Generative Models

Sameer Gabbita & Arnav Jain

May 2022

1 Introduction

Generative models are an unsupervised task which aims to generate new examples from an existing dataset. There are a variety of ways to do this, including Autoencoders and GAN's, both of which have branched out into several submodels which are good at achieving different tasks. Generative models are useful in machine learning for creating synthetic data, cleaning up data, and for prediction models.

2 Autoencoders

2.1 Overview

Autoencoders are an unsupervised machine learning tool that learns to encode the data into a smaller dimensional representation, then decode it back to close to the original image. This inherently teaches the autoencoder to denoise images or data. The encoder does this through encoding correlated data.

2.2 Implementation Details

An autoencoder consists of three parts: the encoder, the bottleneck, and the decoder. In order to train an autoencoder to encode successfully, a reconstruction loss is used, which measures the difference between the encoded image and the produced image. Additionally, a regularizer tends to be added to prevent the autoencoder from memorizing. An easy way to implement autoencoders in code is to reduce the number of nodes in the hidden layer of a neural network so that there is a bottleneck layer.

2.3 Sparse Autoencoders

This type of autoencoder doesn't require a bottleneck, instead it works by only activating certain neurons in the layer, instead of the whole layer.

2.3.1 Implementation

There are two ways to implement the sparse autoencoder. The first method, called L1 regularization, punishes the number of activations performed by the neurons. It works by adding a component to the loss function that minimizes the number of activations, scaled by a factor of λ.

$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i \left| a_i^{(h)} \right| \tag{1}$$

The second method, is a bit more complicated and is called the KL divergence, but also works by limiting activations.

2.4 Applications

One of the major applications of regular autoencoders is denoising images and reducing the dimensionality of data. Furthermore, the encoder-decoder structure can also be used to generate additional fake data and predict the data-point in a larger sequence.

3 Variational Autoencoders

There are a few problems that arise when applying autoencoders. Namely, the distribution to sample data from in the latent distribution is not fixed and we don't know where to pick samples. To address this, the variational autoencoder restricts the latent space values by forcing it to follow a normal distribution, by encoding a mean vector μ and a standard deviation vector σ.

3.1 Applications

Variational autoencoders are often used to reveal underlying data patterns from large data distributions (such as large sequential databases). They are also much powerful at generating synthetic images compared to standard autoencoders. Prior research applications have also used variational autoencoders to debias deep learning models.

4 GANs

Generative Adversarial Networks were introduced in 2014 by Ian Goodfellow where he introduced the idea of a 2-player game (minimax) to create synthetic data. The topic of Vanilla GANs have been covered extensively in prior TJML club lectures. Today, we will look into some more advanced and interesting variations of the network.

4.1 Deep Convolutional-GANs

The initial implementation of the GAN utilized linear layers. As a result, when generating synthetic images, the model could not retain any spatial information and the generated images were easily distinguishable from real images. The Deep Convolutional GAN (DC-GAN) utilizes convolutional-based layers for improved results. The general training process and architecture of the DC-GAN remains the same as the Vanilla GAN. However, in the discriminator, linear layers are replaced by convolutional layers while in the generator, linear layers are replaced by transposed convolutional layers.

4.1.1 Transposed Convolutions

Transposed convolutions are used to upsample images. It is essentially the opposite of a traditional convolution.

4.1.2 Implementation Details

The paper that introduced the DC-GAN architecture suggested a set of hyperparameters that are important to include when developing your own GAN to optimize the training process. They include:

- Use strided convolutions rather than pooling layers in the discriminator (allows the model to learn more information about an image)
- Use batch-normalization in both the generator and discriminator
- Remove fully-connected layers from deeper architectures
- Use the ReLU activation function throughout the generator, except for the output layer when hyperbolic tangent is suggested
- Use the Leaky ReLU activation function throughout the discriminator

4.2 Wasserstein GANs

Due to the unstable training process of GANs, many problems arise. First, mode collapse is the phenomenon when generator gets stuck during training and generates a single class from a larger dataset. Second, the binary cross entropy loss function used to optimize the discriminator to differentiate between real and fake images leads to vanishing gradient problems, causing the training process to essentially halt.

The Wasserstein GAN (W-GAN) has been developed to address the aformentioned problems that commonly arise in DC-GANs. Rather than utilizing a discriminator, which outputs a binary classification of whether an input image is real or fake, the W-GAN replaces it with a Critic network. Instead of outputting a probability, the Critic outputs any real number - signifies the distance between the real data distribution and the distribution of the generator's outputs.

Because the Critic doesn't output a probability, binary cross entropy can no longer be used to train he model. Rather, the W-GAN's loss function is based off of the Earth mover's distance metric (a measure of the similarity between 2 probability distributions). The new W-Loss function is:

$$\mathcal{L} = min_g max_c E(c(x)) - E(c(g(z))) \tag{2}$$

Similar to the standard GAN training process, the W-GAN still trains using a minimax game. The generator wants to minimize the Earth mover's distance while the critic wants to maximize the metric.

4.3 Image Translation

4.3.1 Overview

Image Translation are done through unsupervised learning models called CycleGANs. Image Translation aims to convert one image to another. An example of an Image Translation model is one that inputs a picture of a scene in the summer, and outputs the same picture as if it was taken in the winter, and vice-versa.

CycleGANs are really useful for these tasks as they do not a paired input and output to train on. Instead, it relies on two sets of unpaired datasets taken in different domains. In the example above, it would need a set of pictures taken in the summer, and another in the winter, but these sets do not have to be paired.

4.3.2 Implementation

CycleGANs use are essentially two normal GANs positioned in a cycle. It is composed of two discriminators and two generators. One generator-discriminator pair is used to go from one domain to the next, and the other pair does the opposite. Although this is useful, in order for it to be a correct translation of the same exact scene, what CycleGAN does is use the output of one GAN, as the input for the next. This idea is called Cycle Consistency, and means that an image that starts as a summer scene, when passed through both GANs(aka one cycle), should result in the same image. It does this through a different loss function, called Cycle Consistency Loss.

$$\mathcal{L}_{cyc}(G, F) = E_{x \sim p_{data}(x)} \left[\|F(G(x)) - x\|_1 \right] + E_{y \sim p_{data}(y)} \left[\|G(F(y)) - y\|_1 \right] \tag{3}$$

As you can see, the cycle consistency loss aims to minimize the error when running a data point through the cycle of the two GAN's represented by F and G.

4.3.3 Applications

One popular application of image translation and CycleGANs are for style transfer. Style transfer essentially takes an image, and converts it into the style of something else. This can be used for converting a photograph to the style of a Picasso painting. There is also an ongoing "toy" contest on Kaggle where you can apply a CycleGAN (or Neural Style Transfer) to convert images to the style of paintings created by Claude Monet. Another popular application in the medical community is to convert between different imaging modalities while retaining the same information (i.e. converting MRI scans to CT scans). Coloring gray-scale images and enhancing image quality through super resolution are also popular applications.

4.4 Conditional GANs

4.4.1 Overview

Conditional GANs (CGAN) are GANs that use auxiliary information to augment the output. This auxiliary information are called labels, and modify the output that the GAN produces.

4.4.2 Implementation

The generator aims to generate data for each label. However, the discriminator, rather than just distinguishing between real and fake data, it also learns to distinguish labels that don't match the image. A quick example can be seen with handwritten digits. If the generator produces a 3, but there is a label 4 passed along with it, the discriminator should learn to reject it. As with GAN's the discriminator of the Conditional GAN should still also learn to distinguish real and fake images.

4.5 StyleGAN

Introduced in 2018, the StyleGAN framework is the current state-of-the-art architecture for generating lifelike images. Furthermore, they allow for the generation of a more diverse set of images while allowing for more control about the features of a generated image.

The StyleGAN greatly improves on the generator aspect of the GAN. First, a random vector is first fed through a series of linear layers. The goal of this process is to map a random distribution to the desired distribution of images we want to generate (these fully connected layers are known as a mapping network). This is followed by a synthesis network which generates an image from the mapped distribution. Random noise vectors are injected throughout the synthesis network to increase the diversity of generated images. Furthermore, vectors from the mapped distribution are intermediately injected into the synthesis network to provide more control about the features in the generated image. These mapped vectors go through a process of adaptive instance normalization in the network. The synthesis network also goes through a process known as progressive growth where the generated image gradually gets bigger as it is getting processed through the network. This step is vital for enhancing resolution and image quality.

The results of the StyleGAN was one of the first generative models to raise significant ethical issues for how lifelike produced images are.

4.6 Additional Resources

- VAE: `https://arxiv.org/pdf/1312.6114.pdf`
- Vanilla GAN: `https://arxiv.org/pdf/1406.2661.pdf`
- DC-GAN: `https://arxiv.org/pdf/1511.06434.pdf`
- WGAN: `https://arxiv.org/pdf/1701.07875.pdf`
- CycleGAN: `https://arxiv.org/pdf/1703.10593.pdf`
- Conditional GAN: `https://arxiv.org/pdf/1411.1784.pdf`
- StyleGAN `https://arxiv.org/pdf/1812.04948.pdf`