Core Concepts in Mechanistic Interpretability

Neural Networks as Grown Systems

Unlike traditional software where we write explicit instructions, neural networks develop through a growth-like process. We provide the architecture (like designing a garden trellis) and the training objective (like sunlight), but the actual capabilities emerge through training. For example, while we can write code to detect edges in an image, we can't directly program the ability to recognize cats in any position or lighting. Neural networks develop this ability by growing complex internal representations through exposure to millions of examples. This is why we often can't directly explain how they make specific decisions - we cultivated the capability rather than engineered it.

Features and Circuits

Neural networks develop internal "features" (detectors for specific concepts) that connect into "circuits" (algorithms made of connected features). A practical example: in image recognition, a "car detector" circuit might combine simpler features in a specific recipe - it looks for window-detecting features above, wheel-detecting features below, and chrome/body features in the middle. This hierarchical organization is similar to how human-written software might break down a complex task into simpler, reusable components, but these patterns emerge naturally during training rather than being explicitly programmed.

Universal Building Blocks

Different neural networks, and even biological brains, independently develop remarkably similar fundamental features. In computer vision, every successful model develops curve detectors and edge detectors in its early layers, just like biological visual systems. This "convergent evolution" suggests these features are optimal solutions for processing visual information - much like how both bats and birds independently evolved wings for flight. When training new vision models, you can reliably expect to find these universal building blocks, which helps in understanding and debugging them.

Linear Representations

Neural networks represent concepts as directions in high-dimensional space, enabling mathematical operations on ideas. The classic example is word embeddings, where "king - man + woman = queen" works because gender and royalty are represented as directions that can be added or subtracted. This extends beyond words - in multimodal models, you might find that "image of dog - fur + scales = image of lizard" works similarly. This linear structure makes it possible to manipulate and combine concepts in intuitive ways, similar to how we might mix colors or musical notes.

Superposition

Neural networks can store more concepts than they have dimensions through a mathematical phenomenon similar to radio multiplexing. For instance, a network with 500 dimensions might need to represent thousands of concepts (like all countries, parts of speech, abstract ideas, etc.). It accomplishes this by taking advantage of sparsity - most concepts aren't active simultaneously. Just as multiple radio stations can share the same frequency band by broadcasting at different times, neural networks can encode multiple features in the same "space" by ensuring they rarely activate together. This is like having a single physical bookshelf that can somehow store more books than it has space for by cleverly organizing them.

Monosemanticity vs Polysemanticity

Most neurons in neural networks are "polysemantic" - they respond to multiple, often unrelated concepts, making them hard to interpret. For example, a single neuron might activate for both pictures of dogs and mathematical

equations about derivatives. Recent breakthroughs using "sparse autoencoders" help untangle these mixed signals into clear, interpretable features - like separating a jumbled radio signal into distinct channels. This is crucial for understanding and auditing AI systems, as it lets us see the individual concepts the model is using rather than their scrambled combinations.

Multimodal Features

Advanced neural networks develop features that recognize abstract concepts across different types of input. For instance, a "security vulnerability" feature might activate for both code containing buffer overflows AND images of someone clicking through SSL certificate warnings. A "backdoor" feature activates for both malicious code snippets AND pictures of hidden cameras in devices. This shows how models develop genuine conceptual understanding rather than just surface-pattern matching, similar to how humans can recognize "deception" in both written stories and visual body language.

Mechanistic Interpretability vs Traditional Studies

Studying neural networks offers unique advantages over studying biological brains - we have perfect recording of all neurons, can modify connections at will, and can run unlimited experiments. However, even with these advantages, understanding neural networks remains challenging. Imagine debugging a complex program where you can see every variable's value and modify any line of code, but the program was written in a completely alien programming language - that's similar to the challenge of mechanistic interpretability. This suggests that new tools and approaches may be needed to fully understand both artificial and biological intelligence.

Dictionary Learning

Neural networks often store information in mixed-up ways, with single neurons responding to multiple unrelated concepts. Dictionary learning helps us untangle these mixed signals into clear, interpretable features. The technique works like an automated translation system - it learns to convert complex patterns of neural activity into simple, single-concept features. For example, when applied to large language models like Claude, it can separate a neuron that responds to both cars and dogs into two distinct features, each responding to just one concept. This is similar to how audio software can separate a recording of multiple instruments into individual tracks, making it possible to study and understand each component independently. The breakthrough of this approach is that we don't need to tell it what features to look for - they emerge naturally from the data.