Mechanistic Interpretability

Explainable and Trustworthy AI

Gabriele Ciravegna

Stages of Explainability – Mechanistic

Explainability involves the entire AI development pipeline

Pre-modelling explainability

Before building the model
- Data exploration
- Data selection
- Feature engineering

Explainable modeling

Build inherently interpretable models
- Manage the accuracy and interpretability trade-off

Post-modelling explainability

After model development
- Explaining predictions and behavior of trained models

Contents

Mechanistic Interpretability in CNN

How Neural Networks Globally Reason Internally?

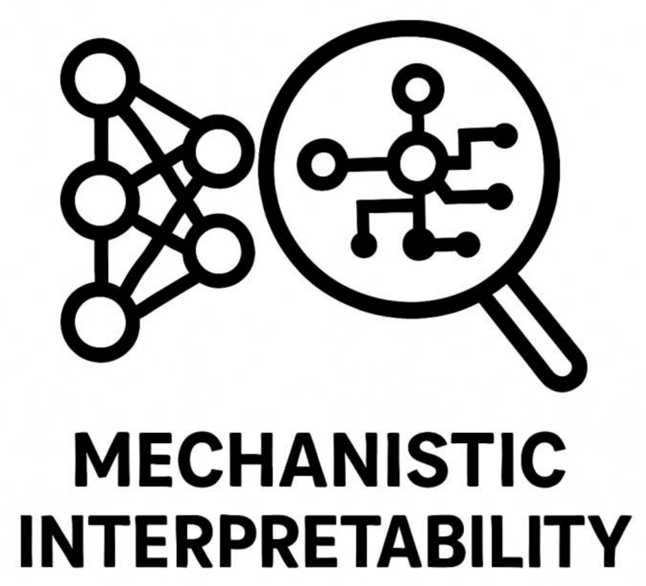
The decision making of NNs is seen often defined **indecipherable:**

- We are unable to interpret NNs inner workings

We can reveal meaningful patterns **zooming in** on:

- Individual neurons
- Their connections

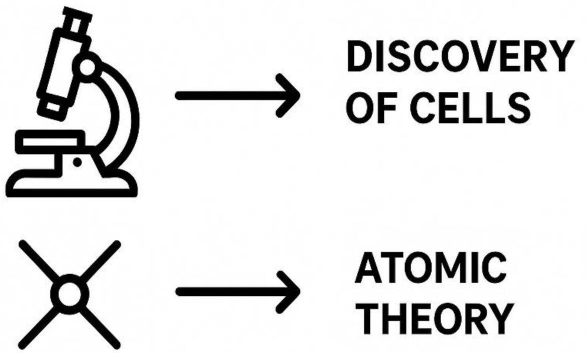
Mechanistic Interpretability helps us break down NN decisions into **globally understandable structures**



The power of *“Zooming In”*[1]

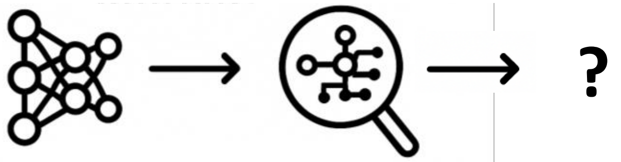
Scientific progress is driven by the ability to zoom into finer details of a given field:

- Microscope --> Discovery of cells --> Cellular Biology
- X-ray cristallography --> DNA structure and molecules --> Atomic Theory



Why don’t we try to do the same with NNs?

- NN visualization --> Computational circuits --> **NN interpretation**?



NNs often studied at a **macro level** but:

- Finer analysis may uncover universal processes within NNs

[1] https://distill.pub/2020/circuits/zoom-in/

Three Speculative Claims

1. **Features**
    - NN representations are composed of **individual features** such as edges, textures, and object parts
    - Each neuron (or combination of neurons) specializes in detecting specific characteristics
2. **Circuits**
    - Neurons form **meaningful interactions**, creating circuits rather than working in isolation
    - **Circuits connect features**, creating complex computations as shape recognition or object segmentation
3. **Universality**
    - Similar **neurons and circuits** appear across **different architectures and tasks**
    - Do certain structures emerge **naturally from training data?**

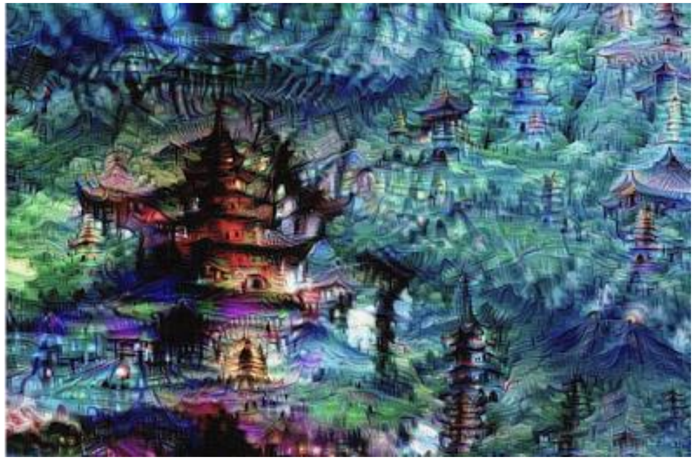
Feature Example - Curve Detectors

Neurons in vision models detect **curves and edges**, a crucial step in visual processing

Supporting evidence:

- **Feature visualization:** Visualizing neurons shows curve patterns
- **Dataset examples:** Neurons activate when encountering curve-related images
- **Synthetic testing:** Creating artificial inputs verifies neuron behavior

Feature Visualization[2]



How?

- Optimizes an input image to strongly activate a specific neuron or layer
- Iteratively adjuste the image until maximazing activation

What do we obtain?

- Generate synthetic representations of the features learnt by the network (neurons or layers)

Deep Dream

- Modifies an existing image to exaggerate the patterns that a NN detects
  - Instead of generating new images from scratch
- Artistic yet insightful look into how models perceive patterns and objects

[2] https://distill.pub/2017/feature-visualization/

Feature Example -Pose-Invariant Dog Head Detector

- Some neurons specialize in recognizing **dog heads from various angles**

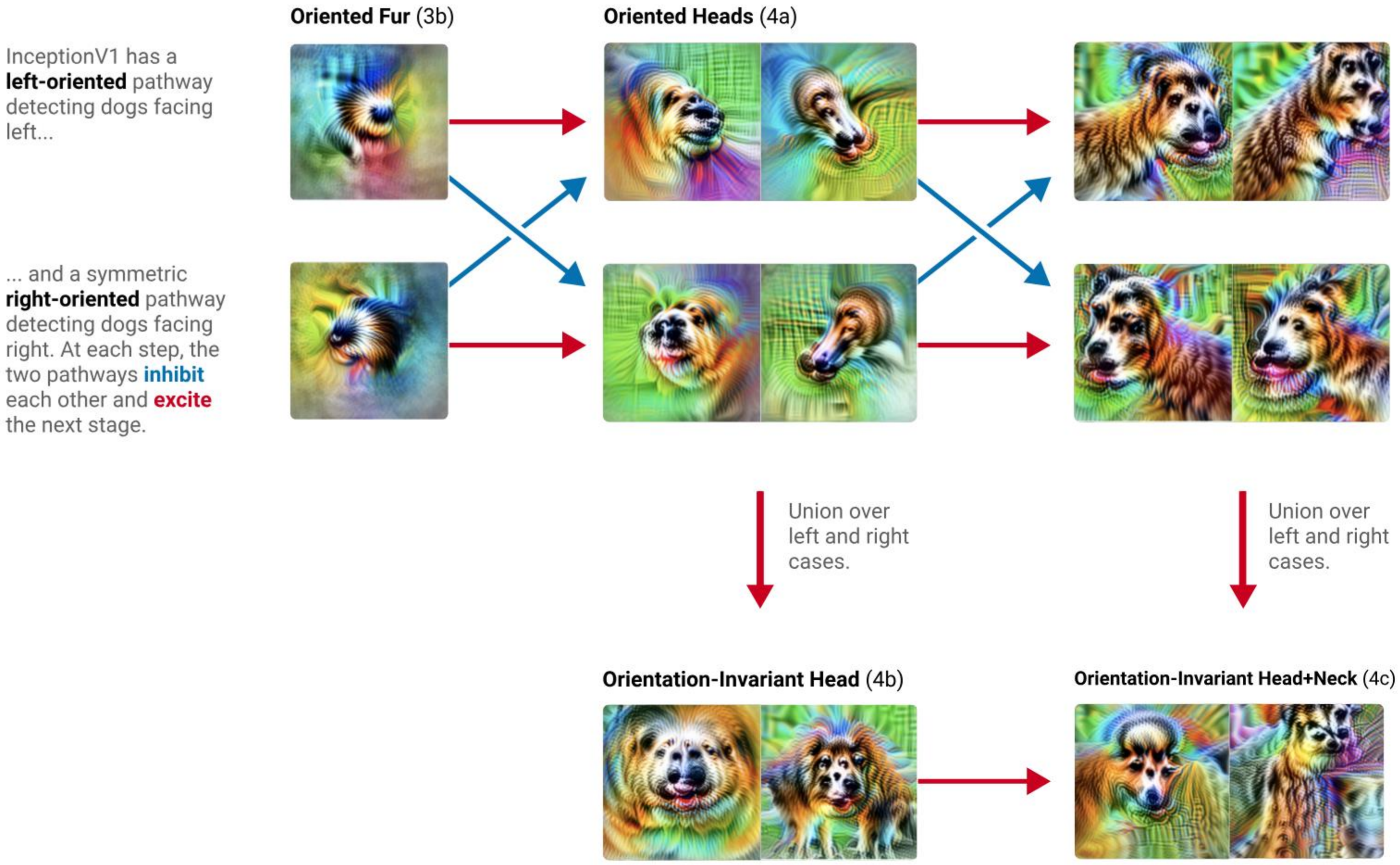
- Networks generalize recognition by forming **“union over cases”**

- This allows models to **learn abstract concepts** beyond pixel patterns

Circuits Motifs in Neural Networks

- Features are connected by weights, forming circuits
- Neural networks develop **consistent circuits**, known as **motifs**:
  - **Equivariance** – Rotation-invariant recognition.
  - **Unioning over cases** – Combining multiple perspectives.
  - **Superposition** – Using neurons efficiently to store information.

Circuit Example: Forming the Pose-Invariant Dog Head Detector



Universality

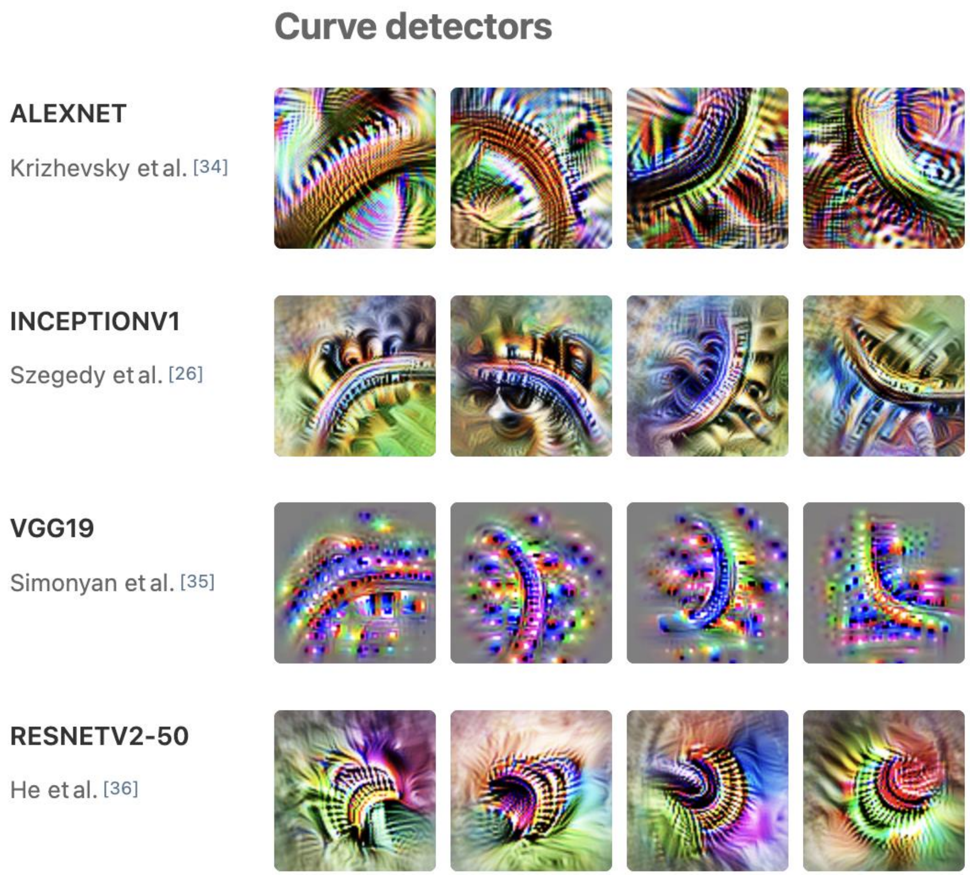
NNs often develop similar features and circuits across **different models and architectures**

Why Universality Matters?

- Suggests neural networks may be converging toward **fundamental computational principles**
- Provides a foundation for **transfer learning**

Examples of Universality:

- **Edge and texture features** appear across all vision model
- **Curve detectors circuits** function similarly in different architectures



Challenges

Polisemanticity

- Some neurons respond **to multiple unrelated features**
- Complicates alignment to human decision-making
- **Superposition:** need to use neurons efficiently to store information

Dataset examples

Required **human annotation to:**

- Inspect neurons
- Create visualizations
- Examine circuits

Universality is not strictly required

- But if it does not hold future research can focus only on **individual models**

Mechanistic Interpretability in LLMs

Different Representation Structure

Lack of Spatial Structure

- Transformers process tokens in a sequence rather than pixels in a grid

  --> We do not have filters that learns patterns tied to specific locations

Dynamic, Complex Interactions

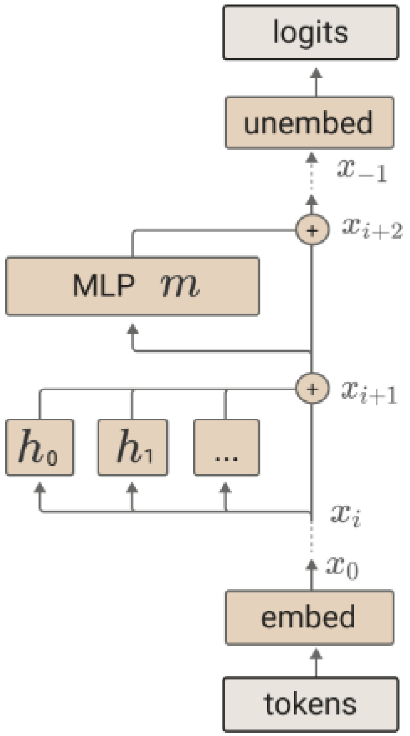
- Attention mechanisms dynamically shift focus based on context
- Transformers learn relationships between tokens

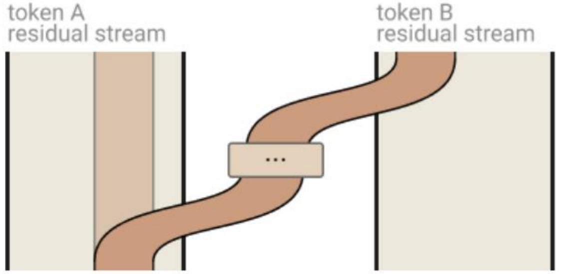
  --> Challenging to attribute a single neuron to a specific feature

  --> Challenging to visualize circuits

Reverse Engineering Transformers

- The Residual Stream
  - Each layer reads its input and writes its output on the *residual stream*
  - Deeper layers do not overwrite previous information but additively build upon it
- Attention heads are independent and additive
  - Attention heads move information from one token to the other (residual stream)
  - Their contribution can be considered quasi-linear and additive
    - only the attention score with the softmax is non linear



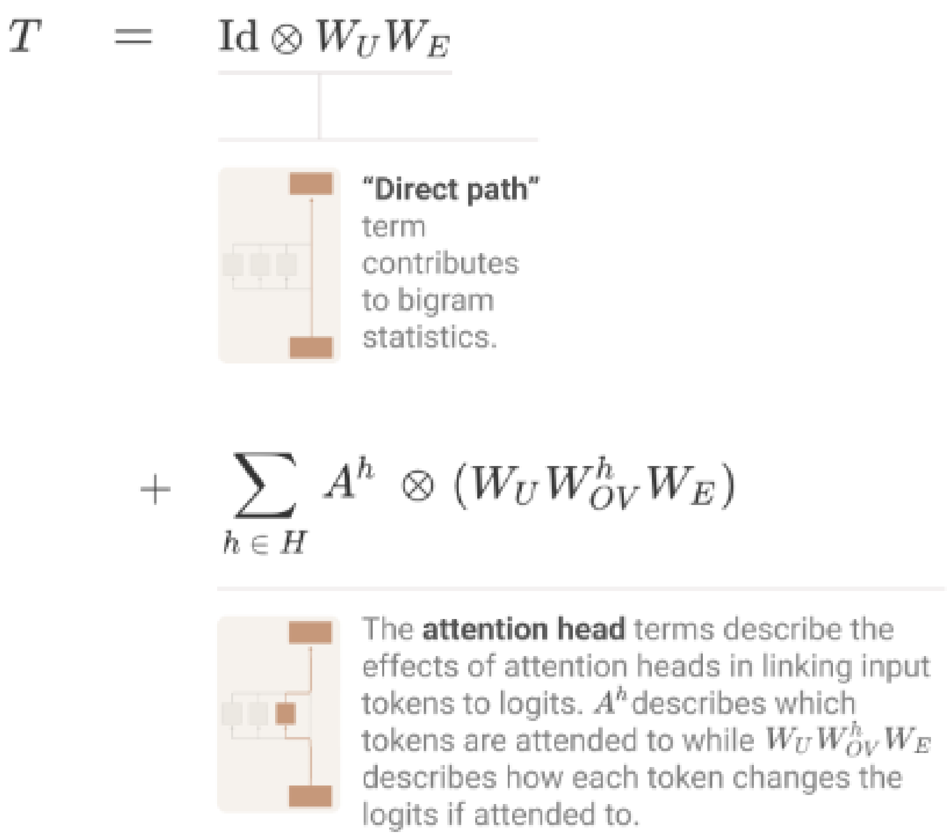


Analysis of Simplified Transformers[3]

- Zero-Layer Transformer
  - No information moves across tokens

  --> it predicts the next token purely based on bigram (i.e. «token pairs») statistics

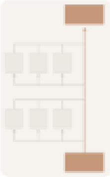
- One-Layer Transformer
  - Keep bigram probabilities (from residual stream)
  - It introduces skip-trigrams
    - Patterns "A...BC": a token earlier in the sequence (A) influences a later prediction (C) despite being separated by (one or more) token B
    - Attention heads selectively attend to earlier tokens to modify predictions accordingly



[3] https://transformer-circuits.pub/2021/framework/index.html

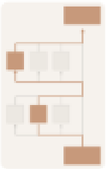
Analysis of Simplified Transformers (2)

$$T = \mathrm{Id} \otimes W_U W_E$$



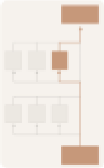
"Direct path" term contributes to bigram statistics.

$$+ \sum_{h_2 \in H_2} \sum_{h_1 \in H_1} (A^{h_2} A^{h_1}) \otimes (W_U W_{OV}^{h_2} W_{OV}^{h_1} W_E)$$



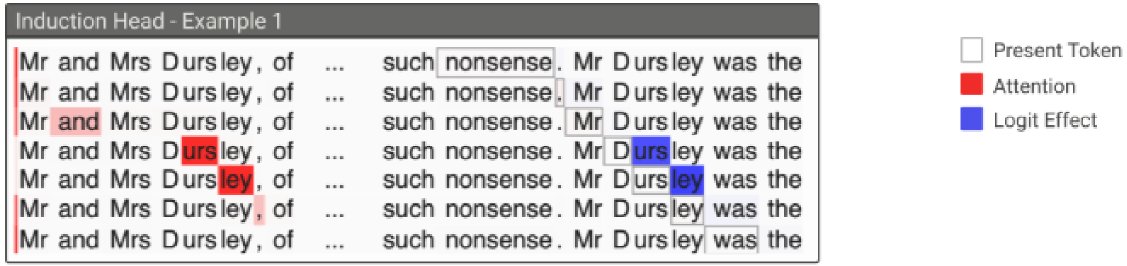
The **virtual attention head** terms correspond to V-composition of attention heads. They function a lot like individual attention heads, with their own attention patterns (the compositon of the heads patterns) and own OV matrix.

$$+ \sum_{h \in H_1 \cup H_2} A^h \otimes (W_U W_{OV}^h W_E)$$



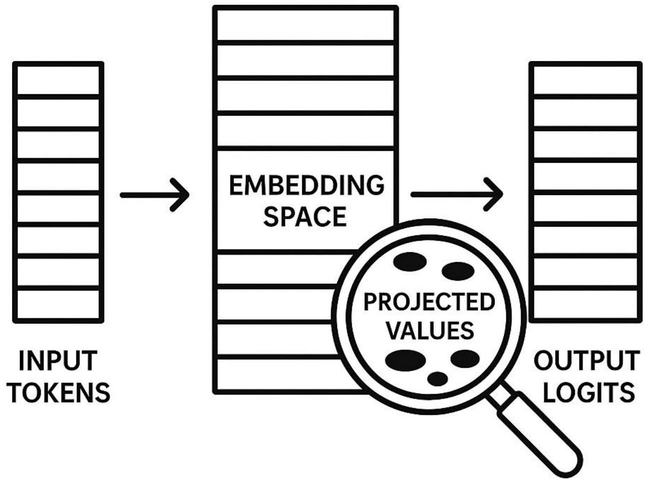
The **individual attention head** terms describe the effects of individual attention heads in linking input tokens to logits, similar to those we saw in the one layer model.

- Two Layer Transformer are composed of:
  - Direct path contributing to next token statistics
  - Virtual and Individual attention heads behaving similarly
    - Virtual attention heads are the linear multiplication of several attention heads
  - Much higher in-context learning by «Induction heads»



Transformer Matrix dimensions

- Embedding step:
  - Size: Nv × Ne (50257 × 1600 for GPT-2 1558M)
  - Purpose: From the vocab space (Nv) into the embedding space (Ne)
- Transformer Blocks:
  - Processing: The 1600-dimensional vector moves through all transformer layers (blocks)
  - Each Block's Output: Another 1600-dimensional vector, progressively refining the representation
- Un-Embedding Step (Final Projection):
  - Size: Ne × Nv (1600 × 50257)
  - Purpose: Transforms the processed 1600-dimensional vector back into vocab space (Nv)

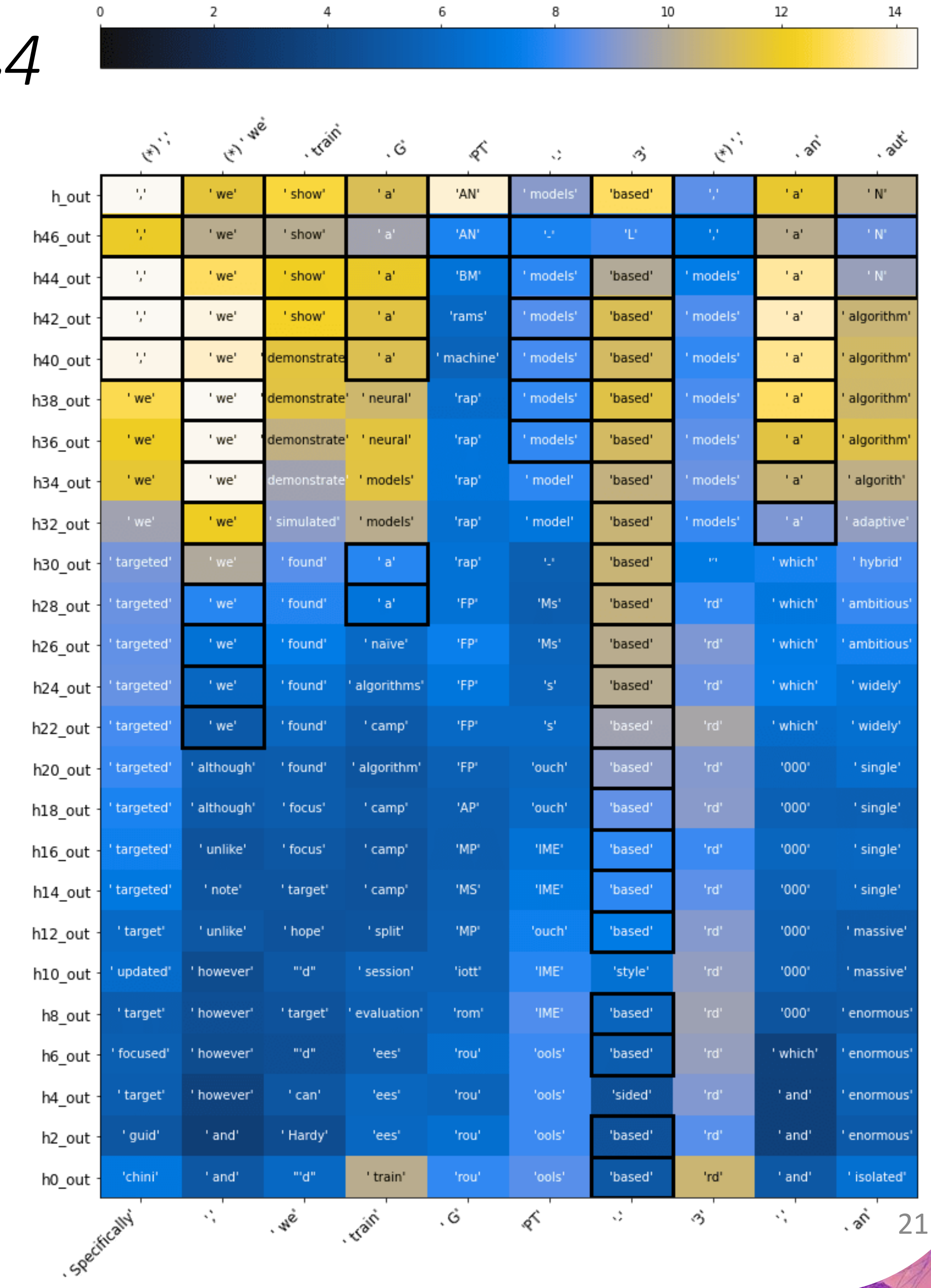


--> We can analyze how the residual stream evolves through the layers

- By appling the *Un-Embeedding* step of the output layer to any intermediate layer!

Analysis under the *Logit Lens*[4]

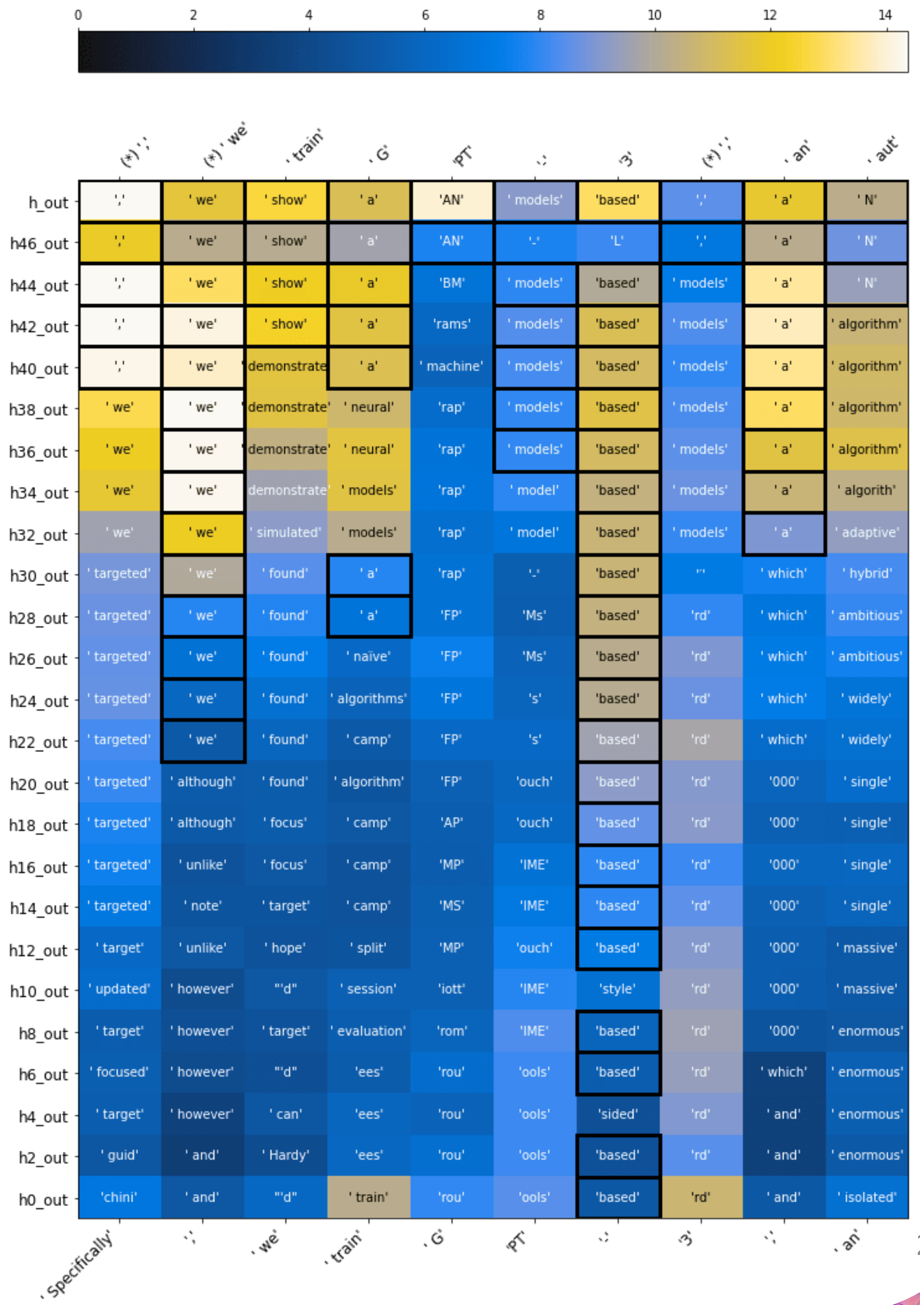
- **Input**: A segment of the GPT-3 paper's abstract
  - Preceding text available but not explicitly visualized
- **Output**: Token predictions (h_out) and correct labels:
  - Correct label: the token following the current input token
  - Correct predictions (*) where the model's top guess matches the expected output
  - DISCLAIMER: Even in output, many predictions are wrong. It does not matter, GPT2 was an early LLM, the underlying process is still valid
- **Intermediate values:** Color-coded **logits**
  - Logits increase as the model refines its predictions and improves its certainty



[4]https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens

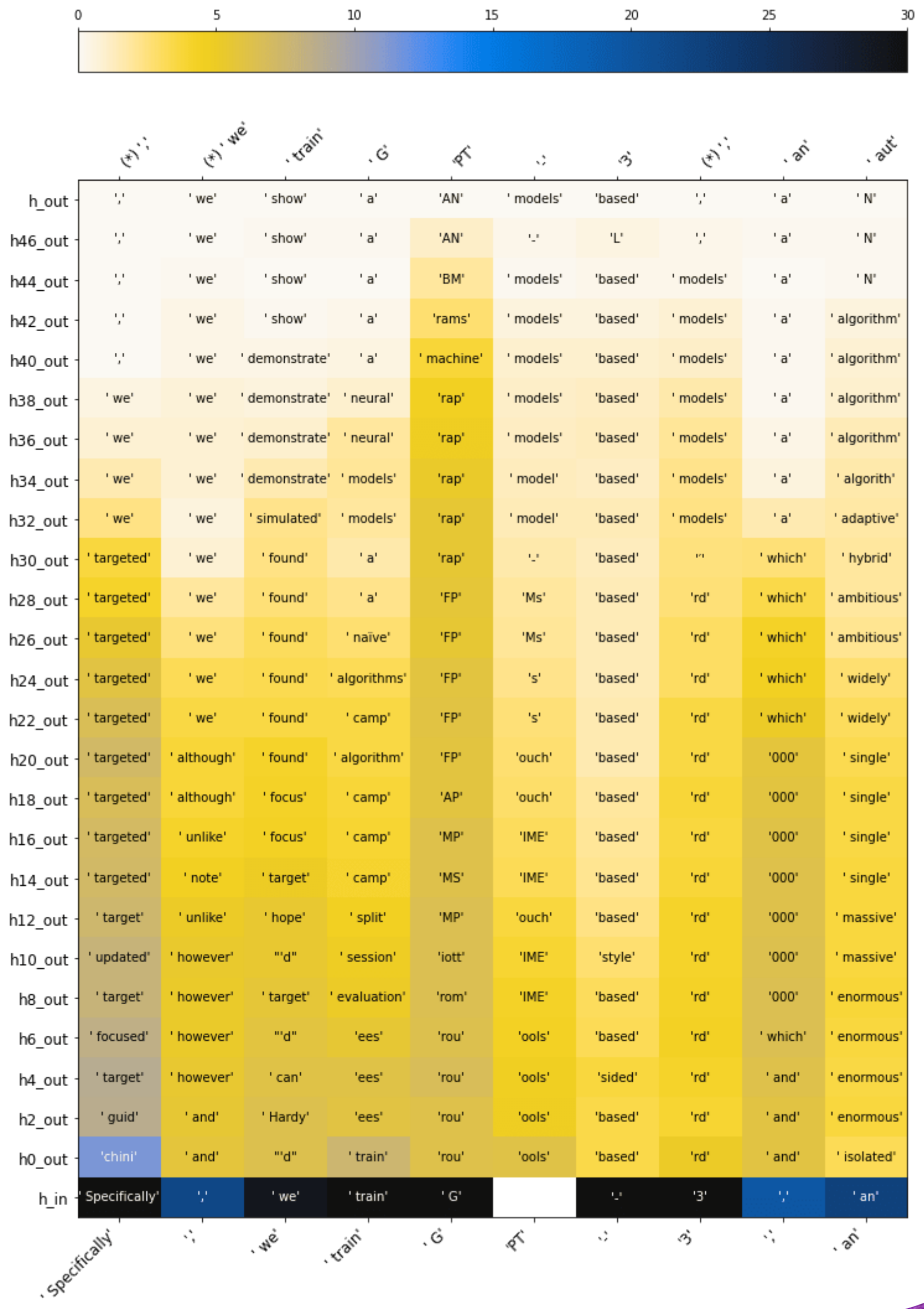
Observations

- GPT "early guesses" are generally wrong but often sensible enough in some way:
  - "We train GPT-3..." *000?* (someday!)
  - "GPT-3, an..." *enormous? massive?* (not wrong!)
- Some early predictions look noisy but gradually become coherent
  - "We train GPT-3, an aut..." *oreceptor?* (later converges to the correct *oregressive*)
- The logit lens reveals **how each step contributes** to the final output



KL divergence plot

- The KL Divergence measure the similarity between two distributions
  - The current predicted output distribution
  - The correct output label distribution
- Input token information is **quickly discarded** after the first layer
  - Inputs are transformed immediately rather than preserved for gradual processing
  - Later layers refine guesses **without keeping direct input reference**
- GPT works more like an iterative **predictive space** refiner rather than an input processing model



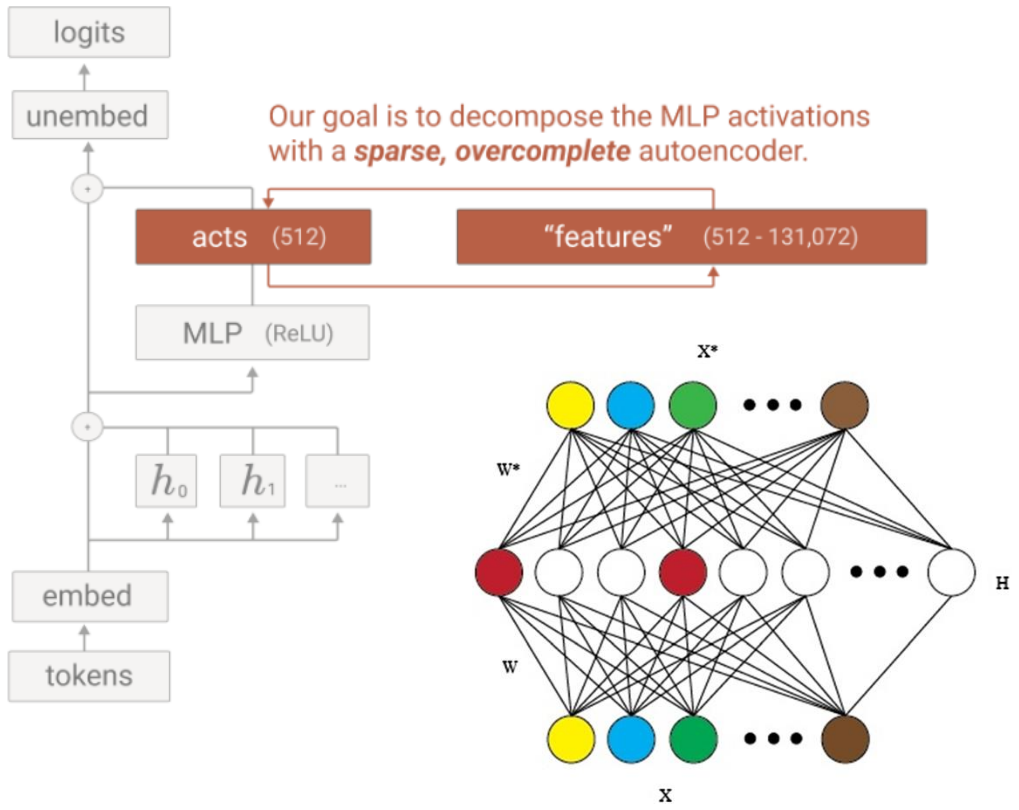
Addressing the Polisemanticity Issue

- Mechanistic interpretability seeks to break neural networks into simpler, understandable components
- **BUT:** Neurons are often polysemantic -- they activate for multiple unrelated concepts
  - E.g., The neuron in Inception v1 responding to both **cat faces and car fronts**.
  - In LLM a neuron can fire for **academic citations, HTTP requests, and Korean text**.

- **Polysemanticity complicates interpretability**, making it hard to assign clear functions to neurons.
- Researchers believes **dictionary learning** may allow extract **monosemantic features**, improving transparency

Towards Monosemanticity: Decomposing Language Models With Dictionary Learning[5]

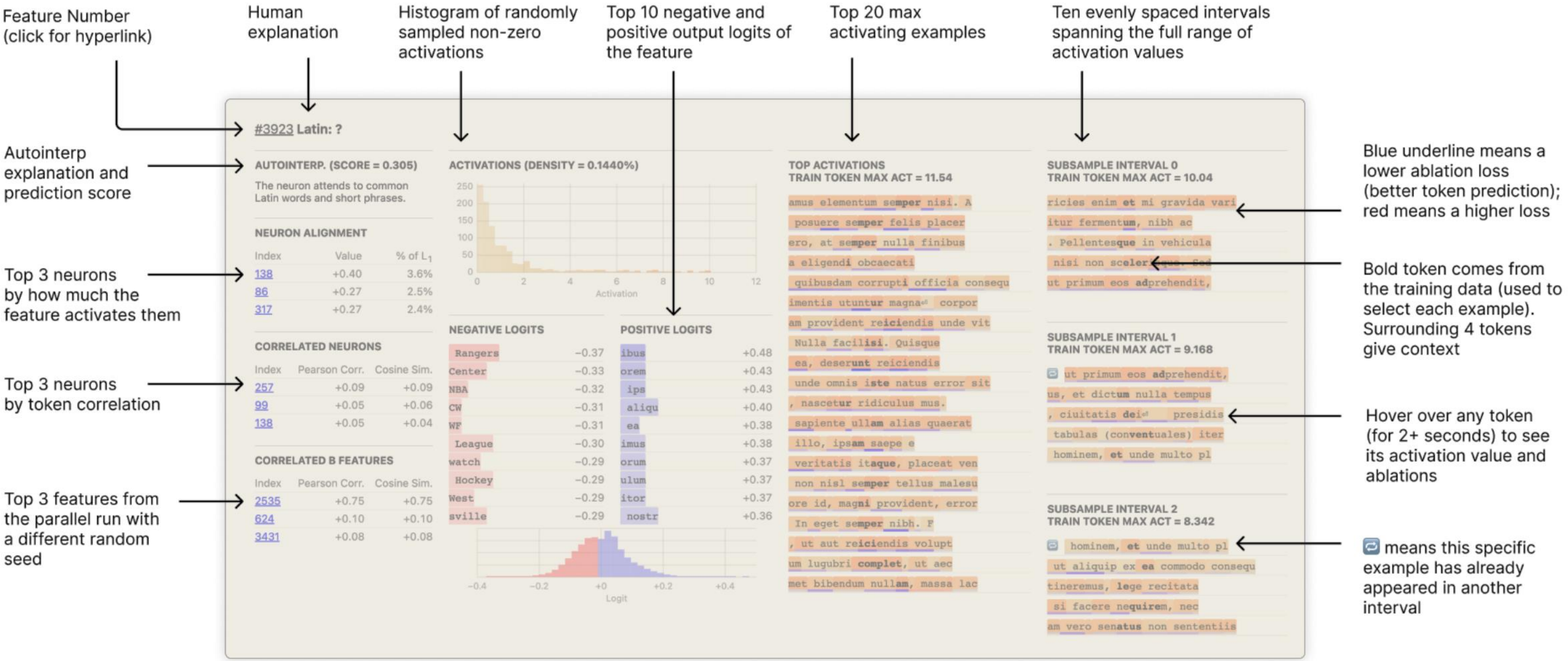
- Feature-Based Decomposition:
  - Instead of analyzing neurons
  - We analyze the entire activations
  - We decompose it into single general **features**
  - It can be applied on **top of MLP** as well!
- Sparse AutoEncoders (SAE)
  - Represent activations as a combination of distinct features instead of single neuron responses
  - Feature expansion: SAE hidden dimensions >> input dimension (8x)
  - Allows to decompose MLP representation and **avoid superposition**



- $Loss_{SAE} = |W_{DEC}(MLP(x)W_{ENC}) - MLP(x)|$

[5] https://transformer-circuits.pub/2023/monosemantic-features/index.html

Let’s find features within an LLM!



Let’s find features within an LLM!

https://transformer-circuits.pub/2023/monosemantic-features/vis/a1.html?ordering=count&search_text=food

Let's see how to use these features to analyze a text

https://transformer-circuits.pub/2023/monosemantic-features/vis/a1-en.html