MIT Press • Open Encyclopedia of Cognitive Science

Generative Modeling

Jun-Yan Zhu[1] Phillip Isola[2]

[1]Carnegie Mellon University, [2]Massachusetts Institute of Technology

MIT Press

Published on: Feb 12, 2025

DOI: https://doi.org/10.21428/e2759450.ff10c6af

Generative models are statistical models that learn to generate data samples following the probability distribution $p(x)$ of data x. For example, a generative model trained on cat images could learn to produce new photos of cats. A useful variant, conditional generative models, can further take instructions c and create data according to the distribution $p(x \mid c)$ in its conditional form. For example, given the instruction, "Make me a video of kids playing on a beach at sunset," a text-conditioned video generative model produces videos corresponding to this description. Generative models are distinguished from other kinds of machine learning models by two properties. First, their output is high dimensional and structured—in other words, it is something that we would call "data" rather than a label or a decision. Examples include text, images, videos, weather patterns, and more. Second, the mapping from input instructions to outputs is one-to-many. Many photos match the text instruction "a beach at sunset," and the model should be able to output the entire distribution of possibilities. Generative models are widely used in computer vision, natural language processing, and machine learning. In cognitive science, probabilistic generative models are commonly used as priors for Bayesian models of perception and cognition.

History

Humans have always been fascinated by creating data. For most of history, this was the domain of artists, musicians, and authors, among others. With the advent of machine learning, the ability to *automate* the creation process, modeling and generating data via algorithms, has become a new focus.

Classic generative models such as the Gaussian mixture model, which models observations as coming from a mixture of multiple different Gaussian distributions, have long been used to analyze data. Later, probabilistic graphical models (Koller & Friedman, 2009) became popular tools for generative modeling, with a focus on discovering latent structures. Notable examples include restricted Boltzmann machines (Smolensky, 1986) and deep belief networks (Hinton, 2009). Since learning a probability density is quite challenging, several methods were introduced to instead learn unnormalized densities (energy functions) or density derivatives (score functions; Hinton, 2002; Hyvärinen, 2005), as these relative quantities are often sufficient for sampling. However, the models of this era were relatively small and only succeeded on modeling simple, low-dimensional distributions.

Since 2014, deep neural networks have become the dominant architecture for generative modeling. Popular methods within this family include generative adversarial networks (Goodfellow et al., 2014), variational autoencoders (Kingma & Welling, 2014), autoregressive models (van den Oord et al., 2016), flow-based models (Dinh et al., 2017), diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015), and score-based models (Song & Ermon, 2019). These methods are effective at modeling the full distribution of high-dimensional data. Their success is due to development of the abovementioned new generative models, advances in neural network architectures, and the availability of massive training datasets.

Core concepts

Generative models need to map from each given input instruction to a *distribution* of outputs. This is a one-to-many mapping problem. The typical way to solve it is to define a *stochastic* mapping, in which each time you run the model, you get a different randomized output. Figure 1 provides an overview of learning and sampling algorithms.

Sampling

How can we define such a stochastic mapping? There are two main methods: sampling with a generator and sampling from a density function. In the first method, we augment the input to the generator with a random variable, often referred to as "noise." The generator, a deterministic function, uses this variable to produce the output directly. The idea is that this variable specifies all the latent factors necessary to determine the output, which are unspecified by the input instructions. For example, in a cat generator, the random variables might determine the cat's pose, lighting, and appearance.

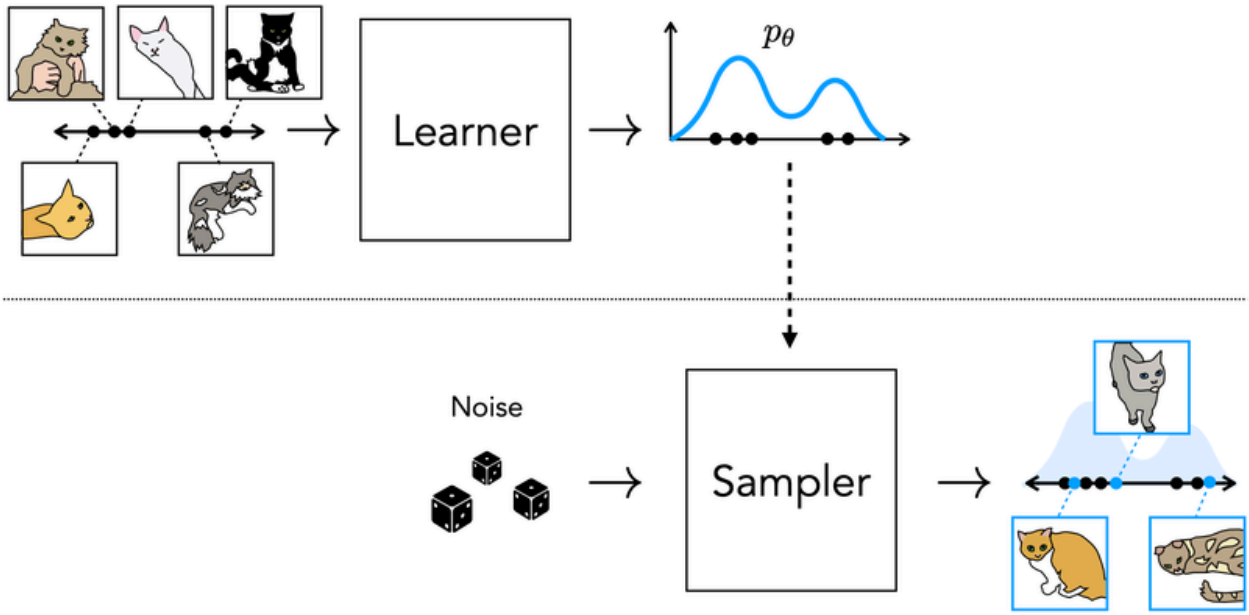

Figure 1
Generative models consist of two phases: (1) learning—from a set of training examples, the model learns a distribution from which these examples could have been sampled—and (2) sampling—new samples are drawn from the inferred distribution.

The other approach is to learn a density function that maps from inputs x to a probability distribution that assigns a scalar to each possible output representing their probability of occurring. We then draw new samples from this distribution with algorithms such as the Markov Chain Monte Carlo (MCMC; Robert & Casella, 2013) or its Langevin-type variant (Neal, 2012) [see Markov Chain Monte Carlo]. The high-level idea is to begin with a randomly initialized data point x and iteratively move it toward regions of higher probability. For example, we can initialize an image with Gaussian noise, where no training

images are nearby. The sampling algorithm then moves this image into a high-density area, where many training images reside, resulting in a more realistic cat image.

Learning

How can we learn such a model? Most generative modeling methods use some form of *maximum likelihood estimation* ($\arg\max_\theta E_x \log p_\theta(x)$), in which we maximize the expected log-likelihood of the training data under the density function p_θ. The idea is to assign more probability density to regions of the output space where the training data is found while reducing the density for regions where there is little or no training data. However, estimating the probability of high-dimensional data is often intractable. To tackle this issue, researchers either rely on architectures with certain restrictions (e.g., invertible networks and sequential data generation) to compute exact likelihood, use approximation methods such as variational methods and MCMC, or adopt adversarial learning, which uses a binary classifier that distinguishes generated samples from real data and trains a generator to fool the classifier.

Applications

In addition to content creation, generative models can be used for a wide variety of applications, including producing synthetic training data for other machine learning algorithms, acting as world models in model-based reinforcement learning, discovering data structures through latent variable modeling, deriving useful representations for downstream tasks, and compressing data.

Questions, controversies, and new developments

Generative models introduce two types of controversies: that they are not good enough and that they are too good. The first deals with issues of bias and hallucinations. *Bias* occurs when a model's outputs systematically differ from what we want, either due to training on data that is biased (compared to the ideal data distribution we want) or due to the model poorly fitting the data. *Hallucination* happens when the model's outputs are inconsistent with reality. For example, a language model might produce sentences that are factually incorrect [see Large Language Models].

The second controversy is that generative models can be too good, producing images, voices, videos, and more that look so real they can fool humans. This can lead to misinformation, as seeing is no longer believing. To address this issue, visual forensics techniques (Farid, 2009) have been developed to automatically detect synthetic images, although this cat-and-mouse game continues as both forensics algorithms and generative models become more powerful.

Broader connections

One notable example of generative modeling in language domains is large language models, which are primarily based on autoregressive models. More broadly, generative models have seen extensive use in defining priors for Bayesian models as they are used to model psychological phenomena [see Bayesian Models of Cognition; Bayesianism].

Acknowledgments

We would like to thank the valuable comments from reviewers and editors. We also thank the generous support from the Packard Foundation, Sloan Foundation, and the National Science Foundation (IIS-2239076 and ISS-2403303).

Further reading

- Goodfellow, I. (2016). NIPS 2016 tutorial: Generative adversarial networks. *arXiv*. https://doi.org/10.48550/arXiv.1701.00160
- Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, *12*(4), 307–392. https://doi.org/10.1561/2200000056
- Song, Y., & Kingma, D. P. (2021). How to train your energy-based models. *arXiv*. https://doi.org/10.48550/arXiv.2101.03288
- Tomczak, J. M. (2022). *Deep generative modeling*. Springer.

References

- Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2017). Density estimation using Real NVP. *arXiv*. https://doi.org/10.48550/arXiv.1605.08803 ↩
- Farid, H. (2009). Image forgery detection. *IEEE Signal Processing Magazine*, *26*(2), 16-25. https://doi.org/10.1109/MSP.2008.931079 ↩
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K.Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. NeurIPS Proceedings. ↩
- Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*(8), 1771–1800. https://doi.org/10.1162/089976602760128018 ↩
- Hinton, G. E. (2009). Deep belief networks. *Scholarpedia Journal*. http://scholarpedia.org/article/Deep_Belief_Networks ↩
- Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, *33*, 6840–6851. ↩

- Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, *6*, 695–709.

  ↩
- Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. *arXiv*. https://doi.org/10.48550/arXiv.1312.6114

  ↩
- Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press.

  ↩
- Neal, R. M. (2012). MCMC using Hamiltonian dynamics. *arXiv*. https://doi.org/10.48550/arXiv.1206.1901

  ↩
- Robert, C., & Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

  ↩
- Smolensky, P. (1986). *Information processing in dynamical systems: Foundations of harmony theory*. University of Colorado.

  ↩
- Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd International Conference on Machine Learning* (vol. 37, pp. 2256–2265). PMLR.

  ↩
- Song, Y., & Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *arXiv.* https://doi.org/10.48550/arXiv.1907.05600

  ↩
- van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. *Speech Synthesis Workshop*, 125.

  ↩