Generative Deep Learning

Zhongjian Lin
University of Georgia

November 17, 2024

Artificial Intelligence

The potential of artificial intelligence to emulate human thought processes goes beyond passive tasks such as object recognition and mostly reactive tasks such as driving a car. It extends well into creative activities.

- Artificial intelligence isn't about replacing our own intelligence with something else, it's about bringing into our lives and work more intelligence—intelligence of a different kind.
- In many fields, but especially in creative ones, AI will be used by humans as a tool to augment their own capabilities: more *augmented intelligence* than artificial intelligence.
- Our perceptual modalities, our language, and our art work all have statistical structure. Learning this structure is what deep learning algorithms excel at.
- Machine learning models can learn the statistical latent space of images, music, and stories, and they can then sample from this space, creating new artworks with characteristics similar to those the model has seen in its training data.

Generative AI

- In the summer of 2015, we were entertained by Google's DeepDream algorithm turning an image into a psychedelic mess of dog eyes and pareidolic artifacts
- In 2016, we started using smartphone applications to turn photos into paintings of various styles
- In the summer of 2016, an experimental short movie, Sunspring, was directed using a script written by a Long Short-Term Memory

Text Generation

We'll explore how recurrent neural networks can be used to generate sequence data. We'll use text generation as an example, but the exact same techniques can be generalized to any kind of sequence data. Sequence data generation is in no way limited to artistic content generation. It has been successfully applied to speech synthesis and to dialogue generation for chatbots.

Brief history

- The LSTM (Long Short-Term Memory) algorithm in 1997.
- In 2002, Douglas Eck, then at Schmidhuber's lab in Switzerland, applied LSTM to music generation for the first time, with promising results.
- In the late 2000s and early 2010s, Alex Graves did important pioneering work on using recurrent networks for sequence data generation. In particular, his 2013 work on applying recurrent mixture density networks to generate human-like handwriting using timeseries of pen positions is seen by some as a turning point.
- Between 2015 and 2017, recurrent neural networks were successfully used for text and dialogue generation, music generation, and speech synthesis.
- Then around 2017–2018, the Transformer architecture started taking over recurrent neural networks, not just for supervised natural language processing tasks, but also for generative sequence models—in particular language modeling . The best-known example of a generative Transformer would be GPT-3, a 175 billion parameter text-generation model trained by the startup OpenAI.
- GPT-4 can solve difficult problems with greater accuracy, thanks to its broader general knowledge and problem solving abilities.

Generate sequence data

The universal way to generate sequence data in deep learning is to train a model (usually a Transformer or an RNN) to predict the next token or next few tokens in a sequence, using the previous tokens as input. For instance, given the input "the cat is on the," the model is trained to predict the target "mat," the next word. As usual when working with text data, tokens are typically words or characters, and any network that can model the probability of the next token given the previous ones is called a *language model*. A language model captures the latent space of language: its statistical structure.

Once you have such a trained language model, you can sample from it (generate new sequences): you feed it an initial string of text (called conditioning data), ask it to generate the next character or the next word (you can even generate several tokens at once), add the generated output back to the input data, and repeat the process many times. This loop allows you to generate sequences of arbitrary length that reflect the structure of the data on which the model was trained: sequences that look almost like human-written sentences.

Language Model

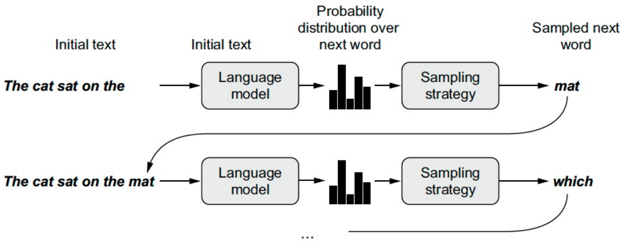


Figure 12.1 The process of word-by-word text generation using a language model

Sampling strategy

When generating text, the way you choose the next token is crucially important.

- A naive approach is greedy sampling, consisting of always choosing the most likely next character. But such an approach results in repetitive, predictable strings that don't look like coherent language.
- A more interesting approach makes slightly more surprising choices: it introduces randomness in the sampling process by sampling from the probability distribution for the next character. This is called stochastic sampling.
- In such a setup, if a word has probability 0.3 of being next in the sentence according to the model, you'll choose it 30% of the time. Note that greedy sampling can also be cast as sampling from a probability distribution: one where a certain word has probability 1 and all others have probability 0.
- Sampling probabilistically from the softmax output of the model is neat: it allows even unlikely words to be sampled some of the time, generating more interesting looking sentences and sometimes showing creativity by coming up with new, realistic sounding sentences that didn't occur in the training data.

Sampling strategy

It doesn't offer a way to control the amount of randomness in the sampling process. In order to control the amount of stochasticity in the sampling process, we introduce a parameter called the *softmax temperature*, which characterizes the entropy of the probability distribution used for sampling: it characterizes how surprising or predictable the choice of the next word will be. Higher temperatures result in sampling distributions of higher entropy that will generate more surprising and unstructured generated data, whereas a lower temperature will result in less randomness and much more predictable generated data.

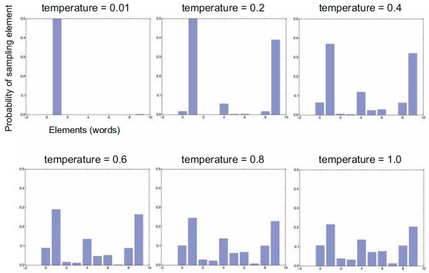


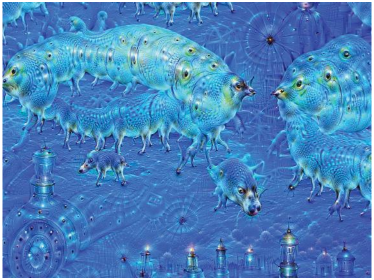
Figure 12.2 Different reweightings of one probability distribution. Low temperature = more deterministic, high temperature = more random.

Generative Deep Learning

- You can generate discrete sequence data by training a model to predict the next token(s), given previous tokens.
- In the case of text, such a model is called a language model. It can be based on either words or characters.
- Sampling the next token requires a balance between adhering to what the model judges likely, and introducing randomness.
- One way to handle this is the notion of softmax temperature. Always experiment with different temperatures to find the right one.

DeepDream

DeepDream is an artistic image-modification technique that uses the representations learned by convolutional neural networks. It was first released by Google in the summer of 2015 as an implementation written using the Caffe deep learning library (this was several months before the first public release of TensorFlow). It quickly became an internet sensation thanks to the trippy pictures it could generate, full of algorithmic pareidolia artifacts, bird feathers, and dog eyes—a byproduct of the fact that the DeepDream convnet was trained on ImageNet, where dog breeds and bird species are vastly overrepresented.



Neural style transfer

In addition to DeepDream, another major development in deep-learning-driven image modification is neural style transfer, introduced by Leon Gatys et al. in the summer of 2015. Neural style transfer consists of applying the style of a reference image to a target image while conserving the content of the target image.

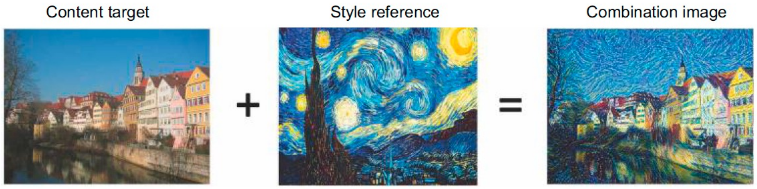


Figure 12.9 A style transfer example

Neural style transfer

Style essentially means textures, colors, and visual patterns in the image, at various spatial scales, and the *content* is the higher-level macrostructure of the image. The idea of style transfer has had a long history in the image-processing community prior to the development of neural style transfer in 2015. But as it turns out, the deep-learning-based implementations of style transfer offer results unparalleled by what had been previously achieved with classical computer vision techniques, and they triggered an amazing renaissance in creative applications of computer vision.
The key notion behind implementing style transfer is the same idea that's central to all deep learning algorithms: you define a loss function to specify what you want to achieve, and you minimize this loss. We know what we want to achieve: conserving the content of the original image while adopting the style of the reference image. If we were able to mathematically define content and style, then an appropriate loss function to minimize would be the following:

$$\begin{aligned} loss =& distance(style(reference\_image) - style(combination\_image))+ \\ & distance(content(original\_image) - content(combination\_image))) \end{aligned}$$

A fundamental observation made by Gatys et al. was that deep convolutional neural networks offer a way to mathematically define the style and content functions.

Content and Style

Content

Activations from earlier layers in a network contain local information about the image, whereas activations from higher layers contain increasingly global, abstract information. Formulated in a different way, the activations of the different layers of a convnet provide a decomposition of the contents of an image over different spatial scales. Therefore, you'd expect the content of an image, which is more global and abstract, to be captured by the representations of the upper layers in a convnet.
A good candidate for content loss is thus the L_2 norm between the activations of an upper layer in a pretrained convnet, computed over the target image, and the activations of the same layer computed over the generated image. This guarantees that, as seen from the upper layer, the generated image will look similar to the original target image. Assuming that what the upper layers of a convnet see is really the content of their input images, this works as a way to preserve image content.

Content and Style

Content

The content loss only uses a single upper layer, but the style loss as defined by Gatys et al. uses multiple layers of a convnet: you try to capture the appearance of the style-reference image at all spatial scales extracted by the convnet, not just a single scale. For the style loss, Gatys et al. use the Gram matrix of a layer's activations: the inner product of the feature maps of a given layer. This inner product can be understood as representing a map of the correlations between the layer's features. These feature correlations capture the statistics of the patterns of a particular spatial scale, which empirically correspond to the appearance of the textures found at this scale.
Hence, the style loss aims to preserve similar internal correlations within the activations of different layers, across the style-reference image and the generated image. In turn, this guarantees that the textures found at different spatial scales look similar across the style-reference image and the generated image.

VAEs and GANs

There are two main techniques in this domain of image generation: variational autoencoders (VAEs) and generative adversarial networks (GANs). The key idea of image generation is to develop a low-dimensional *latent space* of representations. Once such a latent space has been learned, you can sample points from it, and, by mapping them back to image space, generate images that have never been seen before. These new images are the in-betweens of the training images.

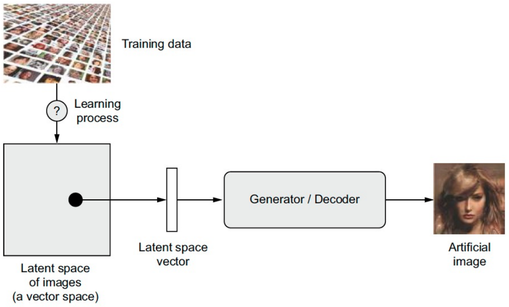


Figure 12.13 Learning a latent vector space of images and using it to sample new images

VAEs

VAEs are great for learning latent spaces that are well structured, where specific directions encode a meaningful axis of variation in the data. VAEs achieves the task of image editing via *concept vectors*. In a latent space of images of faces, for instance, there may be a smile vector, such that if latent point z is the embedded representation of a certain face, then latent point $z + s$ is the embedded representation of the same face, smiling.

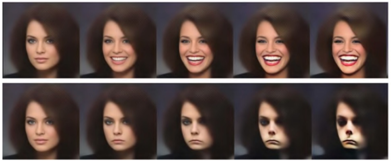


Figure 12.15 **The smile vector**

GANs

GANs generate images that can potentially be highly realistic, but the latent space they come from may not have as much structure and continuity. An intuitive way to understand GANs is to imagine a forger trying to create a fake Picasso painting. At first, the forger is pretty bad at the task. He mixes some of his fakes with authentic Picassos and shows them all to an art dealer. The art dealer makes an authenticity assessment for each painting and gives the forger feedback about what makes a Picasso look like a Picasso. The forger goes back to his studio to prepare some new fakes. As time goes on, the forger becomes increasingly competent at imitating the style of Picasso, and the art dealer becomes increasingly expert at spotting fakes. In the end, they have on their hands some excellent fake Picassos.

- Generator network—Takes as input a random vector (a random point in the latent space), and decodes it into a synthetic image.
- Discriminator network (or adversary)—Takes as input an image (real or synthetic), and predicts whether the image came from the training set or was created by the generator network