Variational Autoencoders and Generative Adversarial Networks
Generative Deep Learning Algorithms In Comparison

Altar Tosun

Universität Kassel, Germany
altartosun06@gmail.com

Abstract. Generative models are very popular in the field of unsupervised learning. They are very successful in learning underlying data distributions of training data and generate new data. This paper presents a detailed study of generative models and how they differ from traditional discriminative models. This survey puts its focus on the two most popular generative models such as Variational Autoencoder (VAE) and Generative Adversarial Network (GAN). Both models are generative models but their application areas are different. The reason for this, that they have mutually exclusive advantages.

Outline

In the first section, we will generally review Generative Adversarial Networks, and in the second section we will continue with Variational Autoencoders. In the third section we will give a comparison between VAEs and GANs, their weaknesses and strengths, which situation is better for which one and their applications. As a small section VAE-GAN, a combination of GANs and VAEs. The paper closes with the fourth section and there will be the summary.

1 Generative Adversarial Networks

An algorithm that converts a scribble into a real picture [9], a known table into a real scene, a description you make into a detailed visual [29], a molecule you are investigating into a functional chemical[14]. Yann LeCun described it as “the most interesting idea in the last 10 years in Machine Learning”.

Generative Adversarial Networks, or GANs for short,were introduced in 2014 by J. Goodfellow and co-authors [7]. GANs are an approach to generative modeling using deep learning methods, such as convolutional neural networks.

Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset. GANs are an intelligent way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated). These two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

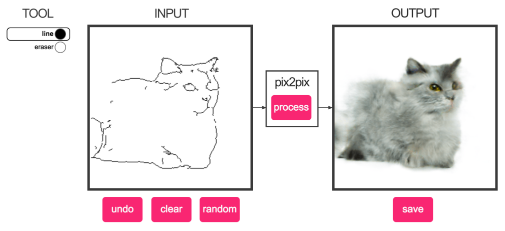

Fig. 1: Generate cat from sketch with Pix2Pix [9]

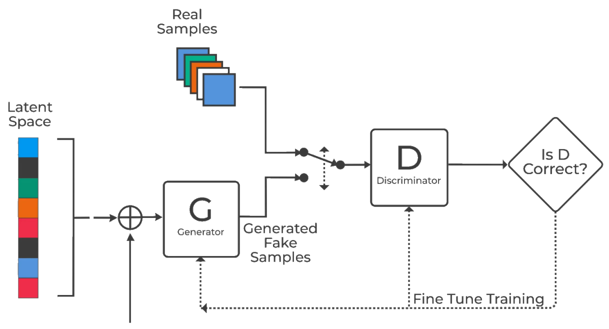

Fig. 2: GANs schema [5]

1.1 The Generator Model

The generator model takes a fixed-length random vector as input and generates a sample in the domain. The vector is drawn randomly from a Gaussian distribution, and the vector is used to seed the generative process. After training, points in this multidimensional vector space will correspond to points in the problem domain, forming a compressed representation of the data distribution. This vector space is referred to as a latent space, or a vector space comprised of latent variables. Latent variables, or hidden variables, are those variables that are important for a domain but are not directly observable. Usually refered latent variables or a latent field as a reflection or compression of a data distribution. That is, a latent space provides compression or higher-level concepts of observed raw data such as input data distribution. In GANs, the generator model applies meaning to points in a selected hidden area; such that new points drawn from the hidden area can be input to the generator model as input and used to produce new and different output samples.

1.2 The Discriminator Model

The discriminator model takes an example from the domain as input (real or generated) and predicts a binary class label of real or generated fake. The real example comes from the training dataset. The generated examples are output by the generator model. The discriminator is a classification model. After the training process, the discriminator model is discarded as we are interested in the generator. Sometimes, the generator can be repurposed as it has learned to effectively extract features from examples in the problem domain. Some or all of the feature extraction layers can be used in transfer learning applications using the same or similar input data.

1.3 Two-players Game

The two models, the generator and discriminator, are trained mutually. The generator creates a batch of samples, and these, along with real examples from the domain, are provided to the discriminator and classified as real or fake.

The discriminator is then updated to get better in discriminating real and fake samples in the next round. The generator is updated, based on the generated samples fooled the discriminator. In this way, the two models are competing against each other. They are adversarial in the game theory sense and are playing a zero-sum game [17]. In this case, zero-sum means that when the discriminator successfully identifies real and fake samples it is rewarded or no change is needed to the model parameters. It causes the generator is penalized with large updates to model parameters.

Alternately, when the generator fools the discriminator, it is rewarded, or no change is needed to the model parameters (weights), but the discriminator is penalized and its model parameters are updated.

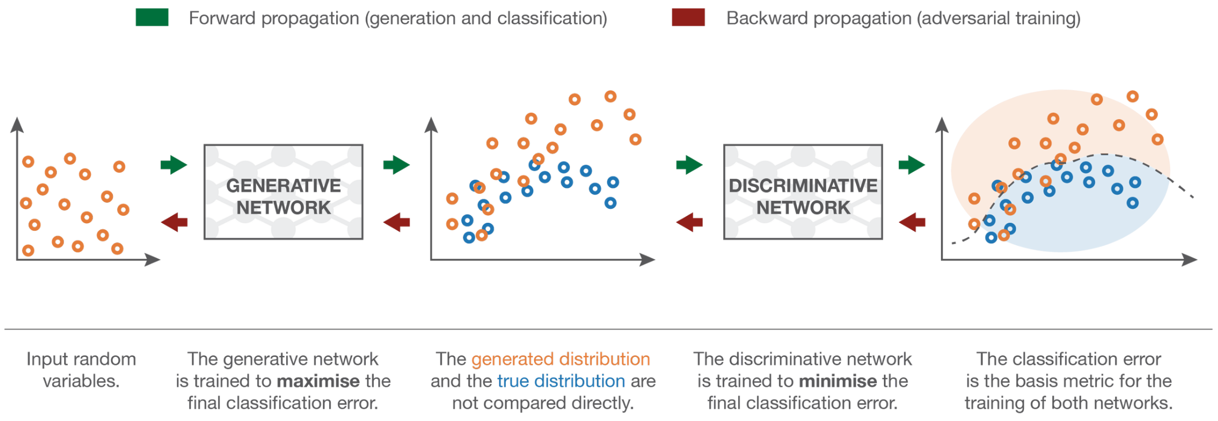
Forward propagation (generation and classification)
Backward propagation (adversarial training)
GENERATIVE NETWORK
DISCRIMINATIVE NETWORK
Input random variables.
The generative network is trained to maximise the final classification error.
The generated distribution and the true distribution are not compared directly.
The discriminative network is trained to minimise the final classification error.
The classification error is the basis metric for the training of both networks.

Fig. 3: Using backpropagation algorithm in order to train the Generative Adversarial Networks [21]

2 Variational Autoencoders

Beside regressors and classifiers, Variational Autoencoders (VAEs) are powerful generative models [12]. VAEs work with remarkably diverse types of data, sequential or non-sequential, continuous or discrete, even labeled or completely unlabeled, making them highly powerful generative tools. Nowadays they are having applications like, generating fake human faces [28], synthetic music [20] and drawing images.

It is possible to generate data in a desired and specific direction. The reason for this is that the latent space allows easy random sampling and interpolation.

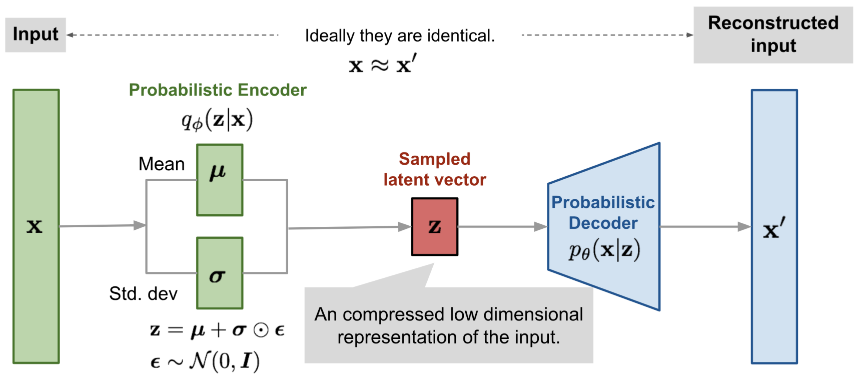
Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
$\mathbf{x}$
$\boldsymbol{\sigma}$
Std. dev
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

Fig. 4: Variational autoencoder schema [26]

VAE's loss function is:

$$L(\theta, \phi; x, z) = \mathbb{E}_{q\phi(z|x)}[\log p_\theta(x \mid z)] - D_{KL}q_{\theta(z|x)||p(z)}$$

Reconstruction loss part: $\mathbb{E}_{q\phi(z|x)}[\log p_\theta(x \mid z)]$ is almost the same as a standart autoencoder step except that here there is an Expectation operator ($\mathbb{E}$) because we are sampling from a distribution.

KL divergence part: $D_{KL}q_{\theta(z|x)||p(z)}$ will be explained below but the main idea is here that the distribution that you are learning is not too far removed from a normally distributed discussion. So force the latent distribution to be relatively close to a mean of zero and a standart deviation of one.

It achieves this by making its encoder outputting two vectors of size n: a vector of μ, and θ. These affect the parameters of a vector of random vairables of length n, with the i-th element of μ and σ are the mean and standart deviation of the i. To obtain the sampled encoding which we pass onward to the decoder:

- μ = [0.1, 1.2, 0.2, 0,8, ...]
- σ = [0,2, 0,5, 0,8, 1,3, ...]
- Intermediate X = $[X_{1\ N(0.1, 0.2^2)}, X_{2\ N(1.2, 0.5^2)}, X_{3\ N(0.2, 0.8^2)}, X_{4\ N(0.8, 1.3^2)}, ...]$
- Sampled Vector = [0.28, 1.65, 0.92, 1.98, ...]

This means, even for the same input, μ and θ, encoding will change a little bit on every single pass simply due to sampling.

The mean vector controls where the encoding of an input should be centered around, while θ controls the "area", how much from the mean the encoding can vary. Random generated encodings are anywhere inside the "circle", the decoder learns all nearby points refer to the same as well. Because of that the decoder does not just decode single, specific encodings in the latent space, but ones that slightly vary too, as the decoder is exposed to a range of variations of the encoding of the same input during training.

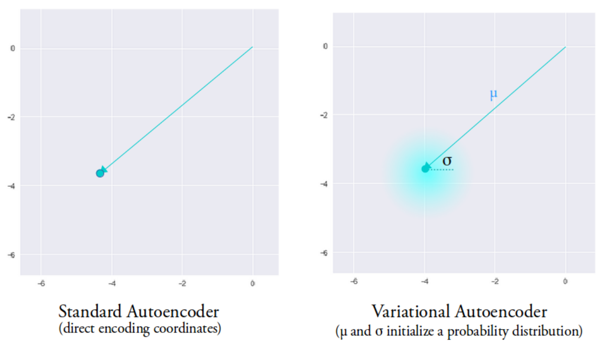

Fig. 5: Effect of μ and θ compared to standart autoencoder [24]

2.1 Kullback–Leibler Divergence

Encodings should be as close as possible to each other, allowing smooth interpolation and enabling the production of new samples.

There is a Kullback-Leibler divergence [13] in the loss function. Kullback–Leibler divergence (also called relative entropy) is a measure of how one probability distribution is different from a second, reference probability distribution. For VAEs, the KL loss is:

$$\sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - log\left(\sigma_i\right) - 1$$

Intuitively, this loss helps the encoder to distrubute all encodings around the center of the latent space. If it tries to cluster them away from the origin, it will be penalized.

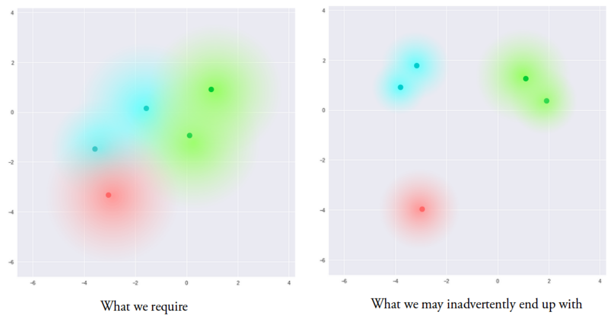

Fig. 6: Effect of KL divergence on encodings [24]

2.2 Summary

The idea of VAE is the most different one in autoencoders [8] but deeply rooted in the methods of variational bayesian and graphical models. If we label the distrubution as p_Θ, parameterized by Θ. The relationship between the input data $\mathbf{x}$ and the latent encoding vector $\mathbf{z}$ can be fully defined by:

- $p_{\Theta(z)}$
- $p_\Theta\left(x|z\right)$
- $q_\Theta\left(z|x\right)$

Find a distrubution $q\left(z|x\right)$ of some latent variables which we can sample from $z \sim q_\Theta\left(z|x\right)$ to generate new samples $x^{'} \sim p_\Theta\left(x|z\right)$.

3 Comparison

Both GANs and VAEs are generative models, which means that learn a given data distribution rather than its density. The key difference is how they achieve it.

VAEs learn a given distribution comparing its input to its output, this is good for learning hidden representations of data, but is pretty bad for generating new data. This is mainly due to the fact that we learn because we learn an averaged representation of the data thus the output becomes pretty blurry.

GANs take an entirely different approach. They use another network (so-called Discriminator) to measure the distance between the generated and the real data. Basically what it does is distinguishing the real data from the generated. It receives some data as an input and returns a number between 0 and 1. 0 meaning the data is fake and 1 meaning it is real. The generators goal then is learning to convince the Discriminator into believing it is generating real data.

The main advantage of GANs over VAEs in generating data is that they can be conditioned by different inputs. For example, you can learn the mapping between two domains: satellite images to google maps [10] or you can teach the generator to reproduce several classes of data: generating the MNIST dataset [16].

Because of these differences, they can be used for different tasks. VAEs are more suitable for compressing data to lower dimensions or generating semantic vectors from it. On the other hand, GANs are more suitable for generating data.

When using generative models, one could simply want to generate random new outputs, that is distributed similar to the training data, it is also possible with VAEs. But more often, you'd like to alter, or explore variations on data you already have, and not just in a random way either, but in a desired, specific direction. This is where VAEs work better than any other method currently available. Because VAEs stores latent attributes as probability distributions. On Figure 7, feature values are Gaussian distributed and takes value between 0 and 1. While creating a new sample, these features can be specialized.

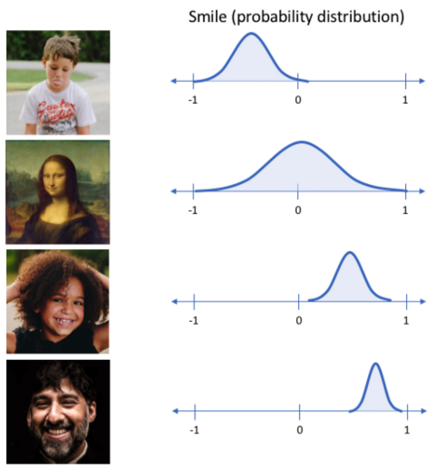

Fig. 7: VAEs stores latent attributes as probability distibutions. [11]

Variational approximations Variational methods define a lower bound

$$L(x;\Theta) \leq \log p_{model}(x;\Theta)$$

A learning algorithm that maximizes L is guaranteed to obtain at least the value of the log-likelihood. For many families of models, it is possible to define an L that is computationally tractable even when the log-likelihood is not. Currently, the most popular approach to variational learning in deep generative models is the VAE. Variational autoencoders are one of the three approaches to deep generative modeling that currently the most popular as of this writing, along with GANs. The main drawback of variational methods is that, when too weak of an approximate posterior distribution or too weak of a prior distribution is used. Even with a perfect optimization algorithm and infinite training data, the gap between L and the true likelihood can result in p_{model} learning something other than the true data. GANs are designed to be unbiased, which means that with a large enough model and infinite data. In practice, variational methods often obtain very good likelihood, but are regarded as producing lower quality samples. There is not a good method of quantitatively measuring sample quality, so this is a subjective opinion, not an empirical fact. See figure 11 for an example of some samples drawn from a VAE. While it is difficult to point to a single aspect of GAN design and say that it results in better sample quality, GANs are generally regarded as producing better samples.

GAN framework can train models that the VAE framework cannot and vice versa, but the two frameworks also have a lot in common. The most salient difference is that, if relying on standard backpropagation, VAEs cannot have discrete variables at the input of the generator because latent attributes are Gaussian disributed, while GANs cannot have discrete variables at the output of the generator.

We see that by training the discriminator, we are able to obtain an estimate of the ratio

$$\frac{p_{data}(x)}{p_{model}(x)}$$

at every point x. Estimating this ratio enables us to compute a wide variety of divergences and their gradients. This is the key approximation technique that sets GANs apart from variational autoencoders and Boltzmann machines.

Classification of generative models can be done based on maximum likelihood estimation. It helps to estimate parameters of a model that best fits the probabilistic density function of training data. There are two types of generative models: implicit and explicit. In explicit model, the task is to learn probability distribution underlying the data explicitly. Explicit model assume some prior distribution about some data and directly find density function in form of likelihood using optimization algorithm. Explicit models falls into two categories: tractable and approximate. Tractable explicit models are computationally tractable. Where else in approximate explicit models are not computationally tractable. So here various methods of approximations are used like variational methods, stochastic approximations etc to define density function. Variational autoencoder falls into this category. Implicit generative models learn data distribution without explicitly defines a density function. It is an implicit because it can only generate samples and cannot evaluate the likelihood of data distribution.

GANs requires finding a "Nash Equilibrium" during training. That is a point in the game between the generator and the discriminator where the game is set to terminate or that there is an end of game point. However there is no concrete algorithm to actually determine this equilibrium end of game point yet. On the other hand VAEs offer a closed form objective. There is a formula that can be used to determine the end of training phase in VAEs.

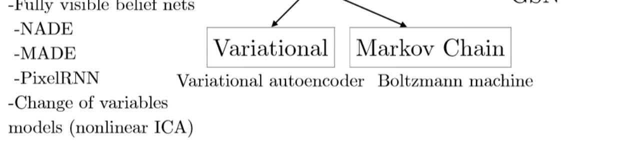

Fig. 8: Taxonomy of deep generative models [6]

Criteria	**VAE**	**GAN**
Learning Type	Convolutional Autoencoder	Convolutional networks with some constraint
Gradient Update	Stochastic Gradient Descent (SGD) with update to reconstruction and KL loss	SGD update to both Generator and Discriminator
Objective	Inference by matching latent data distribution to original data distribution	Learn structural hierarchy of objects in Generator and Discriminator
Performance Metrics	Log-likelihood and error rate	Accuracy and error rate

3.1 GANs In Application

GAN applications have increased rapidly in three years. There are different types of GANs customized for different purposes.

DCGAN (Deep Convolutional GANs)
In this approach CNNs have been used the first time within GANs and achieved impressive results. Before this, CNNs have shown unprecedented results in supervised computer vision tasks. It was a major milestone in GANs research as it introduced major architectural changes to tackle problems like training instability, mode collapse, and internal covariate shift. Since then, numerous GAN architectures were introduced based on the architecture of DCGAN [19].

BigGAN
This is the latest development in GANs for image generation. For the first time, GANs have generated images with high fidelity and low variety gap [2]. Previous highest Inception Score (visual quality of generated images, first introduces in 2016 [23]) was 52.52 and BigGAN achieved an Inception Score of 166.3, which was 100% better.

StyleGAN
StyleGAN is another major breakthrough in GANs research. StyleGAN was introduced by Nvidia. StyleGAN sets a new record in Face generation tasks. At the core of the algorithms are style transfer techniques or style mixing. Apart from generating faces, it can generate high-quality images of cars, bedrooms etc.

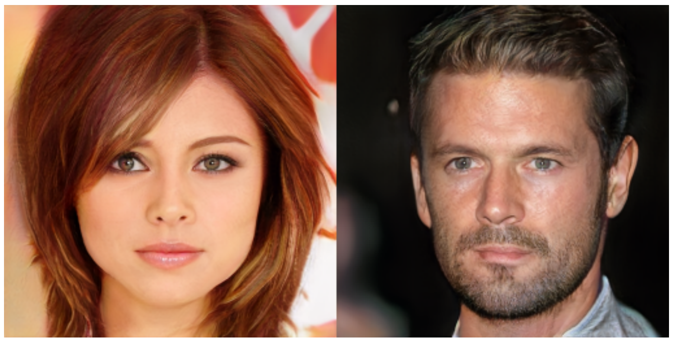
Fig. 9: Two imaginary celebrities that generated by a random number generator.

StackGAN
The definition of a StackGAN is text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks [29]. A StackGAN is a pair of networks that generates realistic looking images when provided with a text description.

CycleGAN
CycleGANs [30] have some use-cases, such as converting photos to paintings, converting a picture taken in summer to a photo taken in winter, or converting pictures of horses to pictures of zebras.

Pix2Pix
For image-to-image translation tasks, Pix2Pix [30] has also shown impressive results. For example converting night images to day images or vice versa, colorizing black and white images, translating sketches to photos and many more.

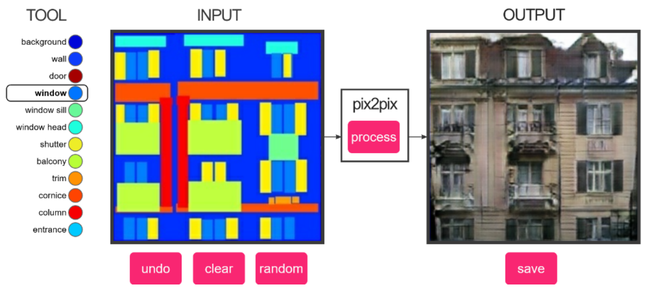

Fig. 10: Building generation with Pix2Pix

Age-cGAN
Face aging has many industry use cases, including cross-age face recognition, finding lost children, and in entertainment and this is what Age-cGAN [1] does.

3.2 VAEs In Application

Face Generation

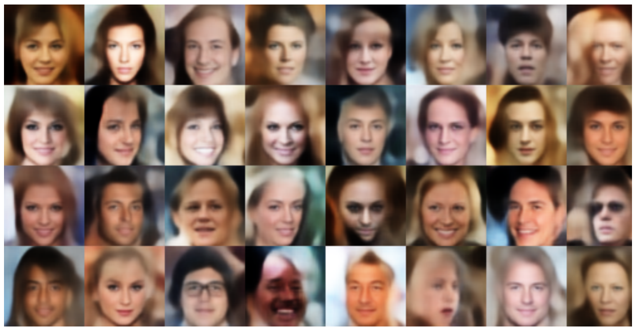

Fig. 11: Generated faces by VAE [28]

Dimensionality Reduction
Representing data in a lower-dimensional space can improve performance on different tasks, such as classification. Indeed, many forms of dimensionality reduction place semantically related examples next to each other, aiding in generalization.

Relationship with principal component analysis (PCA)
If linear activations are used, or only a single sigmoid hidden layer, then the optimal solution to an autoencoder is strongly related to principal component analysis. However, the potential of Autoencoders resides in their non-linearity, allowing the model to learn more powerful generalizations compared to PCA [3], and to reconstruct the input with a significantly lower loss of information.

Information Retrieval
Information Retrieval benefits particularly from dimensionality reduction by searches that can become extremely efficient in certain kinds of low dimensional spaces. Autoencoders were indeed applied to semantic hashing, proposed by Salakhutdinov and Hinton in 2007 [22]. In a nutshell, training the algorithm produces a low-dimensional binary code, then all database entries can be stored in a hash table mapping binary code vectors to entries.

Anomaly Detection
Autoencoders will reconstruct normal data very well, while failing to do so with anomaly data which the autoencoder has not encountered. Reconstruction error of a data point is the error between the original data point and its low dimensional reconstruction, is used as an anomaly score to detect anomalies.

Image Processing
The peculiar characteristics of autoencoders have rendered these model extremely useful in the processing of images for various tasks. For example image denoising [4], for the detection of breast cancer [27], super-resolution [25].

4 VAE-GAN

There is a new model already calls VAE-GAN [15]. As shown in figure 12, it looks like a variational autoencoder,

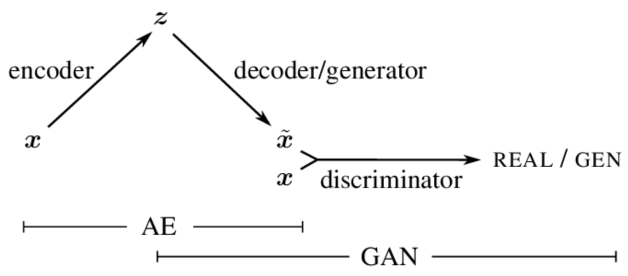

Fig. 12: VAE-GAN schema [18]

the difference is that the decoder is replaced with the generator of the GAN and loss function is calculated using the discriminator. The result is a pretty impressive. Image quality is improved compared to VAE and it outputs more diverse images compared to the GAN. The motivation behind this modification is VAEs tend to produce blurry outputs. This blurriness is related to VAEs's loss function.

$$-\mathcal{L}_{vae} = MSE(x_{decoder}, D^l(x_{real})) + prior$$

$$-\mathcal{L}_{GAN} = E_{x\sim Pdata}\log(D(x)) + E_{x\sim Pmodel}\log(1 - D(x))$$

$$\mathcal{L} = Learnedloss$$

The above equations assume the l^{th} layer of the discriminator for use in the VAE's loss term.The output of the GAN $D(x)$ is a scalar quantity. As a result, calculating the mean squared error (MSE) between the layer outputs gives us the VAE's loss function. The final output of GAN, *D(x)*, can then be used to calculate its own loss function.

5 Summary

This survey provides a comparative analysis of two popular Generative models on basis of their objective, performance and architecture. Both models have their own strengths and weaknesses. GANs and VAEs are both learn a given data distribution rather than density but their processes are different. GANs work like a two player game between generator model and discriminator. VAEs works with an encoder and decoder. VAEs learn by comparing its input to output and it causes blurry output. GANs learn by measuring the distance between the generated and the real data. GANs over VAEs in generate the data is that they can be conditioned by different inputs. VAEs over GANs in generate random new outputs and explore variations. Because of these differences, they can be used in different task.

References

1. G. Antipov, M. Baccouche, and J. Dugelay. Face aging with conditional generative adversarial networks. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2089–2093, Sep. 2017.
2. Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018.
3. Davide Chicco, Peter Sadowski, and Pierre Baldi. Deep autoencoder neural networks for gene ontology annotation predictions. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '14, page 533–540, New York, NY, USA, 2014. Association for Computing Machinery.
4. Kyunghyun Cho. Simple sparsification improves sparse denoising autoencoders in denoising highly noisy images. In *30th International Conference on Machine Learning, ICML 2013*, pages 1469–1477. International Machine Learning Society (IMLS), 2013.
5. Teun de Planque. Generative Adversarial Networks: The Future of Deep Learning? `https://mse238blog.stanford.edu/2017/08/teun/generative-adversarial-networks-the-future-of-deep-learning/`, 2017. [Online; accessed 19-December-2019].
6. Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks, 2016.
7. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.
8. Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In Timo Honkela, Włodzisław Duch, Mark Girolami, and Samuel Kaski, editors, *Artificial Neural Networks and Machine Learning – ICANN 2011*, pages 44–51, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
9. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
10. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
11. Jeremy Jordan. Variational autoencoders.). `https://www.jeremyjordan.me/variational-autoencoders/`, 2018. [Online; accessed 23-December-2019].
12. Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.
13. S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.
14. Denis Kuzminykh, Daniil Polykovskiy, Artur Kadurin, Alexander Zhebrak, Ivan Baskov, Sergey Nikolenko, Rim Shayakhmetov, and Alex Zhavoronkov. 3d molecular representations based on the wave transform for convolutional neural networks. *Molecular Pharmaceutics*, 15(10):4378–4385, 2018.
15. Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric, 2015.
16. Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.
17. Roger B. Myerson. Analysis of democratic institutions: Structure, conduct and performance. *Journal of Economic Perspectives*, 9(1):77–89, March 1995.
18. Praveen Narayanan. Learned Losses in the VAE-GAN. `https://pravn.wordpress.com/category/vae-gan-vaegan/`, 2018. [Online; accessed 13-February-2020].
19. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.
20. Adam Roberts, Jesse Engel, Colin Raffel, Ian Simon, and Curtis Hawthorne. MusicVAE: Creating a palette for musical scores with machine learning.). `https://magenta.tensorflow.org/music-vae`, 2018. [Online; accessed 21-December-2019].

21. Joseph Rocca. Understanding Generative Adversarial Networks (GANs). `https://mse238blog.stanford.edu/2017/08/teun/generative-adversarial-networks-the-future-of-deep-learning/`, 2019. [Online; accessed 21-December-2019].
22. Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969 – 978, 2009. Special Section on Graphical Models and Information Retrieval.
23. Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.
24. Irhum Shafkat. Intuitively Understanding Variational Autoencoders). `https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf`, 2018. [Online; accessed 26-December-2019].
25. T. Song, V. Sanchez, H. ElDaly, and N. M. Rajpoot. Hybrid deep autoencoder with curvature gaussian for detection of various types of cells in bone marrow trephine biopsy images. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 1040–1043, April 2017.
26. Lilian Weng. From Autoencoder to Beta-VAE). `https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html`, 2018. [Online; accessed 5-January-2019].
27. J. Xu, L. Xiang, Q. Liu, H. Gilmore, J. Wu, J. Tang, and A. Madabhushi. Stacked sparse autoencoder (ssae) for nuclei detection on breast cancer histopathology images. *IEEE Transactions on Medical Imaging*, 35(1):119–130, Jan 2016.
28. yzwxx. vae-celebA. `https://github.com/yzwxx/vae-celebA`, 2017. [Online; accessed 5-January-2019].
29. Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
30. Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.