CHAPTER 10

Image generation using generative adversarial networks

Omkar Metri and H.R Mamatha
Department of CSE, PES University, Bengaluru, India

10.1 Introduction to deep learning

Neural networks and deep learning are one of the greatest innovations in the field of artificial intelligence. The reason for the majority of the real tasks at hand such as image and face recognition, speech recognition, object detection, and natural language processing having the best solutions is neural networks and deep learning. Likewise, the majority of machine learning algorithms are great in learning patterns, classifications tasks such as category assignment and regression for interpreting the numerical values based on the available data. But the computers have struggled when asked for data generation. The only way out for gathering data to train the models is collection from different sources or manual creation of the data. This gave rise to generative modeling. In a nutshell, generative modeling is a robust way of understanding the data distributions in an unsupervised manner. These models aim to generate new data by learning true distributions of the data. The data distributions cannot be featured with perfection. As a consequence, with the help of the neural networks and deep learning, it can be approximated as a function of the true distribution. Two elegant architectures used for generation purposes are variational autoencoder (VAE) and generative adversarial network (GAN).

10.1.1 Generative deep learning

A few questions may strike the mind such as the differences between generative and discriminative models, the need of generative models, and others. This section answers these questions. A generative model basically categorizes the data sample based on how the data was generated. In other words, categorizing the data sample based on generation assumptions. On the other hand, discriminative models categorize the data sample based on the differences ignoring the data generation details. If we imagine the speech to language classification task, the generative approach would be learning the languages and classifying based on the gained knowledge, and the discriminative approach would be determining the linguistic differences without learning any language and predicting the language of the speech. For the x inputs and y labels, the generative algorithms would learn the

Generative Adversarial Networks for Image-to-Image Translation
https://doi.org/10.1016/B978-0-12-823519-5.00007-5

joint probability, i.e., $p(x, y)$. Similarly, discriminative algorithms would learn the conditional probability, i.e., $p(y|x)$ [1]. The generative models are being extensively used for producing realistic images of artwork, simulation purposes, time-series data, reinforcement learning as well as for generalizing features. A few prominent models are nuclear autoregressive density estimator (NADE) [2], masked autoencoder density estimator (MADE) [2], pixel recurrent neural networks (PixelRNN), pixel convolution neural networks (PixelCNN), variational autoencoder (VAE), Markov chains, and generative adversarial network (GAN).

PixelRNN made an impact and became one of the promising solutions for image compression, reconstruction, generation, and others [3]. The model basically loads the image and at a given point of time scans one row and one pixel within that row. The idea is to predict the distribution of the next pixel with the possible values. Joint distribution of the pixel is basically the product of the conditional probabilities, thus making a sequence problem. Two types of architecture are experimented with, i.e., row LSTM (long-short-term memory) and diagonal BiLSTMs. The former uses a unidirectional layer of LSTM to scan the image row by row along with one-dimensional convolution. The diagonal BiLSTMs scan the image in the diagonal fashion. To increase the convergence speed and propagation of the signals explicitly, residual connections were added to the architecture. Usually, the realistic pictures have three channels, i.e., RGB. Hence, while predicting a pixel for the R channel, the context is the previously generated pixels to the left and above. Similarly, the context for the G channel remains the previously generated pixels along with the dependency on the R channel. Likewise, the B channel will have dependency on the generated pixels, the R channel and the G channel. To ensure the dependencies, masks are applied, i.e., masked convolution. Also, PixelCNN has been modeled using CNN. The model has been tested on MNIST, CIFAR-10, and ImageNet data. Fig. 10.1 shows the generation samples of CIFAR-10 and 32×32 ImageNet.

The sequential generation is slow, which is a major drawback of pixelRNN/CNN. Also, the training of pixelCNN is faster compared to pixelRNN. Van den Oord et al. [4] present an improvement over the pixelCNN termed a Gated PixelCNN. It is computationally efficient and surpasses the pixelRNN in Ref. [3]. The vanilla PixelCNN ignored the content to the right of the current pixel, termed the blind spot (Fig. 1 in Ref. [3]). The gated architecture took off the blind spots with the help of two convolution stacks: one for a horizontal stack (current row till the current pixel) and the other for a vertical stack (all the rows above the current row). Using conditional modeling of images, the conditional PixelCNN model generated realistic images of different classes. Also, the model was tested on human images. It performed well on generating images of the same person with different postures (Fig. 10.2) and modeling on eight classes (Fig. 10.3). It also demonstrated the use of PixelCNN as an image decoder. The generated samples were of average quality depicting the model's capability of capturing the variations on objects. Another improvement on

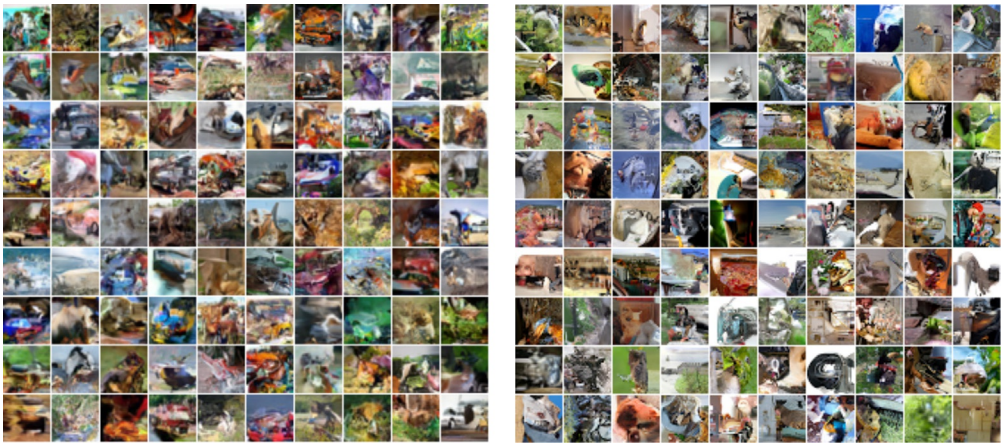

Fig. 10.1 Generation samples of CIFAR-10 (left) and 32×32 ImageNet (right) [3].

Fig. 10.2 Source image (left) and image generation samples (right) [4].

PixelCNN with a logistic likelihood is proposed in Ref. [5]. The next section deals with a brief introduction of autoencoders and a thorough explanation of the VAE.

10.1.2 Variational autoencoder

Autoencoders are feed-forward neural networks where the input is equivalent to the output. The autoencoder has three parts, i.e., encoder, code, and decoder. The input is fed to the encoder which compresses and produces a code. The code is termed latent space representation. In turn, the decoder regenerates the input using the latent representation. The output is a degraded representation of the input [6]. Autoencoders are used for

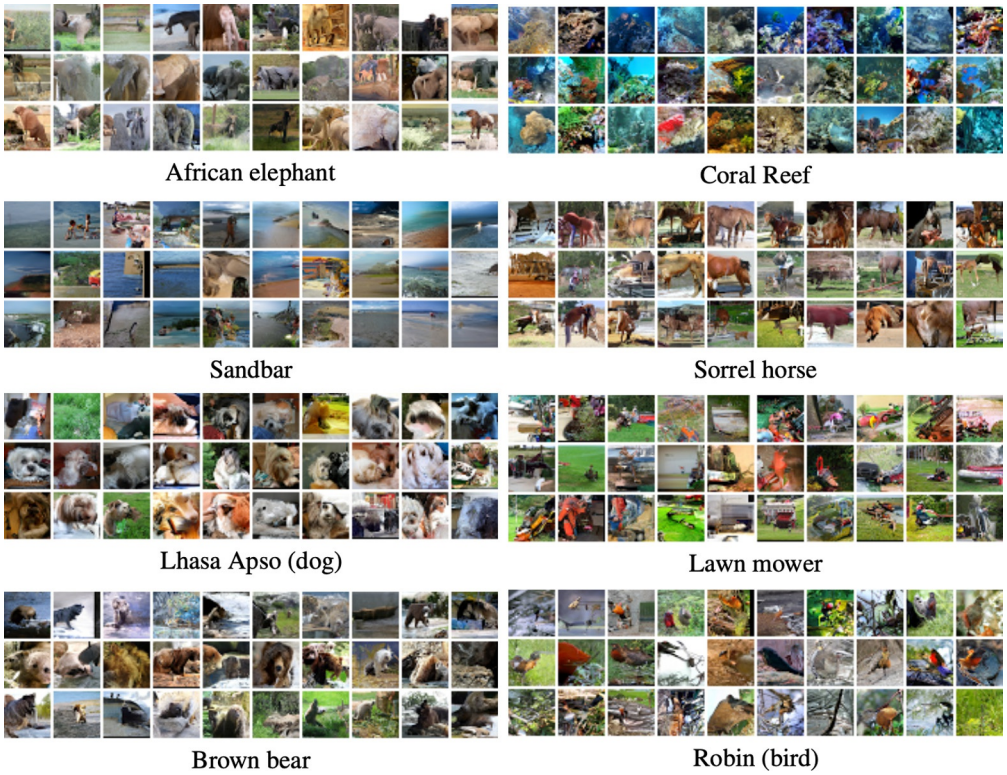

Fig. 10.3 Conditional image generation on eight classes [4].

anomaly detection purposes, information retrieval, image denoising, medical aging, popularity prediction, and others [7].

Many variations of the autoencoders have been used in a variety of applications. The prominent ones are sparse, stacked, denoising, and variational autoencoders. The standard autoencoder encodes the images as latent vectors, trying to memorize the images. Therefore, generating new images is not possible as the task of producing latent vectors depends on the input images. VAE solves the problem efficiently by representing the data into latent space which roughly follows the normal Gaussian distribution. Hence, feeding a randomly sampled data from the distribution to the decoder will generate a new image. To measure the efficiency, two separate losses are calculated. One is the mean squared error (generative loss) and the other is the Kullback-Leibler (KL) divergence measuring the proximity between latent space and normal Gaussian distribution. The simplest method to optimize KL divergence is by allowing the encoder to produce a code of two vectors (means and standard deviations) and thereby picking a sample for generating an image [8, 9].

The availability of huge amounts of data has driven visual forecasting. The drawbacks of the traditional approaches like nearest neighbor algorithms and transferring raw trajectories are computationally expensive, output space with high dimensions, encoding difficulty due to the pixel color variation in the frames, and blurry predictions. The challenges are addressed by predicting the dense pixel trajectories using conditional VAE [10]. The results indicate the model's capability of learning representations with less data and commendable visual forecasting from static images (Figs. 3 and 4 in Ref. [10]). VAEs have been used in the music sector for style transfer between music genres [11]. Also, VAEs have been employed in medicine [12] and anomaly detection [13].

10.2 Introduction to GAN

GANs [14] are an amazing AI innovation fit for making pictures, sound, and recordings that are unclear from the real thing. The section explains GAN in relation to the concept of game theory, i.e., Nash equilibrium, architecture, and training problems.

10.2.1 Nash equilibrium

Nash equilibrium is an important concept of game theory named after the inventor, i.e., John Nash. The best end result is dependent on the behavior and interaction of the participants in the game. In other words, the optimal solution in a noncooperative game is where the player cannot change the initial strategy. Basically, the player does not gain anything by deviating from the initial strategy under the assumption that other players keep the strategies unchanged. The game may include multiple equilibria or none of them [15, 16].

For example, assume two companies A and B. The companies are determining if they should start an advertising campaign to launch their products. If both the companies choose to advertise, each company acquires 100 customers. If only one of them chooses to campaign, then the company will acquire 200 customers. If neither of them campaigns, then no customers are acquired (Table 10.1).

Company A should advertise as it provides a good profit and reward, rather than not advertise. A similar scenario comes up for company B as well. Hence, both the companies opting for advertisement is a Nash equilibrium. Another common example associated with Nash equilibrium is the prisoner's dilemma [16].

Table 10.1 Reward table.

Company A, B	Advertise	Do not advertise
Advertise	100, 100	200, 0
Do not advertise	0, 200	0, 0

10.2.2 GAN and Nash equilibrium

A GAN setup includes two neural network systems in opposition to one another—one to create fakes (generator) and one to spot them (discriminator). The term generative indicates the idea of creating new data depending upon the training data. The term adversarial indicates a gamelike framework with two networks, i.e., generator and discriminator. The generator produces the realistic data which is similar to the training data, whereas the discriminator's task is to identify fake data produced by the generator from the real data coming from the training sample (Fig. 10.4) [18–23]. GAN is built on the concept of a zero-sum noncooperative game, i.e., minimax. In other words, one player maximizes its action and the other player minimizes those actions. Before diving into deep mathematics, Eq. (10.1) represents the variable notations with Eqs. (10.2)–(10.4) representing log-loss, KL divergence, and Jensen-Shanon divergence (JSD) function notations used in the rest of the chapter.

$$\begin{gathered} z \rightarrow \text{Noise vector} \\ x \rightarrow \text{Original training data} \rightarrow x_r \\ G(z) \rightarrow \text{Generator output} \rightarrow x_f \\ D(x) \rightarrow \text{Discriminator output for } x_r \\ D(G(z)) \rightarrow \text{Discriminator output for } x_f \end{gathered} \tag{10.1}$$

$$E(p|\, y) = -\frac{1}{N}\sum (y_i \times \log(p_i)) + ((1-y_i) \times \log(1-p_i)) \tag{10.2}$$

$$D_{KL}(P \,\|\, Q) = \int p(x) \log \frac{p(x)}{q(x)}\, dx \tag{10.3}$$

$$JSD(P \,\|\, Q) = \frac{1}{2} D_{KL}(P \,\|\, M) + \frac{1}{2} D_{KL}(Q \,\|\, M) \tag{10.4}$$

The discriminator is a binary classifier with the intention of categorizing the image as real or fake. As a consequence, the model should showcase high and low probabilities for

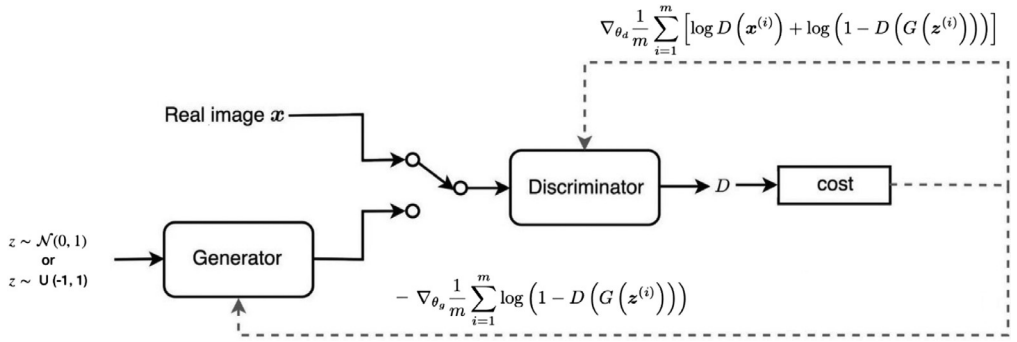

Fig. 10.4 GAN architecture [17].

real and fake data. Hence, the outcome of $D(x)$ and $D(G(z))$ lies in the range 0 and 1. The discriminator's task is to maximize and minimize the probabilities of $D(x)$ and $D(G(z))$, respectively, whereas the generator maximizes the probability of $D(G(z))$. The discriminator and generator aim at achieving Eqs. (10.5), (10.6), respectively. Therefore, the value function of the minimax game is defined as Eq. (10.7). $E(\cdot)$ in Eq. (10.7) is the binary cross entropy, i.e., log-loss function. The noise $\mathbf{z}$ follows normal or uniform distribution. According to game theory, the convergence of GAN is achieved when the discriminator and generator reach Nash equilibrium, i.e., $-\log_e 4$ (details in Section 10.2.2.1).

$$\text{Discriminator}: \max E_{x\sim p(x)}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))] \tag{10.5}$$

$$\text{Generator}: \max E_{z\sim p(z)}[\log D(G(z))] \tag{10.6}$$

$$\text{GAN}: \min_G \max_D V(D, G) = E_{x\sim p(x)}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))] \tag{10.7}$$

10.2.2.1 Nash equilibrium proof

Assume,

$$\begin{aligned} A &= P(x_r) \\ B &= P(x_f) \\ y &= D(x) \end{aligned} \tag{10.8}$$

From the Radon-Nikodym theorem, $G(z)$ can be approximated to x. In addition, from Eqs. (10.2), (10.8), Eq. (10.7) can be written as

$$\min_G \max_D V(D, G) = \int_y A \log y + B \log(1 - y)\, dy \tag{10.9}$$

The optimal discriminator is obtained by maximizing the integrand in Eq. (10.9). Hence, the integrand is rewritten as

$$f(y) = A \log y + B \log(1 - y) \tag{10.10}$$

To find the maximum of Eq. (10.10), $f'(y) = 0$ and $f''(y) < 0$ should satisfy.

$$\begin{aligned} f'(y) &= 0 \\ \frac{A}{y} - \frac{B}{1-y} &= 0 \\ y &= \frac{A}{A+B} \end{aligned} \tag{10.11}$$

Eq. (10.11) boils down to,

$$D(x) = \frac{P(x_r)}{P(x_r) + P(x_f)} = \frac{1}{2} \,|\text{if}\, P(x_r) = P(x_f) \tag{10.12}$$

Eq. (10.12) indicates the discriminator's puzzled situation on feeding original and generated images. Hence, Eq. (10.9) becomes:

$$\begin{aligned}
V(D,G) &= \int_{\gamma} A\log\frac{A}{A+B} + B\log\frac{B}{A+B}d\gamma \\
&= \int_{\gamma} A\log\frac{A}{A+B} + B\log\frac{B}{A+B} + (\log_e 2 - \log_e 2)(A+B)d\gamma \\
&= \int_{\gamma} A\log\frac{A}{A+B} + B\log\frac{B}{A+B} + \log_e 2(A+B)d\gamma - \int_{\gamma}\log_e 2(A+B)d\gamma \\
&= \int_{\gamma} A\log\frac{A}{A+B} + A\log_e 2 + B\log\frac{B}{A+B} + B\log_e 2 d\gamma - \log_e 2\int_{x}(A+B)d\gamma \\
&= \int_{\gamma} A\log\frac{A}{\frac{A+B}{2}} + B\log\frac{B}{\frac{A+B}{2}}d\gamma - 2\times\log_e 2 \\
&= D_{KL}\left(A\,\|\,\frac{A+B}{2}\right) + D_{KL}\left(B\,\|\,\frac{A+B}{2}\right) - 2\times\log_e 2 \\
&= 2JSD(A\,\|\,B) - \log_e 4
\end{aligned} \tag{10.13}$$

Eq. (10.13) is rewritten as

$$V(D,G) = -\log_e 4 + 2JSD\left(P(x_r)\,\|\,P\left(x_f\right)\right) \tag{10.14}$$

From the JSD divergence, $JSD(P(x_r)\,\|\,P(x_f))$ is zero when $P(x_r) = P(x_f)$. Hence, Eq. (10.14) reduces to $-\log_e 4$ proving that Nash equilibrium (global minimum) for the minimax game is $-\log_e 4$.

10.2.2.2 Training problems

Since the advent of GAN, a lot of research has been conducted. Consequently, numerous problems are associated with the training. A few are stated below [17, 24]:

1. Modal collapse: is the collapsing of the generator to produce the same kind of image for the possibly different latent vectors. The aim of the generator is Eq. (10.6). Consider the generator is trained substantially without updating the generator. In this case, the generated images will converge on finding the optimal image fooling the discriminator, thereby becoming independent of **z** (Eq. 10.15). Hence, the gradient turns zero w.r.t. **z** and the mode collapses to a single point. One prominent affecting factor

is the learning rate. It is recommended to use a low learning rate and add noise to real and generated images during training. Mode collapse is a challenging problem to date.

$$x^* = \text{argmax}\, D(x) \tag{10.15}$$

2. Nonconvergence and stability: Due to the diminished gradient, the discriminator becomes more powerful and the generator's gradient vanishes. Hence, GAN does not learn anything. Also, the stability of the model is a major concern. In other words, model parameters destabilize and therefore are responsible for nonconvergence. Lipschitz regularization has shown great success in stabilizing GAN training [25] and a few important considerations during training are cited in Ref. [26].
3. Early stopping and hyperparameters: Early stopping is terminating the training due to an abrupt increase or decrease in loss. Hyperparameter selection like loss function and imbalance between the two components can result in overfitting.

10.2.3 VAE-GAN

Larsen et al. [27] presented VAEGAN architecture by combining VAE and GAN outplaying the trivial VAE. The flow is indicated in Fig. 10.5A. The disadvantage with the VAE is the generation of blurry images. Hence, the loss function of the VAE's decoder is

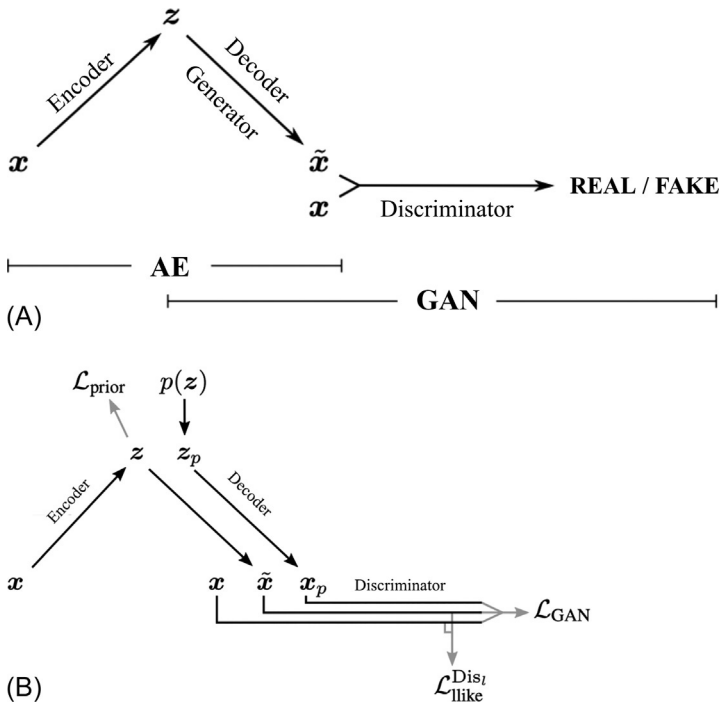

Fig. 10.5 VAE-GAN architecture [27]: (A) vanilla VAE-GAN and (B) VAE-GAN with auxiliary generator.

replaced with the loss metric learned through the GAN. Thus, no assumptions are made regarding the loss function as it uses the GAN's discriminator to categorize the image as real or fake. In other words, the generator (VAE decoder) uses this information to produce less blurry images. The original VAE generates samples which differ in a Gaussian way resulting in blurry images. VAEGAN overcomes the problem by sending the reconstructions and ground truth to the discriminator assuming that one hidden layer of the discriminator will differ in a Gaussian way. Hence, one of the hidden layers of the GAN discriminator is used for VAE loss. The reason for not choosing the final output layer is due to the lack of variation learning between the real and fake.

$$L_{GAN} = \log(Dis(x)) + \log(1 - Dis(Dec(z))) + \log(1 - Dis(Dec(Enc(x)))) \quad (10.16)$$

The architecture is refined by adding an auxiliary generator over the generator cum decoder (Fig. 10.5B). The discriminator is set to receive and classify images from three sources, i.e., original (x), samples from normal distributions (x_p), and VAE ($\tilde{x}$). Hence, the outputs generated by the auxiliary and decoder generator are treated as fake. The objective of the GAN is Eq. (10.16) and the results obtained are better compared to vanilla VAE-GAN (Fig. 10.6).

10.3 Applications

Since the innovation of GANs, they have been extensively used by the experts and researchers. It is one of the remarkable innovations in the field of deep learning. GANs can be used for image editing by reconstructing the image. GANs can be employed for

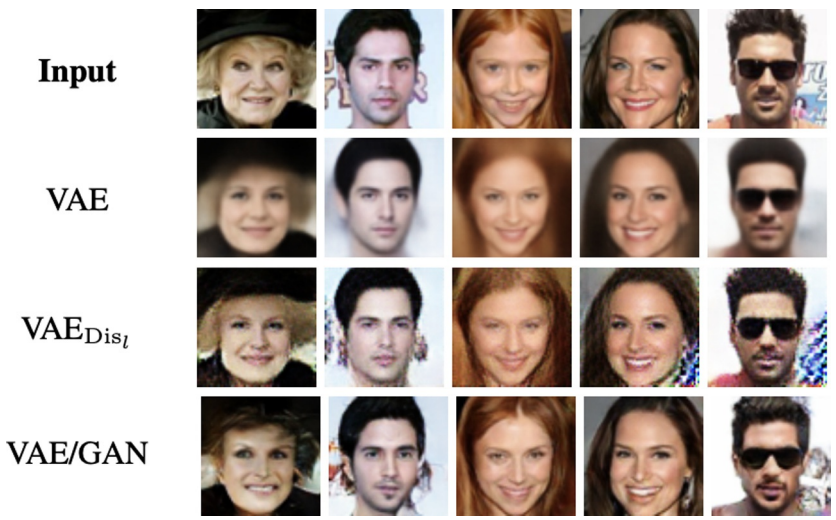

Fig. 10.6 Reconstruction using VAE variations [27].

Table 10.2 Implementation details.

S·no	GAN flavor/link	Paper
1	Pix2Pix, CycleGAN	[19, 20]
2	Pix2Pix	[19]
3	CoGAN	[21]
4	BiGAN	[23]
5	StarGAN	[28]
6	StarGAN V2	[29]
7	SRGAN	[30]
8	Art2Real	[31]
9	Monkey-Net	[32]
10	First Order Model	[33]
11	StackGAN	[34]

strengthening security by creating fake threats and training the model to identify these threats. The availability of data in the healthcare industry is limited. Hence, the GANs can be used for synthetic generation of data. On a similar basis, they can be employed to generate 3D objects and for prediction of future video frames and thereby generating videos. In addition to it, they are making a huge impact in the movie making, gaming, music, and fashion industries. The section introduces a few image translation applications using flavors of GAN. Tables 10.2 and 10.3 provide the implementation and dataset details.

10.3.1 Image-to-image translation using {c, cycle}-GAN

The goal of the image-to-image translation is to align the input image to output image with the help of aligned image pairs. Mirza and Osindero [18] extended the vanilla GAN to a conditional model. The architecture supplied additional information, i.e., class labels to the generator and discriminator by adding an extra input layer. The experiment was conducted on the MNIST dataset. Parzen's window-based log likelihood estimate is calculated and the comparative results are cited in Table 1 of Ref. [18]. On a similar basis, tag vectors were conditioned on the images for the automatic tagging of images. The objective remained the same in Ref. [19] with the discriminator's task unchanged and the generator's task being to fool the discriminator along with nearing the output to that of the ground truth in the L1 sense as L1 supports little haziness. The architecture is named as pix2pix and the experiments have been performed on various datasets like cityscapes, CMP (Centre for Machine Perception) facades, Google maps, edges, sketches, and others. Pix2pix architecture consists of two components, i.e., U-Net generator and PatchGAN discriminator. The U-Net generator is an autoencoder with skip connections. As a consequence of the matching spatial connections between the layers, the skip connections do not require resizing or projections. The aim of the patchGAN

Table 10.3 Dataset description.

S·no	Dataset name/link	Description
1	Open Image V6	9M images with image-level labels, bounding box, and segmentation masks of objects, visual relationships, and localized narratives
2	Cityscapes, Edge2handbags, Edge2shoes, Facades, Maps	Urban street scenes, sketches, buildings, and Google maps
3	CelebA	202,599 number of face images with 40 annotations per image of 10,177 unique identities
4	CelebA-HQ	High-resolution images of CelebA
5	RaFD	67 models with 8 different emotional expressions
6	AFHQ	15,000 high-resolution images of cat, dog, and wildlife
7	Monet2Photo	Monet images
8	Landscape2Photos	Landscape images
9	Portrait2Photo	Portrait images
10	UvA-Nemo	1240 smile videos from 400 subjects (597 spontaneous and 643 posed)
11	BAIR Robo Pushing	59,000 examples of robot pushing motions
12	CUB Bird	6033 images of 200 bird species
13	VoxCeleb	Short clips of video extracted from YouTube videos
14	Oxford-102	102 flower categories with the number of images per class ranging between 40 and 258
15	COCO	COCO is a large-scale object detection, segmentation, and captioning dataset

discriminator is to classify the $N \times N$ patch in an image as real/fake. In addition, it has fewer parameters and is faster when compared to classifying the entire image. The advantage of pix2pix architecture is being generic and learning the objective during training without making any assumptions between two types of images. Hence, it is flexible for various situations. Human scoring and semantic segmentation are the two evaluation strategies [19, 35, 36].

Zhu et al. [20] is a notable extension of GAN architecture with two generator and two discriminator models trained simultaneously. For the simplicity of understanding, d1 and d2 are two domains. One generator takes the input images from d1 and generates images from d2. Similarly, the other generator takes the input images from d2 and outputs images from d1. The discriminators play the same role and generators update accordingly. Cycle consistency is the add-on to this architecture from the machine translation domain. It states that the phrase translated from Kannada to English should translate from English to Kannada with the same efficiency. In case of CycleGAN, the idea is to feed the output of one generator as input to the other generator and the output should be the same as the original image. The reverse is also true. The loss is calculated in two parts, i.e., forward

Table 10.4 Evaluation of different models on Cityscapes dataset.

Rank	Model	Year	Per Pixel Acc (%)	Per Class Acc (%)	Class IOU	Paper
1	Pix2Pix	2016	71	25	0.18	[19]
2	CycleGAN	2017	52	17	0.16	[20]
3	CoGAN	2016	40	10	0.06	[22]
4	SimGAN	2016	20	10	0.04	[21]
5	BiGAN	2016	19	6	0.02	[23]

and backward cycle consistency loss [20, 37]. Impressive applications like collection style transfer, object transfiguration, season transfer, and image generation from paintings are demonstrated as well. A few more noteworthy extensions of GAN are simGAN, coGAN, and BiGAN [21–23]. Table 10.4 summarizes the scores obtained using different flavors of GAN on the cityscapes dataset. Pictorial results are showcased in Fig. 10.7. Pix2Pix (Fig. 10.8A) and CycleGAN (Fig. 10.8B) can be employed for many real applications at hand.

10.3.2 Face generation using StarGAN

Face generation is the task of generating different variations of the face from the existing dataset. The task of generating the face from the given image is related to a particular aspect, i.e., changing the color of the hair, smiling to angry, etc. The important part of face generation is the requirement of high-quality images. Choi et al. [28] use two datasets, i.e., CelebA consists of images with 40 attributes such as hair color, skin tone, etc. and RaFD consists of 8 emotional expressions per face image. The trivial versions are inefficient and less productive for translation among multidomain datasets as it would require $k(k-1)$ generators with k value set to the number of domains. StarGAN is an efficient solution for learning the differences of various domains with the help of a single generator and discriminator. The trivial versions learn fixed translation such as black-to-gray hair while the StarGAN generator is fed with image and domain information, thereby learning to translate the images. The domain is either fed as a binary or one-hot vector and the target domain is randomly generated during the train, giving the control and flexibility of generating the images in any domain during the test phase.

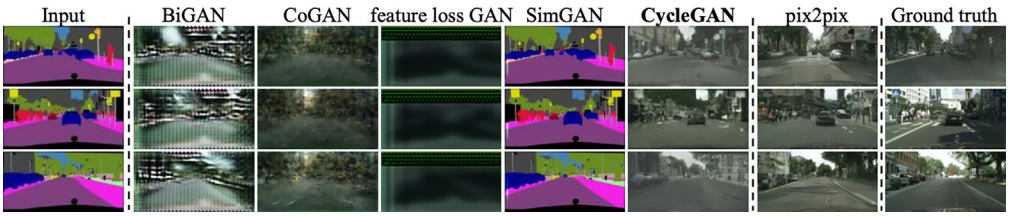

Fig. 10.7 Different flavors of GANs on the Cityscapes dataset [20].

Fig. 10.8 Use of GANs in the fashion industry, paintings, and realistic image generation: (A) Edges to handbags using Pix2Pix [19] and (B) CycleGAN image generation [20].

The approach allows training on multiple datasets by ignoring the unknown labels and concentrating on the necessary labels. Basically, the model learns the features of CelebA along with the RaFD dataset like happy, fearful and incorporates these emotional features on the CelebA dataset (Fig. 10.10A). Two variations have been experimented with and the results depict the success of the model, not only on multiple domains of a single dataset but also on multiple domains across multiple datasets.

The limitation of starGAN is the indication of the domain with a predetermined label as the generator receives a fixed label, thereby producing the same output image for every domain. The same set of researchers [28] came up with the scalable approach to generate images having multiple domains. In other words, the generated image can have more than one domain incorporated. This is achieved by replacing the domain information with the domain-specific style code [29]. To achieve this, two modules are in the architecture. One is the mapping network which transforms the latent code to various style codes and one of them is randomly selected during training. Another module is the style encoder with the functionality of extraction of the style code from the image. StarGAN and StarGAN V2 architecture are presented in Fig. 10.9A and B, respectively. Apart from the scalable approach and good results (Fig. 10.10B), Choi et al. [29] present an AFHQ

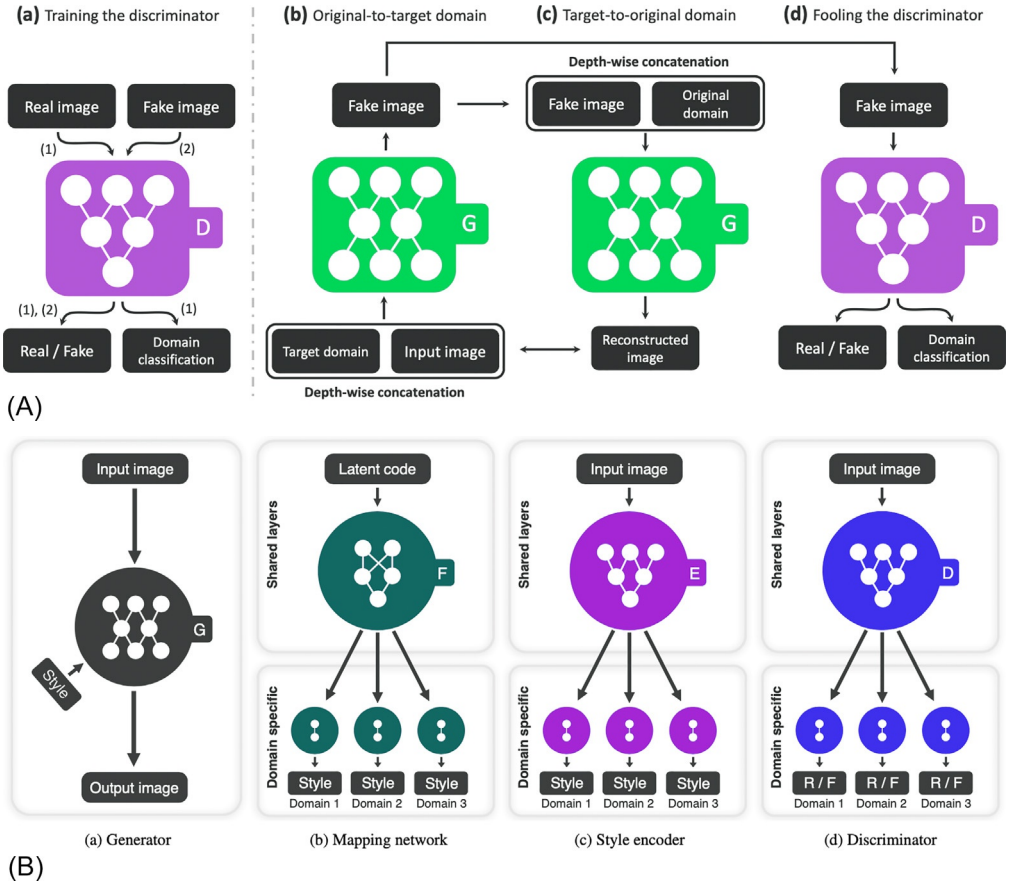

Fig. 10.9 StarGAN architecture: (A) StarGAN [28] and (B) StarGAN V2 [29].

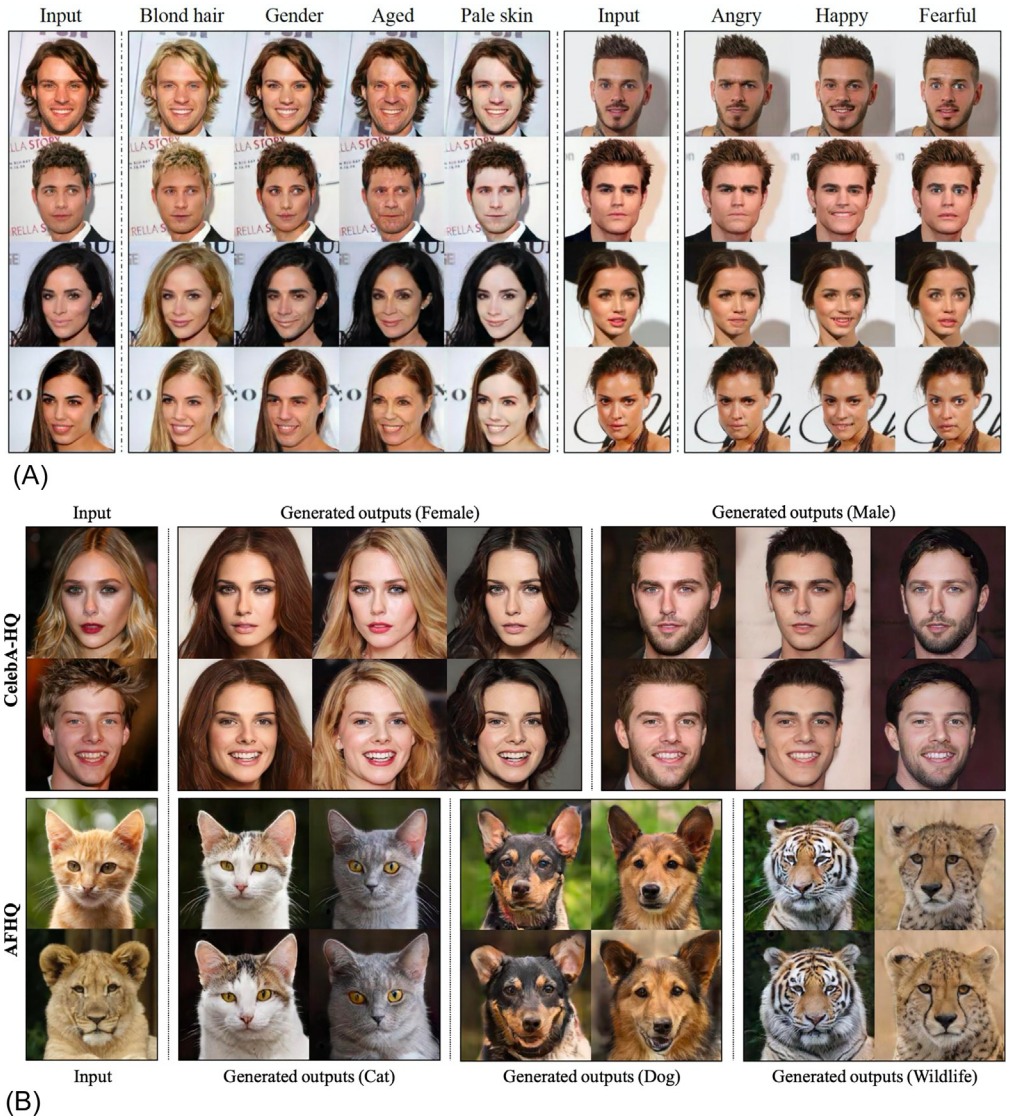

Fig. 10.10 Image-to-image translation: (A) StarGAN [28] and (B) StarGAN V2 [29].

dataset consisting of high-quality animal face images having inter and intra domain differences. The dataset is publicly available.

10.3.3 Photo-realistic images using SRGAN and Art2Real

Single image superresolution (SR) is the task of improving and reckoning a high-resolution (HR) image using a low-resolution (LR) image. Dong et al. [38] is a major

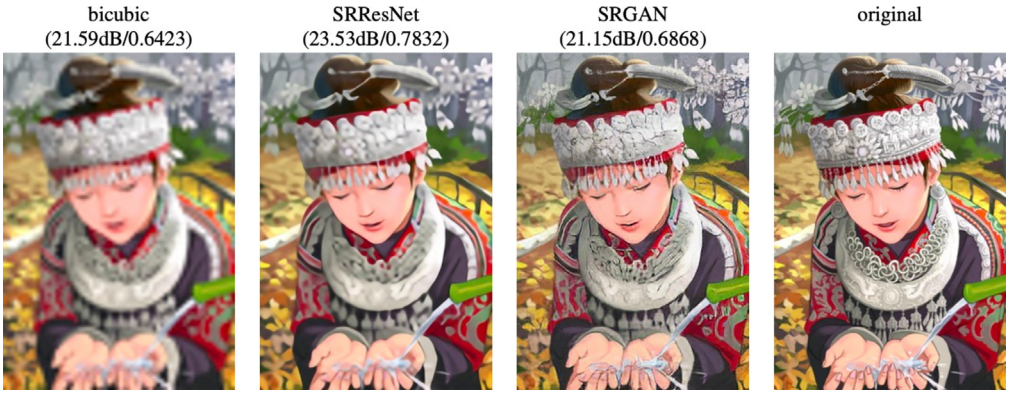

Fig. 10.11 Left to right: Bicubic, SRResNEt, SRGAN, and original image [30].

breakthrough with the proposal of SRCNN. The model structure is simple and achieved SOTA[a] compared to the traditional sparse coding-based SR methods. The major unsolvable problem in SR is recovering the finer texture details when resolved at the upper scale [30]. Ledig et al. [30] proposed SRGAN with the combination of deep learning and adversarial networks. The ultimate goal of the generator is the estimation of the HR image using the LR image, achieved with the help of feed-forward CNN. On the other hand, the discriminator is similar to the VGG network with the exception of dropping off the max-pooling layer throughout the component. The work highlighted the perceptual loss function defined as the weighted sum of content loss and adversarial loss. It accounts for the loss more sensitive to human perception. Content loss is defined as the difference between the features of the generated image and original image such as PSNR (peak signal to noise ratio), MSE (mean square error), and SSIM (structural similarity index). Adversarial loss accounts for the probability that the generated image is real or fake. It is worth mentioning that no objective measures can match the level of human perception (Fig. 10.11). Hence, mean opinion score (MOS) testing was performed where 26 evaluators rated the generated images on a range of 1–5. The MOS test proved that SRGAN results are superior with SOTA[a] performance and measures such as SSIM, PSNR fail to estimate the image quality.

D-SRGAN[b] has also been used in the field of the Digital Elevation Model (DEM). DEM is a 3D representation of the terrain's surface such as the Earth, moon, or asteroid (Fig. 10.12) [39]. The accuracy of the DEM models depends on various factors apart from source, horizontal and vertical precision of the elevation data. DEM data is used for prediction of soil attributes [40], developing the product, decision-making, mapping purpose, 3D simulations, river channel estimation, contour maps creation, and so on [41].

[a] SOTA, State of the art.
[b] D-SRGAN, Dem-Super resolution generative adversarial network.

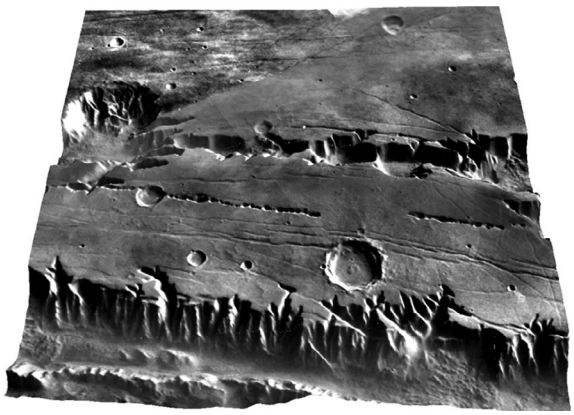

Fig. 10.12 3D rendering of a DEM of Tithonium Chasma on Mars [39].

Lakshmi and Yarrakula [41] present the research in DEM generation and various techniques developed over a decade. A variety of satellite images are available at different resolutions such as spectral, radioactive, and temporal. This reduces the time, cost, and effort required to collect DEM data. SRGAN and D-SRGAN [42] differ in network architecture with D-SRGAN outperforming other methods used in the domain of DEM with the model performing well on a flatter terrain compared to a steeper terrain.

On similar lines, Tomei et al. [31] proposed semantic aware architecture to generate realistic images from the paintings. The architecture is based on memory banks from which realistic details are recovered at patch level (Fig. 10.13). Hence, it includes preparation of memory banks to support generation. One memory bank (B^c) consists of patches of one semantic class such as B^{sky}, B^{rock}. These patches are extracted using a sliding window policy if 20% of pixels belong to the semantic class *c*. Once the preparation of the memory banks is complete, the painting is transformed to a realistic image by pairing generated and real patches. The generated patches are segmentation maps from the original/generated painting and the real patches are memory banks. The dataset consists of paintings from specific artists, artworks from Wikiart, landscape photos from Flickr, and people photos from CelebA. The evaluation metric is Frechet Inception Distance (FID) which measures the difference between two Gaussians and the results outperform in comparison with cycleGAN, Unsupervised Image-to-Image Translation (UNIT), Disentangled Representation for Image-to-Image Translation (DRIT), and style-transferred real methods. A lower FID conveys the realism of the generated images. Fig. 10.14 depicts the qualitative results of the Art2Real framework.

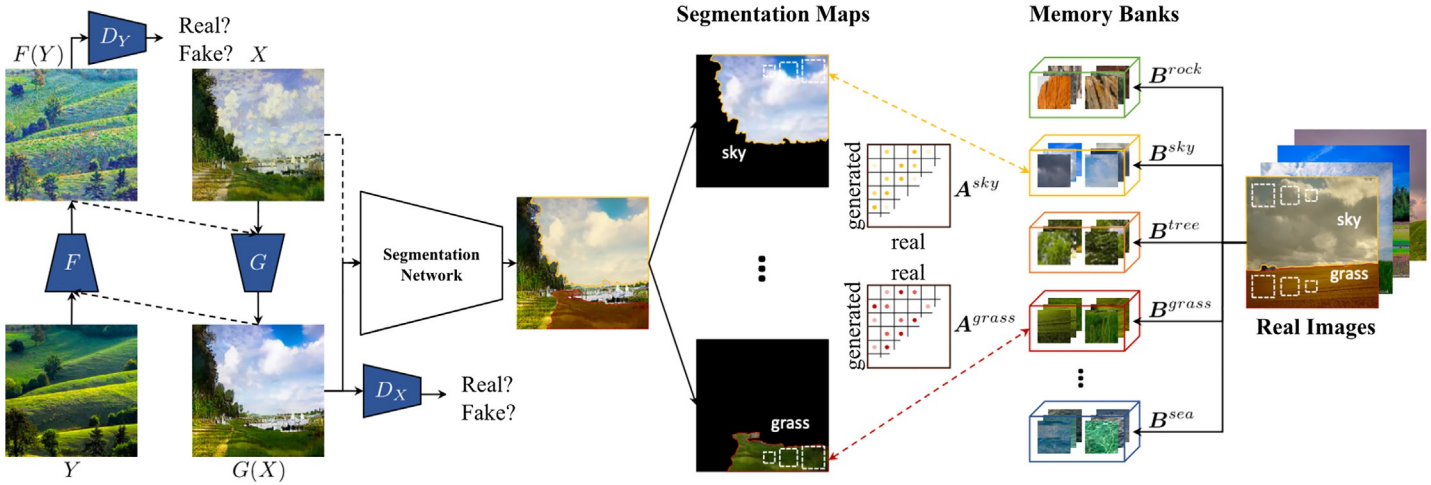

Fig. 10.13 Art2Real architecture [31].

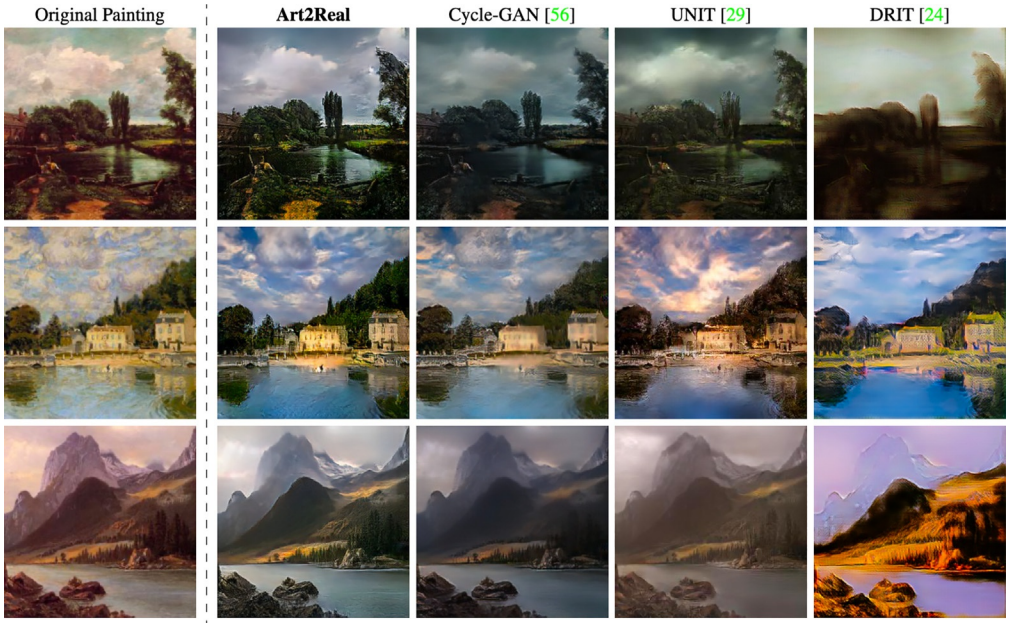

Fig. 10.14 Qualitative results of Art2Real [31].

10.3.4 Image animation and scene generation using monkey net, first-order motion, and StackGAN

Animation in the field of computer graphics and vision is defined as the ability of generating moving images [43]. It is also referred to as computer-generated imagery [44] which usually comprises 3D computer graphics. For 3D animation, objects are rendered on the computer, whereas for 2D figure animation, separate objects are used and the animator moves the specific parts like eyes, mouth based on keyframes. Early works show the use of deep learning and GAN for deep motion transfer [45, 46]. Siarohin et al. [32] generate image animation on the target object given a driving sequence and source image. The requirement of pretrained models for object detection, ground truth data availability, animating any kind of object, and video translations from one domain to another are the few limitations of animations. To address these challenges, it introduced a three-module framework. The first module extracts the object keypoints in an unsupervised fashion. The second module generates motion heatmaps from the keypoints to encode the motion information and the third module incorporates the heatmaps and appearance from the source image to produce a short video. The novelty of the approach is image animation on arbitrary objects and a game plan of transferring the motion information by learning the pixel movement in an unsupervised manner. The architecture follows a self-learning scheme named as Monkey-Net (Fig. 10.15). The UvA-Nemo, Tai-Chai, and BAIR datasets are used for experimental purposes. The evaluation metrics are Average Keypoints Distance (AKD), Missing Keypoints Distance (MKD), AED (Average Euclidean Distance (AED), and FID. The results outperform when compared to the X2face method (Fig. 10.16). The image animation was followed by

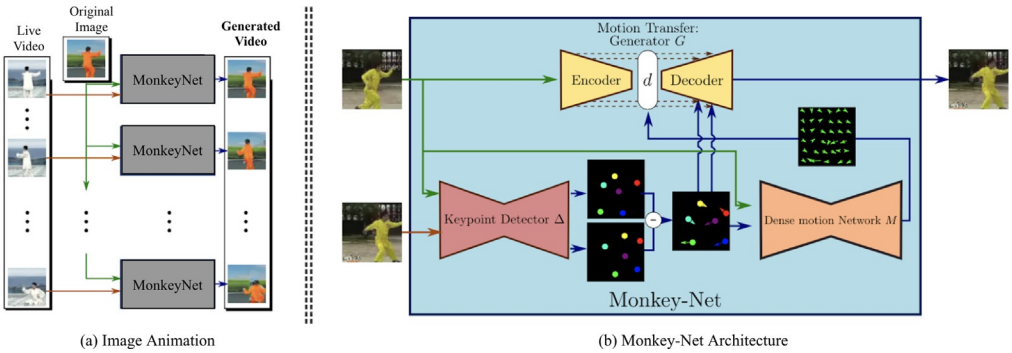

Fig. 10.15 (A) Image animation (B) monkey architecture [32].

image-to-video translation. The qualitative evaluation was performed using amazon mechanical turk (MTurk). MTurk makes tasks like survey, data validation easier by assigning the tasks to distributed forces who perform them virtually. Three videos were shown to users: driving video and two videos generated using X2face and Monkey methods. The Monkey generated videos are preferred more than 80% of times over the X2face method.

The weakness of Monkey-Net is the poor generation quality due to pose changes in case of large objects [33]. To overcome the weakness, the keypoints detector models complex motions with the help of local affine transformations. To improve their estimation, equivariance loss is used during keypoints training. If the driving video contains large motions, then the generator should infer the object parts from the context which are not clear in the input image. Hence, an occlusion-aware generator is set up. Most of the frameworks fail to handle HR data, whereas the experimental results on the HR dataset depict the success of the framework in image animations when compared to other SOTA[a] methods (Fig. 10.16). In addition, the authors have released a new Thau-Chi-HD dataset.

Deep learning in combination with GANs has been used in scene generation as well. Generating quality images from the text is an interesting field of computer vision termed scene generation. Early approaches fail to capture the object and generate low-quality images. Zhang et al. [34] presented a novel approach to generate 256×256 quality images conditioned on text data. The idea is to decompose the problem into subproblems by extracting the primitive shape and colors of the object, thereby producing LR images with the help of given words. In the next stage, HR realistic images are produced using the generated LR images and text descriptions. This is achieved by stacking up two GANs (Fig. 10.17). The first step is converting the text to embedding. The less text description results in the discontinuous latent data, thereby behaving as an obstacle for the generator. Hence, the embedding is condition augmented which is combined with noise to generate an image. The discriminator generates the decision score. The rough LR images along with the text embedding are fed to the second GAN with the generator following the encoder-decoder network with residual blocks. CUB, Oxford-102, and MS COCO datasets have been used for experimental purposes with inception score as evaluation metrics. The results generate realistic images when compared to the existing methods (Figs. 10.18 and 10.19).

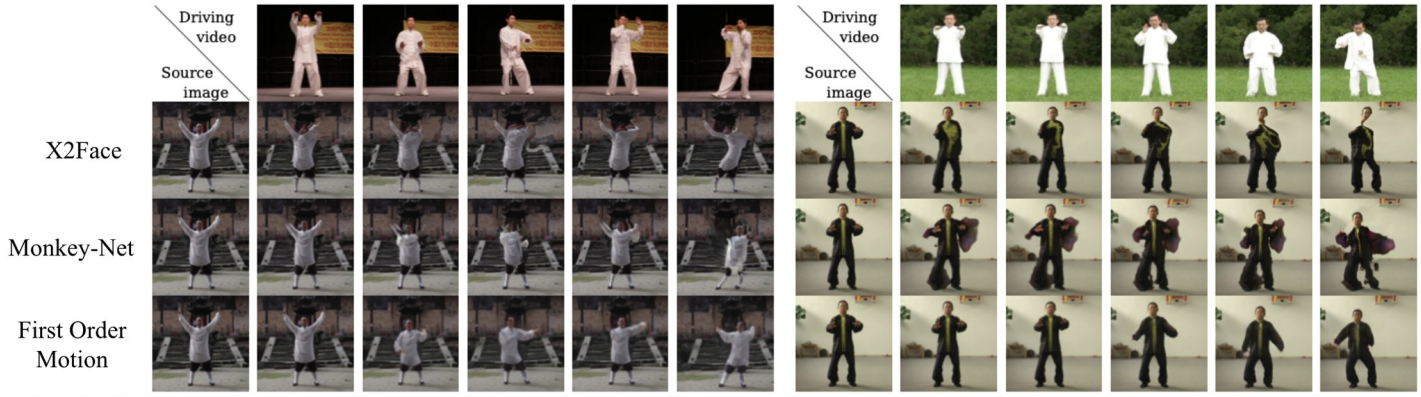

Fig. 10.16 Image animation: reference video (first row), X2face (second row), Monkey-Net (third row), and first-order model (fourth row) [33].

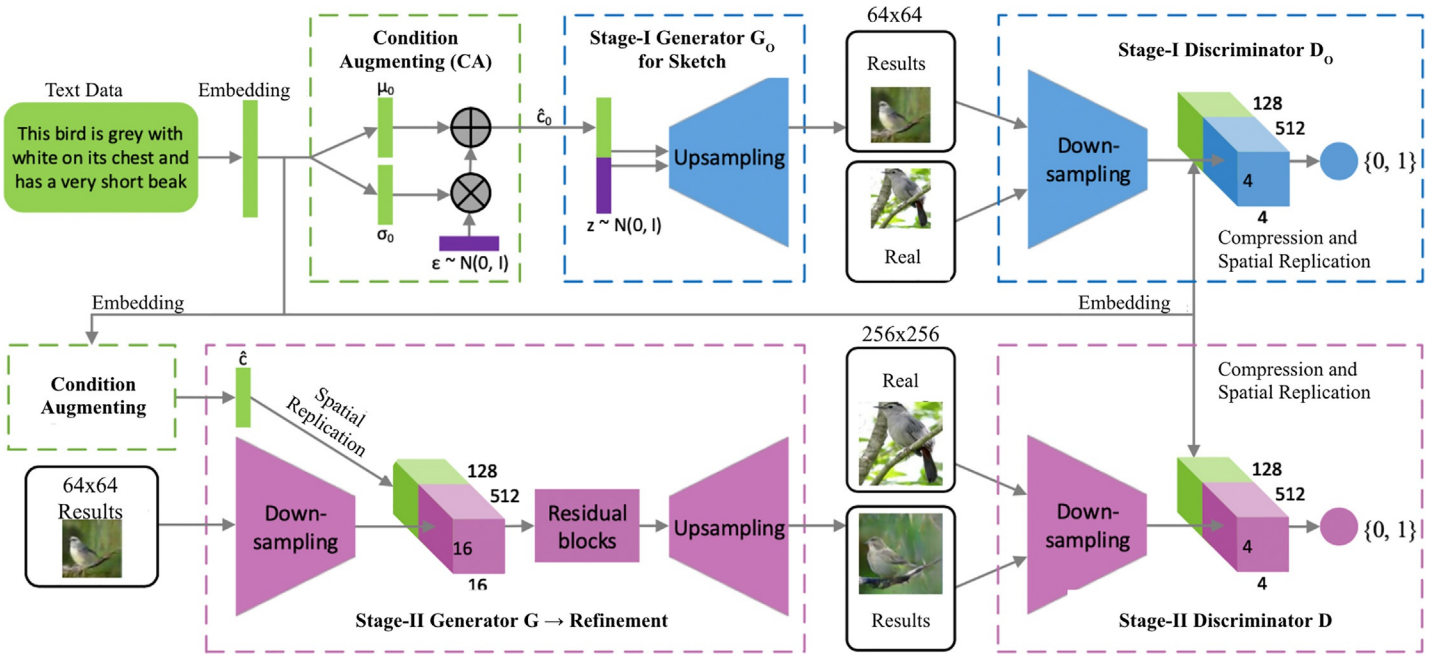

Fig. 10.17 StackGAN architecture [34].

Fig. 10.18 Scene generation using different methods on the CUB dataset [34].

Fig. 10.19 Scene generation on the Oxford-102 and COCO datasets [34].

10.4 Future of GANs

Research in GANs is soaring in image and video generation to a great extent. GANs are a perfect setup to govern the generated samples belonging to the distribution of interest with the help of the adversarial discriminator. The results proved to synthesize facial expressions, swapping the horse with a zebra, fashion industry, paintings to realistic images, animation and scene generation compared to the other SOTA[1] methods. A statement by Facebook AI Research—"GANS are the most interesting idea of the decade"—is somewhat true and experienced to the present day. A good amount of research is currently ongoing in natural language processing and GANs. In addition, the possible future applications are text, audio, and music generation, drug discovery, and medical imaging. With the amount of research ongoing in various fields, GANs remain a promising and prominent solution.

References

[1] A.Y. Ng, M.I. Jordan, On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes, in: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, 2001, pp. 841–848.
[2] M.M. Khapra, CS7015 (Deep Learning): Lecture 22 Autoregressive Models (NADE, MADE), 2020, [Online] Available at: https://www.cse.iitm.ac.in/~miteshk/CS7015/Slides/Handout/Lecture22.pdf. (Accessed 6 July 2020).
[3] A.V.D. Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel Recurrent Neural Networks, 2016. arXiv 2016. arXiv preprint arXiv:1601.06759.
[4] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, Conditional image generation with pixelcnn decoders, in: Advances in Neural Information Processing Systems, 2016, pp. 4790–4798.
[5] T. Salimans, A. Karpathy, X. Chen, D.P. Kingma, Pixelcnn++: Improving the Pixelcnn with Discretized Logistic Mixture Likelihood and Other Modifications, 2017. arXiv preprint arXiv:1701.05517.
[6] A. Dertat, Applied Deep Learning—Part 3: Autoencoders, 2020, [Online] Available at: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798. (Accessed 6 July 2020).
[7] Wikipedia Contributors, Autoencoder—Wikipedia, the Free Encyclopedia, 2020, [Online] Available at: https://en.wikipedia.org/w/index.php?title=Autoencoder&oldid=957588290. (Accessed 6 July 2020).
[8] D.P. Kingma, M. Welling, An Introduction to Variational Autoencoders, 2019. arXiv preprint arXiv:1906.02691.
[9] K. Frans, Variational Autoencoders Explained, 2020, [Online] Available at: http://kvfrans.com/variational-autoencoders-explained. (Accessed 6 July 2020).
[10] J. Walker, C. Doersch, A. Gupta, M. Hebert, An uncertain future: forecasting from static images using variational autoencoders, in: European Conference on Computer Vision, Springer, Cham, 2016, October, pp. 835–851.
[11] G. Brunner, A. Konrad, Y. Wang, R. Wattenhofer, MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer, 2018. arXiv preprint arXiv:1809.07600.
[12] Q. Zhao, E. Adeli, N. Honnorat, T. Leng, K.M. Pohl, Variational autoencoder for regression: Application to brain aging analysis, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, Cham, 2019, October, pp. 823–831.