Surveying Generative models

Farah Aymen
Faculty of Informatics and Computer Science, AI department
The British University in Egypt
Cairo, Egypt
farah194233@bue.edu.eg

Ashraf Adel
Faculty of Informatics and Computer Science, AI department
The British University in Egypt
Cairo, Egypt
ashraf196280@bue.edu.eg

Mohamed Negm
Faculty of Informatics and Computer Science, AI department
The British University in Egypt
Cairo, Egypt
mohamed206069@bue.edu.eg

***Abstract*— In recent years, remarkable progress has been made in generative models, particularly in the fields of computer vision and natural-language processing. The ability of generative models to generate new and diverse samples has resulted in a wide range of applications, such as image and video synthesis, text generation, and music composition. This study investigates generative modeling advances made using Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and diffusion models. Although VAEs and GANs have been widely used for generative modeling tasks in the past, diffusion models have recently emerged as state-of-the-art models. This study provides a detailed analysis of each model, including its strengths and limitations, as well as its applications in image synthesis and video generation. Furthermore, this paper discusses recent developments in diffusion models such as denoising. Finally, this paper emphasizes the importance of future research in generative modeling, as well as the potential for these models.**

Keywords—GANs, VAEs, Diffusion Models

I. INTRODUCTION

Many difficulties have been alleviated as an outcome of generative models. For example, generative models can generate more data that can then be used to train other models or networks. This can be useful for small datasets because gathering more data can be difficult. Texts, images, and audio are generated using generative models. They can also help with image denoising, upscaling, style transfer, and so on. In Section II, the background of various generative models (GANs, VAEs, Diffusion Models) is discussed, along with their architecture, variations, advantages, and drawbacks. In Section III, a comparative analysis is conducted between these models according to many metrics, performance measures, and efficiency, and the advantages and drawbacks of each model are discussed.

II. BACKGROUND

A. Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) first emerged in 2014 when Goodfellow et al. [1] first proposed two networks where one generates data while the other network tries to discriminate whether the output was produced by the first model or a sample from the distribution. First, the generator model learns to generate realistic images by generating images from random noise. The input random noise is sampled using a uniform or normal distribution and then fed into the generator, which generates an image. The generator output, which consists of fake images and real images from the training set, is fed into the discriminator, which learns how to distinguish between fake and real images. The probability that the input is real is the output. The concept behind GANs is to use two primary probability distributions. The probability distribution of the generator, which refers to the distribution from the generator model's output, is one of the main entities. The probability distribution from real images is another important entity. The goal is to ensure that both probability distributions are close, resulting in high-quality output.

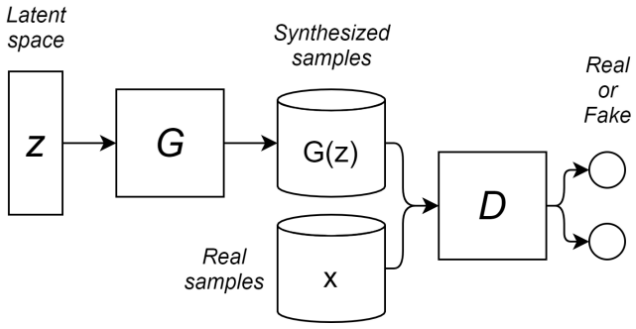


Figure 1. Architecture of GANs

GANs are based on game theory and involve two neural networks competing in a min-max game. The generator attempts to minimize its loss while the discriminator attempts to maximize its loss.

$$L = \min_G \max_D \log(D(x)) + \log(1 - D(G(z)))] \quad (1)$$

One of the advantages of GANs is that it learns the density distributions of data as it learns the internal representations of it. Also, the discriminator is a classifier and can be used for classifying objects. This perk is not found in other generative models. The drawbacks that have arisen with GANs are that they can be unstable, collapse, and produce invariant results. GAN training has been demonstrated to be infamously unpredictable [2]. The proof of convergence of the GANs does not hold in practice because the generator and discriminator are modelled networks; thus, the optimization technique operates in the parameter space of these networks rather than directly learning the probability density function. Furthermore, the game's non-convex-concave nature makes it particularly difficult for the gradient descent ascent (GDA) algorithm to converge, frequently resulting in diverging, oscillating, or cyclic behaviour. A discriminator that is close to optimal would provide meaningful feedback to the generator, resulting in a better generator. In practice, however, the cost of a well-trained discriminator approaches zero. A highly accurate discriminator squashes the loss function to 0 and produces gradients close to zero, providing little feedback to the generator and slowing or stopping learning. As a result, as the discriminator improves at distinguishing between real and generated samples, the updates to the generator become consistently worse. One reason the discriminator easily outperforms the generator is that the supports of real and generated distributions frequently lie on low dimensional manifolds, allowing a perfect discriminator to exist. In such a case, the gradients of a perfect discriminator will approach zero on almost every data point, providing no feedback to the generator. To avoid the vanishing gradients problem,

Goodfellow et al. [1] proposed using a non-saturating (NS) loss for the generator instead of log D(G(z)). Although this modification to the loss alleviates the vanishing gradients problem, it does not completely solve the problem, contributing to more unstable and oscillating training. One of the most common failures in GANs, mode collapse occurs when the generator maps multiple distinct inputs to the same output, resulting in low diversity samples. This is a problem for GAN training because a generator that cannot generate diverse samples is useless for training. Finding and addressing mode collapse is a difficult problem because the cause of mode collapse is deeply rooted in the concept of GANs. In 2020, a research paper proved that GANs suffer from catastrophic forgetting which can be a critical drawback[3]. They ran experiments and identified two main causes for catastrophic forgetting which are: The Information from previous tasks is not carried to/used for the current task. SGD does not use information from previous model distributions, p optimizing, and the discriminator on the current task will degrade its performance on older tasks. Over the years, many types, and variations of GANs have emerged for different applications. In 2014, months after GANs were introduced, Mizra proposed Conditional Generative Adversarial Nets (CGANs) which are considered a conditional version of GANs [4]. The idea behind the CGANs is conditioning the generator and discriminator with extra information by adding an additional layer where this information is fed.

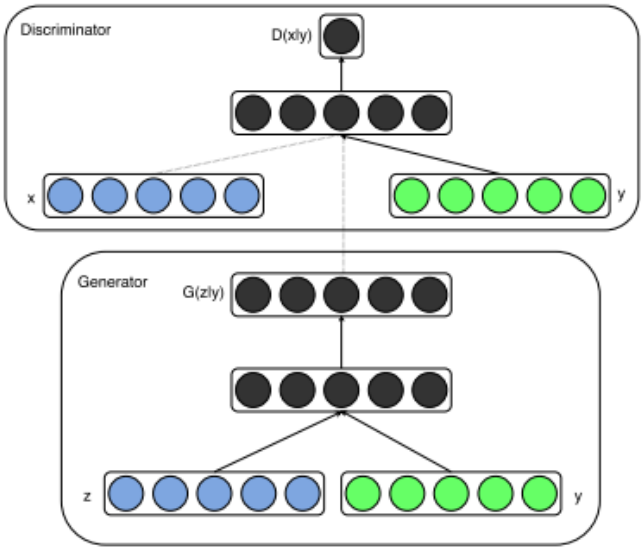


Figure 2. Conditional GANs

The architecture experimented on unimodal data (MNIST) and multimodal data (Labeled Images). The results from the CGANs were extremely primary nevertheless they showed the potential of being used in various applications. Another variation of GANs was introduced in 2016 by Radford and Metz called Deep Convolutional Generative Adversarial Networks (DCGAN) [5]. They proposed that Convolutional Neural Networks (CNNs) were widely used and essential for many applications. Therefore, combining the unsupervised abilities of GANs and the supervised abilities of CNNs can bring potentially great results in generating good general image representations. The model's architecture was composed of a GAN that had a discriminator and a generator but with a batchnorm attached to each. Also, all pooling layers were replaced with strided convolutions (discriminator) and fractional-strided convolutions (generator). Then, the GANs learn filters that supposedly will be used to generate objects. In conclusion, they prove that the filters the GANs learned were used in generating new images. In 2017, Arjovsky et al came up with Wasserstein GANs with the aim to improve traditional GANs. As stated before, GANs suffer from many disadvantages like model collapse. Wasserstein GANs offer higher stability to the training model in comparison to traditional GANs which it hs in reducing the unstable training problem. Therefore, the Wasserstein distance is used unlike the other known traditional distance measurements (e.g., KL-divergence) [6].

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z)] \quad (2)$$

The maximum value in the above equation represents the discriminator's constraint. In general, there is no sigmoid activation function to limit the values to 0 or 1, corresponding to real or fake. WGAN discriminator networks, on the other hand, return a value within a range, allowing it to act less strictly as a critic. The real data is represented by the first half of the equation, while the generator data is represented by the second half. The discriminator seeks to maximize the distance between the real and generated data in order to successfully distinguish the data. Because it wants the generated data to be as real as possible, the generator network aims to minimize the distance between real and generated data. However, one of the major drawbacks of this research paper, which employs a weight clipping method, was discovered to be that it did not always work as well as expected. When the weight clipping was large enough, it resulted in longer training times because the critic took a long time to adjust to the expected weights. When the weight clipping was small, the gradients vanished. Moving on to another variation which is the StyleGANs [7]. Nvidia introduced StyleGANs, which generate artificial images gradually, beginning with a very low resolution and progressing to a high resolution. It controls the visual features expressed in that level by modifying the input of each level separately, from coarse features to fine details, without affecting other levels. The style GAN changes the architecture of the generator significantly due to the incremental growth of the models during training. The StyleGANs generator no longer accepts a latent space point as input; instead, two new randomness sources are used to generate a synthetic image: a standalone mapping network and noise layers. The mapping network's output is a vector that defines the styles that are integrated at each point in the generator model via a new layer called adaptive instance normalization. Control over the style of the generated image is provided by using this style vector. Noise is added at each point in the generator model to introduce stochastic variation. The noise is added to entire feature maps, allowing the model to interpret the style on a per-pixel basis.

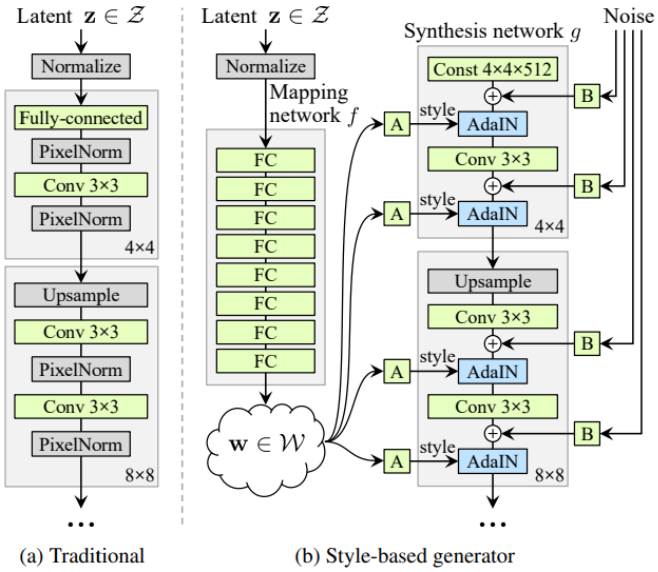


Figure 3. Architecture of StyleGANs compared to traditional GANs

One of the common networks that are used in Image-to-Image Translation is CycleGANs [8]. They are a GAN architecture extension that involves training two generator models and two discriminator models simultaneously at the same time. The first generator receives images from the first domain and outputs images for the second domain, while the second generator receives images from the second domain and generates images for the first domain. The discriminator models are then used to determine how plausible the generated images are, and the generator models are updated accordingly. This extension may be sufficient for generating plausible images in each domain, but it is insufficient for generating translations of the input images. The discriminators are present during the training phase to determine whether images computed by generators appear real or fake. With the feedback of their respective discriminators, generators can improve through this process. In the case of CycleGAN, one generator receives additional feedback from the other. This feedback ensures that an image generated by a generator is cycle consistent, which means that applying both generators to an image sequentially should result in a similar image. Paired training data is made up of training examples that have a correspondence between x and y. CycleGANs, on the other hand, consider unpaired training data, which consists of a source set and a target set with no information about which x matches which y.

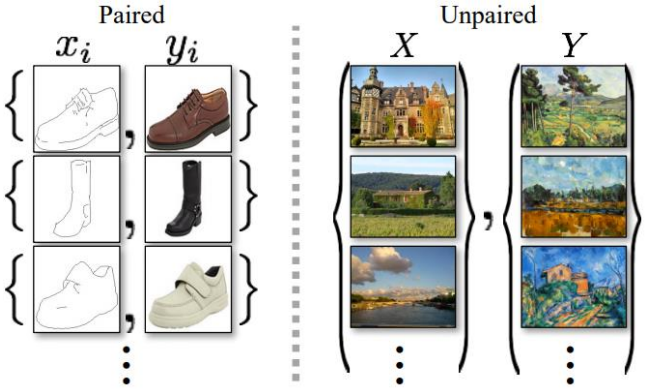


Figure 4. Paired and unpaired Training

B. Variational Autoencoders (VAEs)

An autoencoder is composed of two components which are an encoder and a decoder that use an iterative optimization process to learn the best encoding-decoding scheme. So, at each iteration, the autoencoder architecture is fed some data, and the encoded-decoded output is compared to the initial data, with the error propagated back through the architecture to update the weights of the networks. As a result, the overall autoencoder architecture creates a data bottleneck, ensuring that only the main structured part of the information can pass through and be reconstructed [9].

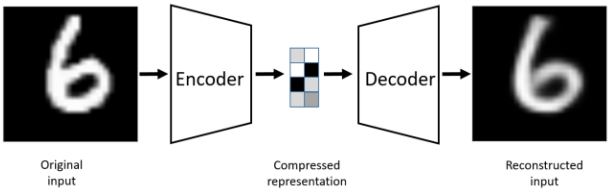


Figure 5. Autoencoders Architecture

The connection between autoencoders and generation may not be clear at this point. Indeed, after training the autoencoder, we have both an encoder and a decoder, but no real way to generate new content. At first glance, we might be tempted to believe that if the latent space is regular enough (well organized by the encoder during the training process), we could pick a point at random from it and decode it to obtain new content. The decoder would then function similarly to the generator in a Generative Adversarial Network. The regularity of the latent space for autoencoders, on the other hand, is a difficult point that depends on the distribution of the data in the initial space, the dimension of the latent space, and the encoder architecture. As a result, ensuring that the encoder will organize the latent space in a smart way compatible with the generative process we just described is difficult (if not impossible) a priori. The autoencoder's high degree of freedom, which allows it to encode and decode with no information loss (despite the latent space's low dimensionality), causes severe overfitting, implying that some points in the latent space will yield meaningless content once decoded. As a result, variational autoencoders appeared. A variational autoencoder is an autoencoder with regularized training to avoid overfitting and ensure that the latent space has good properties that enable a generative process. There has been a significant improvement in the representation capabilities of autoencoders. Variational Autoencoders (VAEs) were used to achieve this[10]. VAEs are generative models that attempt to describe a situation. The generation of data using a probabilistic distribution. Given an observed. We assume a generative model for each data point I i=1 in the dataset X = {x} N of V methods of determining samples. xi conditioned on an unobserved random latent variable z_i, where are the generative distribution parameters. A probabilistic decoder is also equivalent to this generative model. We assume an approximate posterior distribution over the latent variable z_i given a datum xi, which is equivalent to a probabilistic encoder and is governed by the parameters.

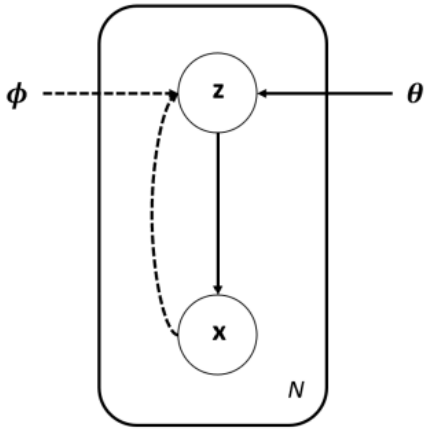


Figure 6. Conceptual representation of VAEs

Finally, for the latent variables zi, we assume a prior distribution denoted by pθ (z_i). The marginal log-likelihood is expressed as a sum of the individual data points log p (x_1, x_2,..., x_N) = N, i=1 log p (x_i), and each point can be rewritten as follows:

$$\log p_\theta(\mathbf{x}_i) = D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}_i)\,||\,p_\theta(\mathbf{z}|\mathbf{x}_i)\right) + \mathcal{L}(\theta,\phi;\mathbf{x}_i) \quad (3)$$

Many types of VAEs have emerged to fill many applications. VQ-VAE is a variational autoencoder that employs vector quantization to generate a discrete latent representation. It differs from VAEs in two important ways: the encoder network produces discrete codes rather than continuous codes, and the prior is learned rather than static. Ideas from vector quantization (VQ) are used to learn a discrete latent representation. Using the VQ method allows the model to avoid posterior collapse, which occurs when latents are paired with a powerful autoregressive decoder and is common in the VAE framework. When these representations are combined with an autoregressive prior, the model can generate high-quality images, videos, and speech, as well as perform high-quality speaker conversion and unsupervised learning [11].

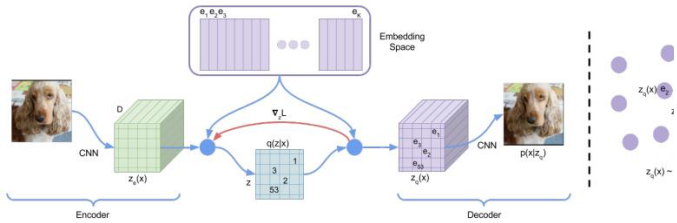


Figure7. VQ-VAE Architecture (Left). Visualization of the embedding space (Right)

The combination of a Generative Adversarial Network (GAN) and a Variational Autoencoder (VAE) has recently been investigated in revolutionary research. In 2016, The paper suggested combining those two architectures (VAE and GAN) into a new generative model called, naturally, VAE-GAN. It proposed combining VAEs and GANs into an unsupervised generative model that concurrently learns to encode, generate, and compare dataset samples, demonstrating that generative models trained with element-wise error metrics, and demonstrating that unsupervised learning produces a latent image representation with disentangled variables of variation [12]. The VAE reconstruction error term (expected log-likelihood) is substituted with a GAN discriminator reconstruction error. Because both VAE's decoder and a GAN's generator work on the latent space z to create the image x, a decoder is used instead of a generator.

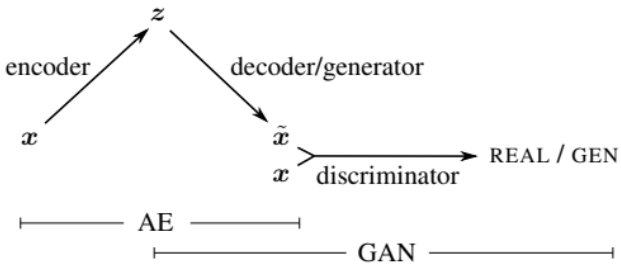


Figure 8. Conceptual Representation of VAE-GAN

The main advantage behind the VAE-GAN is that it helped in solving the invariance problem in the GANs by reconstructing the latent space. Moreover, a paper proposed a hierarchical patch VAR-GAN that generates a diverse set of samples in both the image and video domains [13]. Whereas f-VAEGAN-D2 [14] employs both VAE and GANs in a conditional generative model for any-shot learning. It also demonstrates that the learned features are interpretable. The primary advantage of a variational autoencoder is that it can learn smooth latent state representations of the input data. The generated samples from VAEs are much more blurred than those from GANs due to the injected noise and imperfect reconstruction, and with the standard decoder (with factorized output distribution). Because of its Gaussianity assumption, it produces blurry images and thus suffers from l2 loss. Given the latent variable, the VAE assumes that the output follows a normal distribution, resulting in an l2 loss in the objective function. It has been demonstrated that when data is drawn from multi-modal distributions, the l2 loss results in blurry images. To overcome this limitation, a paper proposed adding a residual block, resulting in Residual-VAE and Multi-Stage VAE [15].

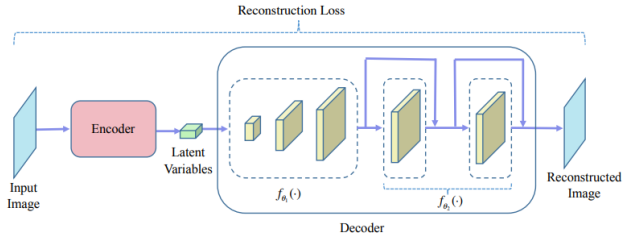


Figure9. Architecture of deep residual VAE

Both networks performed slightly better than the traditional VAE and reduced some of the blurriness.

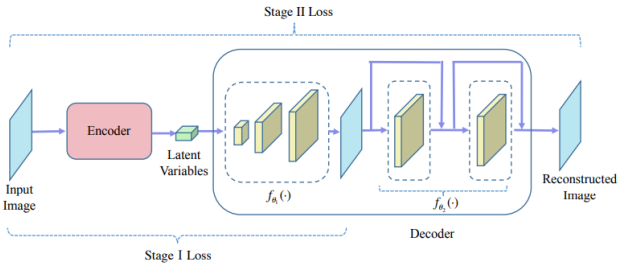


Figure 10. Architecture of multi-stage VAE

C. *Diffusion Models*

Diffusion models have emerged as new state-of-the-art deep generative models. They have broken GANs' long-standing dominance in the difficult task of image synthesis and have shown promise in a variety of domains, including computer vision [16]. Diffusion models are a type of probabilistic generative model that breaks data progressively by injecting noise and then learns to reverse this process for sample generation. Instead of learning the density functions themselves, score-based generative models learn the gradient of the distribution. Stochastic sampling can use such gradient information in reverse to generate diverse samples [17]. The Forward path employs SDE to smoothly transform a complex data distribution to a known prior distribution by gradually injecting noise, progressing from x(0) to x(T), where x(0) corresponds to a data point and x(T) is pure noise corresponding to some prior distribution. The standard Wiener process (also known as Brownian motion) is represented by w, f(t) is a vector-valued function known as the drift coefficient of x(t), and g(t) is a scalar function known as the diffusion coefficient of x. (t). There are several approaches to designing such SDE (as shown in the equation above) to diffuse the data distribution into a fixed prior distribution. The backward path transforms the prior distribution into the data distribution using reverse-time SDE, going from x(T) to x. (0) [18].

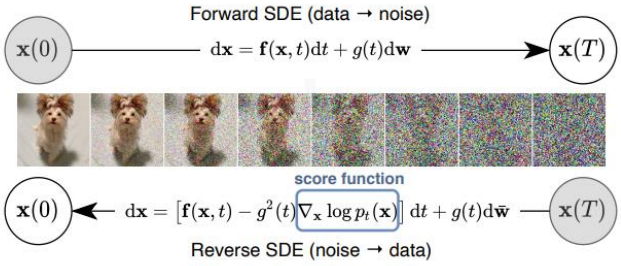


Figure 11. Architecture of Score-based SDE

III. Comparative Analysis

A. *Overall Comparison*

Each generative model had its pros and cons and had its variations that helped in overcoming some of the drawbacks. In a survey paper, it compared various generative models among them the ones discussed in this paper. The metrics it compared with were regarding the time taken to train and test (sample), the FID, the number of parameters, etc. [19].

Table 1 Comparing between various generative models regarding training and test speed, parameter efficiency, and sample quality.

Model	Training speed	Sample speed	# of Params	FID
GANs				
DCGAN[5]	<1/2 day	1 step	<10M	37.11
SyleGAN [7]	>5 days	1 step	<60M	2.42
VAEs				
Convolutional VAE	<1/2 days	1 step	<10M	106.37
VQ-VAE [11]	<5 days	Middle	>120M	-
Denoising Diffusion	<1/2 day	MCMC	<60M	3.17

B. *Diffusion Models Beat GANs*

As seen in Table 1, Diffusion models and GANs are rather close when comparing them. Looking at various applications where the two are tested against each other, Muller-Franzes et al were attempting to generate 2D medical images using the two models [20]. They used the AIROGS challenge train dataset, which contained 101,442 256x256 RGB eye fundus images from approximately 60,357 subjects, 98,172 of whom had "no referable glaucoma" and 3,270 of whom had "referable glaucoma." The second dataset is the CRCDX, which contains 19,958 colour-normalized 512x512 RGB histology colorectal cancer images with a resolution of 0.5 m/px. The third dataset is the CheXpert train dataset, which included 223,414 gray-scaled chest radiographs from 64,540 patients, contained half of the images that were microsatellite stable and half that were microsatellite unstable. After excluding images taken in a lateral position, 191,027 images from 64,534 patients remained. Following the pre-processing, all images were scaled to 256x256 and normalized between -1 and 1. First, consider the classical generative adversarial networks (GANs) proposed by Goodfellow et al [1]. It was intended to use GANs with cutting-edge quality on the respective datasets to allow a fair comparison with the diffusion model. The network used for the CheXpert dataset was a pre-trained progressive growing GAN (proGAN). This GAN's chest x-rays were barely distinguishable by three inexperienced and three experienced radiologists. Furthermore, when compared to traditional augmentation, this GAN architecture has already been used for data augmentation, resulting in higher downstream performance. We used a pre-trained, conditional GAN (cGAN) for the CRCDX dataset. For the diffusion model, they proposed the Medfusion model, which is based on the Stable Diffusion model. It was made up of two parts: an autoencoder that compressed the image space and a DDPM. In two phases, both parts were trained. The Autoencoder was trained to encode the image space into an 8-times compressed latent space of size 32x32 and 64x64 for the 256x256 and 512x512 input spaces, respectively, during the first training phase. During the training phase, the latent space was decoded directly back into image space and supervised by a multi-resolution loss function, as described in the supplemental material.

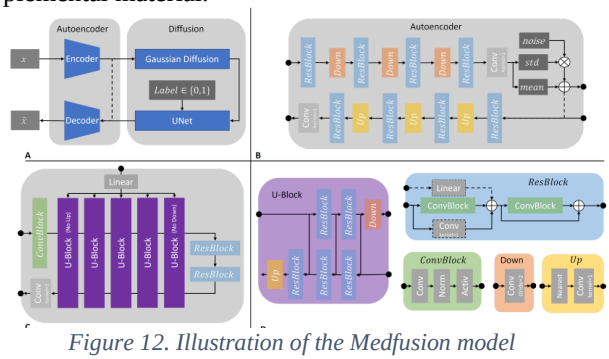


Figure 12. Illustration of the Medfusion model

The final results showed that the Medfusion model has achieved the lowest FID which indicates the high quality of generated images.

Table 2 Comparing results between the GANs models and the Diffusion model on different datasets.

Dataset	Model	FID	Precision	Recall
AIROGS	StyleGan-3	20.43	0.43	0.19
AIROGS	Medfusion	11.63	0.70	0.40
CRCDX	cGAN	49.26	0.64	0.02
CRCDX	Medfusion	30.03	0.66	0.41
CheXpert	ProGAN	84.31	0.30	0.17
CheXpert	Medfusion	17.28	0.68	0.32

Another research was done where they used a diffusion model to generate a talking face based on audio input [21]. To learn the distribution of frames extracted from videos, a diffusion model is trained. Then a video is selected from the training set at random and then a frame from that video. To include temporal information, the corresponding audio sequence is divided into equal-length chunks based on the number of frames in the video. The audio chunks are then encoded into audio embeddings using an audio encoder that has been pre-trained on the LRW dataset.

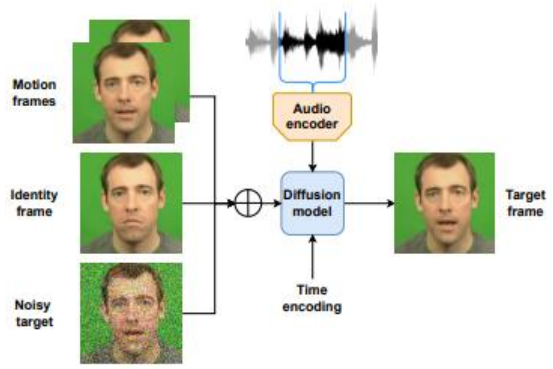


Figure 13. Training a diffusion model to output a generated talking video.

Then the results of this model as compared with other models specifically GANs to deduce that t has achieved better results and produced the most realistic videos out of the other compared models.

C. VAEs vs GANs

Comparing the VAEs with GANs, El-Kaddoury et al conducted a study proving that although VAEs are simpler and quicker to train, their output is generally blurrier than that of GANs [22]. The methods outlined in the paper were applied to three different data sets: MNIST, CIFAR10, and CelebA. We resized the images to a maximum size of 72 72 pixels to reduce the necessary computational work.

Table 3 Results of GANs and VAEs on different datasets.

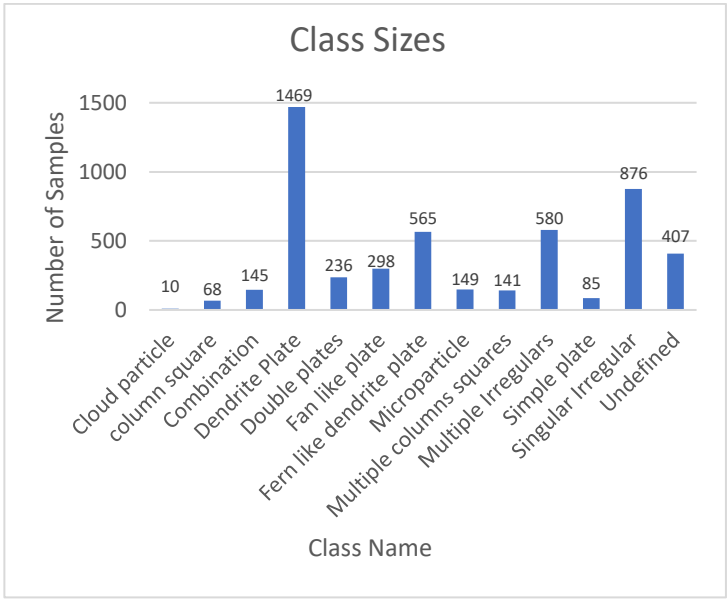
Dataset	Method	Inception Score
MNIST	**VAEs**	**3.32**
	GANs	9.05
CelebA	**VAEs**	**2.78**
	GANs	7.02
CIFAR-10	**VAEs**	**3.0**
	GANs	6.8

As seen in Table 3, VAEs performed better than the GANs on the three datasets. The training time for the two approaches was found to be a significant difference in this study. GANs required longer to train (in terms of both the number of epochs and the running time). Another benefit of VAEs is their consistency and stability.

IV. Methodology

A. Dataset Description

The dataset that has been used is an original made dataset of images of water crystals. It is composed of 13 different classes.



The class that was focused on in this study was the Microparticle class that composed of 149 Images as stated in Figure 14.

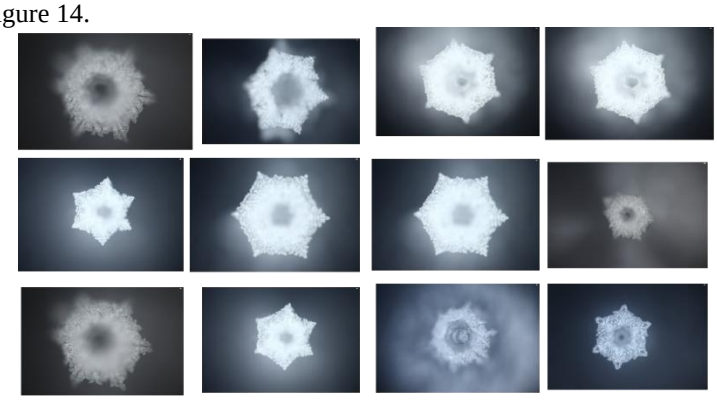


Figure 14. Samples from Microparticle Class

B. Preprocessing Phase

First, due to the different sizes of samples, all images were resized to resolution of 128x128. In one of the variations of the VAE-GANs, a resolution of 64x64 was tested. Second, on the rather small classes data augmentation was applied to increase the size of the data as the generative models were only fed one class out of all 13 classes which was Microparticle.

C. Metrics Used

Loss was used for all variations of both VAE-GANs and diffusion models (DMs), as FID was not suitable as a metric, due to using Google's Inception model [23] to extract

features based on images the model was trained on, which was dissimilar than the water crystal dataset.

D. Model Results on Different Variations

The implementation of the following models is found in the referenced GitHub repository of this project, where "project_media_2" directory includes some of the media present in this report.

I) VAE-GANs

Several variants were tested of the VAE-GANs inspired by [12]. The fixed hyper parameters were using a learning rate of $3e^{-4}$ and using Adam optimizer. The first variant was tested on 10 images from the data of class cloud particle with resizing of 64x64 and number of epochs of count 100.

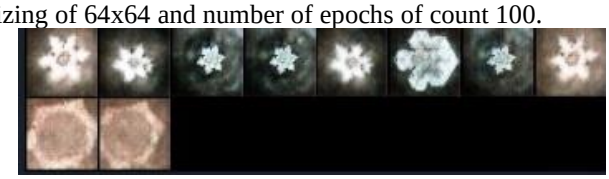


Figure 15. 564x64 images generated on dataset of 10

The results in figure 15 seemed a bit blurry and invariant from the training set that is seen in figure 16. Moreover, the size of the image is too small and pixelated which may contribute to the quality of the results.

Figure 16. 64X64 original dataset of 10

In the next variant the size of the images was increased to 128x128 pixels and was tried on the whole dataset of 149 images with 20 images per batch to generate 20 new images.

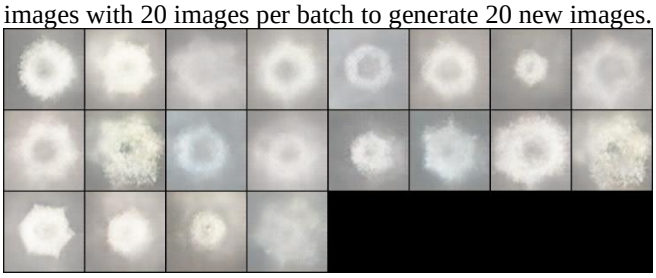


Figure 17. 128X128 images generated on the full dataset of sample size 149 images.

As seen in Figure 17, the new results are still blurry but have better quality and representation of the images. The metrics used for the GANS were the loss metric where Binary cross entropy loss metric was used to calculate the GAN loss, the reconstruction loss and the prior loss. Along the 100 epochs the GAN loss has reached 1.273. while the prior loss reached 0.19 and the reconstruction loss reached 0.2619.

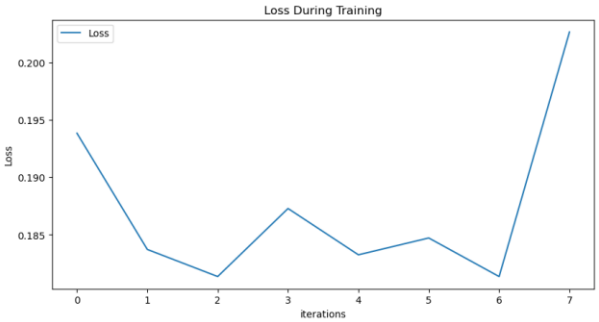


Figure 18. Prior loss

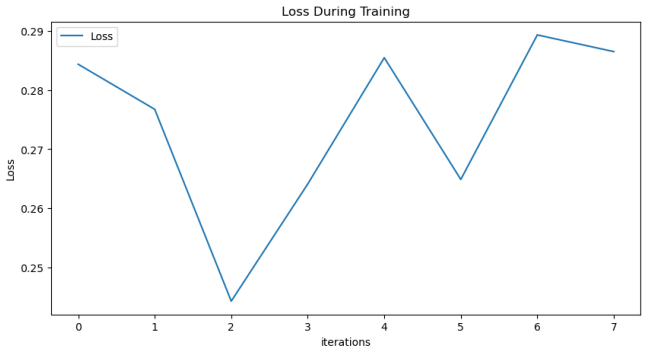


Figure 19. Reconstruction loss

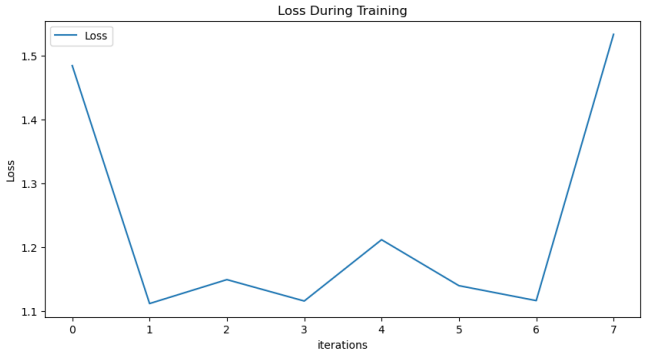


Figure 10. GAN loss

II) Diffusion Models

The following three variants of diffusion models were tested on different hyperparameters. However, it is important to note that the learning rate was fixed to 0.0001. Additionally, the Adam optimizer was used in all model variants. Furthermore, all logging of the results' metrics was done using Tensor board. The first version includes an extra pre-processing step of augmenting the images by rotating, vertical and horizontal flipping, and center-zooming by small values the loaded images in Microparticle folder. The train batch size was 4 images, which we were not able to increase greatly in subsequent model variations due to computation limitations (CLs). The number of epochs was 100, and no gradient accumulation was applied. Moreover, the result of the first DM variant is shown in Figure 21.

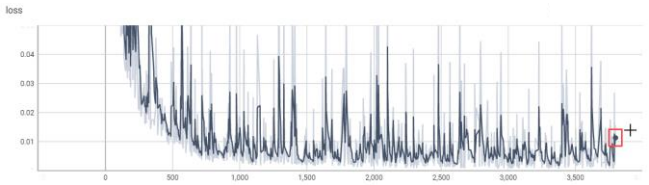


Figure 21. Loss Graph for the first variant of Diffusion model.

Table 4 Loss value of last step

Step	3799
Value	0.01565

Noting that this is the only log graph where the "step" on the x-axis represents the loss in each batch present in each epoch (so since the number of epochs is 100, the batch size is 4, and there are 149 images in Microparticle class, therefore there are 149 / 4 which equals an upper bound of 38 batches, so 100 * 38 total batches that the loss were calculated for, that is why the last step in that graph is step 3,799). However, in the next two loss graphs, the x-axis steps represent the loss of the final batch only in each epoch (for easier visual interpretation). The output results can be seen in Figure 22.

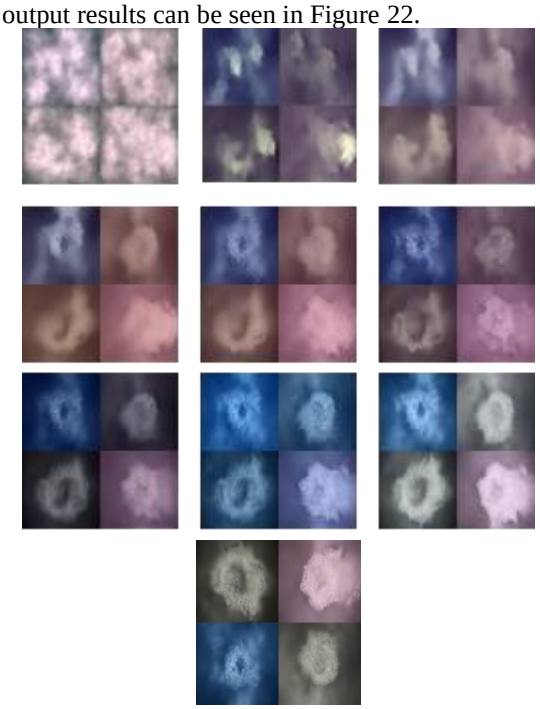


Figure 22. Output of first diffusion model every 10 epochs on 100 epochs total.

Regarding the second DM variant, no augmentation was done, and gradient accumulation was set to 4, which means that the optimization step is done after 16 (4 batches * 4 gradient accumulation steps) images are processed. The results were the following:

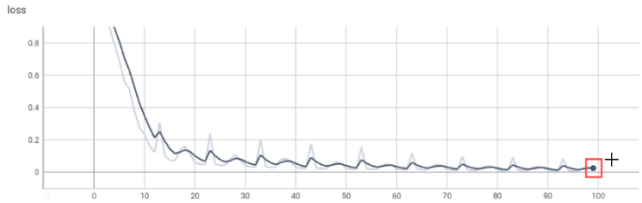


Figure 23. Loss Graph for the second variant of Diffusion model.

Table 5 Loss value of last step

Step	99
Value	0.02525

The variant was run on 100 epochs giving the 4 generated images per epoch. It was run on 100 epochs total.

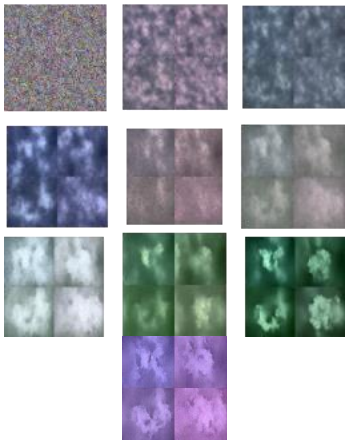


Figure 24. Output of second diffusion model every 10 epochs on 100 epochs total.

Regarding the third DM variant, no augmentation was done, nor gradient accumulation, and the epochs were supposed to be 500, but due to computation limitations, the computing device crashed mid-training before even finishing 80 epochs. In addition, an attempt was made to create a model architecture similar to the stable diffusion model, but unfortunately the computation limitation prevented doing more than 60 epochs.

Figure 25. Unsuccessful attempt of a diffusion model on 60 epochs.

Finally, the first variant was run for 500 epochs giving the best results yet that are shown in Figure 26.

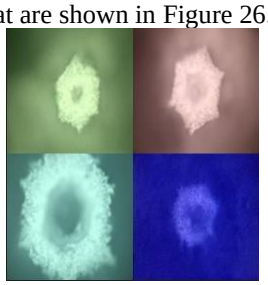


Figure 26. Results of first variant of Diffusion model at 500 epochs

V. Future Work

In the next phase, it is planned to obtain high computational resources to train the diffusion model and the VAE-GANs to obtain better results. Also, the focus will be more on the diffusion models since the results shown by the VAE-GANS that it did not produce much variance far from the original dataset.

VI. Conclusion

In conclusion, Diffusion models have proved that they are state-of-the-art and overcome many of the

disadvantages of the other generative models. Moreover, it has proved to reach high scores using various metrics (e.g., Fréchet Inception Distance, etc.). They have also proved to be better than GANs models although GANs can reach high accuracies and good generation results. However, GANs' drawbacks make them unrecommended to be used. Overall, the two models that will be implemented and compared between are VAE-GANs and Diffusion models because Diffusion models are considered SOTA and VAE-GANs supposedly solve the invariance problem in GANs, and combining VAEs with the high accuracy of GANs may produce high results. Regarding this study's results, it is seen that the VAE-GANs did not show much variance than the dataset as the literature review has proved yet it outputted more structured images than the diffusion models. This can root back to the computational limitations. For the diffusion model, it has provided novel images that are different that the dataset it has been trained on. It may need more epochs and bigger batches to train on to obtain better results.

REFERENCES

[1] I. Goodfellow *et al.*, "Generative Adversarial Networks," *Commun ACM*, vol. 63, no. 11, pp. 139–144, Jun. 2014, doi: 10.1145/3422622.

[2] M. Wiatrak, S. V. Albrecht, and A. Nystrom, "Stabilizing Generative Adversarial Networks: A Survey," Sep. 2019, Accessed: Mar. 24, 2023. [Online]. Available: https://arxiv.org/abs/1910.00927v2

[3] H. Thanh-Tung and T. Tran, "On Catastrophic Forgetting and Mode Collapse in Generative Adversarial Networks," Jul. 2018, [Online]. Available: http://arxiv.org/abs/1807.04015

[4] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," Nov. 2014, Accessed: Mar. 24, 2023. [Online]. Available: https://arxiv.org/abs/1411.1784v1

[5] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, Nov. 2015, Accessed: Mar. 24, 2023. [Online]. Available: https://arxiv.org/abs/1511.06434v2

[6] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," Jan. 2017, Accessed: Mar. 24, 2023. [Online]. Available: https://arxiv.org/abs/1701.07875v3

[7] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," *IEEE Trans Pattern Anal Mach Intell*, vol. 43, no. 12, pp. 4217–4228, Dec. 2018, doi: 10.1109/TPAMI.2020.2970919.

[8] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, pp. 2242–2251, Mar. 2017, doi: 10.1109/ICCV.2017.244.

[9] W. H. Lopez Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, "Autoencoders," *Machine Learning: Methods and Applications to Brain Disorders*, pp. 193–208, Mar. 2020, doi: 10.1016/B978-0-12-815739-8.00011-0.

[10] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends in Machine Learning*, vol. 12, no. 4, pp. 307–392, Jun. 2019, doi: 10.1561/2200000056.

[11] A. Van Den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural Discrete Representation Learning," *Adv Neural Inf Process Syst*, vol. 2017-December, pp. 6307–6316, Nov. 2017, Accessed: Mar. 24, 2023. [Online]. Available: https://arxiv.org/abs/1711.00937v2

[12] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *33rd International Conference on Machine Learning, ICML 2016*, vol. 4, pp. 2341–2349, Dec. 2015, Accessed: Mar. 24, 2023. [Online]. Available: https://arxiv.org/abs/1512.09300v2

[13] S. Gur, S. Benaim, and L. Wolf, "Hierarchical Patch VAE-GAN: Generating Diverse Videos from a Single Sample," *Adv Neural Inf Process Syst*, vol. 2020-December, Jun. 2020, Accessed: Mar. 24, 2023. [Online]. Available: https://arxiv.org/abs/2006.12226v3

[14] Y. Xian, S. Sharma, B. Schiele, and Z. Akata, "f-VAEGAN-D2: A Feature Generating Framework for Any-Shot Learning," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 10267–10276, Mar. 2019, doi: 10.1109/CVPR.2019.01052.

[15] L. Cai, H. Gao, and S. Ji, "Multi-Stage Variational Auto-Encoders for Coarse-to-Fine Image Generation," *SIAM International Conference on Data Mining, SDM 2019*, pp. 630–638, May 2017, doi: 10.1137/1.9781611975673.71.

[16] L. Yang *et al.*, "Diffusion Models: A Comprehensive Survey of Methods and Applications," Sep. 2022, Accessed: Mar. 24, 2023. [Online]. Available: https://arxiv.org/abs/2209.00796v10

[17] J. H. Lim *et al.*, "Score-based Diffusion Models in Function Space," Feb. 2023, Accessed: Mar. 24, 2023. [Online]. Available: https://arxiv.org/abs/2302.07400v1

[18] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-Based Generative Modeling through Stochastic Differential Equations," Nov. 2020, Accessed: Mar. 24, 2023. [Online]. Available: https://arxiv.org/abs/2011.13456v2

[19] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models," Mar. 2021, doi: 10.1109/TPAMI.2021.3116668.

[20] G. Müller-Franzes *et al.*, "Diffusion Probabilistic Models beat GANs on Medical Images," Dec. 2022, [Online]. Available: http://arxiv.org/abs/2212.07501

[21] M. Stypułkowski, K. Vougioukas, S. He, M. Zięba, S. Petridis, and M. Pantic, “Diffused Heads: Diffusion Models Beat GANs on Talking-Face Generation,” Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.03396

[22] M. El-Kaddoury, A. Mahmoudi, and M. Majid Himmi, “Deep Generative Models for Image Generation: A Practical Comparison Between Variational Autoencoders and Generative Adversarial Networks,” in *Mobile, Secure, and Programmable Networking*, É. Renault, S. Boumerdassi, C. Leghris, and S. Bouzefrane, Eds., Mohammedia, Morocco: MSPN, Apr. 2019, pp. 13–20. [Online]. Available: http://www.springer.com/series/7411

[23] C. Szegedy *et al.*, “Going Deeper with Convolutions,” Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.4842