Chi Tian

- *Does numerical relativity still depend on advances in topological theory or is it possible to bypass the hairy mathematics to make feasible predictions/simulations?*
  If the advances in topological theory means advances in fundamental mathematics, fundamental mathematics did not seem to push the development of numerical relativity as much as numerical techniques did (reformulation of Einstein's equations, clever ways to set up meshes, smarter coordinate choices, etc.). It seems the complications of numerical relativity come from the complexity of the theory of GR itself; and thus it cannot be bypassed I think.
- *Are any of the tools and methods that were developed to solve problems specifically in numerical relativity applicable to problems in other domains?*
  I do not know of any applications right now. But one possibility way I can image cloud be generalizing the idea of BSSN to other unstable but constrained hyperbolic systems (not sure if other similar systems exist other than GR).
- *What are some common methods to solve Einstein's Equations numerically and what are the assumptions made in these methods?*
  Most common methods are based on the 3+1 decomposition. The 3+1 decomposition is casting Einstein's equation into another form, so no assumption is made (but is subjected to different numerical instabilities).
- *How are boundary conditions taken care of in numerical relativity research?*
  That largely depends on the problem you are interested. If you are interested in a cosmological scale problem, the boundary is usually periodic because this might be the best thing we can do to mimic our universe considering the real size of the universe. For an astrophysical problem (black holes, neutron stars), boundary conditions which can absorb any outgoing waves with minimum reflection might be the best choice (search Sommerfeld boundary condition).
- *What type of math & technology do you need to study numerical relativity?*
  Other than the math you need to understand GR, just some knowledge about PDEs.
- *What primarily limits how complex of a system you can simulate with a computer? Memory, storage, cpu cores, the time it takes to set it up, just general computation time, or something else?*
  It largely depends on how the problem you are interested scales when you add more memory, cpu or gpu cores. We say it scales WELL when the improvement you will get is proportional to the computational resources added to the simulation. For example, the large scale structure simulations usually scale well, so it will make use of almost all resources that are given, and give you a proportional speed-up. However, the astrophysical problem (black holes, neutron stars) usually does not scale well and thus it is not limited by the computational resources (but is limited by the computational method or the problem itself).
- *Is numerical relativity limited by accuracy in some cases? I'm not a physics person, but I feel like I've heard people talk about going to some ridiculously small time scale before Einstein's*

equations break. Can these early stages of the universe still be solved numerically or are the numerical models accurate enough?
It is always very numerically challenging to resolve both very large and very small scale at the same time, even with AMR. But one can always check the numerical convergence to make sure the numerical result is trustworthy or not. Even in the early universe there are some regimes and periods that can be modeled numerically. If you are asking at what scale that you can still trust GR itself, you have to estimate possible quantum corrections case by case (needs some knowledge of quantum field theory and quantum gravity); once they are large, you do not trust GR anymore (maybe also not the quantum gravity theory since there's no proved quantum gravity theory currently).

- *When articles refer to "solving" Einstein's equations, what exactly is being solved? For example, in the extreme case of two black holes falling into each other, what are we trying to understand?*
  It's the metric components ($g_{\mu\nu}$) that are solved; they describe the property of spacetime (they define distances in 3+1 dimensions), that's all you have to know to calculate trajectories of particles inside this spacetime. In the case of binary black holes, you are solving for the metric components for this system, for example. Once you know this, you will be able to calculate exactly the trajectories of test particles around the binary black hole system.

Shaden Smith

- *How do you ensure the physics of a problem is taken into account with Deep Learning methods?*
  A common technique not limited to deep learning is to encode it into the objective function that you are minimizing. You can do this in a "soft" way by adding a penalty to the objective function (called a regularization), or in a "hard" way by constraining the solution space to be physically feasible. For example, you might constrain solutions to be non-negative if you know it can't take negative values in nature.

  Here are some examples more specific to deep learning: 1901.06314, 2006.10503

- *It seems that a lot of Microsoft's Deep Learning research has gone into language recognition and fluency. To what extent are DL language models transferrable between different languages? Does language development begin from scratch for each new language?*
  This is an important question. A large portion of natural language processing research is done with English and a few other languages, and thus it's not always clear how transferable the techniques are. That being said, things like language translation are powered by deep learning now and some networks are designed with many languages in mind. Large training datasets often include multiple languages, to some extent at least.

  A bit more on language availability can be found in NLP.

- *How we can design a neural network to solve a particular problem? (For example, the number of nodes, what each node represents, if there exists a connection between two nodes, how we are going to change weight based on data, etc)*
  It's still something of an art, and for a given task the state-of-the-art is always changing! In short, I think it's easiest to think of it in a mathematical way. One simplified example:

  Say you want to map some items of set X (e.g., pixels of an image) into items of set Y (e.g., labels "cat", "dog", "car", ...). You could build a matrix L of dimension $|X| \times |Y|$, where each element (i, j) is the "signal" of mapping $X[i]$ to $Y[j]$. A new image m could be mapped to labels via $L^T * m$. The rows/columns of this matrix are the nodes of the neural network.

If you measure error in some way (cross entropy is common), you can use gradient descent to "learn" the matrix L. Training is ultimately just minimizing some objective function.

- *What do you recommend to someone who wants to learn more about machine learning/deep learning?*
  I'm sure there are some great courses available at your university! There are also many publicly available courses. Coursera has some fantastic resources, including a famous course.

  Google also provides free GPU compute time in Colab!

- *Do you know of any applications of AI in the healthcare field?*
  Many! There are a few articles at thegradient.pub/tag/healthcare. In short, there are many applications ranging from voice transcription, records analysis, diagnoses/treatment recommendation, radiography, and drug discovery. Naturally, it's important to emphasize that we don't want to just start training neural networks to replace doctors. These tools should augment and enhance healthcare.

- *At what point does more training data not help a model? I would imagine it'd be some asymptotic function where eventually more data isn't very helpful. Can a model be over-trained/are there any detriments from just too much data?*
  Your intuition is correct, and this is a tough problem (like most other problems touched on in these great questions). There are a few ideas at play here:

  Data *quality* is also super important. A trillion examples of the same concept will probably not result in a model that performs well, unless you are testing a very specific thing.

  Every model has some amount of capacity, which is a measure of capability or "amount of stuff it can learn". Capacity is an abstract idea, but for some problems we understand it roughly scales with the model size (in terms of number of parameters). There's a lot of complexity that goes to increasing model capacity though, and different domains have different "scaling laws".

  Ultimately, what makes a model useful is its ability to generalize and perform well on new data. As we train a model, we use a validation set of data that is not found in the training set. When quality on the validation set ceases to improve, we know future training may make the model too specific to the training data (called overfitting). There are also many techniques to help avoid overfitting, usually called regularization. Tikhonov regularization is probably the most common example.

  The asymptotic long-tail to training is actually usually very useful to an extent! Neural networks often take a long time to "settle" into a good local minimum, and so quality can continue to improve for a long time. We often slowly decay the learning rate (i.e., step size) of stochastic gradient descent to help this process of settling in.

- *Are the current AI algorithms at the limit of their efficiency/computational complexity? Basically, is there much work left to do tweaking the algorithms or is more data and better GPUs all that's left to get more out of AI?*
  Definitely not at their limit! This is the area that my group is most active. There's a ton of exciting and impactful research on more efficient network design. Some examples:

Model sparsity: encourage the model weights (such as the elements of the L matrix above) to be mostly 0. That allows us to skip storing and computing on certain elements. This is popular in a few areas such as sparse attention

Another form of sparsity is to only engage some parts of the model for each input: 2101.03961

Can we learn the same things with less data?

Can we split the model in certain ways, or change the problem such that we communicate across machines less often? For example, pipeline parallelism.

- *Are there AIs that can get around the "I am not a robot" captcha things?*
  Probably! One trend of captchas that you might have encountered is when you are told to select some images out of a grid (e.g., "select all of the traffic lights"). In addition to comparing your response against others to determine if you act humanlike, your responses are used for training data! Captchas like that are a very effective way to crowdsource labeled training data for things like self-driving cars.