1 Properties of Generative Models

People are familiar with classifying a machine learning model as supervised, unsupervied or reinforcement learning models based on the model's data type and model structure. Another way to distinguish the machine learning models is based on the model's output, and classify it as discriminative model or generative model.

1.1 Generative vs Discriminative Models

For the discriminative model, the task is to identify the labels of a given image. For example, an image classification model is a discriminative model. In other words, the discriminative model is trying to learn the distribution of the label with given data, which is p(y—x). Generative model, on the other hand, will output a new data sampled from the learned distribution. The learned distribution can be the joint distribution of data and label P(x, y) or simply the data itself p(x), if no labels are provided.

1.2 Why we need Generative Models

One significant advantage of the generative model is that it can create new data. If we train the model correctly, the new data will capture the structures of the original data. Therefore, if we want to do data forecasting or let the model doing a creative job, such as generate a piece of music with similar styles of a famous composer, then the generative model will do a great and surprisingly good job.

2 Examples of Generative Model

Variational Autoencoder and Generative Adversarial Networks are mainly used models in many areas with all kinds of generative models. In the following sections, we will briefly explain the structure of these two models, how they are trained, and their properties.

2.1 VAE

2.1.1 Model Structure

Variational Autoencoder, also called VAE, has an encoder-decoder structure that comes from autoencoder. The autoencoder contains two parts, an encoder that maps the input data into a latent space and a decoder that generates output from latent space. In general, the latent space has a lower dimension than the input data. The transformations of the autoencoder model are linear that we can view the encoder and decoder as two matrices. The output of the autoencoder usually has the same dimension as input data.

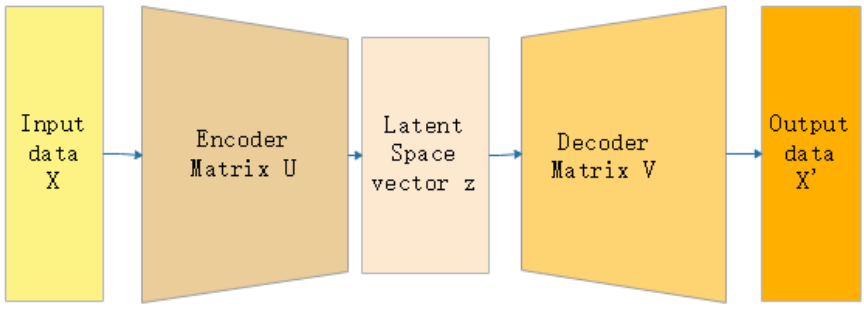

Figure 1: Autoencoder Structure

For VAE, we increase the model complexity by replacing encoder and decoder with deep neural networks,

where the encoder learns the distribution of latent states given the input $p_\theta(z|x)$, and the decoder learns the distribution of output given the sampled latent vectors $q_\theta(x|\hat{z})$, where $\hat{z} \sim p_\theta(z|x)$. The choice of deep neural networks can be variate according to different tasks, such as adding LSTM cells for time series data or an MLP for general purposes.

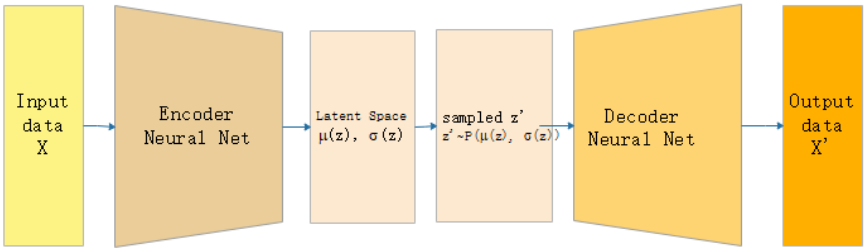

Figure 2: VAE Structure

2.1.2 Training Process

The VAE is trained with back propagation, however the loss function is bit different to traditional MLP neural nets. The loss function of VAE is defined as:

$$Loss = ||x - \hat{x}|| + KL(N(\mu_z, \sigma_z), N(0, 1))$$

Where $||x - \hat{x}||$ represent the reconstruction loss between the ground truth x and the generated output $\hat{x}$. The KL-Divergence loss represents the difference between the learned distribution $p_\theta(z), \theta = (\mu(z, \sigma(z)$ and the prior distribution of z $p_(z)$. In general, we assume the prior of z is a standard multi-Gaussian distribution.

One interesting tricks during VAE model is the reparameterization trick used for sample vector z' from $p_\theta(z)$. If we sample from $p_\theta(z)$ directly, then during the back propagation step, we need to take the gradient of $p_\theta(z)$ w.r.t. $\mu(z)$ and $\sigma(z)$ separately, and this step could be annoying. Instead of sampling directly, we can have:

$$z' = \mu(z) + \epsilon * \sigma(z), \epsilon \sim N(0, 1)$$

By using this trick, now we have $\frac{\partial z'}{\partial \mu} = 1$ and $\frac{\partial z'}{\partial \sigma} = \epsilon$, which will save a lot time during back propagation.

2.1.3 Model Features and Limitations

For VAE, the latent space can be view as a feature space. We can add some feature vectors to the latent vector of input data to generate output with these features, such as given an image of a person's face, we can add a feature vector of glasses to the input image's latent vector. Decoding from this augmented latent vector, the output image will be this person wears glasses.
Since VAE has an encoder-decoder structure, the latent space size will be a hyperparameter that can significantly influence the model performance. Suppose the latent space size is too small. In that case, the decoder cannot gain sufficient information to produce the output similar to the input data. Moreover, if we want to use VAE to generate the image, the output may not have a good performance with large image size.

2.2 GAN

2.2.1 Model Structure

Generative Adversarial Network contains two different neural nets: generator and discriminator. The generator is a generative model that tries to produce output similar to the input data, the input to generator is a random noisy vector. The discriminator is a discriminative model that tries to identify whether the input data is from a training set or generated by a generator. We can view GAN as a zero-sum game played by a generator and a discriminator, where the generator tries to fool the discriminator, and the discriminator tries not to be fooled by the generator.

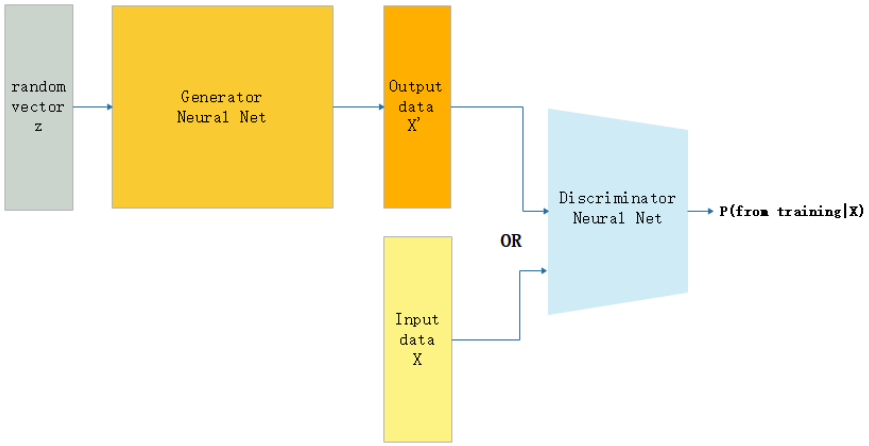

Figure 3: VAE Structure

2.2.2 Training Process

For GAN, the goal is to train a minmax function[4]:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1-D(G(z)))]$$

We will train the discriminator and generator separately. For discriminator, the cost function is the cross-entropy loss of classification between real and fake input data:

$$J_D = E_{x\sim p_{data}(x)}[-logD(x)] + E_{z\sim p_z(z)}[-log(1-D(G(z)))]$$

For generator, we will use a modified loss to avoid the saturation problem. A saturation problem is when the generator has a bad output, the output of discriminator will be close to 0, which makes the cost of generator be flat that the model cannot learn properly.

$$J_G = E_{z\sim p_z(z)}[-log(D(G(z)))]$$

In particular, during the training loop, we first randomly select a batch of data from the training set as x and generate a batch of random vector z to compute the cost of discriminator J_D and update the discriminator. After the discriminator is updated, we will re-sample a batch of random vectors z' to compute the generator loss J_G and update the generator. Since both discriminator and generator are updated during the training, in practice, we can observe that as the iteration goes, the loss of generator may not decrease, but the output quality will increase.

2.2.3 Model Features and Limitations

GAN has a zero-sum game structure, and it contains both generative and discriminative models. In practice, people use the GAN-based model to generate high-quality images, and the output images can have a larger size with better quality compared to VAE output.
However, since GAN generates images from the random input vector, we cannot control the output image by modifying the latent input vector. In other words, we do not have a method to control the output image features like VAE. Although this problem can be solved by train another VAE to capture the latent space representation of some features, it will increase the total model complexity and the training cost.

3 Applications of Generative Models

After seeing a lot theoretical explanation, we will show some applications based on VAE or GAN.

3.1 VAE

As we mentioned in the VAE sections, we could control the decoder's output features by modifying the latent vector passed to it. Here is a VAE model that used CNN as the encoder and decoder to generate images of human faces with different facial expressions and other features such as wearing sunglasses.

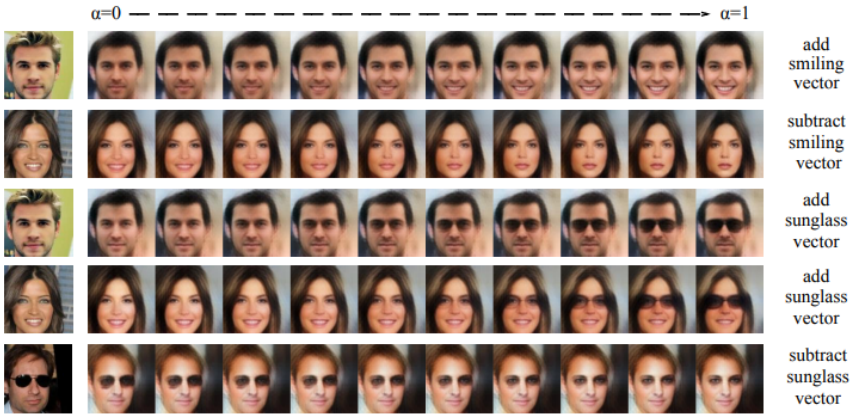

Figure 4: Output of VAE decoder by adding feature vectors to latent vector, cite from [5]

3.2 DC-GAN

Similar to the VAE model we introduced above, we have Deep Convolutional GAN(DC-GAN) that uses deep convolutional neural networks as the discriminator and generator[6]. CNN can reduce the model size while maintaining the model performance. Therefore CNN is wildly used for models that need to process image-like data. DC-GAN can be used to generate realistic images from a batch of random latent vectors. The style of generated images is determined by the training data.

Figure 5: Output of DC-GAN for bedrooms, cite from [6]

3.3 Style-GAN

Style-GAN is a more recent and powerful GAN model that can generate an image by mixing an image's style with a base image.[7] The generator of Style-GAN has a similar structure to VAE in that it contains two parts. The first part is an MLP network that maps the input latent vector z to another latent space W, and the second part is a multi-layer CNN with skip connections that generate the mixed-style image. Although Style-GAN is powerful and can produce high-quality images, it has some problems. One problem is that it works well only if the generated image contains one face. When the output image contains multiple human faces, some generated faces will blur or distort or lose the standard human face structure.

Figure 6: Output of Style-GAN, cite from [7]

4 Summary

This report explains the generative model and provides some examples and applications of VAE-based and GAN-based models. Here is short summarization of the two models.

VAE

- Encoder-Decoder Structure with Neural Networks
- Sample from Latent Space with reparameterization tricks
- Using KL-Divergence to minimize the prior and posterior distribution of latent space
- Using ELBO as loss function, Loss = Reconstruction loss (MSE) + Regularization loss(KL-divergence)

GAN

- Generator and Discriminator with Neural Networks
- Train Discriminator first, then Generator
- Generate output from random noisy vector
- Using cross entropy loss for Discriminator, and modified generator loss to avoid saturation problem

References

[1] Brownlee, J. (2019, July 19). *A gentle introduction to generative adversarial networks (gans)*. Retrieved April 07, 2021, from https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/.

[2] Jeremy Jordan. (2018, July 16). *Variational autoencoders*. Retrieved April 07, 2021, from https://www.jeremyjordan.me/variational-autoencoders/

[3] Rocca, J. (2021, March 21). *Understanding variational Autoencoders (vaes)*.. Retrieved April 07, 2021, from https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. Generative Adversarial Networks, 2014; arXiv:1406.2661.

[5] Xianxu Hou, Linlin Shen, Ke Sun and Guoping Qiu. Deep Feature Consistent Variational Autoencoder, 2016; arXiv:1610.00291.

[6] Alec Radford, Luke Metz and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015; arXiv:1511.06434.

[7] Tero Karras, Samuli Laine and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018; arXiv:1812.04948.