Research Directions in Interpretability

Here are collected thoughts on research directions in interpretability I think would be impactful. A good starting point is to begin with application areas and end goals, then backpropagate into research that could potentially be fruitful for those areas and end goals. Therefore I will begin with downstream research areas I'm interested in, starting with general philosophy, and then move into more specific project ideas.

Philosophy

In general, I meant what I said in my statement of purpose. It's clear that large foundation models have learned useful things about the world. This is true for language, but I think vision and multimodal (and video, which I think will develop over the coming years!) is just as important, and possibly often easier from a research perspective because visual information is just so much more semantically rich from a (human) cognitive perspective than language information. Using prompts to access that information feels to me like an incredibly coarse method of information acquisition, and I think I want one of the central pushes of my PhD to be towards finding richer ways of accessing that information.

Motivations

This is useful for a number of reasons not necessarily transparent to computer scientists with a background entirely in deep learning. I'm coming from a biomedical data science / traditional machine learning / data science background. In that world, a big source of hesitation in using deep learning - besides the obvious bulkiness involved in training a big model over running `sklearn.decomposition.PCA.fit_transform()` and the lack of usefulness in low-data regimes - is that it's much more difficult to find interpretable features. I'm coming out of a very computational drug discovery startup, for instance. In that world, language models trained on RNA are starting to make their appearance (for instance, https://www.profluent.bio/ is active in that space, and my company has started to play with SpliceBERT). Biologists and chemists can't access the information in these models from prompting, since the whole point is that the model has learned something rich about how RNA sequences are organized in a language that they don't know. However, these models are clearly learning interesting and nontrivial information about biology, as evidenced by the downstream tasks (branchpoint prediction, splice site prediction, variant effect inference, nucleotide clustering...) they were able to get surprisingly good results on. A microscope to peer at this information directly would be incredibly useful for scientists.

In computer vision, my girlfriend's company is another great example. They are using straightforward classification architectures - think EfficientNets and ResNets - to classify veterinary X-rays. One of their big challenges, particularly in moving into human data, is that doctors need to be able to trust the model. In order to trust the model, there needs to be a great deal of transparency in how the model is making its decisions. They've been able to take advantage of pixel attribution methods, e.g., SHAP, LIME, and GradCAM, to get some insight. But these methods don't tell the whole story.

Sometimes (e.g., GradCAM, according to her) don't have easily-accessible implementations that can hook into existing codebases. For cases where the problem is wild-west levels of implementation for useful techniques, I could also see myself at some collecting all of these methods into a single codebase with a methods paper attached, and then trying to get that codebase PR'd into a major library on huggingface or something torch-adjacent. This worked well during my master's in the graph world for graspologic, during which time a labmate PR'd his work into scipy. A common problem in academia is that useful methods are often locked up in a single unmaintained codebase rather than disseminated, and I'd love to both avoid and help fix that.

Here are a few areas in which better interpretability would be very useful:

- customer-facing medical applications
- legal applications
- self-driving
- drug discovery
- all of science
- people who want to know why their model isn't training

Basically, anywhere in which a central push is to *learn something* rather than *do something*, and any application area in which there's a high cost for mistakes.

Interests

With the above impact motivations in mind, there are also things that just *strike my fancy*, and which I think are interesting and fun to think about. Good projects, in my opinion, will also contain elements of this. A somewhat representative but not at all exhaustive list:

- Constrained optimization methods which create model edits that respect the data manifold. Riemannian optimization methods, for instance, or using spatial derivatives to move around on manifolds. Methods which take advantage of the tangent space and local linearity.
- In the same vein, activation steering and vector arithmetic in the latent space. Anything that takes advantage of the geometric properties of algebraic spaces to perform some useful and

unintuitive task.
- The whole set of LoRA/DoRA/ReLoRA etc. PEFT methods are interesting for me largely because I have a geometric intuition for low-rank updates -- e.g., making linear updates which are constrained to a low-dimensional hyperplane on the original subspace.
- I have a side-interest in explorable explanations and interactive documents, e.g., distill.pub, betterexplained, anything by brett victor, anything by flowingdata.
- Petar Veličković and friends have been working on mathematical frameworks for categorical deep learning recently, extending geometric deep learning to use the tools of category theory to attempt to build a language of architectures. It looks interesting; I don't know at first glance how it would be useful in interpretability.
- My areas of mathematical interest include causal inference (e.g., judea pearl's work), high-dimensional statistical inference, information theory, information geometry, and differential geometry. My intuition is that all of these areas contain useful ideas for interpretability research.

Research Directions

With all of the above in mind, here is an equally non-exhaustive list of directions and/or projects. Some of the below is more fleshed out than others; I basically just threw something down whenever I had an idea. My hope is that we (e.g., me and david) can quickly whittle down ideas that aren't feasible and ideas which don't have clear impact, incorporate david's and then expand into concrete projects with clear goals.

- Papers which explore the same idea, but under a lot of different model architectures, implementation details, etc. Finding a set of golden retriever ear detecting neurons for a ConvNext trained on exactly ImageNet using exactly AdamW at exactly a learning rate of 1e-4 is much more interesting if it's also true for vision transformers and whatever else, using different optimizers and hyperparameters. Which results generalize most strongly?
    - Example: efforts to transfer interp results in GANs to diffusion models, efforts to transfer results in transformers to mamba...
    - addendum to this: to what extent do the same concepts apply when we change modalities? Can we discover ideas in interpretability in text, and do they also apply to vision, video, audio...
- Volumetric and video models are cool and largely underexplored right now, primarily (as far as I can tell) because of compute constraints. Possibly soon, particularly with video, this will be interesting to explore (although moving from $O(n^2)$ to $O(n^3)$ is a big jump). A paper which applies existing image-interpretability techniques to video models could be fun to write.
- For LLMs: Can we find or create a 'factuality' or 'confidence' direction in activation space, in a way that lets us add that information into model responses?

    - for instance, you ask why some code doesn't work, and the LLM responds, and above the response there is something that says "the model is 60% sure that this answer is correct".
    - The loss function for this particular example would involve measuring the empirical distribution of the actual amount of times it was correct, and then looking at the difference.
- What kinds of interpretability questions can we ask about the kinds of updates PEFT methods are constrained to make under the low-rank regime? Is the difference between a low rank update and a full rank update just a quantitative change, or is there a qualitative difference in the kinds of updates that can be made?
    - In the same vein: What are we actually doing geometrically to weight matrices by adding low-rank information into them? I suppose (via geometric intuition rather than formal proof, so possibly wrong) that higher-magnitude low-rank updates causes weight matrices to be closer to low rank themselves. Do we care about this?
- Something that would be very useful if solved is "densifying" context windows. Say you're at the edge of a very large context window. Is there a way to map the set of token embeddings you're conditioning on to a lower rank matrix, or to a single information-rich vector, or just anything more compressed, so that you can still condition on the context without running out of space? This is not necessarily interpretability research, but it is interesting.
- the attention operation, viewed as a linear transformation, has access to at most a hyperplane of dimension `m` when operating on an embedding matrix where `m` is the number of tokens.
    - Multi-headed attention takes subsets of dimensions, and moves the `m` vectors around within those subsets. Then they are concatenated.
    - is this linear? The softmax in the attention matrix isn't, but can you pull out all the softmaxes?
    - If the above is true, can you shuffle things around algebraically so that the splitting, attention-updating, and concatenating all happen in a single matrix multiplication (block attention matrices * head concatenations), since it's all one linear transformation?
    - if you can do the above, is there a technique that makes doing it this way faster, since then you're taking advantage of optimized matrix multiplication on the gpu as much as possible?
- My competitive advantage is a strong geometric intuition for linear algebra and being able to take ideas from spectral graph theory. How many ideas that I know well from the graph stats textbook could possibly be useful in deep learning for interpretability research?
    - MASE and OMNI embeddings - joint representation learning for multiple graphs (project nodes from different graphs into the same subspace) - possibly useful for comparing attention heads
    - I wonder if the spectral embedding of the Laplacians of attention matrices tells us anything
    - graph matching - maybe not useful? I could potentially see an experiment where we look at whether attention heads become harder or easier to graph-match (e.g., the loss function gets bigger or smaller) in deeper layers.

    - random graph models and their properties, e.g., erdos-renyi, SBMs, RDPGs, etc - probably not useful here. Maybe as a statistical framework. Not sure. Don't see it right now.
- Mixture of Depths is an interesting paper which commits to the idea: "why apply every transformer block to every token when presumably some tokens need more computation than others?". An interesting interpretability follow-up to this would be: Which tokens ended up needing more compute than others, and what does that tell us about what is easy and difficult to compute for a language model?