:08
AC-GAN	Odena *et al.* (2017)	8:24±0:08
SGAN	Huang *et al.* (2017)	8:58±0:13
WGAN-GP	Gulrajani *et al.* (2017)	8:68±0:13
Splitting GAN	Grinblat *et al.* (2017)	8:85±0:08

The second set of non-curated images are given in Fig. 6. The scores are reported in two methods. In the first method, the highest score obtained when the training was underway is given with±symbol and this refers to the standard deviation obtained by the inception score calculator. In the second method, the mean and standard deviations are obtained from the highest scores that are observed in training. The second method was found more appropriate since it produced better inception scores and the data was not augmented in any way. The inception scores are presented in Table 2. Table 3 gives the values for label-conditioned tests (Dumoulin *et al.*, 2016; Durugkar *et al.*, 2016).

It is clear from the above figures and tables that the generator synthesized MNIST values are in three colour channels. Using the method suggested by Radford *et al.* (2015), a pre-trained classifier was used to classify the digits. More than 100 images were generated and these are shown in Fig. 5 and 6. The KL divergence and histogram was also calculated. It is possible to cover all modes with low divergence. The QGAN-GP loss for the networks indicates that more modes are covered than the GAN loss. Divergence of KL gives more accurate results and measures than raw counts. When batch normalisation was used instead of individual normalisation, the results for equalised learning rate and pixelwise normalisation gave better results. When a minibatch std dev layer was added, the score improved more and the discriminator capacity was increased to 0.5%.

Figure 7 and 8 show the generated images of successful and unsuccessful results. These are examples of CelebA HQ and CelebA which were obtained by mirror augmentation for the tests. The sliced Wasserstein distance was used along with the Freschet Inception Distance (FID) and they were generated from 50 kb pictures. The figures show the images for the LSUN categories.

Different networks were trained for each and the same parameters were used. Training was done with the images and since the small images had less training data, mirror augmentation was used. For Fig. 7, the nearest neighbours were seen in the training data by using the feature-space distance. Activations were obtained from the VGG layers and crops were used to compare and exclude image background while searching for facial features. More 1024×1024 pictures were developed with CelebA HQ and SWD x 103 was applied for different levels. The FID was calculated for 50 k pictures and the value was 7.3. Please refer to images in Fig. 7.

While the pictures in Fig. 7 are replicas of the original data, Fig. 8 shows some aberrations. These were

Fig. 6: Second samples of non-curated pictures

non-converged and the pictures show a number of problems such as excessive pixilation, aliasing, loss of sharpness, excessive graininess and other problems. These errors are mainly due to flaws in the dataset. The applications, however, attempted to replicate them as they were given. CelebA is found to be appropriate for such work, since, the training pictures had problems such as aliasing, blur, compression and the generator could not reproduce them effectively. The differences between the training were highlighted by selecting a low-capacity network. The training was stopped after the total capacity was reached. Therefore, the images indicate non-converged pictures (Huang *et al*., 2017).

The study shows that the results for GAN have succeeded and stable training is possible. Some amount of stability for larger resolutions was also, achieved. However, photo-realistic images are still to be achieved. A better understanding and development of attributes such as anti-aliasing, image sharpness, clarity, visual presentation is yet to be achieved. It was also seen that some data sets were corrupted and this was not detected until the training and generation was completed. Therefore, methods to check for compatibility must be developed. Improvement is needed in the microstructure of the images and the system must learn to differentiate between curved and straight features of images. In addition, all the work was done on the face; work on objects such as furniture, buildings, animals, flowers and other types of objects, clouds, running water needs to be carried out. CelebA HQ provides acceptable results and ways of combining multiple methods needs to be investigated.

The study shows the link between progressive growth as SWD measures and the raw images. When training configurations with and without progressive growing is seen, then the progressive variant provides

Fig. 7: Samples of labelled images

Fig. 8: Unsuccessful sampling results

two advantages. The first is that converging with better results and the training time is reduced considerably. This improvement in convergence is important, since, it indicates a type of learning, forced on the network capacity that increases. If progressive growing of layers was not used, then all layers would be loaded and it would be difficult to find representations for the variations. When progressive loading is used, it is possible that the low-resolution layers would be converged earlier. The two networks would need to only refine the images and pass them to the next layer. Therefore, the highest values reach its optimal status and remain constant for the duration of the training.

It was also, seen during the experiments that the increase in speed of progressive growing rises when the output resolution increases. The training progress was measured in the number of images of the discriminator which is taken as a function of time when the final resolution of 1024×1024 is reached. Therefore, progressive growth is faster, since, the networks are less loaded and evaluation is faster at the beginning. After the full resolution is reached, the image throughput is the same for both methods. The study was successful in generating high resolution outputs. It is seen that larger pixel GAN results indicate that the quality of images show variation. The 8 Tesla V100 GPU was used for training and after training for 4 days, major differences between successive iterations were not evident. Therefore, the minibatch concept is useful and it depends on the output resolution to retain the memory used to acceptable levels. The optimum scores of CIFAR 10 reached 8.87 for label conditions and 7.9 for unsupervised. This variation is explained by the presence of redundant data in JPEG images in unsupervised conditions while with labelling, the data is filtered and cleaned. The goal is to improve the score of unsupervised conditions where filtering and clean-up is done through routines and the score are improved to more than 9.5.

The results also, show the extent to which the networks differentiate between content and style. Content refers to the visual attributes of the face such as size, colour and shape. Style refers to elements such as background and pose of the subjects. It is seen from the study that backgrounds do not create any issues, unless they are of the same colour as the face or the hair. In the samples taken, the background contrasted with the hair and skin colour. The subjects were facing the light in Fig. 5, some had shadows on the face. It is seen that when the contrast is clear, the networks do not get confused, since, a clear demarcation is available. It is possible to get clear images. In any case, most of the subjects had contrasting backgrounds, though the black colour hair tends to mix with the background after the conversion is over. Therefore, CelebA and GANs preserves the background and other details. This aspect needs further research.

Since, the same input data was used noise distribution was the same, it appears that the learning enhances as the resolution increases. It is possible that when the resolution increased, the amount of data that was sampled was higher, leading to enhanced learning and increased fidelity. It also, appears that the networks develop some amount of generalisation ability, since, backgrounds, colour, face shape, textures and other multiple attributes were converted. Some results that are not presented in this study showed faces as blobs and gross distortions of the facial features were seen. The reason for this distortion can only be attributed to corrupt data incomplete image information and other errors.

It is also, possible that the images were earlier of a very low resolution and later increased by data providers before the images were placed in the data set. As indicated earlier, a system is needed to review the data and suggest methods to pass data before accepting them for the test. One possible problem can arise when a representation of the generators distribution pg over the data x and the multilayer value needs to the synchronised with the generator during training. If the generator is trained more than the discriminator, then G can reduce to excess values and G then collapses.

CONCLUSION

The study presented findings from experiments using GANs with datasets of faces. Several experiments were run to generate faces based on the data. Advanced generator and discriminator networks were trained and used in the research. A new approach was used and this involved adding low-resolution layers, training the two networks and gradually adding layers of higher resolution until an output of higher resolution was generated. The results while satisfactory are far from perfect and more effort is needed in training the two networks. Correct data of the required fidelity and purity is needed, since, corrupted data produced distorted figures. The generated images were not affected by different backgrounds, skin and hair colour and with orientation of the subjects, since, some were facing the camera while others presented profiles. When the contrast between background and the face is clear, the networks do not get confused and produce images of good quality. Some more research is recommended to train the networks to generate images of objects and animals. The conclusions are that the approach of adding multiple layers and training the networks hold promise for further work.

REFERENCES

Arjovsky, M., S. Chintala and L. Bottou, 2017. Wasserstein generative adversarial networks. Proceedings of the 34th International Conference on Machine Learning, July 7-10, 2017, London, UK., pp: 214-223.

Ba, J.L., J.R. Kiros and G.E. Hinton, 2016. Layer normalization. Mach. Learn., 1: 1-14.

Chen, B.C., Y.Y. Chen, Y.H. Kuo and W.H. Hsu, 2013. Scalable face image retrieval using attribute-enhanced sparse codewords. IEEE. Trans. Multimedia, 15: 1163-1173.

Dai, Z., A. Almahairi, P. Bachman, E. Hovy and A. Courville, 2017. Calibrating energy-based generative adversarial networks. Proceedings of the ICLR 2017 Track International Conference on Learning Representations, April 24-26, 2017, Toulon, France, pp: 1-17.

Denton, E.L., S. Chintala and R. Fergus, 2015. Deep Generative Image Models Using a OBJ Laplacian Pyramid of Adversarial Networks. In: Advances in Neural Information Processing Systems, Cortes, C., N.D. Lawrence, D.D. Lee, M. Sugiyama and R. Garnett (Eds.). Curran Associates Inc., New York, USA., pp: 1486-1494.

Dumoulin, V., I. Belghazi, B. Poole, O. Mastropietro and A. Lamb *et al*., 2016. Adversarially learned inference. Mach. Learn., 1: 1-18.

Durugkar, I., I. Gemp and S. Mahadevan, 2016. Generative multi-adversarial networks. Mach. Learn., 1: 1-14.

Ghosh, A., V. Kulharia, V.P. Namboodiri, P.H. Torr and P.K. Dokania, 2018. Multi-agent diverse generative adversarial networks. Proceedings of the 2018 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2018), June 18-22, 2018, CVPR, Salt Lake, Utah, pp: 8513-8521.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu and D. Warde-Farley *et al*., 2014. Generative adversarial nets. Proceedings of the 27th International Conference on Neural Information Processing Systems, December 08-13, 2014, ACM, Montreal, Canada, pp: 2672-2680.

Gregor, K., I. Danihelka, A. Graves, D.J. Rezende and D. Wierstra, 2015. Draw: A recurrent neural network for image generation. Comput. Vision Pattern Recognit., 1: 1-10.

Grinblat, G.L., L.C. Uzal and P.M. Granitto, 2017. Class-splitting generative adversarial networks. Mach. Learn., 1: 1-10.

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin and A.C. Courville, 2017. Improved Training of Wasserstein GANs. In: Advances in Neural Information Processing Systems, Guyon, I., U.V. Luxburg, S. Bengio, H. Wallach and R. Fergus *et al*. (Eds.). Curran Associates Inc., New York, USA., pp: 5767-5777.

He, K., X. Zhang, S. Ren and J. Sun, 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE. Trans. Pattern Anal. Mach. Intell., 37: 1904-1916.

Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter, 2017. GANs trained by a two time-scale update rule converge to a local NASH equilibrium. Proceedings of the 2017 31st Annual International Conference on Neural Information Processing Systems (NIPS 2017), December 4-9, 2017, Long Beach, California, USA., pp: 1-38.

Huang, X., Y. Li, O. Poursaeed, J. Hopcroft and S. Belongie, 2017. Stacked generative adversarial networks. Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), July 21-26, 2017, Honolulu, Hawaii, USA., ISBN:978-1-5386-0458-8, pp: 1866-1875.

Ioffe, S. and C. Szegedy, 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. Proceedings of the 32nd International Conference on Machine Learning, July 07-09, 2015, Microtome Publishing, Lille, France, pp: 448-456.

Karras, T., T. Aila, S. Laine and J. Lehtinen, 2017. Progressive growing of GANs for improved quality, stability and variation. Neural Evol. Comput., 1: 1-26.

Khosla, A., W.A. Bainbridge, A. Torralba and A. Oliva, 2013. Modifying the memorability of face photographs. Proceedings of the IEEE International Conference on Computer Vision (ICCV'13), December 1-8, 2013, IEEE, Washington, DC., USA., ISBN:978-1-4799-2840-8, pp: 3200-3207.

Kingma, D.P. and M. Welling, 2013. Auto-encoding variational bayes. Mach. Learn., 1: 1-14.

Nair, V. and G.E. Hinton, 2010. Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, ACM, Haifa, Israel, ISBN:978-1-60558-907-7, pp: 807-814.

Ngiam, J., A. Khosla, M. Kim, J. Nam and H. Lee *et al*., 2011. Multimodal deep learning. Proceedings of the 28th International Conference on Machine Learning (ICML-11), June 28-July 2, 2011, Bellevue, Washington, USA., pp: 689-696.

Odena, A., C. Olah and J. Shlens, 2017. Conditional image synthesis with auxiliary classifier GANs. Proceedings of the 34th International Conference on Machine Learning (ICML'17) Vol. 70, August 6-11, 2017, JMLR.Org, Sydney, Australia, pp: 2642-2651.

Radford, A., L. Metz and S. Chintala, 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. Mach. Learn., 1: 1-16.

Reed, S., Z. Akata, X. Yan, L. Logeswaran and B. Schiele *et al*., 2016. Generative adversarial text to image synthesis. Neural Evol. Comput., 1: 1-10.

Salimans, T. and D.P. Kingma, 2016. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In: Advances in Neural Information Processing Systems, Lee, D.D., M. Sugiyama, U.V. Luxburg, I. Guyon and R. Garnett (Eds.). Curran Associates Inc., New York, USA., pp: 901-909.

Simonyan, K. and A. Zisserman, 2014. Very deep convolutional networks for large-scale image recognition. Master Thesis, Cornell University, Ithaca, New York.

Sohl-Dickstein, J., E.A. Weiss, N. Maheswaranathan and S. Ganguli, 2015. Deep unsupervised learning using nonequilibrium thermodynamics. Mach. Learn., 1: 1-18.

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna and D. Erhan *et al*., 2013. Intriguing properties of neural networks. Comput. Vision Pattern Recognit., 1:1-10.

Theis, L., A.V.D. Oord and M. Bethge, 2015. A note on the evaluation of generative models. Mach. Learn., 1: 1-10.

Tu, Z., 2007. Learning generative models via discriminative approaches. Proceedings of the 2007 I nternational IEEE Conference on Computer Vision and Pattern Recognition, June 17-22, 2007, IEEE, Minneapolis, Minnesota, USA., pp: 1-8.

Yang, J., A. Kannan, D. Batra and D. Parikh, 2017. LR-GAN: Layered recursive generative adversarial networks for image generation. Comput. Vision Pattern Recognit., 1: 1-21.

Yu, F., A. Seff, Y. Zhang, S. Song and T. Funkhouser *et al*., 2015. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. Comput. Vision Pattern Recognit., 1: 1-9.

Zeiler, M.D. and R. Fergus, 2014. Visualizing and Understanding Convolutional Networks. In: Computer Vision, Fleet, D., T. Pajdla, B. Schiele and T. Tuytelaars (Eds.). Springer, Cham, Switzerland, ISBN:978-3-319-10589-5, pp: 818.