Generative Deep Learning

Gilles Pilon

2024-05-12

Improving life one Python script at a time

Gilles Pilon

Contents

1 Abstract

Just as mastering the science behind cooking empowers chefs to innovate in the kitchen, understanding generative deep learning unlocks a world of creative possibilities. Generative models are not just about following pre-built recipes; they are about using knowledge of algorithms and data to push boundaries. By understanding how these models learn from vast datasets and identify patterns, we can guide them to generate entirely new content, from creating never-before-seen images to composing novel music pieces. We will delve deeper into the exciting world of generative deep learning and its potential to revolutionize various fields of visual media, music and audio, drug discovery, materials science, engineering and design, and personal creativity, to help everyone learn and improve.

2 Preface

I am a Python developer and data scientist. I started my career as a process engineer in aluminum rolling mills, transitioned to data science in the food industry, and then to artificial intelligence and machine learning.

3 Generative Deep Learning

There are two core concepts to understanding generative deep learning: generative modeling and deep learning.

3.1 Generative Modeling

"Generative modeling is a branch of machine learning that involves training a model to produce new data that is similar to a given dataset." (Foster 2023, p. 4)

Discriminative modeling

Analogy Imagine you are training a group of people to recognize the plays and poems written by William Shakespeare. You have a dataset of all of his works, as well as many works by many other authors. Each work in your dataset that was written by Shakespeare would be labeled with a 1 and all other works would be labeled with a 0. The group would learn that certain words, phrases, and styles are more likely to indicate that a work was written by Shakespeare. In supervised machine learning an algorithm learns from labeled data, called the training data, to make predictions or classifications. The group learns how to discriminate between these two groups (1, 0) and determines the probability that a new document shown to the group has a label of 1, that is, a work that was created by Shakespeare. This is called discriminative modeling.

Generative modeling

Analogy Imagine your training dataset contains only the works of William Shakespeare. Each work is called an observation. Each observation has many features, such as individual words or groups of letters. You train a generative model on this dataset to determine the rules of the complex relationships between the words in the works of Shakespeare. Now you can sample from this model to create new, realistic plays or poems that did not exist in the dataset, but look like they were created using the same rules as the original data. A generative model must also be probabilistic. We sample many different variations of the output; we do not get

the same output every time, as we would with a discriminative model. A generative model has a random component that influences the outputs generated by the discriminative model. We want our model to generate new observations that look as if they were from the training dataset.

A discriminative model is easier to create, less demanding on the software and hardware, and has many use cases for solving problems. A model created to optimize the components of a industrial mixture is beneficial in a setting where the variability of the components changes constantly. A model to determine if a cancer is present in an image can speed up diagnosis. It is much easier to train a model to predict if a text file was written by a famous author or if an image was painted by a famous artist, than it is to build a model to generate text resembling that author or to create an image that resembles the artist. In the last decade software librairies have improved, new ones created, and hardware to solve specific computations in machine learning process data faster.

Generative modeling framework This is a structure of our goals for our generative model. (Foster 2023, pp. 10–11)

- **Training data** Gather a dataset of observations that were generated according to an unknown distribution.
- **Modeling** Build a generative model that mimics the unknown distribution and sample from the generative model to generate observations that appear to have come from the unknown distribution.
- **Accuracy** Determine the accuracy of the generative model. A high accuracy means that a generative observation looks like it came from the unknown distribution.
- **Generation** It should be easy to sample a new observation from the generative model.
- **Representation** It should be possible to understand the high-level features in the dataset of observations are represented by the generative model. Each observation in the training data is described (mapped) using a lower-dimensional latent space.

3.2 Deep Learning

"Deep learning is a class of machine learning algorithms that uses multiple stacked layers of processing units to learn high-level representations from unstructured data." (Foster 2023, p. 23)

Structured data These data are organized in a predefined format, like rows and columns in a spreadsheet. Supervised learning algorithms excel at finding patterns in these data because the features are well-defined. Many machine learning algorithms require structured data.

Unstructured data These data lack a predefined format, like text , images, or videos. These data may have a spatial structure (image), a temporal structure (audio, text), or both a spatial and temporal structure (video). The individual observations (pixels, letters, frequencies, etc.) are very uninformative, that is, the data are granular. Various algorithms perform poorly with such granular data and the spatial or temporal structure. Deep learning is especially useful for unstructured data.

Deep learning is performed with any system that employs many layers to learn high-level representations of the training data. Most deep learning uses artificial neural networks that have multiple stacked hidden layers. Each layer contains units that are connected to the previous layer through a set of weights. The most common type of layer is the fully connected layer (also known as the dense layer) which means every unit in a layer is directly connected to every

unit in the previous layer. There are several types of artificial neural networks. We will consider the multilayer perceptrons and the convolutional neural network.

Multilayer perceptrons (MLP) All adjacent layers are fully connected. The input is transformed by each layer in turn until it reaches the output layer. Each unit applies a transformation to its inputs and passes the output to the next layer. A single unit in the final layer outputs the probability that the original input belongs to a specific category. The transformation involves a weighted sum of the inputs. The deep neural network finds the set of weights for each layer by training the network. Text, audio, or images are processed through the network and the predicted outputs are compared to the truth. If there are errors in the prediction these are propagated backward through the network and the weights are adjusted to find those which improve the prediction; this is called backpropagation. An artificial neural network learns features from the training data without human guidance in order to minimize the prediction error.

Convulational neural network (CNN) This type of neural network takes into account the spatial structure of the training data. This is particularly useful for images. The image is converted to a single vector before passing it to the dense layer. A convolution is a mathematical operation of sliding a filter across the input data (typically an image) and performing element-wise multiplication. The result is a new value that captures how well the filter matches the features in a specific region of the input. A convolutional layer takes the input and applies convolutions with multiple filters to generate feature maps. It is a collection of filters and the values stored in the filters are the weights learned through training. Each filter in the layer detects a specific feature in the input data. By stacking multiple convolutional layers, a CNN can learn increasingly complex features from the input image, ultimately leading to object recognition or other image analysis tasks.

4 Methods of Generative Modeling

Before delving into the methods of generative modeling, there is one key concept to understand. A **probability density function** is used to specify the probability of a random variable falling within a particular range of values, as opposed to taking on any one value. Instead of a single guess, the model provides a range of possibilities with their corresponding likelihoods.

There are two different approaches of generative modeling:

Implicit density models These produce a random (also called stochastic) process of generating data. These models do not calculate the probability density function.

Explicit density models These constrain the model-building process so that the probability density function is easier to calculate.

4.1 Generative Adversarial Networks (GAN)

This is an implicit density model. A GAN is a battle between two adversaries. Imagine you have two art *specialists*, one a talented counterfeiter (generator) and the other an art critic (discriminator). They are locked in an artistic battle.

The counterfeiter (generator) keeps creating new forgeries, trying to mimic the style of famous artists (data). The critic (discriminator) examines each forgery and tries to determine if it is a real painting or a fake. Over time, the counterfeiter gets better at creating convincing fakes, while the critic becomes a sharper judge. This competition pushes both to improve. The counterfeiter learns the subtleties of the artist's style, and the critic hones their ability to detect even the most minor discrepancies.

This is how a GAN works. One neural network (generator) creates new data (like images or music) based on training data. The other network (discriminator) tries to distinguish the generated data from real data. Through this adversarial process, the generator becomes adept at producing data that closely resembles the real thing, while the discriminator keeps the generator on its toes by constantly improving its detection abilities. This makes GANs powerful tools for generating new, realistic data for various applications, from creating new images to composing novel music pieces. They constantly push the boundaries of what artificial intelligence can create, blurring the lines between the artificial and the real.

4.2 Variational Autoencoders (VAE)

This is an explicit density model. VAEs can be thought of as artistic compressors. Imagine you have a box filled with different kinds of toys (training data), and you want to compress them into a smaller box (latent space) while still being able to recreate them (generate new toys) later.

A regular autoencoder would simply shrink the toys down and then try to inflate them back to their original size. This process is messy and the resulting toys might be blurry or miss some details.

A variational autoencoder is more sophisticated. It encodes the toys into a special kind of compressed space that not only captures their size but also some of their key features. It is like having a box filled with colored blocks (latent space) where each block represents a specific type of toy (a red block for cars, a blue block for dolls, a green block for plush toys). By sampling from this box of blocks and using a decoder, a VAE can generate new, never-before-seen toys that share characteristics with the originals. This makes them useful for tasks like creating realistic images of faces or coming up with new music samples that fit a particular style.

4.3 Energy-based Models (EBM)

This is an explicit density model. EBMs are like happiness detectors for data. Imagine a landscape with hills and valleys, where the height of the ground represents the model's "energy" for a particular data point. Data points that fit the model well (real or desirable) would be in valleys with low energy. Data points that do not fit (outliers or noise) would be on hills with high energy.

The core idea of EBMs is to define a function that assigns an energy score to every possible data point. The model then learns to adjust this function so that low energy regions correspond to the kind of data it is trying to model. Think of it like sculpting a clay model. You keep smoothing the clay (adjusting the energy function) to create a landscape with deep valleys where you want the features of your model to reside. Data points that land in these valleys are considered likely or desirable by the model.

EBMs are a powerful tool for creating realistic images or music. By sampling data points from low energy regions, they can produce new data that adheres to the patterns learned from the training data. Additionally, their probabilistic nature allows them to estimate the likelihood of any generated data point.

4.4 Diffusion Models

This is an explicit density model. Imagine you have a clear image and slowly add random noise to it, making it increasingly blurry and unrecognizable over time. This *noise addition process* is essentially what a diffusion model does in reverse.

The model is first trained on real data by learning this *noise addition process.* It essentially learns how to corrupt clear images with controlled noise step-by-step. Then, to generate new data, the model starts with pure noise and reverses this corruption process one step at a time. Think of it like slowly removing fog from a landscape photo. With each step, the model removes some noise, revealing more and more details of the underlying image. By learning the process of adding noise, the model can learn how to remove it effectively, ultimately creating a brand new image that resembles the training data.

Diffusion models are particularly adept at capturing complex structures and realistic details in data like images or audio. Their ability to learn the data distribution through a denoising process makes them powerful tools for various generative tasks.

4.5 Autoregressive Models

This is an explicit density model. An autoregressive model is like a mathematical equation that takes a series of past values as inputs and outputs a prediction for the next value in the sequence. By analyzing historical data, the model learns these underlying patterns and uses them for forecasting. Autoregressive models are like fortune tellers who predict the future based on the past. Imagine a fortune teller who uses tarot cards (past data points) to predict your upcoming week (future outcomes). They analyze the cards you draw (past values) and based on the patterns they have seen before (relationships between past and future events), make predictions about your future (future values).

In statistics and machine learning, autoregressive models use past values of a time series to predict the next one. They are particularly useful for data that exhibits some dependence on its history, like stock prices or weather patterns. However, autoregressive models have limitations. Just like a fortune teller cannot predict everything, these models can struggle with significant changes or unexpected events. They assume the future will resemble the past to some extent, which might not always be true.

4.6 Normalizing Flow Models

This is an explicit density model. Imagine you have a simple distribution of data, like a bunch of balls clustered in the center of a sandbox (latent space). A normalizing flow model can transform this simple distribution into a more complex one, like sculpting the sandbox into intricate shapes (complex data). Think of it as a flexible mold. You can start with a simple shape and keep applying transformations to create more intricate and varied forms, all while maintaining the ability to go back to the original shape if needed. This makes normalizing flow models powerful for generating realistic and diverse data while providing valuable insights into the probability of the generated data.

The normalizing flow model achieves this by applying a series of reversible functions, like pushing and pulling the sand with tools. Each function modifies the distribution in a controlled way, gradually shaping it into the desired form. The key here is that these functions are invertible, so you can always reverse the steps to get back to the original distribution. This invertibility is crucial because it allows the model to not only generate new data points (like scooping out new shapes in the sandbox) but also efficiently calculate the probability of any

data point it generates. This makes normalizing flows attractive for tasks where understanding the likelihood of generated data is important.

5 Deep Learning Software

Creating generative deep learning models requires powerful, feature-rich software frameworks. TensorFlow (Google Brain 2024b) with Keras (Google Brain 2024a) are exceptionally good Python packages.

5.1 TensorFlow

It is an open-source software package for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. (Google Brain 2024b)

5.2 Keras

It is open-source software for deep learning with a high-level application programming interface (API) that makes it very simple to train and run artificial neural networks. It provides numerous methods that can be combined to create very complex deep learning applications. It runs on top of TensorFlow and other packages. (Google Brain 2024a)

6 Abbreviations

7 Glossary

A

application programming interface It is a way for two or more computer programs to communicate with each other. It is a type of software interface, offering a service to other pieces of software. 10

artificial intelligence See deep learning. 11

artificial neural network It is a network of artificial neurons or nodes. 10

D

deep learning It is a type of artificial neural network containing multiple hidden layers. 10

deep neural network See deep learning. 10, 11

designed experiment It is a method to change factors simultaneously (inputs) using a structured pattern, in order to determine their effect on some response(s) (output(s)). 10

E

exogenous variable Exogenous variables are terms in a model. They are caused by factors outside the system. Other names are x, independent variable, regressor, design, explanatory variable, and feature (machine learning). In $y = mX + b$, x is the exogenous variable. In a designed experiment, it is a factor that you can control, want to control, or is controlled at low cost. 10

H

hidden layer It is a synthetic layer in a artificial neural network between the input layer (the exogenous variables) and the output layer (the prediction). Hidden layers typically contain an activation function (such as ReLU) for training. A deep neural network contains more than one hidden layer. 10

K

Keras It is a open-source software for deep learning with a high-level application programming interface that makes it very simple to train and run artificials neural networks. It can run on top of TensorFlow and other packages. 8

M

machine learning It is an approach to making repeated decisions that involves algorithmically finding patterns in data and using these to make models that deal correctly with new data. There are fourt types of machine learning: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. 11

O

open-source software It is computer software that is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose. Open-source software may be developed in a collaborative public manner. Open-source software is a prominent example of open collaboration, meaning any capable user is able to participate online in development, making the number of possible contributors indefinite. The ability to examine the code facilitates public trust in the software. 10, 11

P

prediction It is the output of a model. 10

R

reinforcement learning It is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward. 10

S

semi-supervised learning It is an approach to machine learning that combines a small amount of labeled data with a large amount of unlabeled data during training. Semi-supervised learning falls between unsupervised learning (with no labeled training data) and supervised learning (with only labeled training data). It is a special instance of weak supervision. 10

supervised learning It is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. The five steps of supervised learning are: 1. import the classes you plan to use, 2. instantiate the transformers, classifiers, regressors, and selectors, 3. train the model on an existing dataset, 4. tune the hyperparameters, and 5. predict the response for a new dataset. 10

T

TensorFlow It is an open-source software package for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. 8, 10

U

unsupervised learning It is a type of algorithm that learns patterns from untagged data. The hope is that through mimicry, which is an important mode of learning in people, the machine is forced to build a compact internal representation of its world and then generate imaginative content from it. 10

8 References

Foster, David (2023). *Creative Deep Learning. Teaching Machines to Paint, Write, Compose, and Play*. 2nd ed. Sebastopol, CA: O'Reilly Media, Inc., p. 456.

Google Brain (2024a). *Keras*. URL: https://github.com/keras-team/keras.

– (2024b). *TensorFlow*. URL: https://github.com/tensorflow/tensorflow.