Subject: Re: Politics and AI
Posted by Wayne Parham on Thu, 30 Jan 2025 22:42:16 GMT
View Forum Message <> Reply to Message

Truthfully, there are a lot of completing ideas in AI, and they aren't aligned with governments or politics or finance. The competing ideas are about things like how to build a network, its topologies, its layers and its flow.

I don't see the relevant issues surrounding the DeepSeek systems (and many other new AI developments) as being American versus Chinese. I don't see them as Capitalist versus Socialist. I don't even see them as proprietary versus open architecture.

In fact, "open source" doesn't really make sense for a neural network. Source code is programing rules and neural networks don't have them. So you can't have "open" source for a neural network because it doesn't have source code at all.

Neural networks are trained.

There is support code for interacting with the neural network. That can be made open source, but it's actually trivial. And you can distribute the weight values of the trained model - the actual "brain" of the network - but that's not an open architecture. It's a result of training, and distributing results is how proprietary systems are distributed.

So unless organizations publish all the training materials and the procedures they choose to load them and optimize them - what is published isn't at all "open source."

That's what you see in the DeepSeek models.

Not knocking them at all. Just saying that's how this works.

Techies aren't the ones talking about this primarily from a political or financial perspective. Other people - the talking heads - are the only ones making a fuss. Those are the ones that annoy me, basically because they either don't know what they're talking about or they're spinning things or most likely both.

Open architectures - both in hardware and software - just mean the internals are open for viewing and inspection. They aren't hidden or protected. There are licensing agreements that can be used to maintain intellectual property rights or to extend them in a limited fashion, or the open source stuff can be simply made free for anyone to use in any way. But it is distinguished from proprietary material by the nature of not being a "black box" that you can't see inside.

This whole mechanism - the description "open source" - has no meaning in a neural network. You don't program them with human-readable rules. Instead, you train them. They bias themselves, generating a lot of internal values called "weights." The collection of weight values isn't the same thing as source code, so you can't really call this "open source." It has no meaning here.

A neural network is definitely a black box.

That's one of the difficulties when working with neural networks. It's inherently hard to "look inside" a trained network to understand what sort of "reasoning" it applied to make a "decision."

So when a model is distributed - even openly - it's hard to see what you've got. You certainly cannot determine what the training was. That part is definitely not "open." Using "open source" to describe a neural network is non-sequitur. Makes no sense.

I don't really care about the semantics, normally, but it does make an interesting problem that I do care about. It is very hard to troubleshoot a neural network because it's not a state-engine or a rules-engine or any other kind of deterministic set of rules.

It also makes testing them difficult. You pretty much have to just run the model, try it out, and hope for the best. But the only things you can know for sure are the things you've asked it. You cannot know for sure how it will respond to things you haven't asked and you cannot even know that it will respond the same way twice.

As for the need for GPU and FPU chips made by companies like Nvidia and others, that's also an interesting technical problem that has understandable repercussions. Changes have been coming for a long time.

For precision mathematical calculations, one needs a good floating point processor or FPU. Lots of "traditional computers" have an integer processor that's separate from a floating point processor. Been that way since the early 1960s. Became really popular in the 1980s. Use a floating point processor for doing math, where fractional values are needed.

Similarly, once computers were used for graphics, it was helpful to have a separate graphics processor or GPU. It had its own memory and processor, because screen operations require a lot of memory as a "map" for what's displayed on the screen, and the GPU processor can do the math for making lines and curves and even 3D operations. That stuff needs an FPU, but when used specifically for graphics, it's called a GPU.

Segue to neural networks: The "weights" in the network have fractional values. So while a neural network could be built from the ground-up, using digital neurons, they are generally simulated using existing chips having a von Neumann architecture. When doing so, the floating point variety is chosen. That's why FPU and GPU chips have been selling like hotcakes in the AI world.

One thing that's bugged all of us for the last decade or so is that while the network weights are fractional, they don't benefit at all from high-precision. They work just fine with low-precision values. So a lot of calculation work is done by the floating point processor that isn't needed. It has been clear for quite some time that a specific type of low-precision FPU chip is desirable for neural networks. It's been brewing for years, and you see some of that poking out right now.

Most technical people using neural networks understand this, but "using what we've got" made the use of FPU and GPU processors convenient. That's why makers of FPU and GPU chips got a boost in sales for a while. It's also what has spawned the new breed of "AI chips." They just

have limited floating point processors. They don't need the precision for what they do.

So it isn't political. It's just an evolution of the state of the art.
