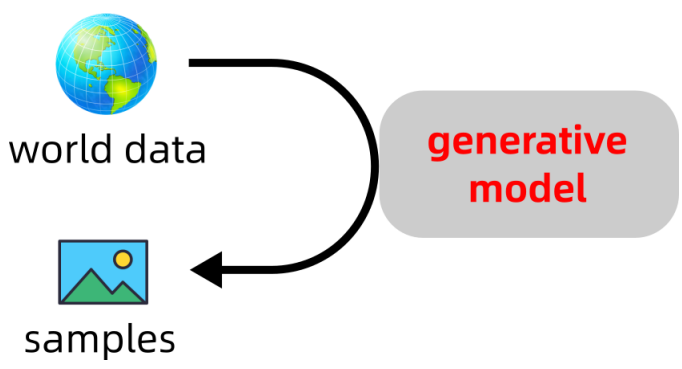
AGI = RL + LLM

Quick review of generative models

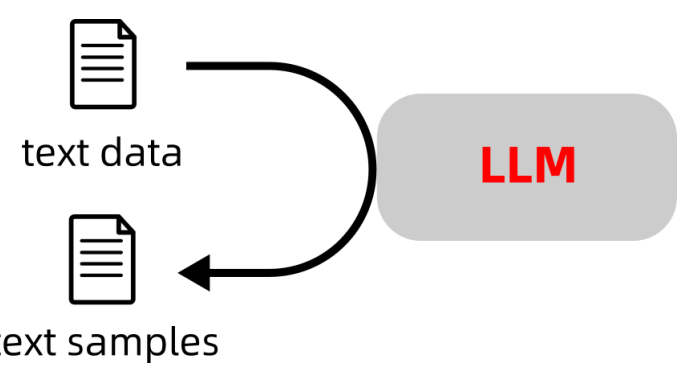
(This quick review is based from the book "The Science of Deep Learning" [Iddo Drori, 2023] with my own simplifications)

A generative model, as opposed to a classifying model, is one that learns the probability distribution of the data and outputs samplings from the learned distribution:



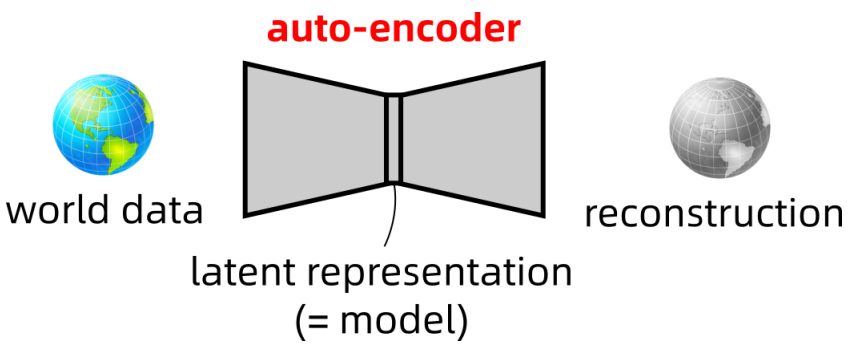
(1)

LLMs are a special case of generative models:



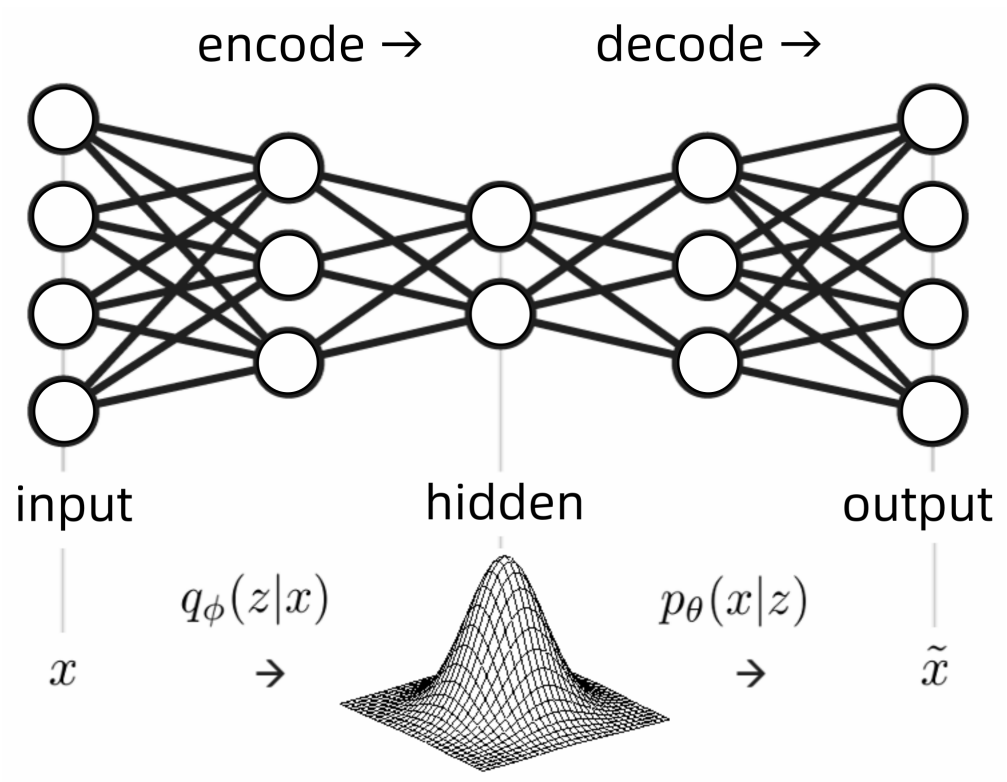
(2)

One class of generative models are auto-encoders, which forces information to flow through a narrow bottleneck, thus compressing the data into a compact, latent representation:



(3)

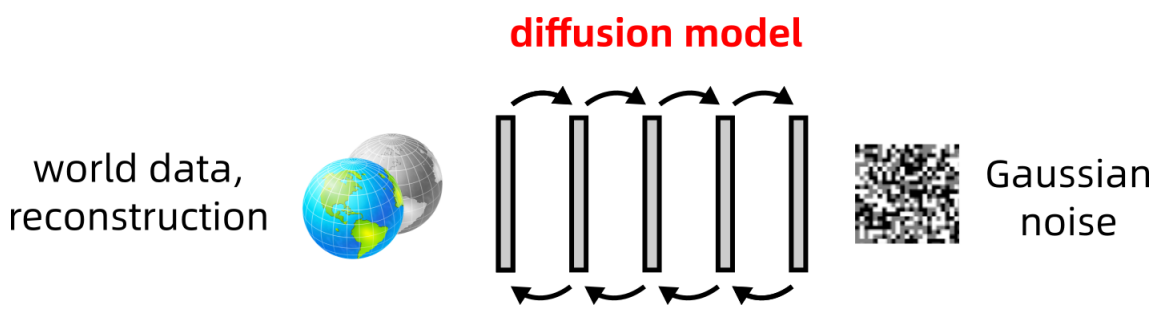
Of which, the VAE (variational auto-encoder) uses variational methods to find a probability distribution $q_\phi(z|x)$ that approximates the true 'posterior' distribution $q(z|x)$:



(4)

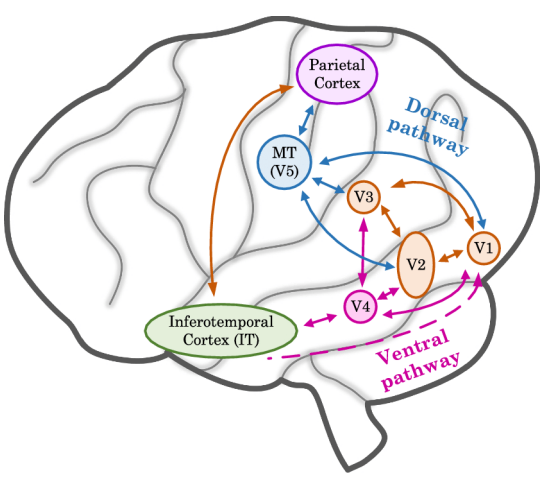
One variational inference algorithm recently proposed is SVGD (Stein variational gradient descent) which exploits efficiency in reproducing kernel Hilbert space.

Another generative model is the diffusion model, whose latent representation is distributed among its many layers:



(5)

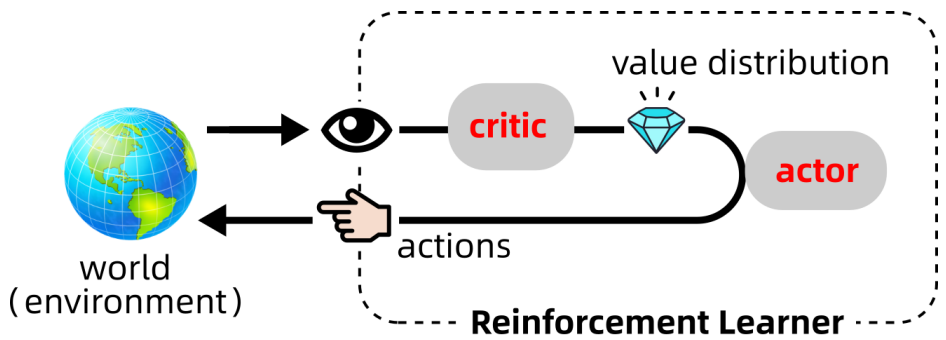
Interestingly, the human brain also has the structure of an auto-encoder.



(6)

What is AGI?

In my opinion, AGI should be developed under the framework of RL (reinforcement learning), which tries to find an optimal policy that acts in an environment, that maximizes the total reward over a time horizon:



(7)

State-of-the-art RL algorithms tend to have an actor-critic structure, that simultaneously learns value functions (denoted Q or V) and policy functions (denoted π).

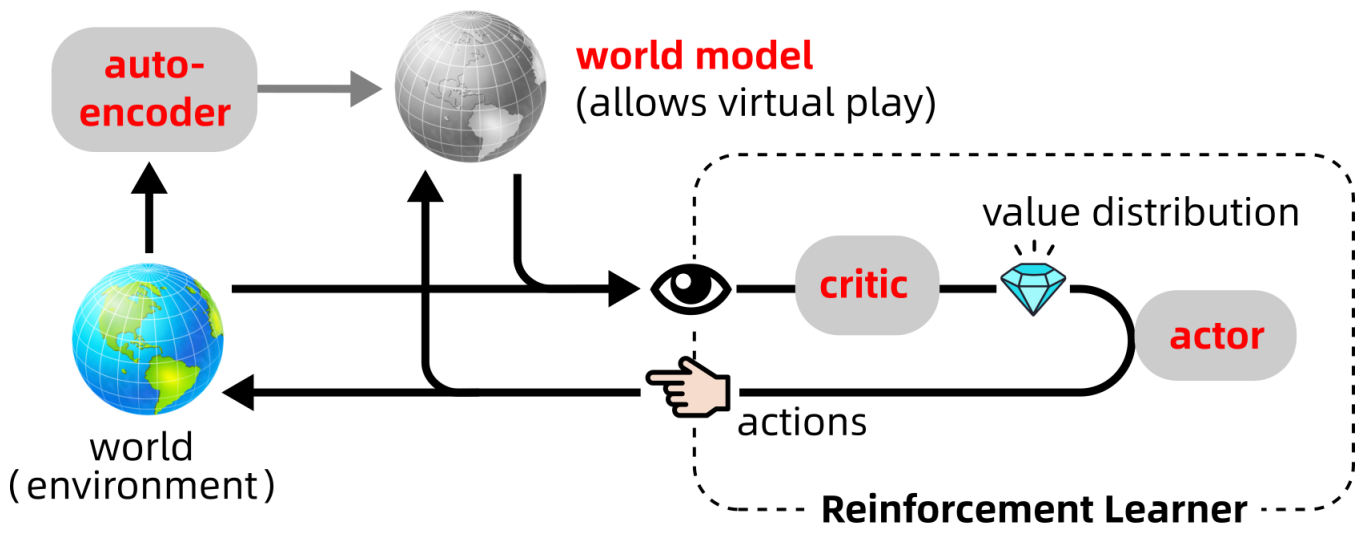
I tend to favor the SAC (soft actor-critic) algorithm for AGI because it has an elegant theoretical underpinning based on entropy maximization. This ensures that the resulting policy not only maximizes rewards, but also maximizes randomness so as to explore the environment adequately.

In its most general form, an RL algorithm tries to maximize the following Bellman objective:

$$\max_{\pi} \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim p(\cdot|s_t,a_t)}} \left[\sum_t \gamma^t R(s_t, a_t) \right] \tag{8}$$

The solution consists of two probability distributions: the policy π and the world model p.

The actor and critic seem distinct from the world model. The world model learns to predict the next state from the current state and action, ie. $p(s_{t+1}|s_t, a_t)$. As such, it is not influenced by rewards or "value judgements". We could say that the world model seeks only the "truth", whereas the actor-critic seeks values [1]. How might a world model help RL be more efficient? Perhaps by letting the actor-critic explore the world model virtually:



(9)

This is significant in physical environments, where physical actions are much more costly. Ironically, most current AI training environments are already virtual, so this may not bring about large improvements.

[1] but we can also let the world model predict rewards as well, in which case its role partly overlaps with the critic.

③

Text world

We want to find the "path of least resistance" to bootstrap an AGI. Riding on the success of LLMs, we may want to train AGIs on a purely text-based environment.

An auto-encoder such as BERT would read a text and produce a latent representation (which could be in natural language, as NL is also a form of symbolic logic in a general sense).

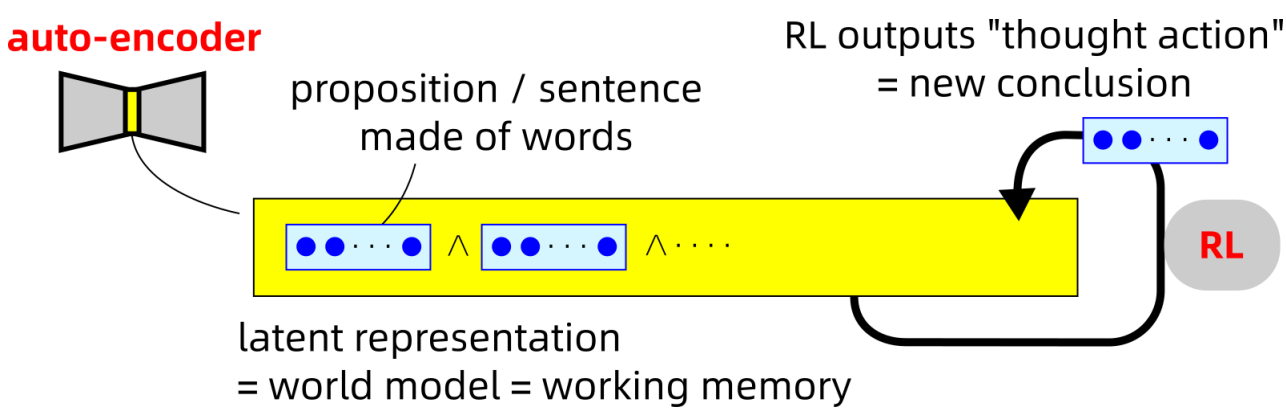
The latent representation is the "working memory" of the intelligent system and its representation is more efficient for inferences than raw input.

The actor-critic of RL works in the latent representation (ie, a model of the world, taking virtual actions). When a good virtual action is found, the algorithm would output an actual action.

RL as "thinking"

Why take so much trouble to combine RL and LLM? The benefit is that RL will find a way to optimize its internal "thinking" to achieve logic coherence. This will cure the problem of LLM hallucinations.

This is feasible if we include "thinking" as "mental actions" in RL:



(10)