I’m looking for feedback on this chapter draft for the 2nd edition of Modeling Mindsets:

- Where you get **confused** or lost or have an unanswered question
- Where you **disagree** or have different experiences
- Where you start to **get bored** and feel like skipping/giving up
- Anything you find **especially interesting or helpful**

You can leave **inline comments** in this Google Doc (easiest for me), or send an e-mail with the comments to readers@christophmolnar.com

This doc is shared with other beta readers. If you prefer to work in your own copy, just create a copy and share it with me later.

Thanks!

Christoph

---

13. Self-Supervised Learning – Reconstruct Unlabeled Data

Premise: The world is best approached by reconstructing it.

Consequence: Automatically generate labels from unlabeled data and learn models that encode the world.

The royal watchmaker accepted three new apprentices. The supervised learner asked for feedback at every step, while the reinforcement learner needed occasional rewards and punishments. The self-supervised learner vanished into the storage room. The watchmaker was horrified to discover that the self-supervised learner had taken apart and attempted to rebuild all the watches. Eventually, however, the self-supervised learner would become the greatest watchmaker of all time.

Admittedly this book’s short stories are not fully my own creation. I had help. For inspiration and cutting the clutter, I relied on self-supervised machine learning. I used large language models (GPT-3 in particular) – machine learning models trained on huge amounts of text to generate new text and serve as chatbots.

Self-supervised is the youngest member of the machine learning family. While it was first mentioned in 1978 and some ideas go back to the 80s, self-supervised learning took off around 2016 lagging behind the deep learning trend. It started out as a fancy approach to feature engineering for supervised models. The initial motivation: Labeled data is expensive, but unlabeled data is cheaply available. Self-supervised machine learning thrives on large unlabeled datasets and helps learn meaningful representations of the data, useful for downstream tasks such as classification. Then came generative AI. Today AI can summarize articles, serve as chatbot, and create images – all made possible with self-supervised learning.

Let's explore this versatile mindset.

13.1 Learning is based on reconstruction

The core idea of self-supervised learning: predict one part of the input data from another part. A reconstruction of the data points. To create the training data, the modeler starts with the data set. The modeler "removes" information from the data points and the model has to learn to reconstruct that information. There is lots of room for creativity when creating the reconstruction task: blur it, mask it, slice it, grey it, cut it, spin it, augment it. For example:

- Cut off the last token of a text snippet. Reconstruction task: Predict the next token from the text snippet.
- Gray out an image. Reconstruction task: Predict the colored image from the gray version.
- Remove links from a network graph. Reconstruction task: Predict missing links between nodes given the partial network.

But how is reconstruction helpful? What can trained self-supervised models be used for? Machine learning in general is a rather task-oriented mindset, and self-supervised learning is no exception.

Direct application: Some of the self-supervised reconstruction tasks directly translate into an application. For example, colorization is a potential feature of photo editing software. Other examples are inpainting (filling missing spots in images) and suggesting connections in friend networks (link prediction).

Generate new data: Some self-supervised approaches can generate new data. Large language models, for example, can be made into chatbots. Another example is image generators that create images from text descriptions.

Representation learning: Self-supervised models typically have a "bottleneck" in their model architecture that forces them to represent the input data as a vector. This vector can be used in downstream tasks. For example for classification, but also in search applications. Again useful when little labeled data is available, but lots of unlabeled data. The difficulty for the modeler is to figure out a reconstruction task that forces the model to learn *useful* representations.

To achieve this goal, the modeler can use one of 3 approaches: generative, contrastive, or adversarial. All are based on deep learning and share some parts of architecture. All architectures use an encoder to create an information bottleneck: the model is forced to learn a compact yet informative representation of the data to fulfill the task. They also differ in how the task is represented and respectively which loss is used (Liu et al. 2021).

13.2 Reconstruct by generating

Generative neural network architectures consist of an encoder and a decoder, a combination also called the generator, see Figure 13.1. While the job of the encoder is to turn a data point (e.g. an input text) into a representation vector, the job of the decoder is to create a new data point, such as the next token. Training is based on "reconstruction-losses" that compare how different the generated data point is from the original.

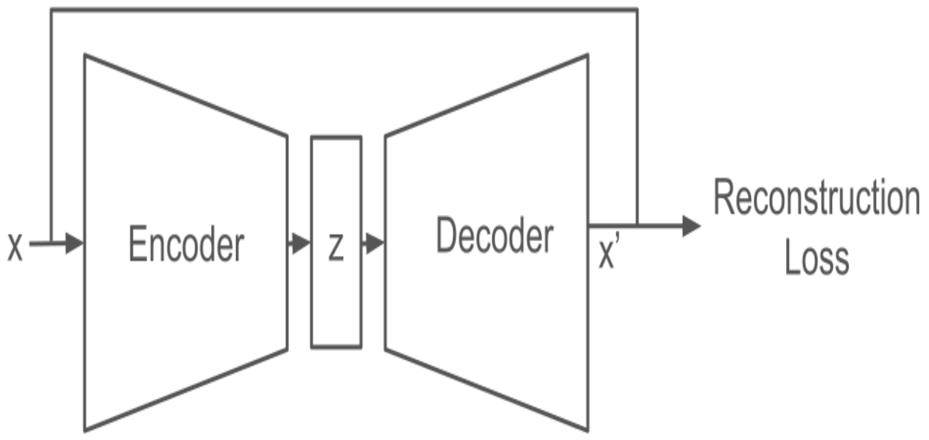


Figure 13.1: Generative architecture. Based on Liu et al. (2021), CC-BY 4.0

A famous example of generative learning is the large language model GPT-2 (Radford et al. 2019) which is short for generative pre-trained transformer 2. GPT-2 was trained on BookCorpus (Zhu et al. 2015) a dataset with 7k self-published books plus a dataset with 8 million websites. Here's how the reconstruction tasks and training data were generated for GPT-2:

- Tokenization: The text is split into smaller parts (word pieces)
- Slicing: The tokenized text is sliced at different positions.
  - The sequence on the left is the input.
  - The first token on the right is the target.

For illustration, with the text "No, this is Patrick!" and word-level tokenization, the training pairs are:

- Input: "No," -> Target: "this"

- Input: “No, this” -> Target: “is”
- Input: “No, this is” -> Target: “Patrick”

These pairs are used to train the model by comparing the predicted tokens with the actual tokens using negative log-likelihood loss (=reconstruction loss).

Generative models like GPT-2 have two uses: generating new data and representation learning. For representing features, the modeler only needs the encoder part of the (already trained) network. Put in a data point, and get back a hopefully meaningful encoded vector, which can be used for training supervised models or search.

13.3 Reconstruct by contrasting

But there are more ways to make a model learn about the world than reconstructing the same input. Such as the contrastive approach. Models in the **contrastive** approach have two inputs instead of one, see Figure 13.2. The encoder produces a representation for both. Then the discriminator part of the network compares the two representations using a contrastive loss. Then the representation(s) are fed into a discriminator that compares the two using a contrastive loss function. The discriminator is typically a shallow neural network. Compared to the generative approach, the contrastive approach works just fine without a decoder. But that also means that contrastive approaches typically lack the ability to generate data.

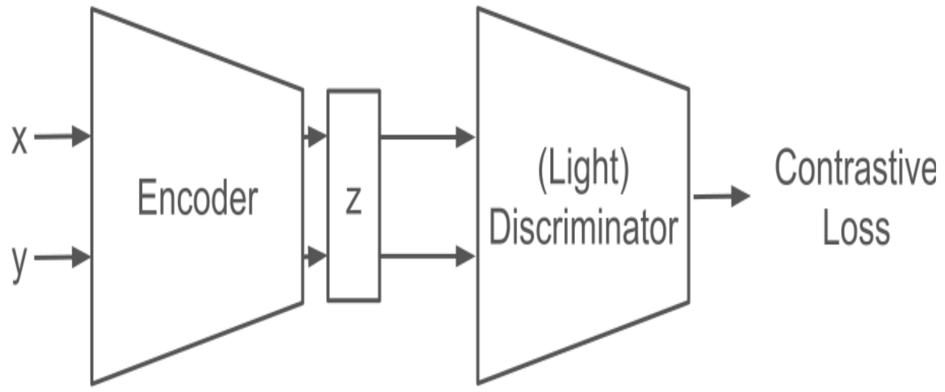
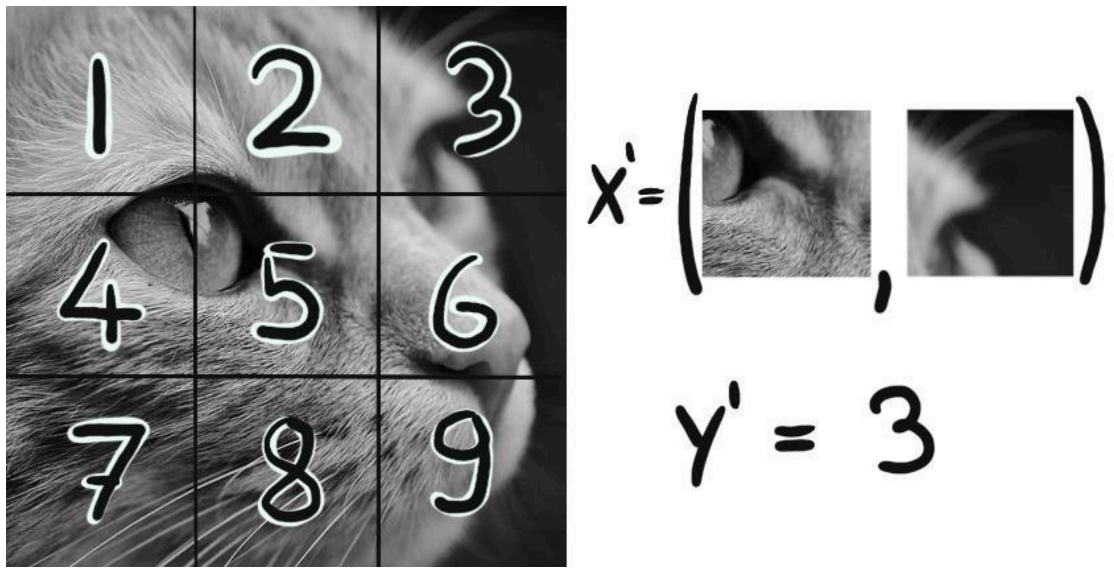


Figure 13.2: Contrastive architecture. Based on Liu et al. (2021), CC-BY 4.0

The contrastive approach is useful for learning feature representations to be used in some downstream tasks, like classification. One example of helping with image classification is context learning (Doersch et al. 2015): First, each image is split into 9 patches. Given one of the 9 patches, the goal is to predict for one of the other patches which position it is in. So the first image provides the context, and for the second we want to guess the position.



It’s not a directly useful task unless we need help with 3 by 3 puzzles. Instead, the hope is that learning the positional arrangement makes the model learn feature representations that are useful for other tasks.

13.4 Reconstruct by competing

Then there’s the **adversarial** approach, a hybrid also called **generative-contrastive**. Adversarial architectures first start with a generator: encode x and decode it again. But it also has a discriminator, usually a somewhat larger network that compares the x and the x’ using distributional divergence (see Figure 13.3). An example of generative-contrastive approaches is generative adversarial networks (GANs) Goodfellow et al. (2020). The job of the generators (consisting of encoder-decoder pair) is to produce “fake data”. Then the discriminator gets two examples, a real data point and a fake one, and has to find out which one is real. Both the generator and discriminator are trained in tandem. They “co-evolve”.

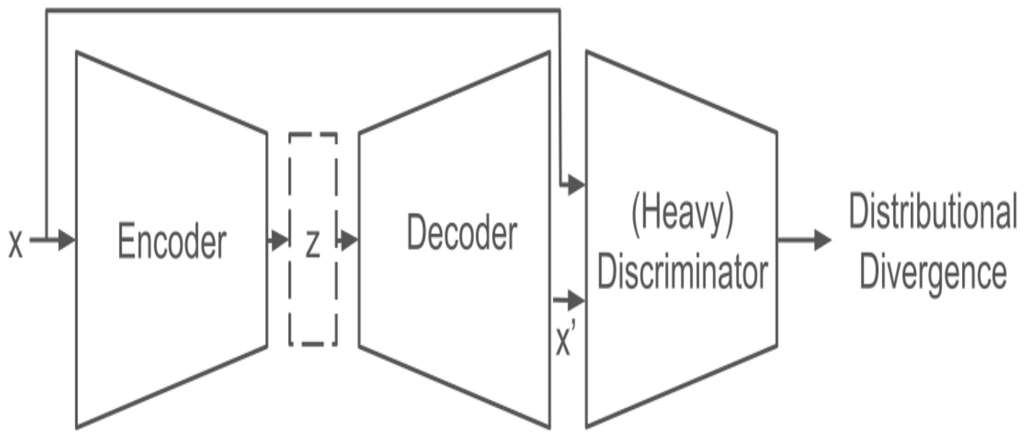


Figure 13.3: Adversarial architecture. Based on Liu et al. (2021), CC-BY 4.0

13.5 Self-supervised learning and other mindsets

Self-supervised learning is a typical task-oriented machine learning mindset. Many people will categorize it as unsupervised learning, but it's somewhere between supervised and unsupervised. The model DOES have a ground truth from which to learn, only it's automatically generated. One point in favor of self-supervised = supervised. But in many cases, modelers don't care about the "prediction" of the self-supervised model, but rather want the model to learn a representation – this makes it more akin to unsupervised learning. To make matters more confusing: Self-supervised learning needs deep learning to exist. It wouldn't work without the expressiveness of neural networks, end-to-end learning, the flexibility. But there are two factors that make it a distinct mindset: The huge community, especially around generative AI and that self-supervised learning has outgrown its role as an auxiliary approach but serves as standalone modeling. Some people are even convinced that self-supervised learning may lead to artificial general intelligence (AGI) (Dickson 2022). But not me.

13.6 Strengths & Limitations

+ No labels needed.

+ Easier to evaluate than your typical unsupervised learning task.

+ Profits from having vast amounts of data.

+ Possibly emergent behavior

+ Produces more generally applicable solutions, especially compared to supervised learning which produces task-specific narrow solutions.

- High computational costs to get useable results. Especially GPUs.

- Usually needs lots of data.

- Tasks may be difficult to formulate

- Fast-moving field with very different approaches (Balestriero et al. 2023)