Inside the Black Box : A Primer on Explainabili & Interpretability

Imagine peering inside an LLM to see how it processes inputs, transforms features, and generates responses. This level of transparency is the goal of interpretability and explainability in AI.

SIDDHANT RAI

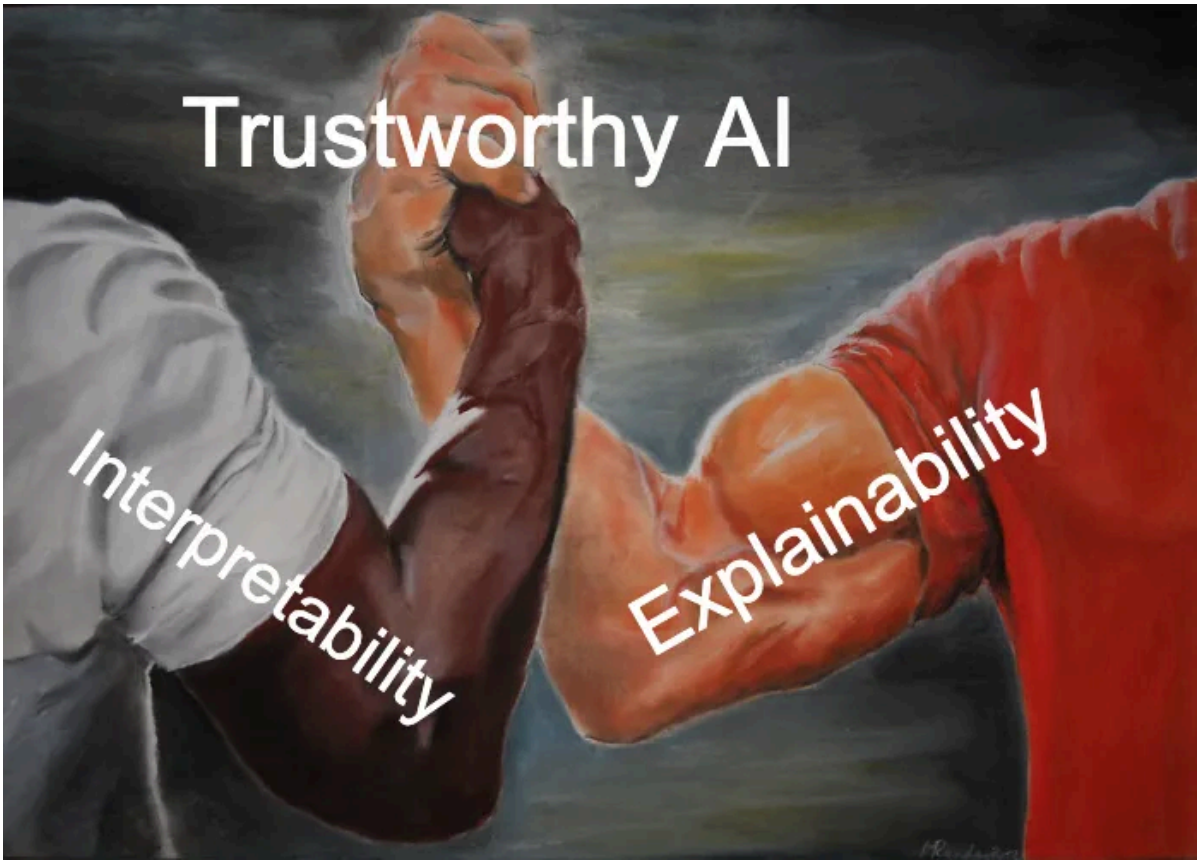
MAY 20, 2025

Table of content

1. Introduction

As AI systems become increasingly complex, especially with models like LLMs, understanding their decision-making processes becomes crucial. Two key concept

this context are **interpretability** and **explainability**. While they are often used interchangeably, they address different aspects of model transparency.



Interpretability: Understanding the Model's Inner Workings

Interpretability refers to the extent to which a human can comprehend the intern mechanics of a model. It's about understanding how the model processes inputs t produce outputs, focusing on the model's parameters (θ) and structure. For instanc a linear regression model, interpretability is straightforward, we can examine the coefficients to understand how each feature influences the outcome. However, as models become more complex, like deep neural networks, achieving interpretabili becomes more challenging due to intricate architectures and non-linear transformations.

Explainability: Deciphering the Model's Decisions

Explainability, on the other hand, focuses on the model's behaviour concerning specific inputs. It's about understanding *why* the model made a particular decisior

a given data point, essentially modeling P(y|x). This involves analyzing the relation between input features and the model's output, often using post-hoc methods to provide insights into the decision-making process. For example, when a model classifies an email as spam, explainability seeks to identify which features (like cer keywords or sender information) led to that classification.

The Interplay Between Interpretability and Explainabilit

While interpretability and explainability address different facets of model transparency, they are inherently interconnected. Understanding a model's intern mechanics (interpretability) can aid in explaining its decisions (explainability), and versa. However, it's important to note that a model can be interpretable without b easily explainable and vice versa.

Consider a car. Interpretability is akin to understanding how the engine works. The mecha the fuel injection, the transmission. Explainability is like knowing why the car took a spec route, based on traffic, weather, or driver preference.

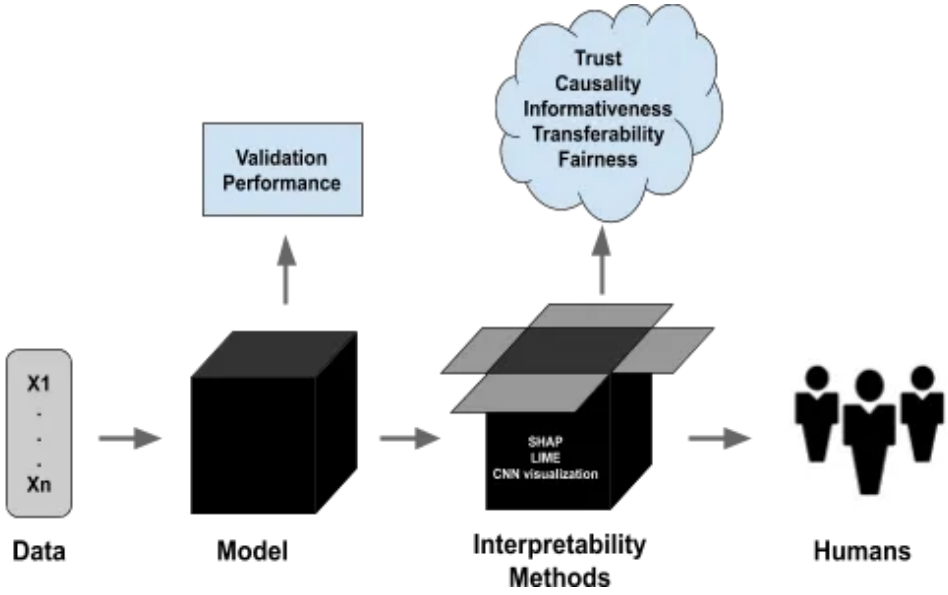
In practice, achieving both high interpretability and explainability is challenging, especially with complex models. Therefore, selecting the appropriate balance betv the two depends on the specific application and the need for transparency.

2. The Ideal Scenario for Interpretabili

Let’s imagine a setup where a machine learning model doesn’t just give you a resu walks you through the reasoning. The kind of transparency where if you ask, “Wh the model reject this loan?” you get something like:

- Income is below threshold → risk goes up
- EMI defaults in the last 3 months → risk goes up
- Long employment history → risk goes down
- Final verdict: high risk

This is the ideal world of interpretable models. Each input feature has a direct anc understandable impact on the output. There's no guessing, no second-guessing. T model structure aligns with how we reason and audit decisions.



What makes this ideal powerful?

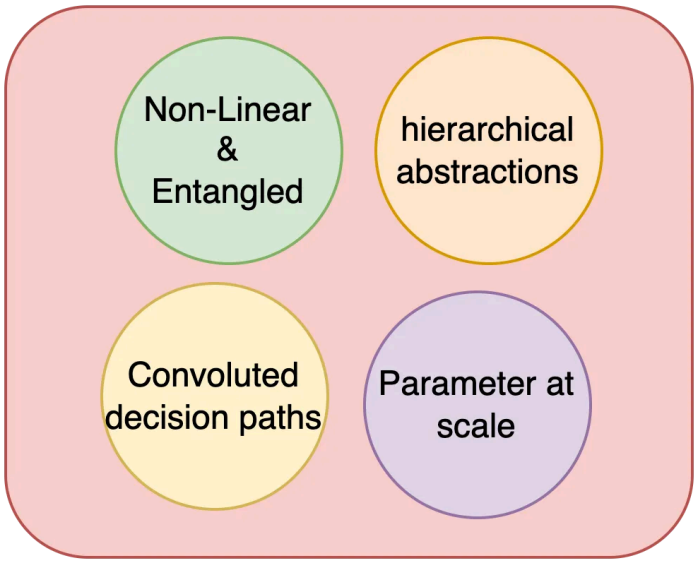
- You can trace every prediction like following a recipe
- Each part of the model has a clear, isolated effect
- There are no entangled dependencies across hundreds of layers
- You don't need special tools to "interpret" it, the model is readable by design

Models like linear regression, decision trees, and rule-based systems live in this id world. They are not just simpler; they're **built to explain themselves**.

Why This Ideal Works for Small Models but Breaks at Sc

So if this clarity is so useful, why don't we use it everywhere?

Because as tasks become more complex like language modelling, image recognitio reasoning across long contexts, the simple models hit a wall. And that's when we t to deeper, larger architectures. But here's the problem: **interpretability doesn't sc with model size**.



Here's what starts to fall apart:

1. **Feature relationships become non-linear and entangled**
   In small models, each feature works independently or in clean combinations. deep models, one feature might only matter when two others are present. *You say "feature X increased risk" instead, "feature X activated in presence of Y and late concept Z."*
2. **The model invents its own abstractions**
   Deep models don't just learn what you give them. They build intermediate features layer after layer that are useful for the task, but meaningless to huma These latent features aren't "age" or "income" anymore; they're high-dimensi blobs with no name.
3. **No clear decision path exists**
   A decision tree follows a visible path. A neural network? It passes data throug dozens of layers, non-linearities, and projections. The final output is a result o everything working together, not a traceable chain of rules.
4. **Too many parameters to inspect**
   Models like LLMs operate at a scale where even if individual neurons *could* be interpreted, there are millions (or billions) of them. You don't just lose transparency; you lose the ability to look at the system in its entirety.

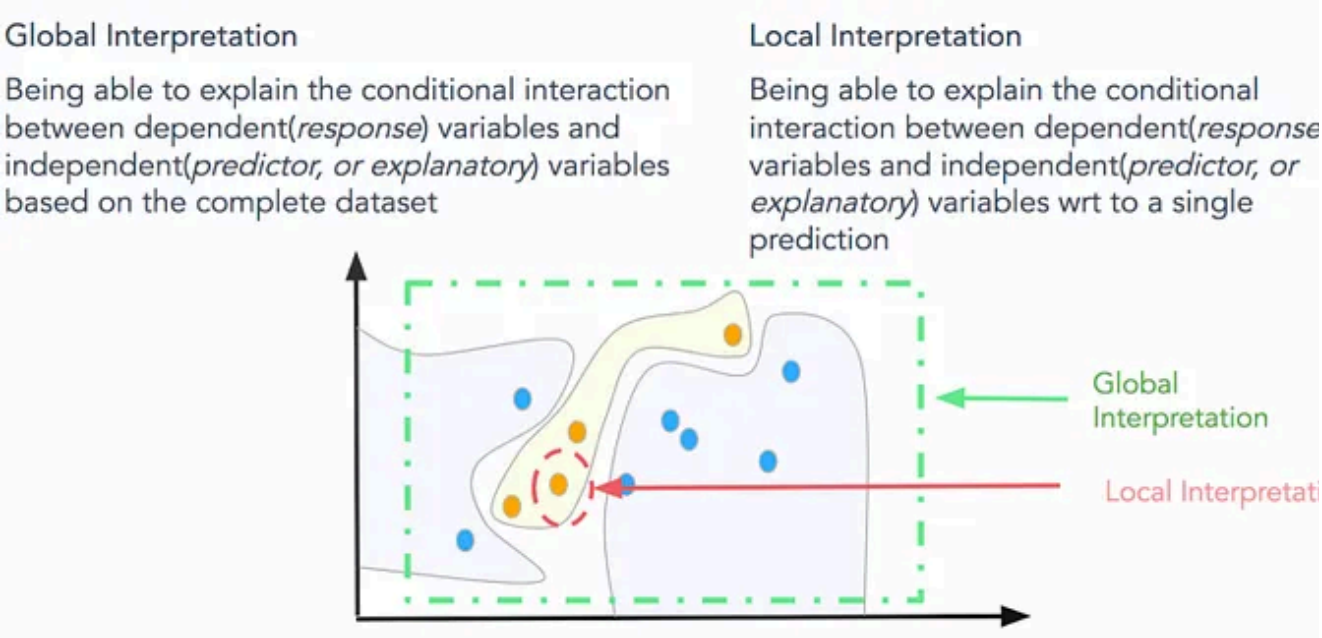
So while small models offer explanations as a side effect of their design, large moc need **external effort** to become even *marginally interpretable*. The explanation is no longer part of the model, it's a separate research challenge.

3. Interpretability Through the Lens of Model Evolution

Let's slow down and reflect on what interpretability *actually* looks like across diffe model families. As we shift from linear models to LLMs, the notion of "interpretability" transforms from being an inherent, structural trait to something need to reverse-engineer. It's like comparing a clock with exposed gears to a seale quantum chip. You don't just lose the ability to trace decisions; you lose the *very language* in which decisions were originally expressed.

3.1 Local vs Global Interpretability

Every single model agnostic interpretability technique either falls under local or g interpretability. Model could be called locally interpretable if only certain section model could be understood whereas if you could infer and understand the entire n space then it could be categorized as globally interpretable model. Generally, mod which are smaller and have non-overlapping features (as in naive-bayes) are consic to be globally interpretable, whereas complex models are locally interpreatble.



https://www.comet.com/site/blog/model-interpretability-part-1-the-importance-and-approaches/

This explanation could also be thought of from the perspective of data itself, whic according to me should fall in category of explainability, but, is mostly talked in junction with interpretability. Anyways, basically Global explainability aims to pr a **bird's-eye view** of how a model works **across the entire dataset**. It answers quest like *"Important feature to the model on average", whereas* Local explainability focuses understanding **individual predictions**. It answers "*Why did the model predict X for t particular instance?".*

3.2 Interpretability in Traditional Machine Learning Moc

These are models where interpretability is not just possible, it's **built in**.

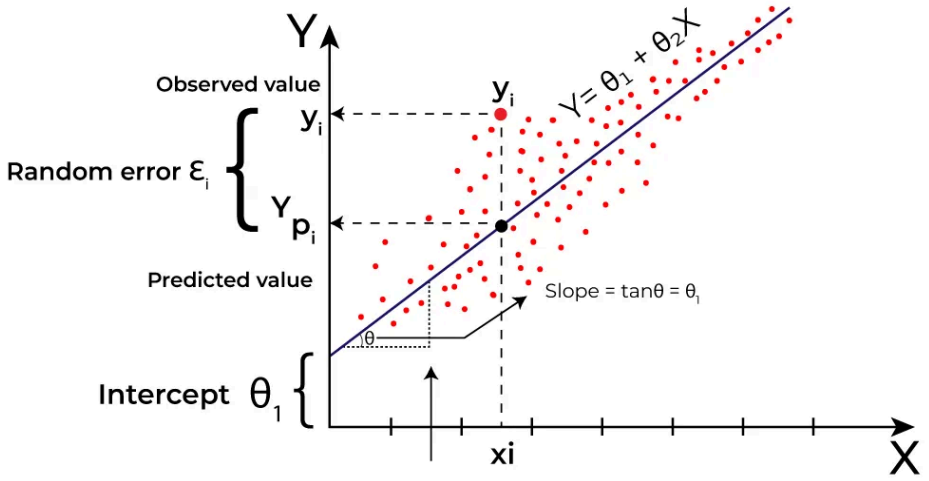
Why?

Because traditional ML models map inputs to outputs using **explicit**, **human-aligı** and **mathematically traceable** rules.

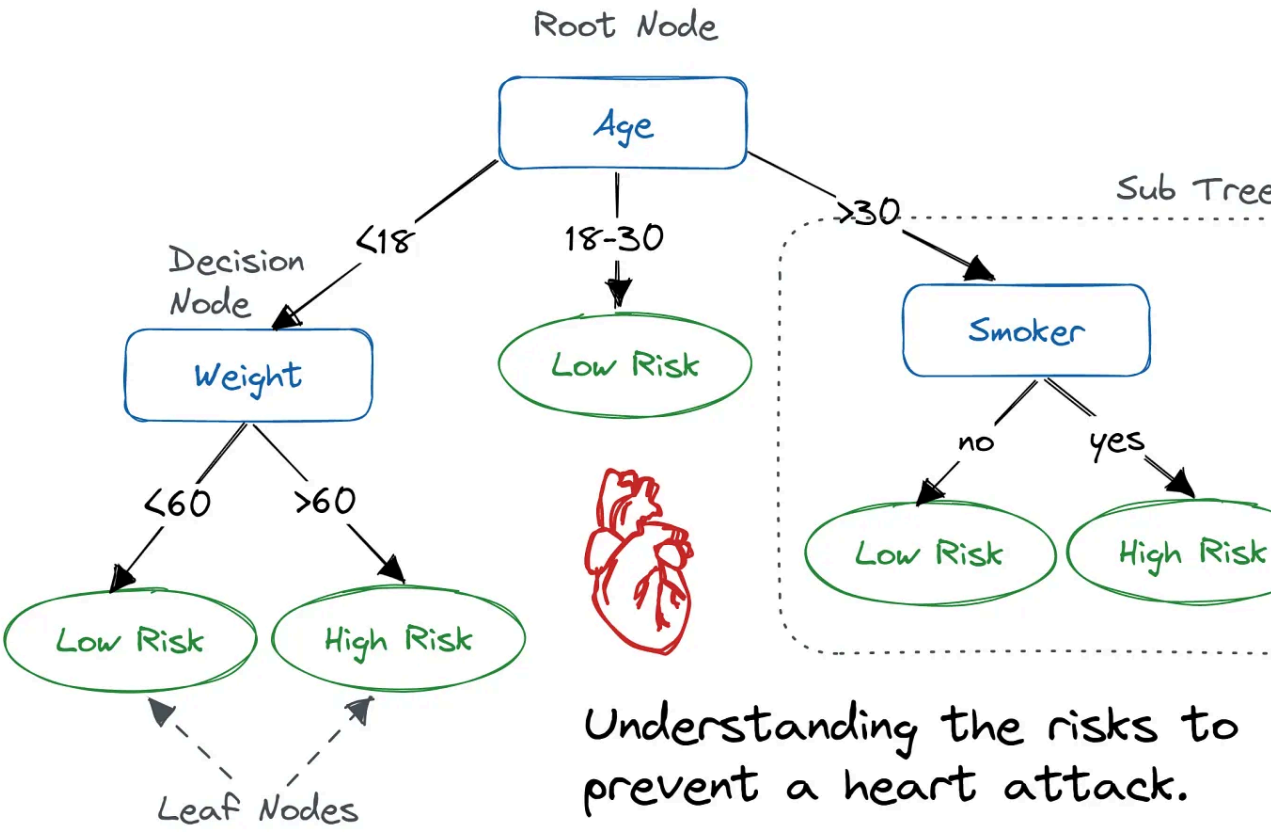
Let's break that down.

- **Linear/Logistic Regression**: These models operate on one principle, weighted summation. Each feature has a coefficient that literally says: "if you increase t feature by 1, this is how much the outcome will change." The output is nothin

but an algebraic expression. There are no hidden states, no complex representations. You can look at the weight matrix and know exactly what the model is doing.



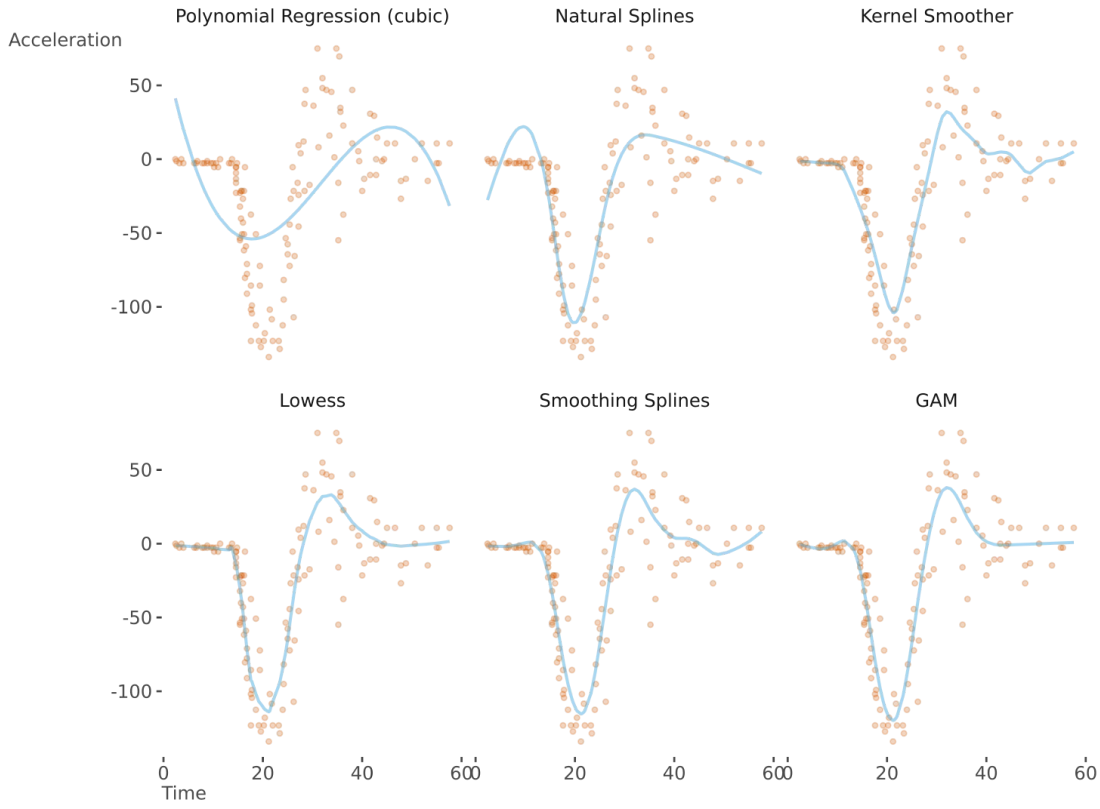
- **Decision Trees and Rule-Based Models**: These give you full control-flow visibility. At each split, the model checks a feature and makes a binary decisio The final decision is just the result of walking down a path, no black box, no hidden abstraction. You could even print the entire model logic as an if-else statement.



https://www.datacamp.com/tutorial/decision-tree-classification-python

Think “expert systems.” Every decision the model makes is governed by hand-crafted or learned rules; “if blood_pressure > X AND cholesterol > Y → risk = high.” This is how humans often reason themselves, so there's immediate alignment between model logic and human understanding.

- **GAMs (Generalized Additive Models)**: These strike a middle ground. Instead each feature having a fixed weight, you let it have a simple curve. So age migh influence prediction non-linearly, but each feature still contributes independe You can visualize these curves and say: “Oh, the model thinks age 30–40 is optimal.”



- **Monotonic Constraints**: Sometimes domain knowledge tells us, "more of this should always mean more of that." You can encode this constraint into the mo (e.g., loan default risk increases with debt). These constraints don't just impro generalization, they reinforce *trust* by making model behavior predictable.

Why it's interpretable: Because the logic is exposed, modular, and aligned with hc we reason both mathematically and cognitively. *The structure is the explanation. Her the easiest way to make an interpretable system is to follow occams razor.*

3.3 Interpretability in Deep Learning Models

Once we step beyond traditional ML models, we enter the deep learning zone and is where interpretability begins to dissolve.

You may still see inputs and outputs, but the "why" becomes murky. It's like watcl a magic trick without knowing where to look, hence, too much is happening behir the curtain.

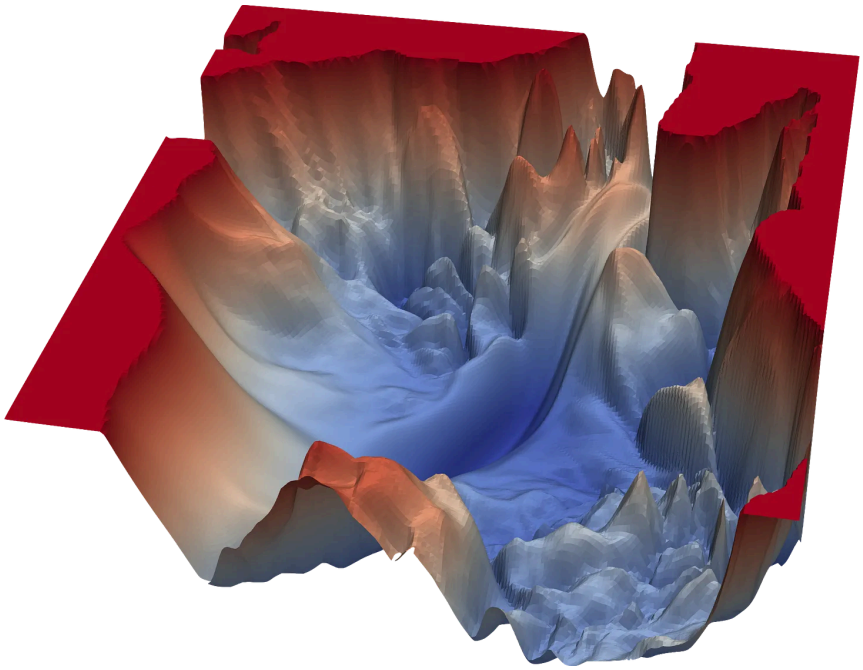
Let's explore **why deep learning models are fundamentally harder to interpret**, ar what that means across different architectures.

Why Deep Models Are So Hard to Interpret

1. High Complexity

A deep model isn't just a function; it's a *stack of hundreds* of them. You're looking a layers upon layers of transformations, each one passing nonlinear outputs to the n With thousands or millions of parameters, the model no longer learns "rules", it le **representational flows**.

Unlike trees or rulesets, where each part plays a specific logical role, deep models distribute decision-making across layers. You can't just isolate a single neuron and "this one handles fraud detection" as its effect only makes sense in coordination w the rest.



2. Non-linearity and Feature Abstraction

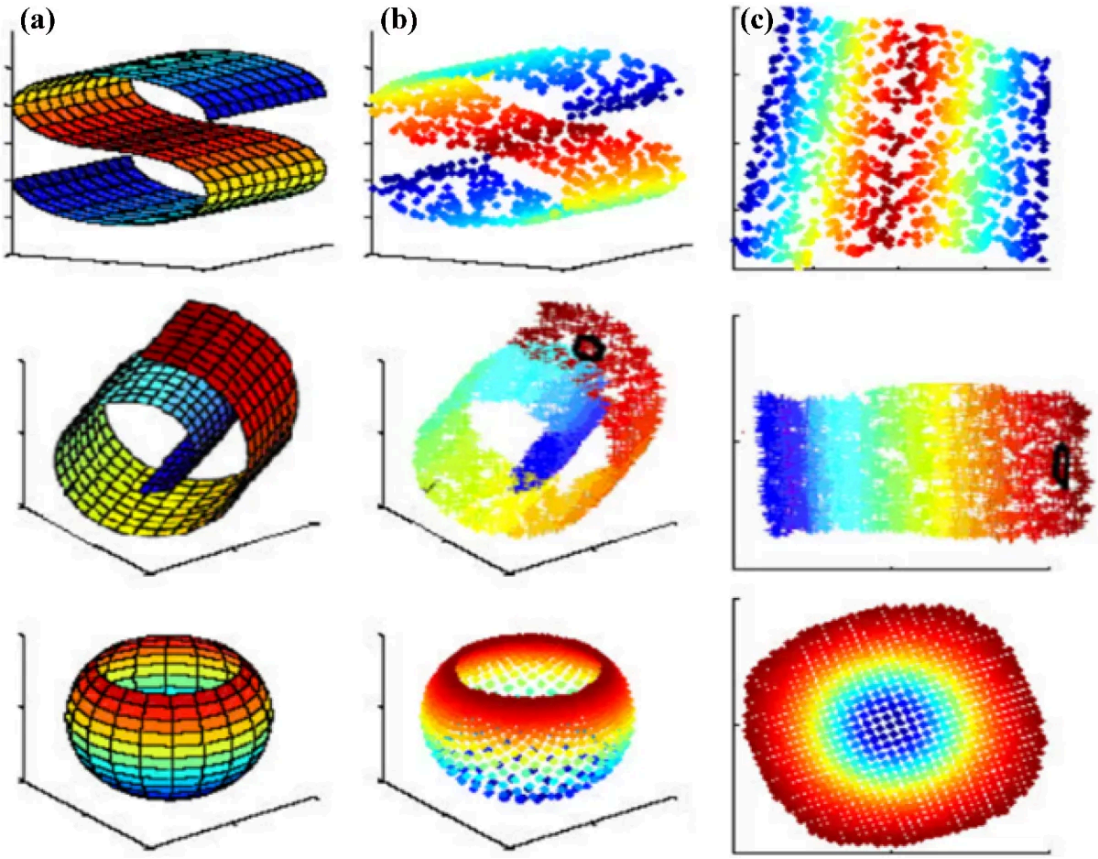
The more non-linearity you stack (e.g., with ReLUs, GELUs, or softmaxes), the mo your model behaves in abstract ways.

The model doesn't just learn "if-then" relationships anymore, it builds its own int vocabulary: neurons activate for shapes, textures, sentiment, rhythm, and even grammars... but only internally. These abstractions often don't align with human concepts, which makes explaining them extremely hard.

3. Lack of Explicit Structure

In traditional models, structure is visible and intuitive. But deep models don't mal decisions based on conditions, they operate through numerical gradients. There's evident tree of logic to follow, no coefficients to inspect. Even if two inputs are ve similar, a deep net might treat them completely differently based on the subtleties internal activations. This lack of alignment between model mechanics and human logic is one of the root causes of their opacity.

4. Curse of Dimensionality



Deep models often work in very high-dimensional input and latent spaces (especia in vision and language).

Imagine you're trying to visualize how a 1000-dimensional vector moves when you tweak a word in a sentence. There's no spatial intuition for that. So even minor inp perturbations can lead to drastic, unintuitive changes in output. For sure we can b the dimensions down through neighbour preserving algos like t-SNE and UMAP, still it destroys a lot of feature space and alignment.

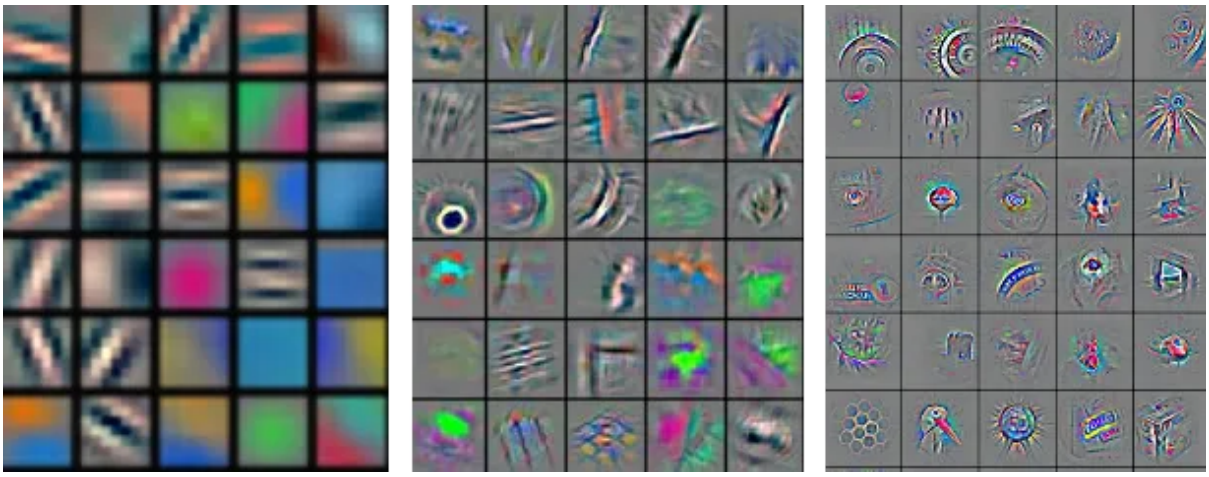
Adversarial attacks are a great example of this: change one pixel (imperceptible to human), and suddenly the image gets classified as “gibbon with sunglasses.”

Architecture-Specific Interpretability: A Closer Look

Convolutional Neural Networks (CNNs)

CNNs were among the first architectures where interpretability got serious attent (especially in vision), as the underlying mechanism was composed of feature learni kernels, which could be visualized to see edges, corners etc. in early layers and mo abstract features in latter layers.

- **Feature Visualization**: Each convolutional layer learns filters that detect patte Early filters capture edges or color gradients. Mid-level filters detect textures shapes. Deep filters detect object parts or entire objects (like dog ears or whee



- **Why this matters**: You can look at activation maps and say, “this filter fires or texture”, which gives you a handle on what the network “sees.”

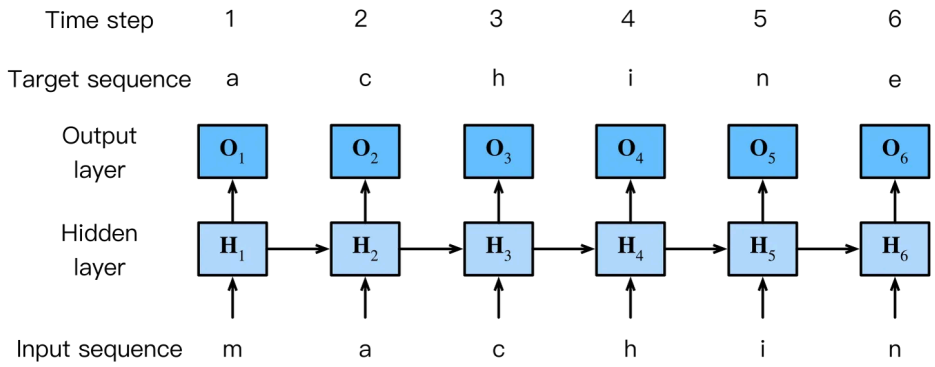
But...

- **Interpretability gets harder with depth**: As layers go deeper, filters stop responding to clear patterns. Instead, they combine dozens of vague signals. Y no longer see “this filter detects ears” instead, it’s “this filter lights up when fe other vague filters light up together.”
- **Feature Map Overlap**: Many filters learn overlapping or redundant features, making it hard to assign clear meaning to individual channels.

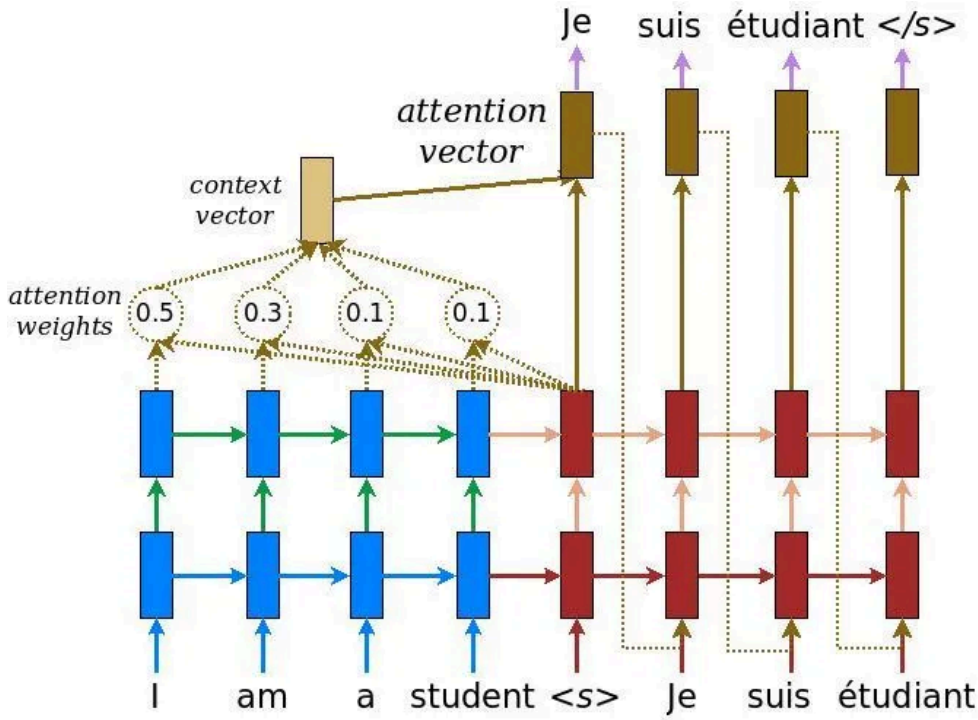
- **Visualization as approximation**: You can visualize activations, but it doesn't always tell you what's *important* for the decision, just what's present.

Recurrent Neural Networks (RNNs)

RNNs, designed for sequential data like text or speech, introduce time into the equation. That brings interpretability challenges of its own.

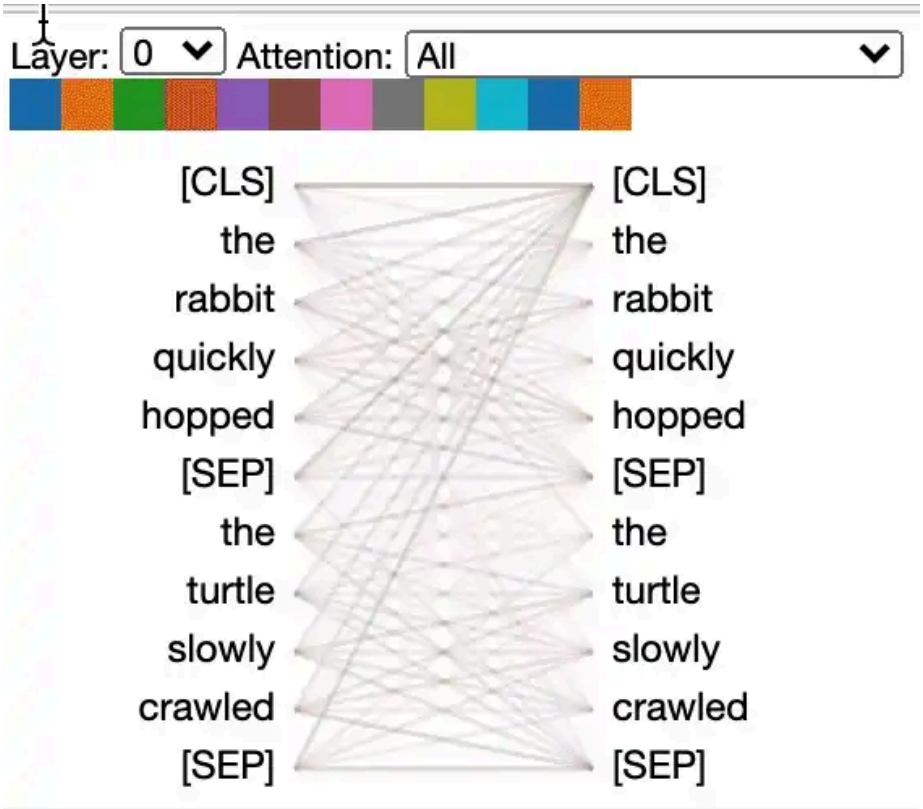


- **Hidden States are Temporal Summaries**: RNNs maintain a hidden state vecto that updates over time. But this vector is a compressed summary of all past to hence it's not easy to unpack.
- **What makes this hard?**: You don't get token-by-token visibility. You get one l blob (the hidden state) that stores everything. So understanding what the mod "remembers" or "forgets" is non-trivial.
- **Interpretability Boost: Attention Mechanisms**: Attention lets models "look b at past tokens directly instead of relying entirely on hidden state memory. Thi allows us to visualize which past words were considered relevant for the curre output (especially in decoding phase).



Transformers and the Self-Attention Mechanism

Enter transformers, the base architecture behind LLMs and things get even more abstract as well as complex.



- **Self-Attention**: Every token can attend to every other token. You get a dense matrix of attention scores across tokens. This forms the basis of the model's ability to contextualize meaning.
- **Is attention interpretable?** Kind of. You can visualize which tokens attend to which in early layers, this can resemble patterns (e.g., subjects linking to verbs But attention scores are not always aligned with *importance*. They show *interac* not *causal influence (this we will cover in further section)*.
- **Multi-head Attention**: The complexity scales again. Each attention head learn track different types of information like syntax, alignment, order, entities. But heads can overlap in function, cancel each other out, or specialize in obscure/abstracted ways.

In summary,

Interpretability in deep learning is a spectrum:

- For shallow CNNs, you can peek into the filters.
- For deeper models, you need to simulate, visualize, or guess.
- For transformers, you often have to treat the model like a black box and test it a scientific experiment.

The biggest shift is this: in deep learning, **meaning is emergent**. You don't assign you discover it.

That's why we moved from interpreting models by reading weights to probing the patching activations, and visualizing hidden flows. Because once the models becam powerful enough to generalize from raw data, they stopped using human-interpret concepts altogether.

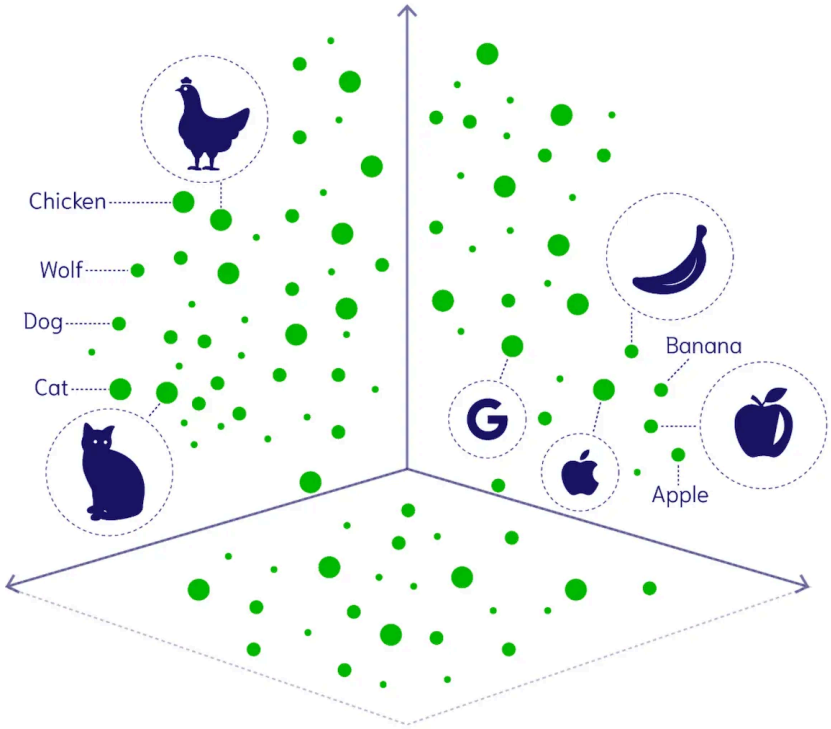
3.4 Interpretability in LLMs (Large Language Models)

Finally, we reach the **most powerful and the most opaque** systems called as LLMs These are models like GPT, Claude, Gemini. They operate over text, but beneath t surface, they're performing some of the most complex geometric reasoning known AI.

But Why are they even harder to interpret?

1. Embedding Space Is a New Language

Tokens are mapped to vectors in 1000+ dimensions. Words like “doctor” and “nurs end up near each other, not because they share a spelling, but because they appear similar contexts. You can’t interpret these vectors directly as they’re alien coordin in a latent semantic space.



2. Transformers are Function Stacking Machines

Every token goes through:

- **Embedding → Positional Encoding**
- **Multi-head Attention**: Which allows each token to peek at every other token, multiple ways
- **Feedforward Layers**
- ...And this repeats across **dozens of layers.**

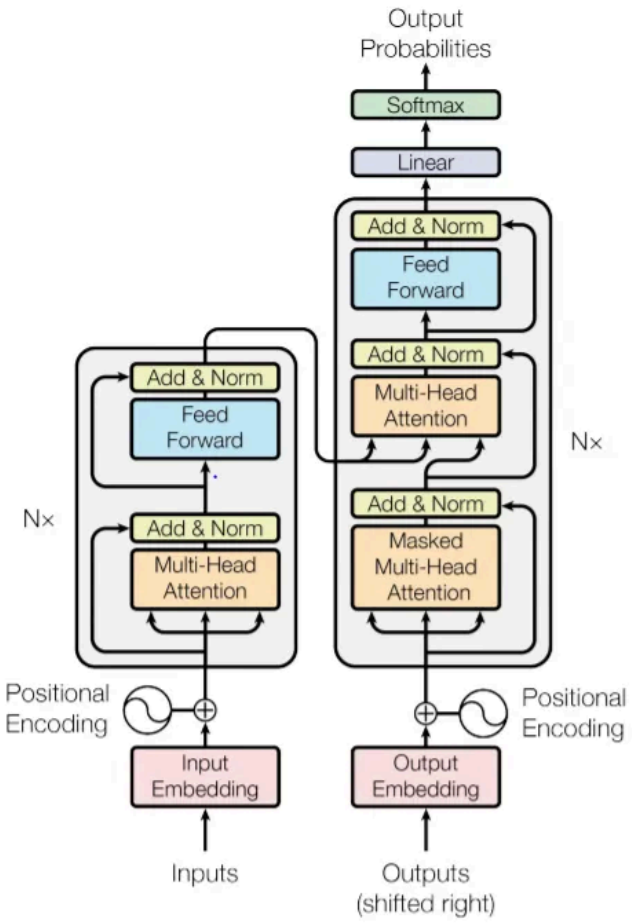


Figure 1: The Transformer - model architecture.

Each layer transforms the token representation. By the end, the representation isn word, it's a **context-aware thought-fragment**.

3. Attention Doesn't Always Explain

You might think, "let's look at the attention weights to see what the model focuse on." That works sometimes; but attention can be diffused, and it doesn't always ali with influence (causal vs correlation). Also, attention heads can develop specializa that don't correspond to human logic (e.g., tracking line breaks or counting parentheses).

Interpretability methods for LLMs

Researchers have started to dissect LLMs using **probing and attribution**:

- **Probing Classifiers**: A *probing classifier* is a small supervised model trained on internal representations (activations) of an LLM to detect if a specific **concept**

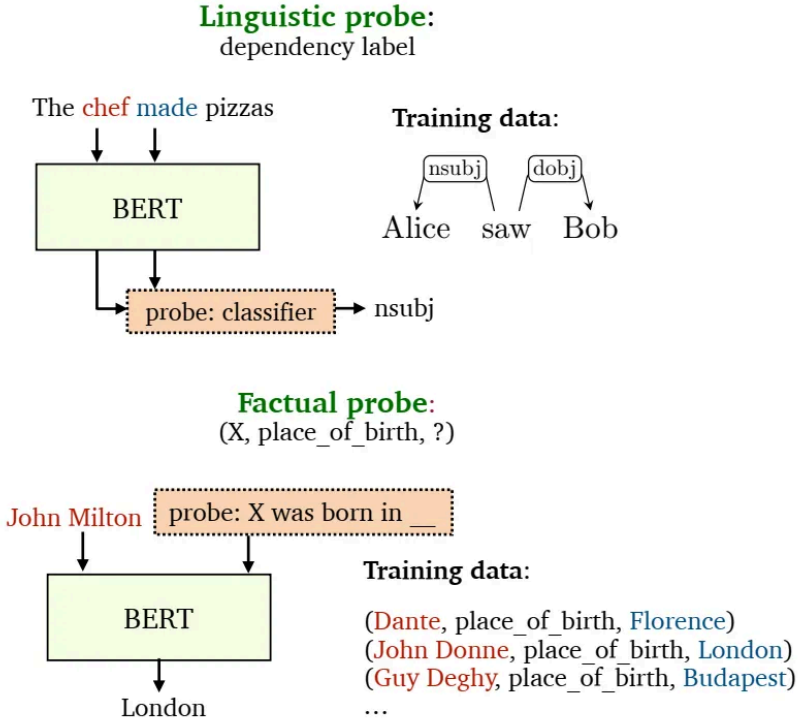
linearly encoded there.

Imagine you take a frozen LLM (no gradient updates), and at a specific layer, y record the hidden state vectors for different inputs. For example, say you feed model sentences with varying **sentiment** (positive/negative). You then train a t logistic regression classifier on these activations to predict sentiment labels. If the classifier performs well, it means the sentiment is linearly recoverable f that layer's representation. So you're not saying "the LLM understands sentim but that "some part of the model's activations encodes information about sentiment".

It helps answer questions like:

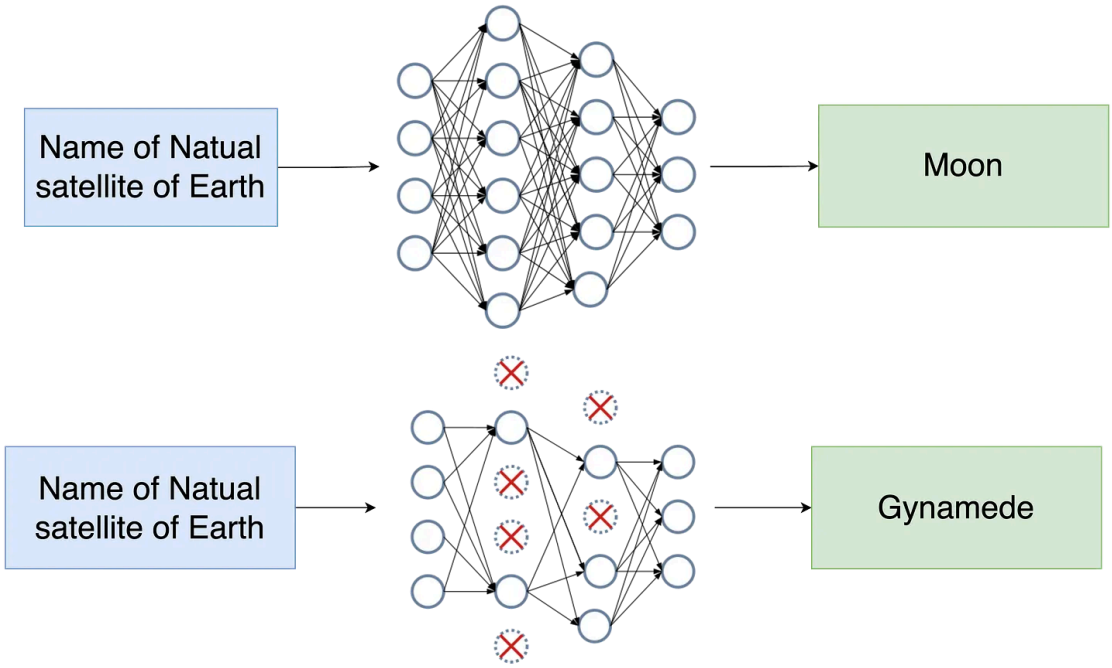
- *Which layers encode syntax vs. semantics?*
- *Where does the model store world knowledge?*
- *How abstract is the representation at different depths?*

It's like using an X-ray to scan a building, you're not walking through it, but y detecting what kind of stuff is inside at different floors.



- **Neuron Editing**: Neuron editing involves manually altering the **activation val** of individual neurons or groups of neurons and observing how the model's ou changes.

Let's say a neuron activates strongly when the input says "Paris is the capital o France". You find that changing that neuron's value leads the model to instead complete the sentence with "Berlin is the capital of Germany". That neuron, t likely encodes **geographic factual recall** or at least contributes heavily to it.



In practice:

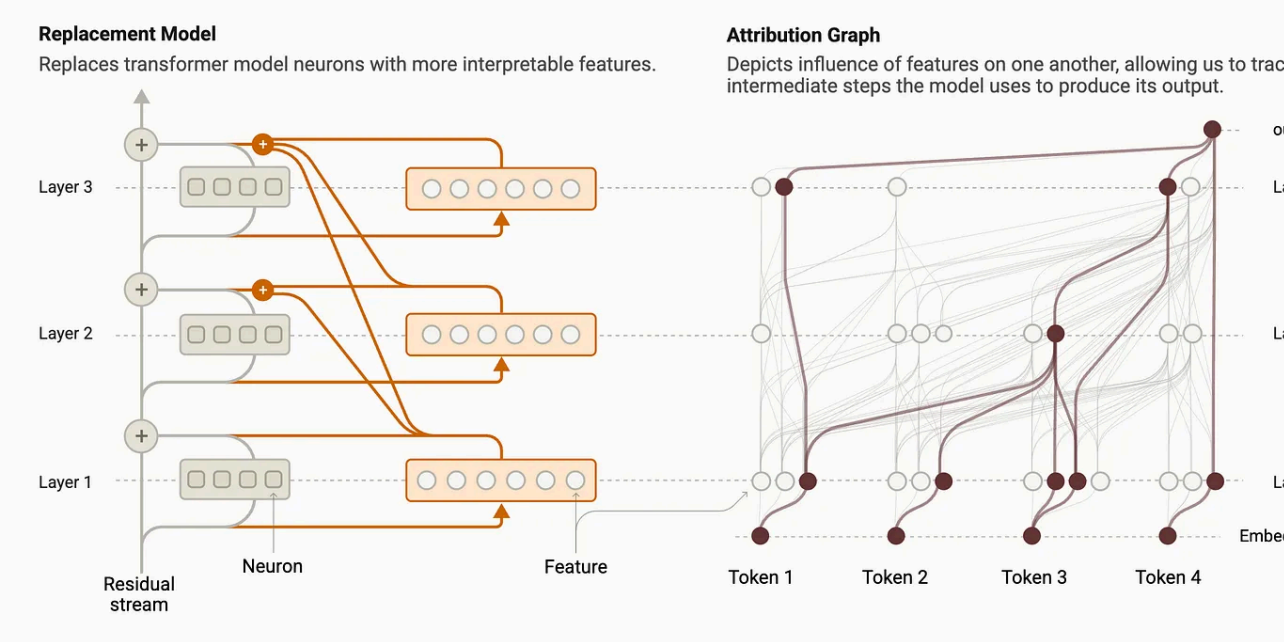
1. Identify neuron activations using representative prompts.
2. Modify (increase/decrease) the activation of a particular neuron.
3. Observe the shift in output distribution.

This is conceptually similar to **adversarial attacks** in vision models where slig perturbations to pixels drastically change the output. Here, we're perturbing t internal activations instead of inputs.

It helps uncover **fact-storing neurons**, **bias-carrying units**, or **decision switch** It's also one of the first steps toward *neural editing*, the idea that you could cor factual errors in an LLM by updating a small number of internal weights or activations.

Here, you're not retraining the model; you're performing surgical edits. This opens the door to model debugging and personalization without full fine-tuni

- **Attribution Graphs** (**Anthropic**): This is one of the most frontier-level interpretability tools for LLMs. **Attribution graphs** trace how a model's outpu say, a specific word and is causally composed from signals across the network which attention heads, MLPs, and layers contributed most.
  Anthropic's method builds a **graph of influence**, not just a layer stack. It invol
  - Recording the activation pathways that influence a particular output toke
  - Quantifying how much each intermediate component contributed to that output.
  - Building a tree/graph-like structure showing how information flowed.



Hence, Instead of looking at neurons in isolation, you start modeling **distribu causality**: "This output happened because token A influenced token B via hea and neuron Y."
This aligns strongly with the idea of **credit assignment** and **traceability**. And massive LLMs, attribution graphs allow us to model internal **compositional reasoning** i.e., how the model builds up knowledge from parts.
Think of it like a **supply chain audit** for the model's output. You trace the orig of the decision back to every component that influenced it. These trace how certain outputs are formed by aggregating contributions from neurons across layers. It's like forming a causal graph inside the network.

- **Token Attribution**: Most decoder-style LLMs (like GPT-3, GPT-4, OpenELM, GPTQ, basically the open-weights models) return not only the predicted token but also a **log-probability distribution over all tokens** at every step. This lets u ask:
    - How likely was the predicted word?
    - What were the runner-ups?
    - Did the model *really* believe in this choice, or was it close?

    But, How does this help explainability?
    Imagine a model outputs: “The Eiffel Tower is in Berlin.” Using token-level logprobs, you might discover that model gave *"Paris" a 45% chance, but, “Berlin* *chosen due to random sampling or temperature*.
    You now know this wasn't a *hard hallucination*, but a *soft confusion*. This lets yo differentiate between:
    - **Confident mistakes** (model is truly wrong),
    - **Ambiguous outputs** (multiple plausible tokens),
    - **Forced completions** (prompt drift or context error).

    You can compare logprobs across decoding strategies (greedy, beam, sampling debug hallucinations, align temperature settings, or trace why a token was pic over another.

Why it’s fascinating? LLMs don’t just store knowledge, they store *computational pathways*. Interpretability is no longer about logic. It’s about understanding the **flo representation** in a neural circuit. This doesn’t just satisfy curiosity. It helps with:

- **Fact checking** (where did this answer come from?)
- **Bias auditing** (what triggers problematic outputs?)
- **Model editing** (how do I fix one fact without hurting others?)
- **Safety tuning** (can I shut down dangerous behavior without neutering intelligence?)

Mechanistic Interpretability: From Attribution t Actual Computation (quick overview)

When we talk about interpretability, we often mean *understanding what the model attends to or uses* — via saliency maps, SHAP values, or feature attributions. But th still mostly **correlation-level understanding**. You're pointing at *what* matters, not *it's processed*.

Mechanistic interpretability takes it further. It asks *"Can we reverse-engineer what model is doing step-by-step and understand the actual circuits of computation that lead t output?"*. It's the difference between saying "The model pays attention to the word in this sentence." vs. "Head 6.2 in layer 8 negates sentiment by routing 'not' to the target noun, and MLP 9.4 implements a compositional logic gate that flips polarit

Core Methods of Mechanistic Interpretability

1. Activation Patching

Take a base prompt (e.g., "Paris is the capital of ___") and swap in activations from different prompt (e.g., "Berlin..."). If the completion changes, the patched compone (layer, head, neuron) is causally responsible. This isolates *where* in the network a specific computation is happening.

2. Path Tracing and Attribution Graphs

Used by Anthropic and others, this involves tracing **which attention heads or neu contribute most to a final output**. The idea is to create a *computation graph inside t model*, capturing not just who participated but *how* similar to causal graphs in SC

3. Neuron Editing & Ablation

Manually boost or suppress certain neurons and observe the output change. Think like "lesion studies" in neuroscience; if flipping one neuron changes "Paris is the capital..." to "Berlin is the capital...", you've found a **fact-storing unit**. This is espec useful for discovering memorization or bias neurons.

4. Sparse Autoencoders: Compressing and Naming the Hidden Space

Perhaps the most promising recent tool is the use of **sparse autoencoders** to make hidden layers more interpretable, especially in transformer MLP activations.

How it works (in brief):

1. You take activations from the MLP layers of an LLM (e.g., GPT-2).
2. You train a **sparse autoencoder** (a model that compresses the high-dimension activations into a smaller), sparse latent space and reconstructs the original activations from that space.
3. The sparsity constraint ensures each latent variable (called a *dictionary elemen* fires only for specific, interpretable patterns.
4. Once trained, each latent can be **interpreted as a concept** example, "country name", "year token", "negative sentiment cue", etc.

Think of it as creating a **semantic vocabulary** for the neural space; a vocab thats human-annotatable and mechanistically traceable.

Why this is powerful:

- You move from uninterpretable floating-point vectors to **human-readable uni**
- You can then **trace which sparse features cause which behaviors**, rather than working with all 3072 neurons at once.
- These interpretable features can then be used for **circuit discovery**, **bias correction**, or **fact editing** at a much more manageable level of abstraction.

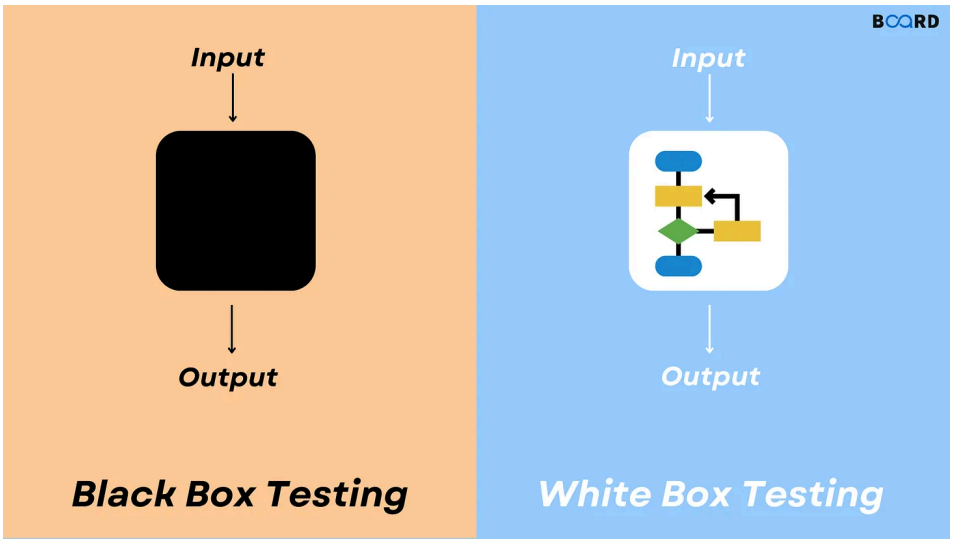
This is exactly how Anthropic's *Transformer Circuits* project and OpenAI's *Toy Mod of Superposition* operate (by sparsifying messy activations into modular, interpretab building blocks). Mechanistic interpretability is what bridges neural models with **program-like reasoning**. Instead of just attributing outputs to inputs, it helps us:

- Understand what internal parts of the model do (**like source code analysis**)
- Reuse or constrain circuits (**like APIs**)
- Detect and remove unwanted behaviors (**like debugging**)
- Build safety guarantees grounded in computation, not just statistics

4. Explainability: From Prediction to Persuasion

The goal of explainability is to understand what features or components in an inpu led to a particular output. While interpretability is model-centric, explainability is **data-point specific**. You're not trying to understand the model in general, but why *specific input* triggered *that specific output*.

4.1 White-box vs. Black-box Explainability



Conventionally, a complex model means black box and less complex model is supp to be white-box. As seen from the definition below,

White-box Models

White-box models are inherently interpretable. Think of decision trees, rule-based systems, or linear models. These models expose their logic which you can trace ac each step and reproduce the decision using just a notebook and a pencil. Explanat here is not an afterthought it is part of the model's DNA.

Black-box Models

Neural networks, ensemble methods, and large language models are black boxes. feed them input, they return an output, but the "why" is hidden inside millions (or

billions) of parameters and internal flows. For these models, we **build explanation post-hoc**, either by probing the model or by creating external surrogates.

But, In modern day context of models, these definitions starts becoming even more bifurc as service (like ChatGPT from openAI, claude and sonnet from Anthropic), could be considered as Black box models as even the weights/activations are hidden, we only have access to final output. Whereas open-weights models like Llama, Mistral, even the Yolov8 (object detection) that runs on your local system are examples of white-box models. Any a like model setup could not be played with (as we don't have control over weights) hence gradient based methods are ineffective, the only available option is data-level methods lik SHAP.

That said, Let's walk through the most important techniques.

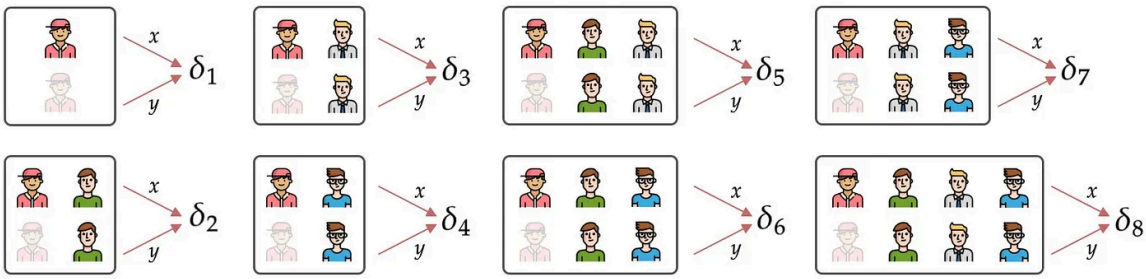
4.2 SHAP (SHapley Additive Explanations)

What is it?

SHAP is a model-agnostic, local explanation method that attributes a model's out prediction to each input feature using **Shapley values** from cooperative game theo

How does it work?

Shapley values measure the **average marginal contribution** of a feature across all possible combinations of features. In a game, each player contributes to the payou a model, each feature contributes to the prediction.

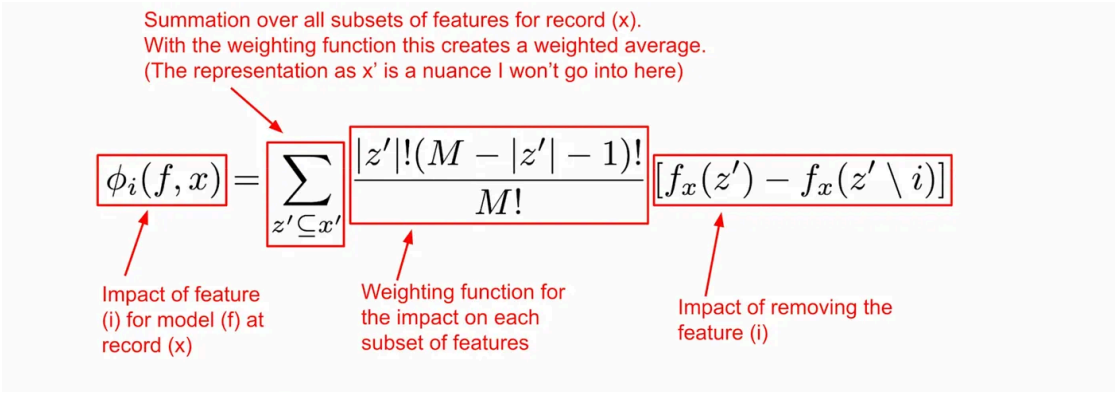


The Shapley value for member

is given by:

$$\phi_i = \frac{\delta_1 + \delta_3 + \delta_4 + \delta_5 + \delta_6 + \delta_7 + \delta_8}{8}$$

Mathematically, for a feature iii, its Shapley value ϕi is:



Where:

- M: set of all features
- Z': subset of features excluding iii
- f(z): model output using only features in SSS

Steps:

1. Enumerate all subsets S⊆N{i}
2. For each subset, calculate marginal contribution as `f(Z∪{i}) - f(Z))`.
3. Weight each by its proportional importance

4. Sum these to get the feature's Shapley value

Example:

Let's say you're predicting loan default probability.

- Income contributes +0.2
- Age contributes -0.1
- Credit score contributes +0.3
  Then the total prediction (baseline + SHAP values) might be:
  → 0.5 = 0.1 (baseline) + 0.2 + (-0.1) + 0.3

Each SHAP value explains "how far and in which direction" a feature shifted the prediction from the baseline.

Strengths:

- Theoretically grounded
- Works on any model
- Gives consistent global and local views

Limitations:

- Computationally expensive (approximations like TreeSHAP helps though)
- Assumes independence between features

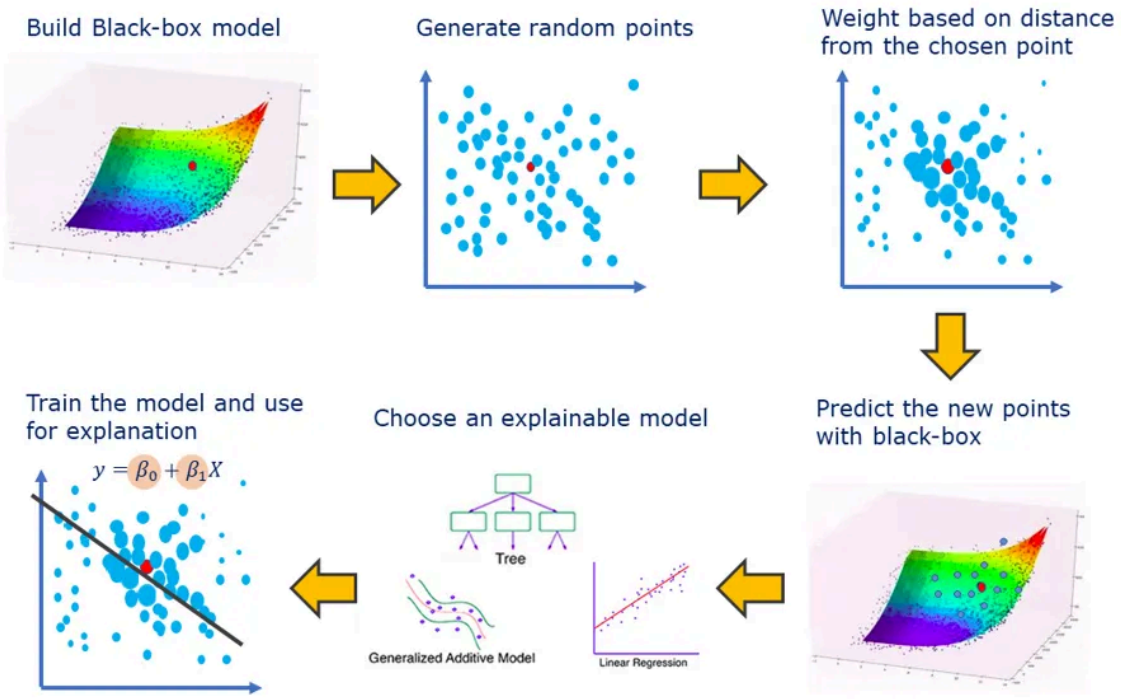
4.3 LIME (Local Interpretable Model-Agnostic Explanations)

What is it?

LIME is a local approximation method. It fits a simple, interpretable model (like li regression) around the neighborhood of the data point being explained.

How does it work?

LIME perturbs the input data and observes changes in the model's prediction. It u
this local dataset to train a linear model that mimics the original model's behavio
around that point.



Steps:

1. Select the input x to explain
2. Generate perturbed versions x′ around x
3. Get the model predictions f(x′)
4. Weight each x′ based on proximity to x (e.g., using an RBF kernel)
5. Fit a sparse linear model g(x′) to mimic f(x′)
6. Use coefficients of g to explain prediction at x

Example:

If the original input is an email classified as spam, LIME perturbs the words by removing them and sees how the prediction changes. The local linear model migh

- Presence of “free” = +0.4
- “Congratulations” = +0.3

- “Unsubscribe” = -0.2
  → Total decision: spam

Strengths:

- Intuitive and flexible
- Doesn’t require model internals
- Supports text, images, tabular data

Limitations:

- Local approximation might not match global logic
- Explanations can vary with sampling randomness

4.4 Integrated Gradients

What is it?

Integrated Gradients is a **white-box**, gradient-based attribution method. It attribu feature importance by integrating gradients along the path from a baseline to the actual input.

Core idea:

Instead of taking the gradient at one point (which may be zero or noisy), take the **average gradient** along a straight line from a neutral baseline to the actual input.

Formally:

$$IG_i(x) = (x_i - x_i') \cdot \int_0^1 \frac{\partial f(x' + \alpha \cdot (x - x'))}{\partial x_i} \, d\alpha$$

Where:

- x: actual input
- x′: baseline input (e.g., all zeros or average)
- f: model function

Steps:

1. Choose a baseline input (e.g., black image or empty text)
2. Linearly interpolate between baseline and input
3. Compute gradients along the path
4. Average gradients and scale by input difference
5. The result is the feature attribution

Example:

For an image classifier predicting “zebra”, you start with a black image and slowly morph into the zebra image, collecting gradients. The stripes (important regions) will have higher accumulated gradients.

Strengths:

- Works well with deep nets
- Axiomatic guarantees (sensitivity, implementation invariance)

Limitations:

- Requires differentiability
- Choice of baseline is critical

4.5 DeepLIFT (Deep Learning Important FeaTures)

What is it?

DeepLIFT is a backpropagation-based attribution method designed to overcome t limitations of gradient-based methods (like vanishing gradients or ReLU saturatio attributes the difference between a neuron's activation and a reference activation, propagating that "contribution" backward through the network.

Core Mechanism

While gradients measure *how* a change in input affects the output, DeepLIFT measures *how much* the actual value of a neuron contributed compared to a referer input.

The formula for the contribution of input `xi` is:

$$C_{\Delta x_i} = \frac{\Delta x_i}{\Delta y}$$

Where:

- `Δxi=xi−xi0` (difference from reference input)
- `Δy=y−y0`

Then contributions are propagated backward using specially defined "multipliers' which are analogous to gradients, but defined based on deltas.

Steps:

1. Define a **reference input** x^0 (e.g., a neutral input or average value)
2. Compute activations for both actual input and reference
3. Calculate deltas Δxi, Δy.
4. Backpropagate contributions using chain rule on deltas
5. Aggregate to get input attributions

Example:

Say you have an image classification model and a pixel changes the score for "cat" +0.2 compared to a gray reference image. DeepLIFT will assign that +0.2 credit ba

the relevant pixels that caused it, even if their gradient is zero.

Strengths:

- Doesn't suffer from vanishing gradient issues
- Fast and works well with ReLUs
- Captures influence rather than just sensitivity

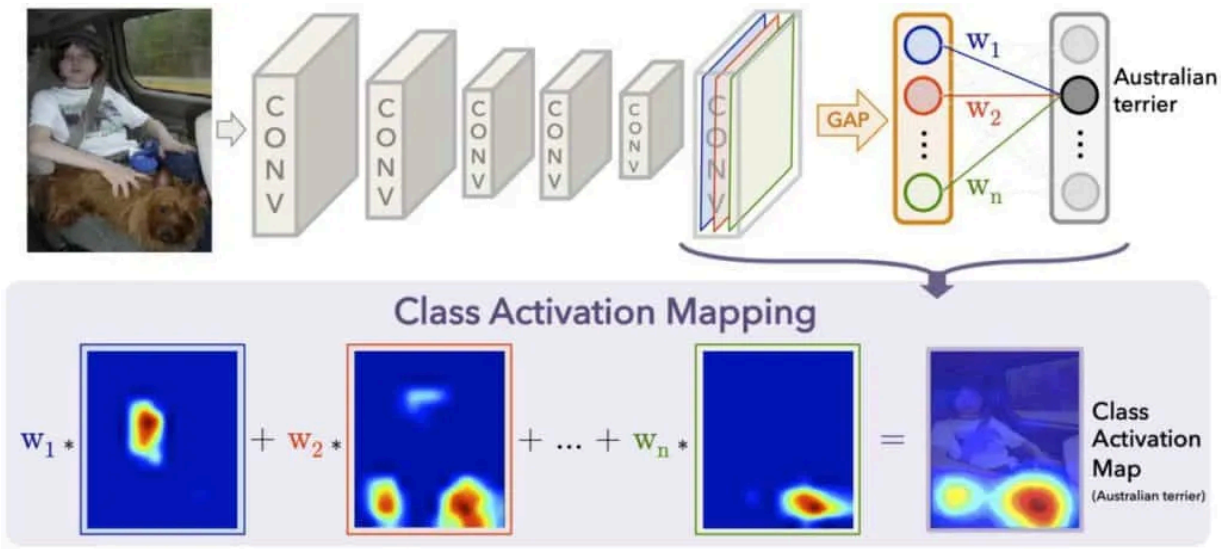
Limitations:

- Requires defining a meaningful reference input
- Implementation is more complex than gradients

4.6 Grad-CAM (Gradient-weighted Class Activation Mapping)

What is it?

Grad-CAM is a visualization technique for CNNs that highlights regions in the in (usually images) that contributed most to the model's decision.



Core Mechanism

It computes the gradient of the class score w.r.t. feature maps in the last convoluti layer, then uses a weighted combination of those maps to produce a **class-specific**

heatmap.

The formula for the importance weight αk of feature map Ak:

$$\alpha_k = Z_1 \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

Then the Grad-CAM map is:

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k A_k\right)$$

Steps:

1. Forward pass: compute prediction score y^c
2. Backward pass: compute gradients of y^c w.r.t. convolutional features A^k
3. Average the gradients to get αk
4. Weighted sum of the feature maps using αk
5. Apply ReLU and overlay on input

Example:

In an image of a lion, Grad-CAM may highlight the mane and face, as these are th areas that most activated the “lion” class neurons.

Strengths:

- Intuitive, visually compelling
- Doesn’t require architectural changes
- Generalizes across CNN-based models

Limitations:

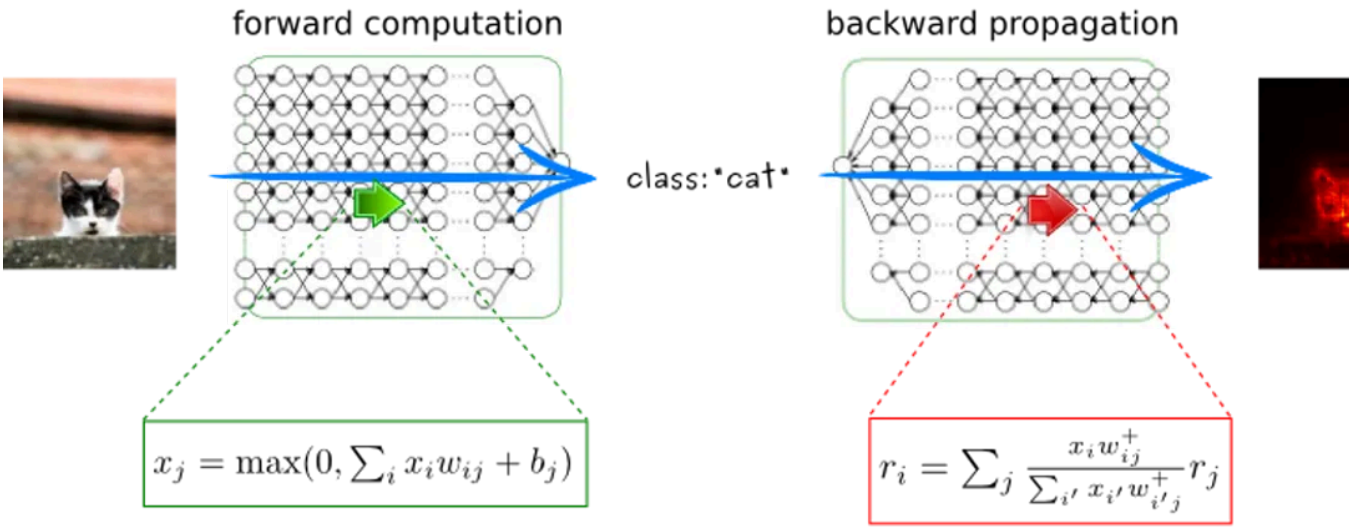
- Works best with spatial models (e.g., images)
- Not suitable for fully connected layers or NLP tokens
- Can blur important fine-grained details

4.7 Layer-wise Relevance Propagation (LRP)

What is it?

LRP is a rule-based method that propagates the prediction score backward to inpu features, redistributing the relevance in proportion to their contribution.

LRP is neither based on gradients nor on perturbations. It uses a "conservation ru to ensure total relevance is preserved as it flows back.



Core Mechanism

If neuron j in layer l is connected to neuron i in layer l+1, then relevance is distribu as:

$$R_j = \sum_i \frac{a_j w_{ji}}{\sum_k a_k w_{ki}} R_i$$

Where:

- aj: activation of neuron j
- wji: weight from neuron j to i
- Ri: relevance of output neuron

Steps:

1. Forward pass: compute the output
2. Backward pass: redistribute output score to neurons below using propagation rules (Z-rule, Epsilon-rule, etc.)
3. Repeat until reaching input layer
4. Visualize or report input attributions

Example:

For a digit image classified as “5”, LRP can highlight the curved stroke in the top-l that differentiates “5” from “6”.

Strengths:

- Strong theoretical guarantees (relevance conservation)
- Robust in detecting subtle patterns in vision models

Limitations:

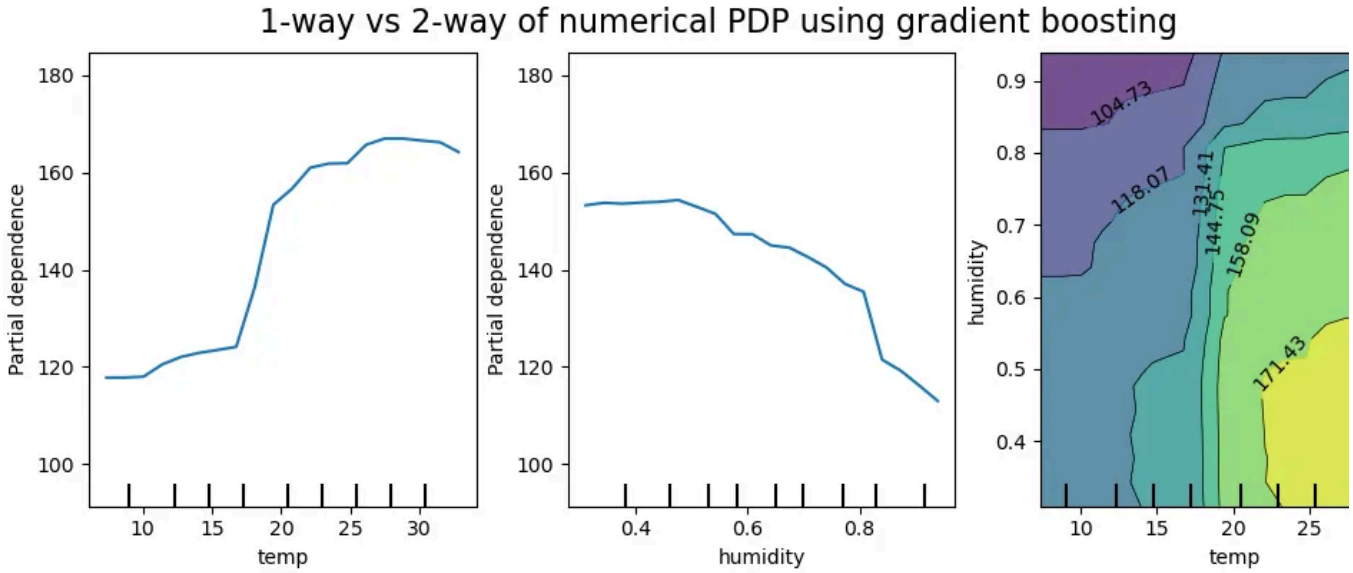
- Sensitive to rule choice
- Complex to implement for modern architectures (especially transformers)

4.8 Visualization Based Explainability

These methods help answer “What is the general effect of an input feature on the model’s output?”. They are **global** in nature as they try to explain the model's beha *across the dataset*, not just for one input. These tools are especially useful in tabula structured data where humans reason in terms of **feature-behavior relationships**.

Partial Dependence Plots (PDPs)

A PDP shows how the predicted output changes as a single feature varies, while keeping all other features fixed. It visualizes the **marginal effect** of a feature on th output.



How it works (algorithm):

1. Select a feature xi to study.
2. For each value v in a grid:
    - Replace xi with v for all instances in the dataset (or a subset).
    - Predict using the model.
    - Average the predictions.
3. Plot v vs. average prediction.

Mathematically:

$$\text{PDP}(x_i = v) = \frac{1}{n} \sum_{j=1}^{n} f(x_1^{(j)}, \ldots, x_{i-1}^{(j)}, v, x_{i+1}^{(j)}, \ldots, x_d^{(j)})$$

Example:

In a loan prediction model, a PDP of income might show that higher income steac decreases default probability, plateauing after a certain point.

Strengths:

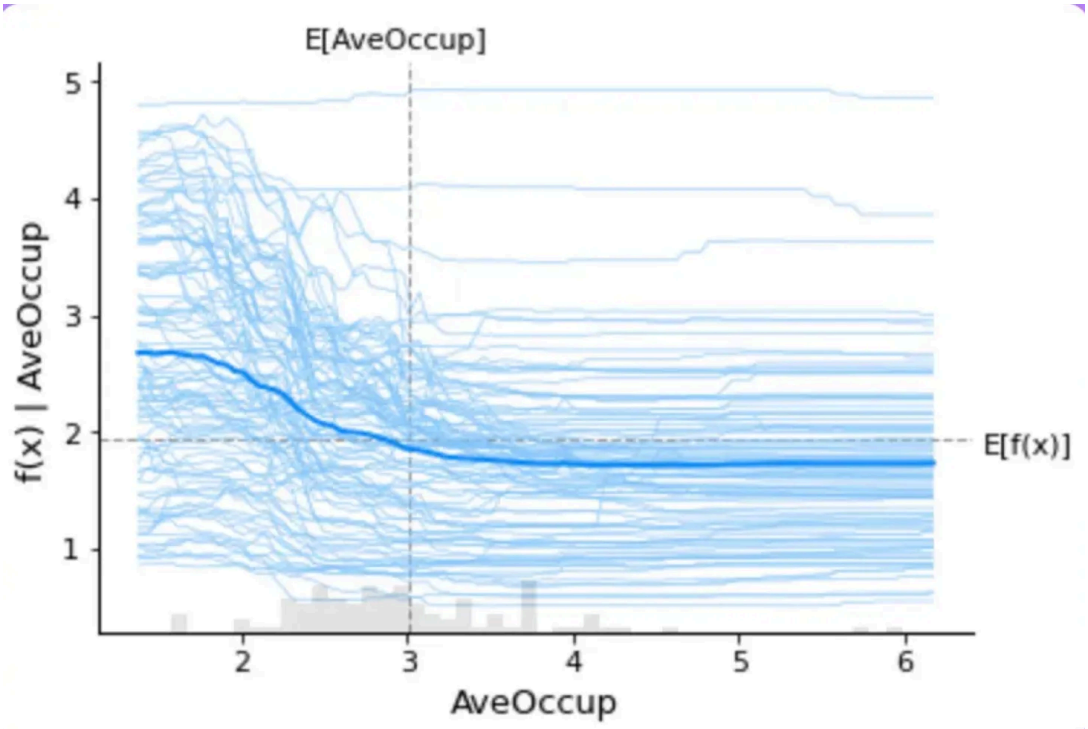
- Simple to interpret
- Global trend visibility

Limitations:

- Assumes feature independence
- Unrealistic feature combinations if correlated (e.g., high age + low income)

Individual Conditional Expectation (ICE) Plots

ICE plots extend PDPs by showing the *individual-level effects* of a feature. Instead c averaging, you plot a line per instance to see how predictions shift per input.



How it works:

1. Select a feature xi
2. For each instance:
    - Vary xi while keeping the rest fixed
    - Record predictions
3. Plot one line per instance

Example:

For some people, increasing education might increase salary, while for others it m not (e.g., due to other features like experience). ICE reveals these interactions, whi PDP would just average them out.

Strengths:

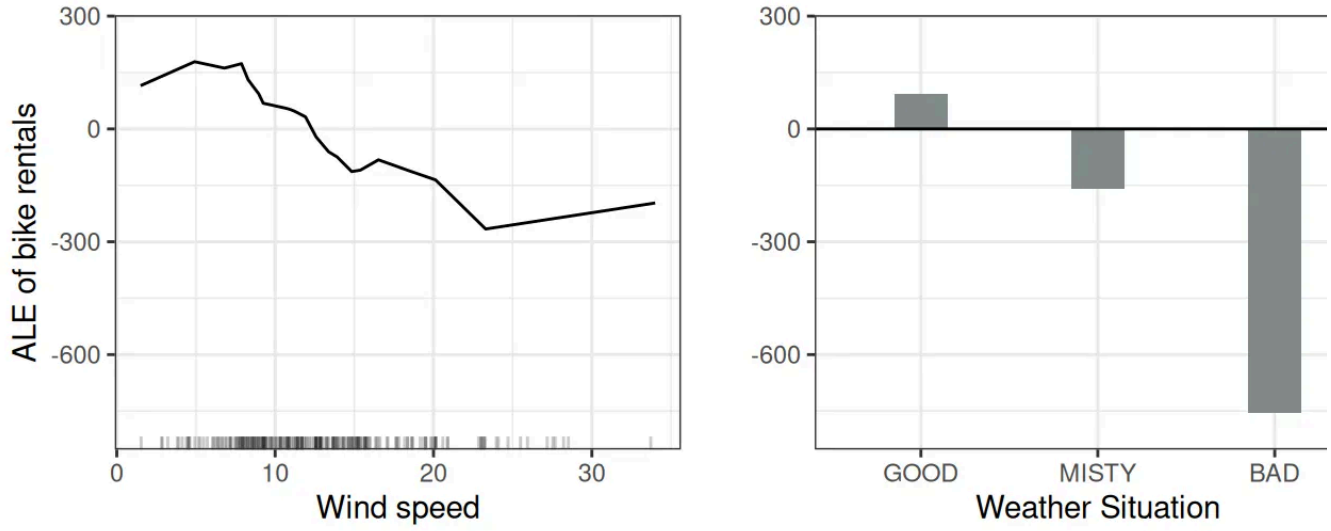
- Captures heterogeneity in model behavior
- Shows individual effects

Limitations:

- Can be overwhelming to read
- Still inherits assumptions of fixed feature substitution

Accumulated Local Effects (ALE)

ALE plots improve on PDPs by accounting for **feature correlation**. Rather than averaging across the full dataset, ALE computes **local differences** within the featu distribution *(it's sort of like auto-correlation analysis)*.



How it works:

1. Divide the feature domain into intervals
2. In each interval:
    - Compute the local change in prediction when xi increases
    - Average these differences only within that interval (no unrealistic data poi
3. Accumulate the local effects to form a global picture across time intervals

Mathematically:

$$\mathrm{ALE}(x_i) = \sum_{k=1}^{K} \frac{1}{|D_k|} \sum_{x \in D_k} \left[f\left(x_i^{(k)}\right) - f\left(x_i^{(k-1)}\right) \right]$$

Where Dk is the set of samples in interval k, f(x_k) and f(x_k+1) are model predicti
kth and k+1th time interval; respectively.

Example:

Suppose income and age are correlated. PDP might show odd jumps when setting
arbitrarily. ALE avoids this by only comparing income ranges where those ages
actually occur.

Strengths:

- Robust to correlated features
- More realistic interpretations

Limitations:

- Harder to interpret than PDP
- Still global in nature, not instance-specific

4.9 Causal Explanation Methods

Most explainability techniques, including SHAP, LIME, and Grad-CAM, tell us w
correlates with a prediction. They answer: “Which features influenced the output?”
this doesn’t necessarily imply causation. A feature may show up as important simp
because it's statistically associated with the outcome, not because it actively cause
This is especially critical in regulated or sensitive domains like healthcare, crimin
justice, education, and finance where explanations have real-world impact.

Consider a model that flags applicants from a certain zip code as higher risk. SHA
might rank zip code as highly influential. But that doesn’t mean changing the zip
causes risk to decrease. What if zip code is just a proxy for income or education le
In that case, acting on it would be misleading and potentially unfair.

To make truly actionable and fair decisions, we need to ask causal questions like:

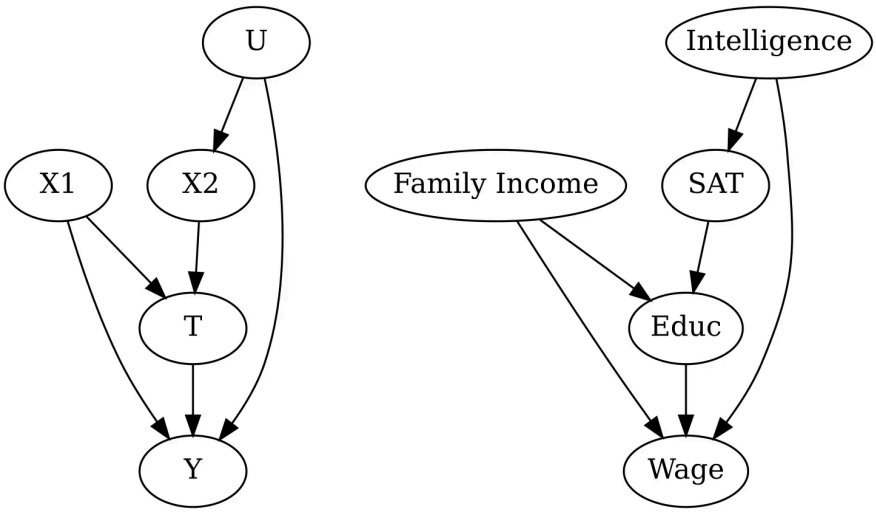
- *Did this variable cause the outcome?*
- *If we changed this variable, would the output change too?*

Causal explanations are fundamentally about **agency**. They answer questions like:

- What features can a person change to alter the outcome?
- Was the decision made on a justifiable basis?
- Are the important variables truly responsible for what happened?

Structural Causal Models and do-Calculus

The formal framework for answering these questions is built on **Structural Causal Models** (**SCMs**). SCMs represent relationships between variables using a **causal gr** which is a directed acyclic graph where nodes are variables and edges represent di causal influences.

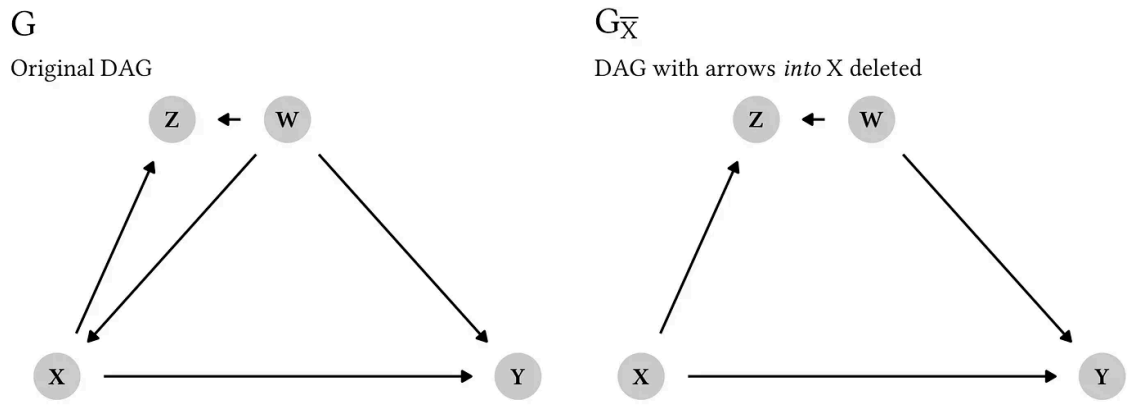


This allows us to differentiate two kinds of reasoning:

- **Observational**: What is P(y | x)?
- **Interventional**: What is P(y | do(x))?

This is the essence of **do-calculus**, developed by Judea Pearl. The key idea is that **observing** a variable (x) and **intervening** to set it (do(x)) are not the same. Interveni

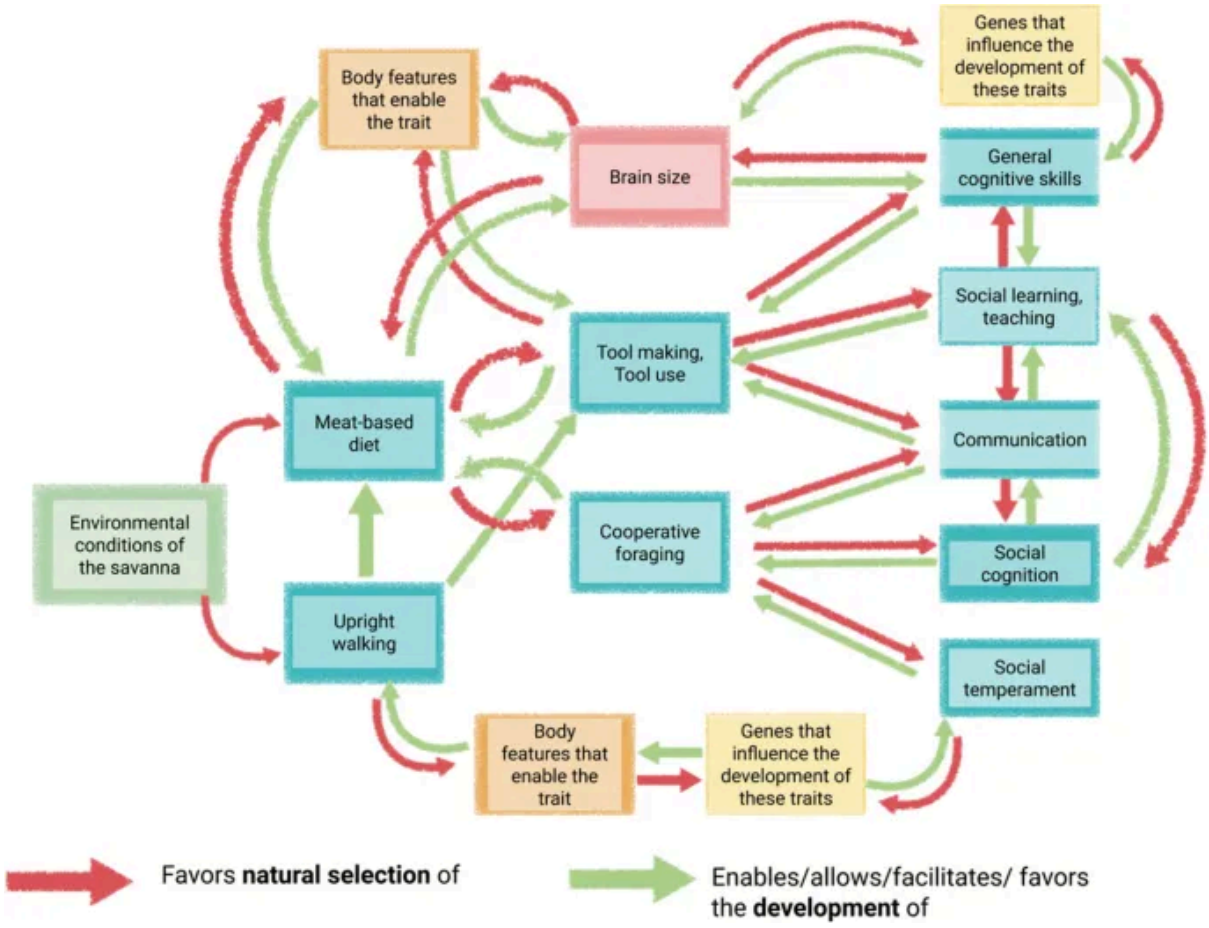
breaks the natural flow of the causal graph and simulates a new world where the variable is manually controlled; not just observed.



In the explainability context, this gives us a deeper form of reasoning: not just why model predicted something, but what would have happened *if we had changed a spe input* in a meaningful, causal way.

Causal Pathways and Mediation

Causal analysis also helps us understand **how** a feature influences the outcome. Do education improve loan approval because it directly reflects responsibility? Or doe work indirectly, by increasing income which then affects credit score?



This is handled by **mediation analysis**, which decomposes the total effect of a vari into:

- **Direct effects**: How much the feature affects the outcome on its own.
- **Indirect effects**: How much it works *through* another feature (a mediator).

Knowing these paths helps policymakers and model designers prioritize the right interventions.

Causal Discovery and Domain Transfer

In many cases, we don't know the causal graph. That's where **causal discovery** methods come in. They attempt to learn the graph structure from data using statis tests, conditional independence assumptions, or score-based optimization.

Another application is **Invariant Risk Minimization** (**IRM**), a technique for identi features whose relationships with the output stay consistent across domains or

environments. For example, income might reliably predict repayment across cities zip code may not.

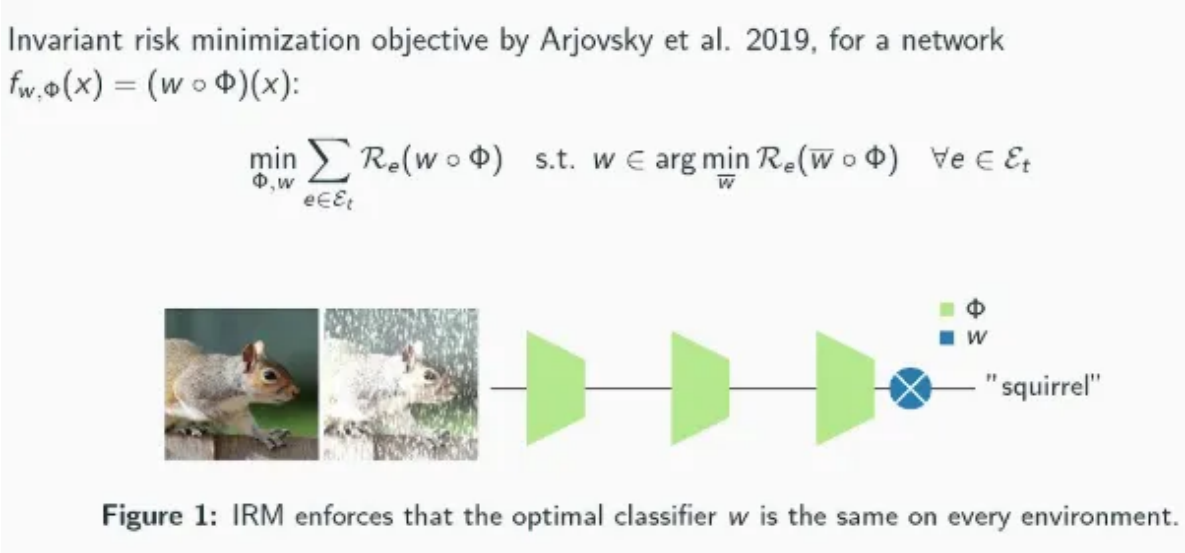


Figure 1: IRM enforces that the optimal classifier w is the same on every environment.

In typical machine learning, we use **Empirical Risk Minimization** (**ERM**); minimi the average loss over training data, and hope the model generalizes. But ERM doe distinguish between **causal signals** and **spurious correlations**.

Source: http://bit.ly/3Yjy3EX

Let's say shorter reviews are often negative in Country A. ERM may learn "short = bad". But this breaks in Country B, where reviews are short and positive. The mod fails to generalize. IRM flips the question. It says: "What if we trained a model wh the same classifier works well in multiple environments, despite surface-level

correlations changing? Instead of minimizing average risk, IRM looks for features are **invariant** i.e., causally useful, not just correlated.

Hence, IRM learns a representation Φ(x) such that, for each environment eee, the s classifier w minimizes risk. Find Φ such that it minimizes following

$$w \in \arg\min_{\bar{w}} R^e(\bar{w} \circ \Phi) \; \forall e$$

In practice, this is approximated using a gradient penalty that enforces flatness of around w, forcing invariance. IRM-based models are more robust and avoid spuric correlations, improving generalization and fairness.

4.10 Counterfactual Explanations

Counterfactual + Causal Reasoning in LLMs and Vision Systems

Modern AI systems, particularly large language models and vision foundation mo are incredibly capable, but often operate as **opaque black boxes**. Causal reasoning counterfactual explanations can be powerful tools to make them more transparent controllable, and aligned.

Counterfactual Explanations

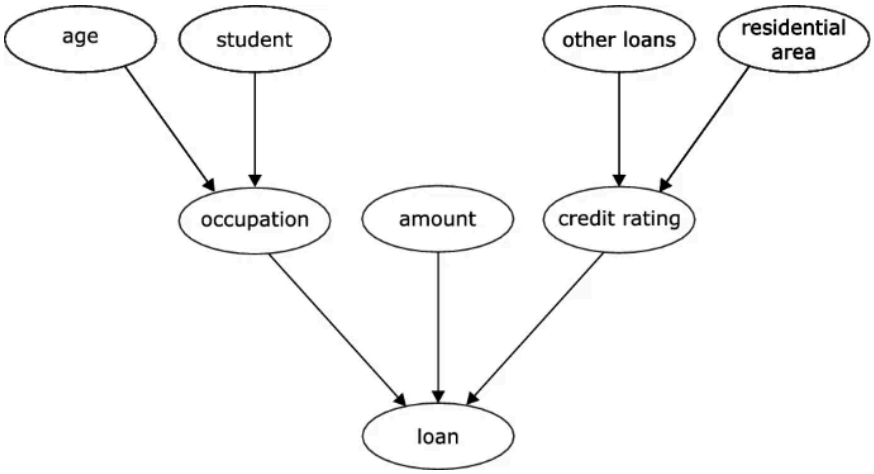
What is it?

Counterfactual explanations try to answer things like *"What minimal change to the i would have led to a different outcome?"*. They're useful when you want to **suggest actionable steps** to reach a better decision (especially in finance, HR, healthcare, e

Core idea:

Find a nearby input x′ such that:

- $f(x') \neq f(x)$ (outcome changes)
- x' is "close" to x (minimal change)
- x' is realistic (exists in the data distribution)



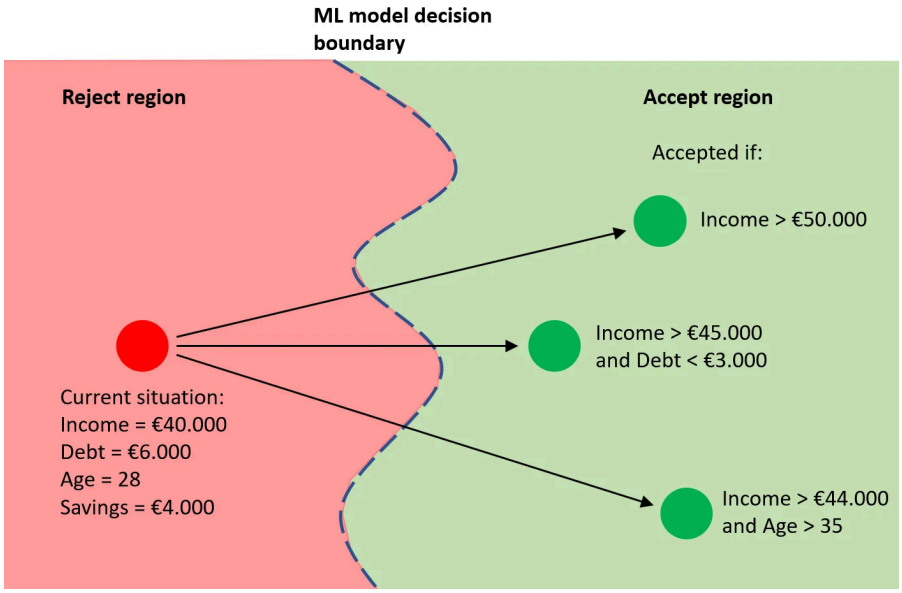
Algorithm (generic optimization-based form):

Minimize:

Where:

- First term ensures output flips
- Second term keeps it close to original
- Third term ensures realism (e.g., using GANs or VAEs)

for example, If your loan application was denied, a counterfactual might say: *"If yo income were $5,000 higher and debt $2,000 lower, your approval chance would be 85%."*



Variants:

- GAN-based counterfactuals (data realism)
- Diverse counterfactuals (multiple suggestions)
- Actionable counterfactuals (only editable features are changed)

Strengths:

- Actionable and user-centric
- Doesn't require full model access

Limitations:

- Can be expensive to compute
- May suggest impossible changes (e.g., age)

Causal Traces in LLMs

In LLMs, causal interventions can help answer: *Why did the model generate this toke What caused it to mention this fact instead of another?*

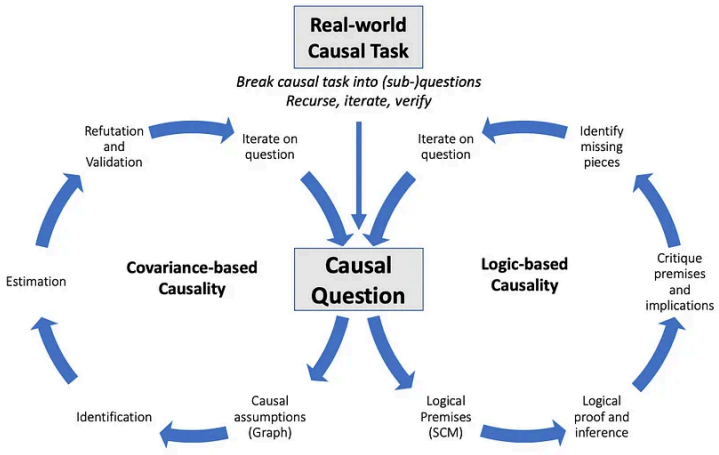


Figure 1: When tackling real-world causal tasks, people strategically alternate between logical- and covariance-based causal reasoning as they formulate (sub-)questions, iterate, and verify their premises and implications. Now, LLMs may have the capability to automate or assist with every step of this process and seamlessly transition bteween covariance- and logic-based causality.

Recent techniques like **activation patching**, **causal mediation through attention heads**, and **probing latent paths** are beginning to treat LLMs as causal systems. F instance:

- By replacing the activations of one layer with those from a counterfactual inp (e.g., “Paris is the capital of <mask>”), we can trace **which parts of the networ cause factual completions**.
- Anthropic and others have used **attribution graphs** to visualize how attention flows cause final outputs.

These methods combine SCM-style reasoning with architectural interpretability t pinpoint where and how tokens emerge.

Vision Models: Interventions and Counterfactuals

In vision systems, counterfactuals can answer: *What part of the image made it a “zeb instead of a “horse”?*

This goes beyond Grad-CAM heatmaps. Using **causal masking**, **intervention-base saliency**, or **generative inpainting**, we can:

- Replace regions of the image to see if the class changes

- Measure counterfactual sensitivity: which regions, when changed, flip the prediction

Some works also generate **counterfactual images** using GANs or VAEs; example, turning a blurry tumor scan into a sharp one to see if the prediction changes. This helps verify robustness and detect fragile, artifact-driven behavior.

Bridging Causal Theory with Neural Practice

Ultimately, combining **counterfactual logic** with **neural attribution** gives us the b of both worlds:

- Explanations grounded in action: "If you change this, that happens."
- Decomposition of blame or credit: "This path in the network was responsible.
- Alignment with human mental models: "Why did it say this?" vs. "What woul have said instead?"

As foundation models become agents, causal explainability might be key to understanding not just **what they do**, but **why they behave the way they do** and hc we might influence, debug, or constrain them.

5. Evaluation and Challenges in Explainability

Imagine two models give different explanations for the same prediction. Which or better? Which is more faithful to the model's actual reasoning? Which one is fair, stable, or even usable?

Explanations are only as good as the metrics we use to **evaluate** them. This sectior explores the key dimensions on which explanation quality is assessed and where current techniques fall short.

Fidelity

Fidelity measures how accurately the explanation reflects the true behavior of the model. It answers "Does this explanation tell the truth about what the model is

actually doing?". For example, a LIME explanation uses a local surrogate model. Fidelity would measure how closely that linear model approximates the real mode the neighborhood of the input. *High fidelity = high trust in the explanation.* Low fidel explanations may look plausible but be misleading, which is a kind of explanatory overfitting.

Stability

Stability (or robustness) measures whether small changes to the input produce **wil different explanations**. If you slightly alter a sentence or image, and the top SHAI features change completely, that suggests **instability** even if the model output didı change much. "Are explanations smooth and consistent under small perturbations This is especially important when explanations are used in safety-critical domains you don't want your rationale to fluctuate because of noise or typos.

Consistency

Consistency compares explanations across **different models** that behave similarly two models produce almost the same predictions, they should have **similar explanations** for the same input. "Do models that agree in output also agree in explanation?" This is a useful sanity check when choosing between architectures during ensembling. If explanations differ wildly, it may mean one model is picking spurious signals.

Comprehensibility

Not all explanations are useful to humans. **Comprehensibility** measures whether t explanation is **understandable**, **concise**, and **contextually aligned** with how humai reason. An explanation with 500 SHAP values may be accurate but unreadable. "C human make sense of this explanation and act on it?" This is especially important human-in-the-loop systems or regulatory settings where explanations must be presented to end users or auditors.

Local Accuracy

For local explanation methods (like LIME or SHAP), we care about **how well the explanation approximates the model around a single data point**. *If we use the*

explanation to simulate the model locally, how close is it to the real output?. This can be measured using metrics like R^2, cosine similarity, or prediction divergence.

Representativeness

Some explanation methods highlight the most influential **examples** or **clusters** lik prototype-based or exemplar explanations. Here, the question is : *Do the selected examples truly represent the behavior of the model?*

If the model relies on edge cases or rare modes, choosing a few prototypes might oversimplify things. Representativeness ensures the user sees the *right* examples n just easy ones.

Faithfulness

Faithfulness checks whether the explanation is causally linked to the prediction.

For example:

- If a feature is said to be important, does removing it change the output?
- If attention is used as an explanation, does masking attended tokens reduce performance?

This is often tested through **ablation**, **occlusion**, or **insertion** tests by systematical removing or replacing input parts to test whether predicted importance aligns wit functional relevance.

Challenges and Open Questions

Evaluating explanations isn't just about metrics, it's a philosophical and practical challenge.

Some of the big problems:

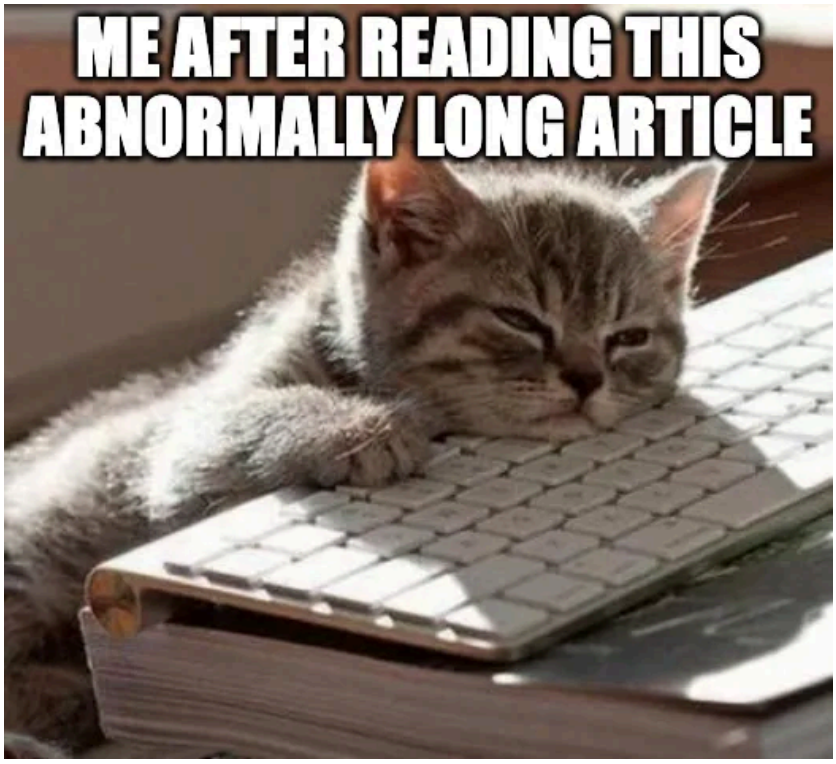
- **Ground truth is missing**: There's often no correct explanation to benchmark against.

- **User needs vary**: What's comprehensible to a doctor may not work for a data scientist.
- **Multiple valid explanations**: A prediction might have many reasonable justifications; now, which one should be shown?
- **Explanations can be gamed**: Just like models can overfit, explanations can be made to *look* faithful while hiding true logic.

This leads to an important principle: **explanations should not just be visual or plausible; they must be tested like any model component**. That includes adversar testing, human studies, and integration into real workflows.

6. Conclusion and Future Directions

With this we end this very looooong yet necessary article, and first of all congratulations if you have reached till here (I hope you are not tired by the sheer volume of content). I would suggest you to treat this article as a structured referen and re-visit the articles in pieces.



We began this article with a simple but powerful idea: *what if you could truly unders what goes on inside your AI model; not just what it predicts, but why it predicts that way?* doing so, we explored two deeply intertwined yet distinct concepts that are **interpretability** and **explainability**. Interpretability gave us tools to understand m by design; from linear models and decision trees to attention maps and activation visualizations. Explainability, on the other hand, gave us tools to post-hoc analyze predictions, especially when the model was too complex or opaque to unpack dire We walked through foundational methods like SHAP, LIME, Integrated Gradients DeepLIFT, and Grad-CAM, understood how visualization tools like PDP, ICE, and ALE reveal global feature behavior, and dived into the emerging world of **counterfactual and causal reasoning** which is the only lens that can answer questi like: *What should I do to change the outcome?*, *What really caused this decision?*

And we didn't stop there. We asked: *Can we trust these explanations?* Are they stable faithful, actionable, or just plausible-looking visualizations? This led us into evaluation frameworks like fidelity, comprehensibility, robustness, and fairness, reminding us that XAI is not just about introspection, but accountability.

Where Do We Go From Here?

Despite all this progress, explainable AI is far from solved. In fact, it's just enterin most interesting phase. Some directions that will shape the next era:

1. **Causal + Neural Fusion**: We're only scratching the surface when it comes to causal interventions in neural systems. Future architectures may explicitly mo causality, allowing built-in reasoning about interventions and effects.
2. **Multi-level Explanations**: A good explanation is not a one-size-fits-all thing. What a compliance auditor needs is very different from what an ML engineer needs. Future explainability systems must adapt their granularity and languag based on audience.
3. **Human-AI Collaboration**: Explanations will become part of human-AI loops, enabling **dialogue**, **clarification**, and even **challenge**. Think: "Why did you say that?" followed by "What if I told you X instead?" Future agents must support interactive, conversational explanation.

4. **LLMs Explaining Themselves**: As LLMs grow in abstraction, we'll need them explain *not just token output*, but **reasoning chains**, **factual sourcing**, and **self-correction**. Tools like chain-of-thought, trace attribution, and token-level influence graphs are early signs of this future.
5. **Ethics and Auditing**: As AI increasingly touches society in hiring, lending, education, justice where explainability is not optional. It becomes part of **governance**, **auditing**, and **ethical oversight**. The models that thrive in this we will be the ones that don't just perform, but can explain their performance.

7. Final Thought & Reference

In many ways, explainability is the bridge between **math and meaning**; between a matrix of weights and a human decision. As models get larger, deeper, and more capable, this bridge only becomes more important. We can't peer into every param but we can and we must design systems that let us ask questions, simulate alterna and understand consequences.

Because in the end, intelligence without understanding is just behaviour. And bui systems we can explain means building systems we can trust.

Further reads / references

Interpretable ML Book
Knowledge probing article
PDP vs ICE plots explained
Anthropics Biology of LLM blogs
Causal ML
Mechanistic interpretability

That's all for today.

Follow me on LinkedIn and Substack for more such posts and recommendations, t then happy Learning. Bye 👋

6 Likes

Discussion about this post

Comments Restacks

Write a comment...

Learning Journey May 21

Liked by Siddhant Rai

Hi, I enjoy newsletter from you guys so much, but I would like to point out that with dark mode hard to read black text in images with transparent background, I hope you guys can design the as white background so they are compatible with dark mode too!

LIKE (1) REPLY

1 reply

1 more comment...