

# AI/large language models and interpretability

---

Complex Systems 530 - Marisa Eisenberg

# Large language models

---

- ChatGPT (UM's version: <http://umgpt.umich.edu>)
- Llama
- Bard

# How are large language models (LLMs) like Chat-GPT structured?

---

- GPT = Generative Pretrained Transformer
- Generative: generates the next word
- Pretrained: what it sounds like, i.e. trained ahead of time
- Transformer: the type of neural network model that many LLMs use
  - Built to recognize and work with sequences of inputs to generate the next thing (you can do this with other stuff too, e.g. images —> video, genetic sequences, etc.)

# LLMs

---

- Transformer models - how they work
- What are they trained on, what are they trained to do
- What they do/don't do



# How are large language models (LLMs) like Chat-GPT structured?

---

- LLMs are built to predict the next word in a sequence

“Once upon a \_\_\_\_”

seems like it should fill in the blank with “time” but not “armadillo”.

$$P(\textit{time} \mid \textit{once}, \textit{upon}, \textit{a})$$

$$P(\textit{word}_n \mid \textit{word}_1, \textit{word}_2, \dots, \textit{word}_{n-1})$$

- Note the input sequence in a chat-GPT session is the whole conversation so far

# Text example, temperature

---

- Chat-GPT 2
- Input: “Complex systems is a fascinating”

- Top 5 most probable outputs:

and	6.8%
concept	5.4%
subject	3.6%
topic	3.5%
way	3.2%

- But if you just take the most probable output every time?

# You get sort of repetitive nonsense

---

```
In[10]:= NestList[StringJoin[#, model[#, "Decision"]] &, "Complex systems is a fascinating", 15]
```

```
Out[10]= {Complex systems is a fascinating, Complex systems is a fascinating and,  
Complex systems is a fascinating and complex,  
Complex systems is a fascinating and complex system,  
Complex systems is a fascinating and complex system.,  
Complex systems is a fascinating and complex system. It,  
Complex systems is a fascinating and complex system. It is,  
Complex systems is a fascinating and complex system. It is a,  
Complex systems is a fascinating and complex system. It is a system,  
Complex systems is a fascinating and complex system. It is a system that,  
Complex systems is a fascinating and complex system. It is a system that is,  
Complex systems is a fascinating and complex system. It is a system that is not,  
Complex systems is a fascinating and complex system. It is a system that is not only,  
Complex systems is a fascinating and complex system. It is a system that is not only a,  
Complex systems is a fascinating and complex system. It is a  
system that is not only a system, Complex systems is a fascinating  
and complex system. It is a system that is not only a system of}
```

# You get sort of repetitive nonsense

---

```
In[11]:= StringReplace[Nest[StringJoin[#, model[#, "Decision"]] &,
    "Complex systems is a fascinating", 100], "\n" .. → " "]
```

```
Out[11]= Complex systems is a fascinating and complex system. It is a system that is not only a
system of complex systems, but also a system of complex systems. It is a system
that is not only a system of complex systems, but also a system of complex
systems. It is a system that is not only a system of complex systems, but also a
system of complex systems. It is a system that is not only a system of complex
systems, but also a system of complex systems. It is a system that is not only
```



# You need to inject some randomness (temperature)

---

```
In[12]:= NestList[StringJoin[#, model[#, {"RandomSample", "Temperature" → .8}]] &,
  "Complex systems is a fascinating", 7]
```

```
Out[12]= {Complex systems is a fascinating,
  Complex systems is a fascinating new, Complex systems is a fascinating new area,
  Complex systems is a fascinating new area of,
  Complex systems is a fascinating new area of research,
  Complex systems is a fascinating new area of research.,
  Complex systems is a fascinating new area of research.
, Complex systems is a fascinating new area of research.}
```

```
In[29]:= NestList[StringJoin[#, model[#, {"RandomSample", "Temperature" → .8}]] &,
  "Complex systems is a fascinating", 7]
```

```
Out[29]= {Complex systems is a fascinating, Complex systems is a fascinating way,
  Complex systems is a fascinating way to, Complex systems is a fascinating way to think,
  Complex systems is a fascinating way to think about,
  Complex systems is a fascinating way to think about the,
  Complex systems is a fascinating way to think about the physical,
  Complex systems is a fascinating way to think about the physical and}
```

# Although it can take you in some funny directions depending how you set the level of randomness

---

```
In[32]:= StringReplace[Nest[StringJoin[#, model[#, {"RandomSample", "Temperature" → .8}]] &,
  "Complex systems is a fascinating", 100], "\n" .. → " "]
```

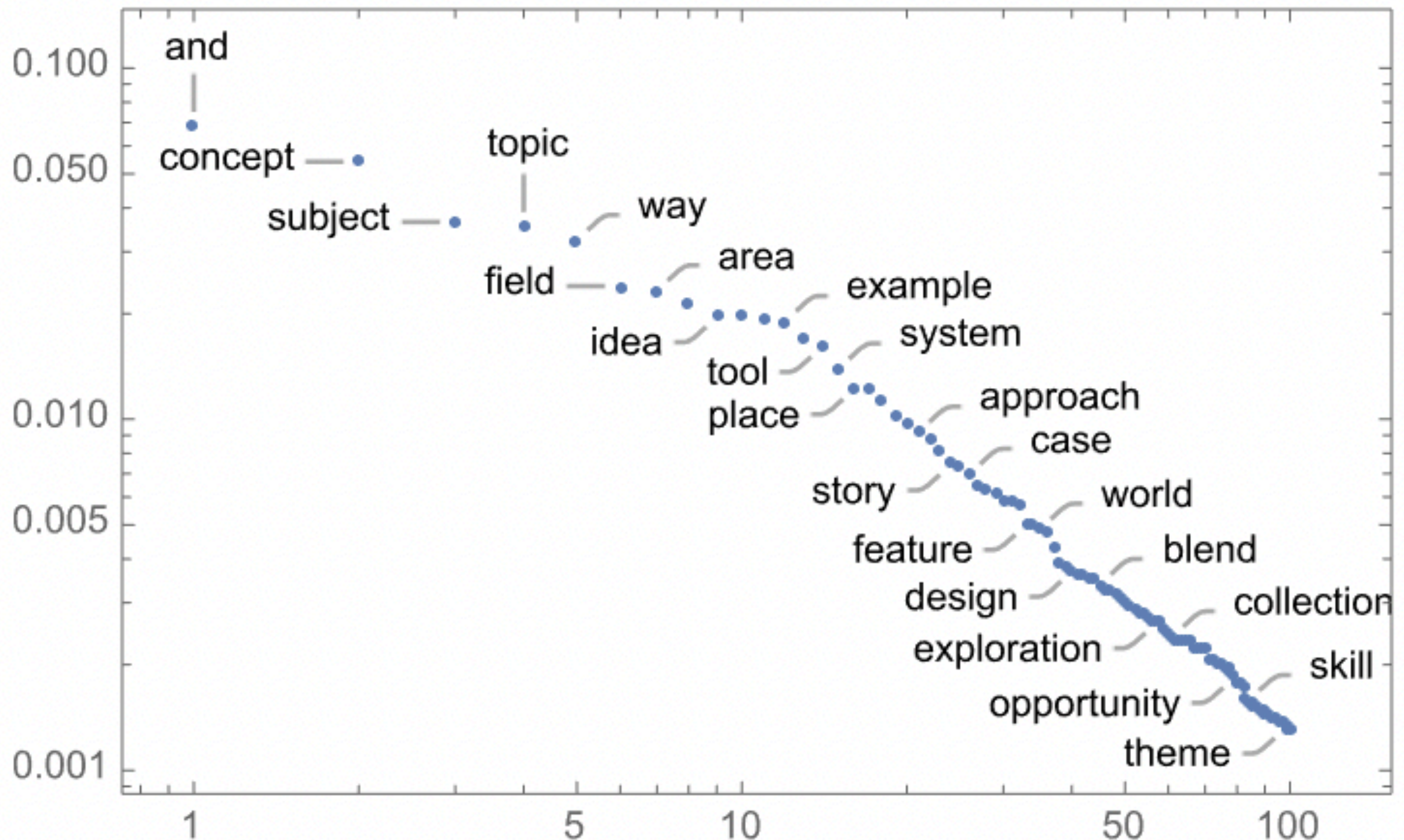
```
Out[32]= Complex systems is a fascinating space, stimulating the imaginative
imagination of the X-Men and the X-Men Unlimited universe. With our brilliant
X-Men Unlimited Team of characters, we're bringing you the fastest, most
creative and fun ways to experience the Marvel Universe. We've launched our
first comic series with X-Men Unlimited, empowering the X-Men Unlimited
franchise in a weird and wonderful new way. We've become a staple in
comic book culture, and we think this is a great time to announce that
```

```
In[33]:= StringReplace[Nest[StringJoin[#, model[#, {"RandomSample", "Temperature" → .8}]] &,
  "Complex systems is a fascinating", 100], "\n" .. → " "]
```

```
Out[33]= Complex systems is a fascinating case study in the revolutionary development of
automation. A few weeks ago, I was the guest of a techno-industrial symposium at TED.
I was at the "tech-filled" stage and talked about the future of automation, entwined
with the work of the American industrialists and the great techno-industrialists who
wanted to make 'better' things. This was an event that was the culmination of a long
process of exploration that was taking place in the past 10 years, starting with the
```

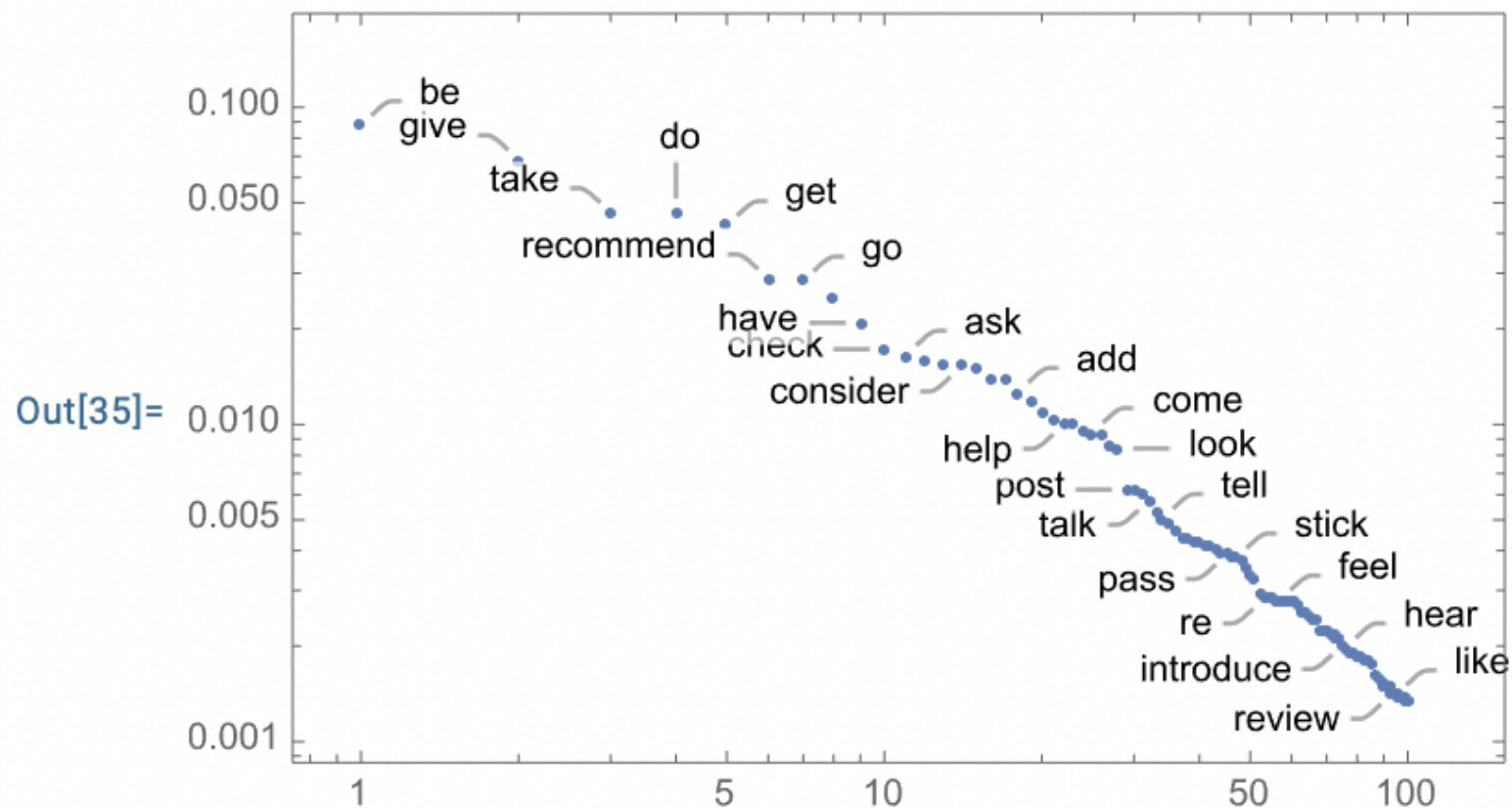


# Distribution of possible next words (notice it gets power-law-y)



# Distribution of possible next words (notice it gets power-law-y)

```
In[35]:= ListLogLogPlot[
  Take[ReverseSort[model["I am very behind on lecture prep so I should definitely",
    "Probabilities"]], 100], Frame → True]
```





# How do you figure out word probabilities?

## Thinking about n-grams

---

- letter n-grams get better as we increase n

0	on gxeeetowmt tsifhy ah aufnsoc ior oia itlt bnc tu ih uls
1	ri io os ot timumumoi gymyestit ate bshe abol viowr wotybeat mecho
2	wore hi usinallistin hia ale warou pothe of premetra bect upo pr
3	qual musin was witherins wil por vie surgedygua was suchinguary outheydays theresist
4	stud made yello adenced through theirs from cent intous wherefo proteined screa
5	special average vocab consumer market prepara injury trade consa usually speci utility

- word n-grams too (random vs 2-gram examples)

of program excessive been by was research rate not here of of other is men  
were against are show they the different the half the the in any were leaved

cat through shipping variety is made the aid emergency can the  
cat for the book flip was generally decided to design of  
cat at safety to contain the vicinity coupled between electric public  
cat throughout in a confirmation procedure and two were difficult music  
cat on the theory an already from a representation before a

# n-grams and word probabilities

---

- Why not just use really long n-grams? Should generate longer sequences of words with the right overall probabilities
- But we don't have nearly enough data! Not enough English text in the world to do this
- Web: few hundred billion words  
Digitized books: ~hundred billion words.

# n-grams and word probabilities

---

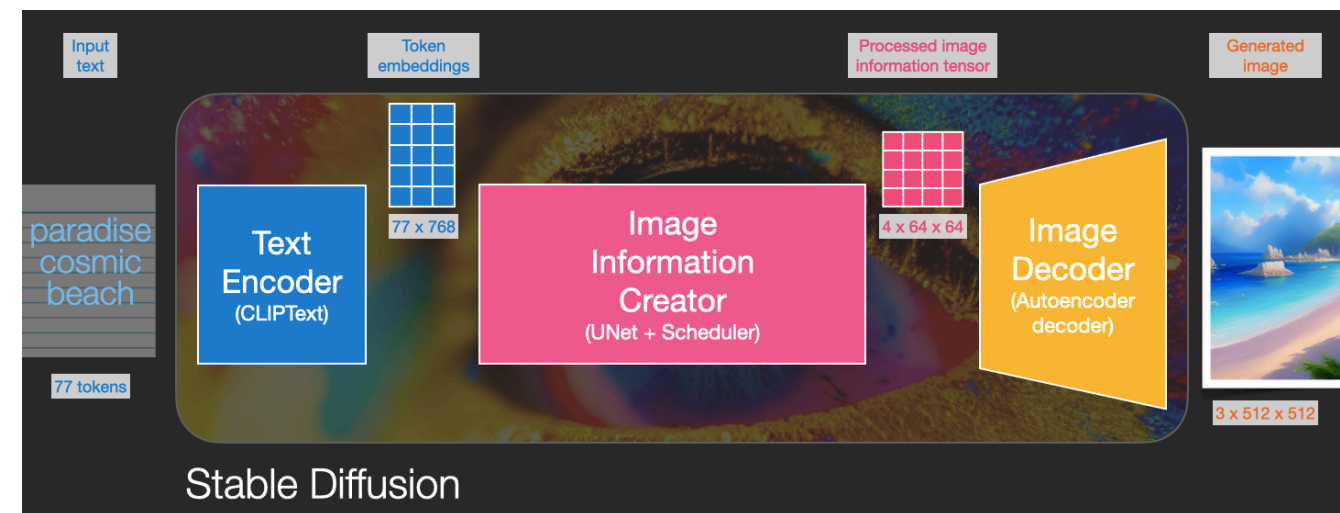
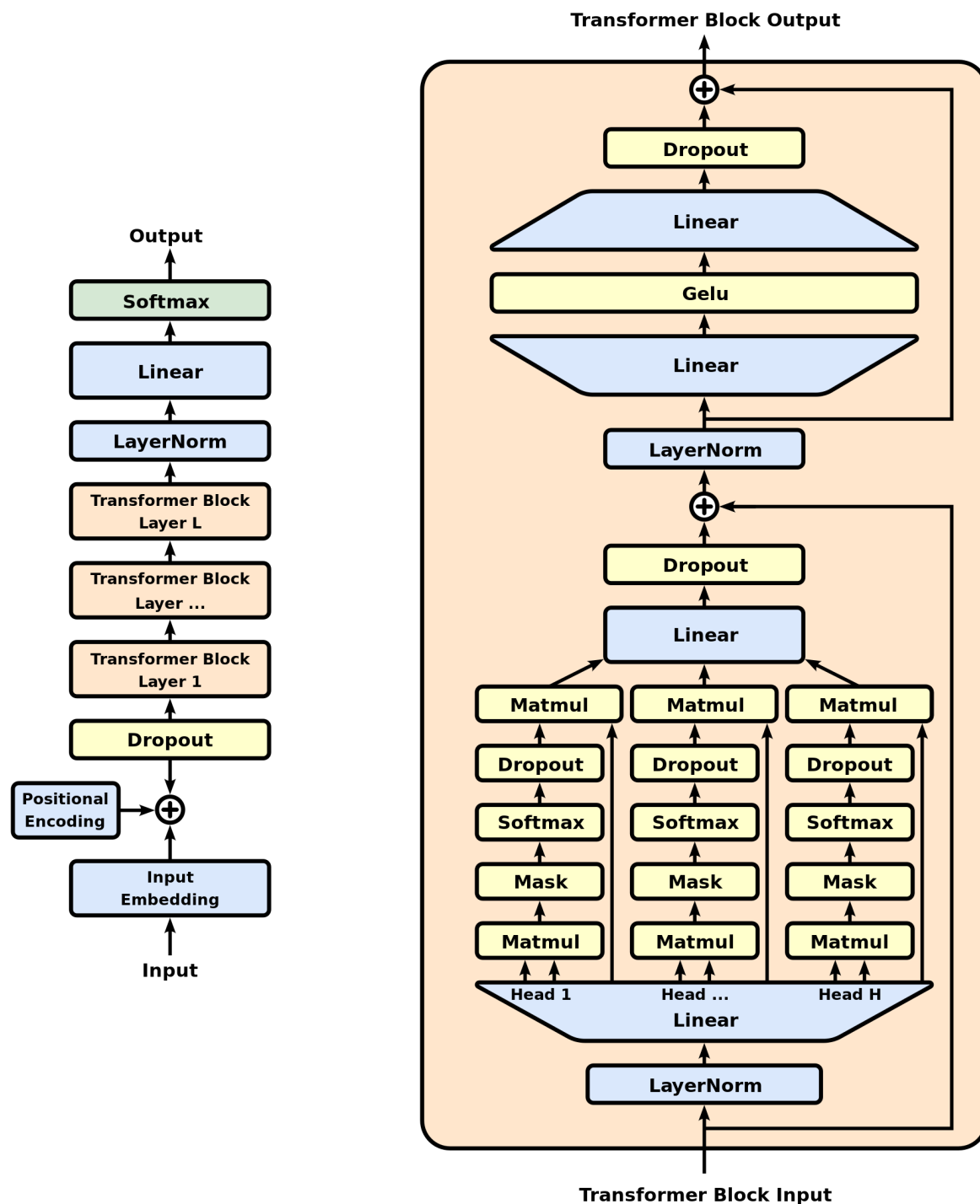
- 40,000 common words  $\rightarrow$  number of 2-grams = 1.6 billion
- Number of possible 3-grams = 60 trillion
- 10 words? 20? More than would ever be possible to write down even if that's all anyone ever did
- Instead build a model that lets us estimate the sequence probabilities even if we've never seen that exact input string—this is what LLMs do

Okay, so how do we do it? Usually something like:

---

- **Tokenize**: get the data in a numerical form we can work with
- **Embed/encode** the inputs
- **Process** them (attention, transformers, etc) - various blocks of neural networks here
- **De-embed/decode**
- Spit out the next word you chose

# Modular structure (blocks) in large AI models



# Tokens

---

- There are ~50,000 commonly used words in the English language
- So we can take a piece of text and represent it as a sequence of numbers that encode which word
- But we do something slightly more efficient—tokens rather than words (this is part of why LLMs can sometimes produce weird made up words)

# Tokens

---

```
prompt <- "The best way to attract bees"  
tokenizer$tokenize(prompt)
```

```
tf.Tensor([ 1  450 1900  982  304 13978  367  267], shape=(8), dtype=int32)
```

```
prompt |> tokenizer$tokenize() |> tokenizer$detokenize()
```

```
tf.Tensor(b'The best way to attract bees', shape=(), dtype=string)
```

```
show_tokens(prompt)
```

1	450	1900	982	304	13978	367	267
""	"The"	"best"	"way"	"to"	"attract"	"be"	"es"

# More frequent tokens get lower IDs

## Token examples

```
show_tokens("ing")
```

```
1 2348
"" "ing"
```

```
show_tokens("working")
```

```
1 1985
"" "working"
```

```
show_tokens("flexing")
```

```
1 8525 292
"" "flex" "ing"
```

```
show_tokens("wonking")
```

```
1 2113 9292
"" "won" "king"
```

```
show_tokens(seq(50, len = 10))
```

```
50 51 52 53 54 55 56 57 58 59
"/" "0" "1" "2" "3" "4" "5" "6" "7" "8"
```

```
show_tokens(seq(100, len = 10))
```

```
100 101 102 103 104 105 106 107 108 109
"a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

```
show_tokens(seq(1000, len = 10))
```

```
1000 1001 1002 1003 1004 1005 1006 1007 1008 1009
"ied" "ER" "stat" "fig" "me" "von" "inter" "roid" "ater" "their"
```

```
show_tokens(seq(10000, len = 10))
```

```
10000 10001 10002 10003 10004 10005 10006 10007
"ång" "citep" "Ill" "rank" "sender" "beim" "pak" "compat"
10008 10009
"occurs" "diese"
```

```
show_tokens(seq(20000, len = 10))
```

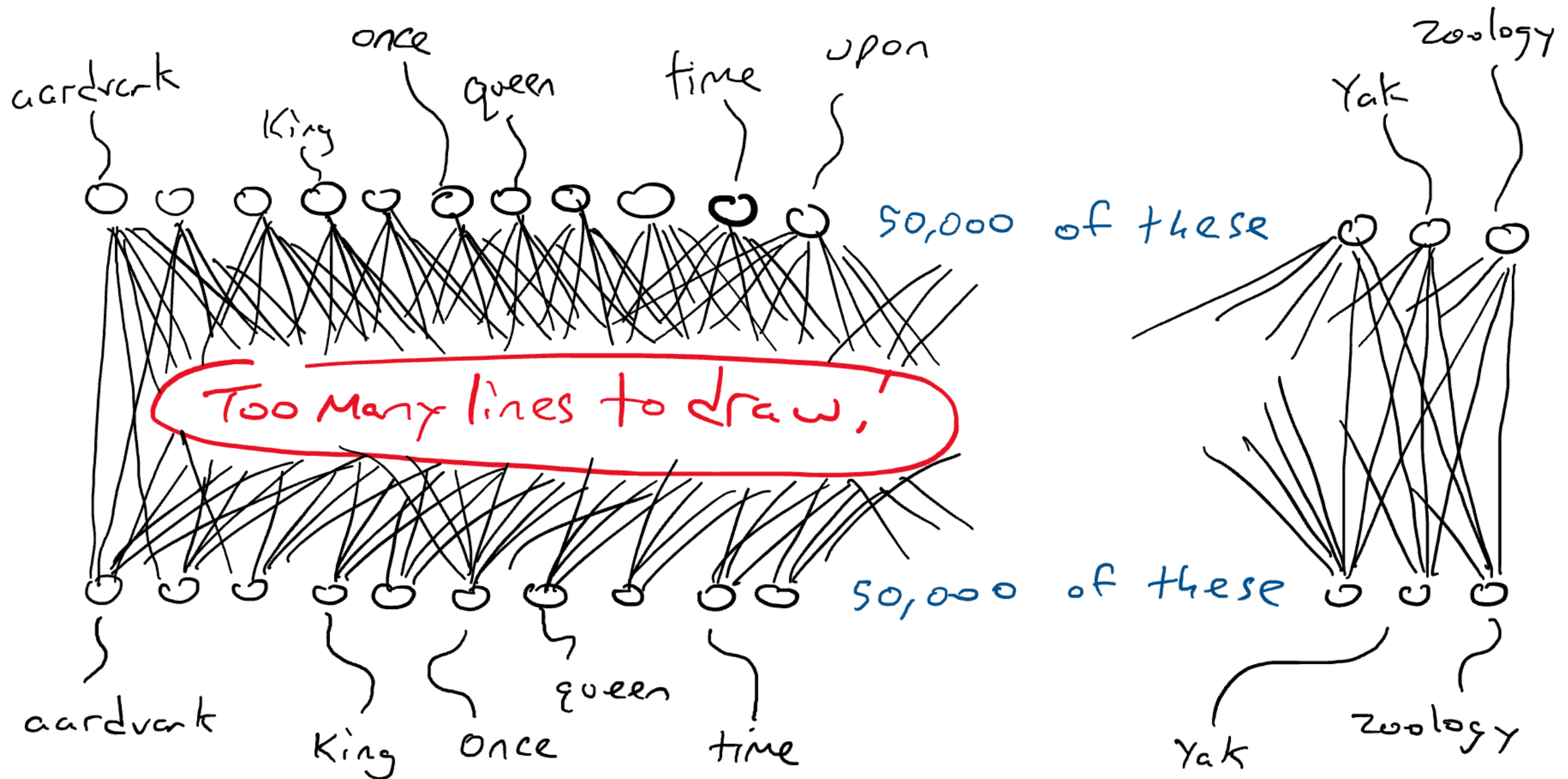
```
20000 20001 20002 20003 20004 20005 20006 20007
"admit" "Comment" "cтя" "Vien" "қи" "permut" "cgi" "crit"
20008 20009
"Console" "ctic"
```

```
show_tokens(seq(to = as.integer(tokenizer$vocab_size()) - 1, len = 10))
```

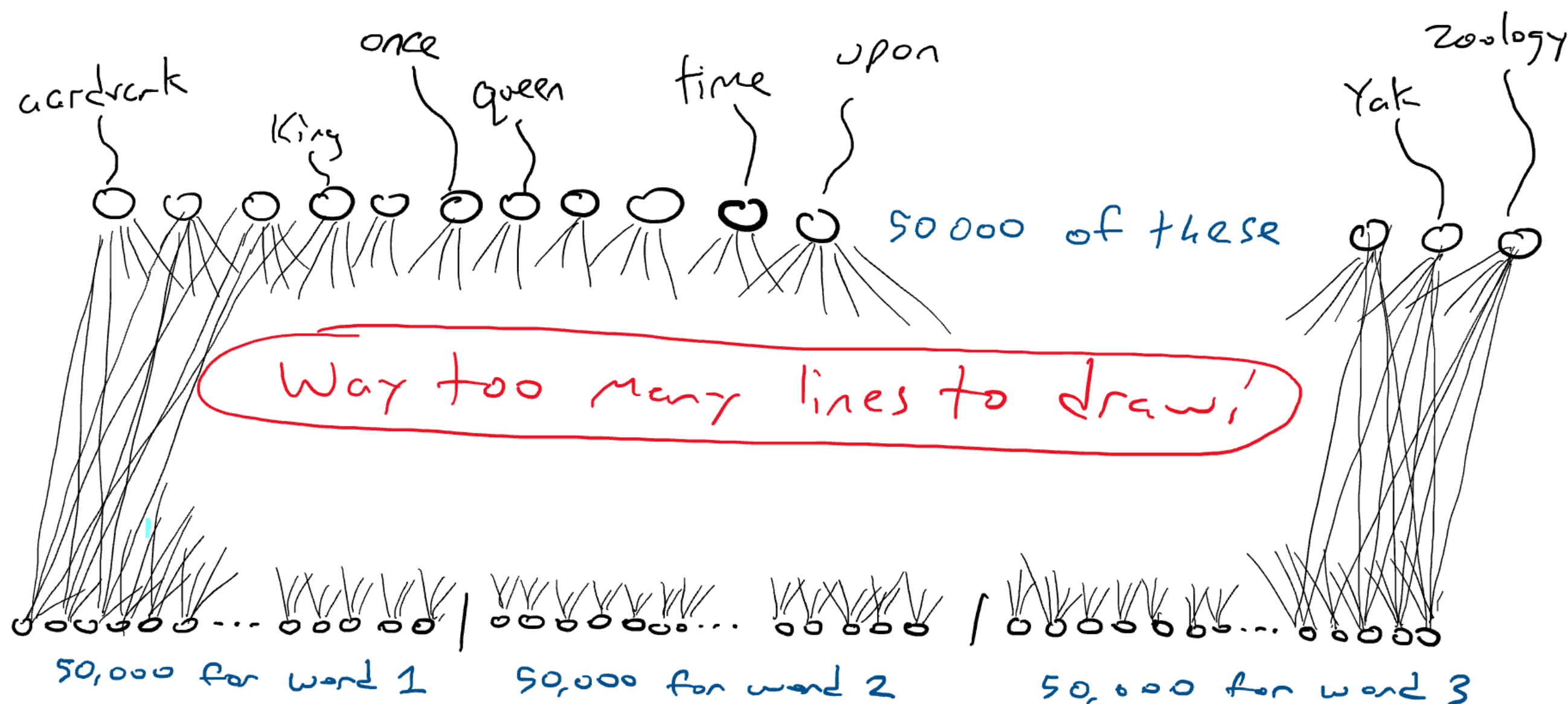
```
31990 31991 31992 31993 31994 31995 31996 31997 31998 31999
"으" "げ" "べ" "辺" "还" "黄" "왕" "收" "弘" "给"
```



# Encoding/embedding: importance of dimension reduction



# Even worse for longer input sequences

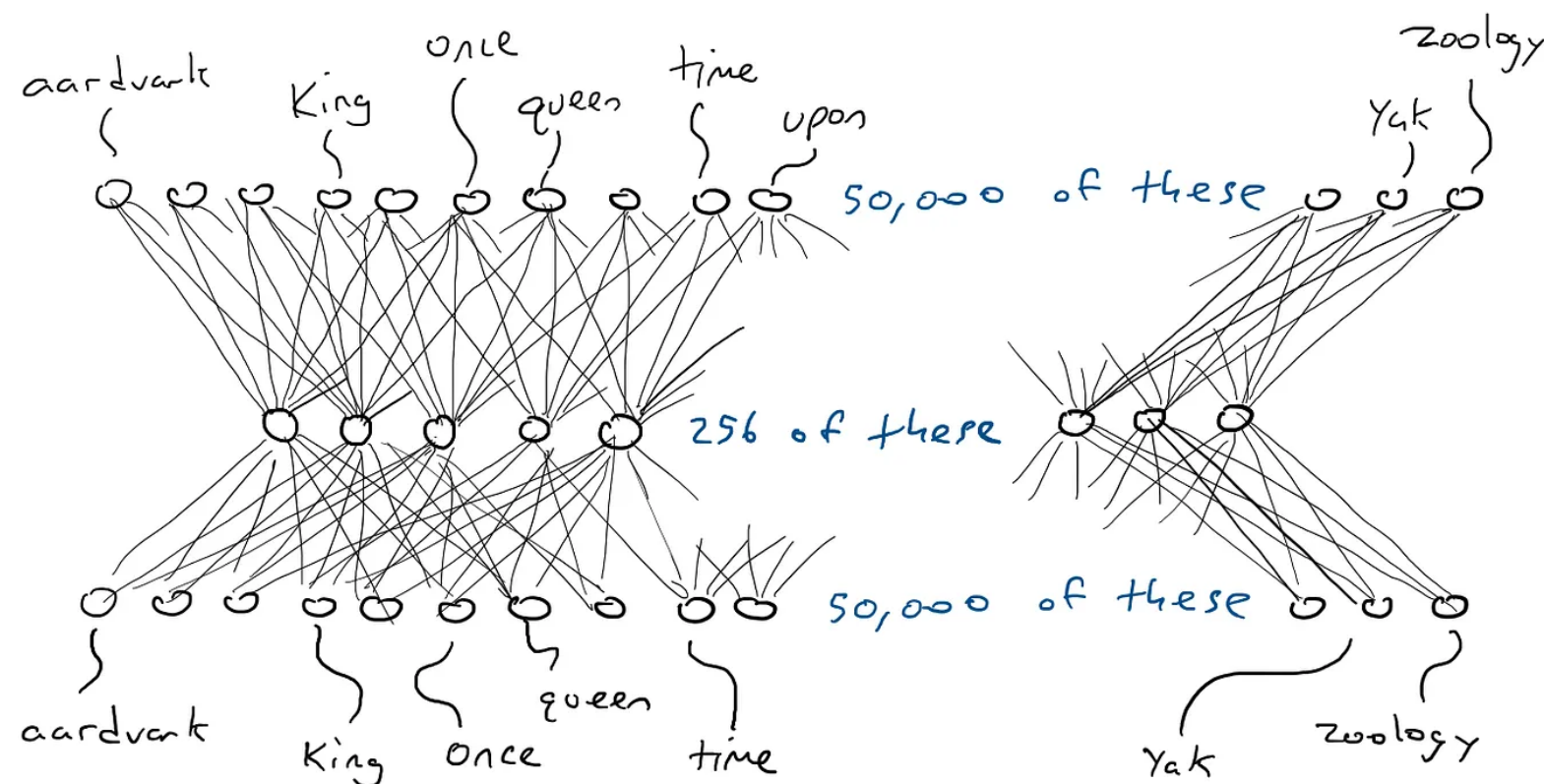


# But many things functionally act the same in human language—we don't need such high dimension

- Anything that means royalty can probably lead to the same word (e.g. throne vs. chair/toilet/horse/etc)

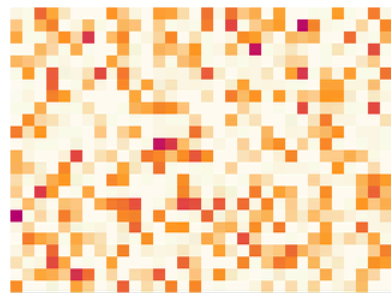
The king sat on the \_\_\_  
The queen sat on the \_\_\_  
The princess sat on the \_\_\_  
The regent sat on the \_\_\_
- This is part of how LLMs learn to understand language!

- Forcing the model to reduce to a lower dimension means we learn common structures

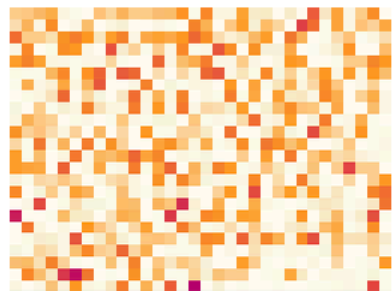


# Embedding helps capture meaning

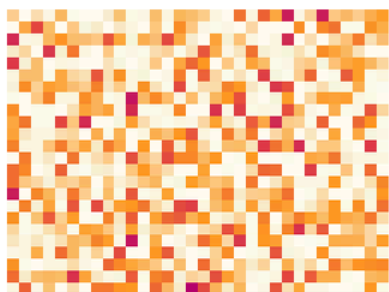
---



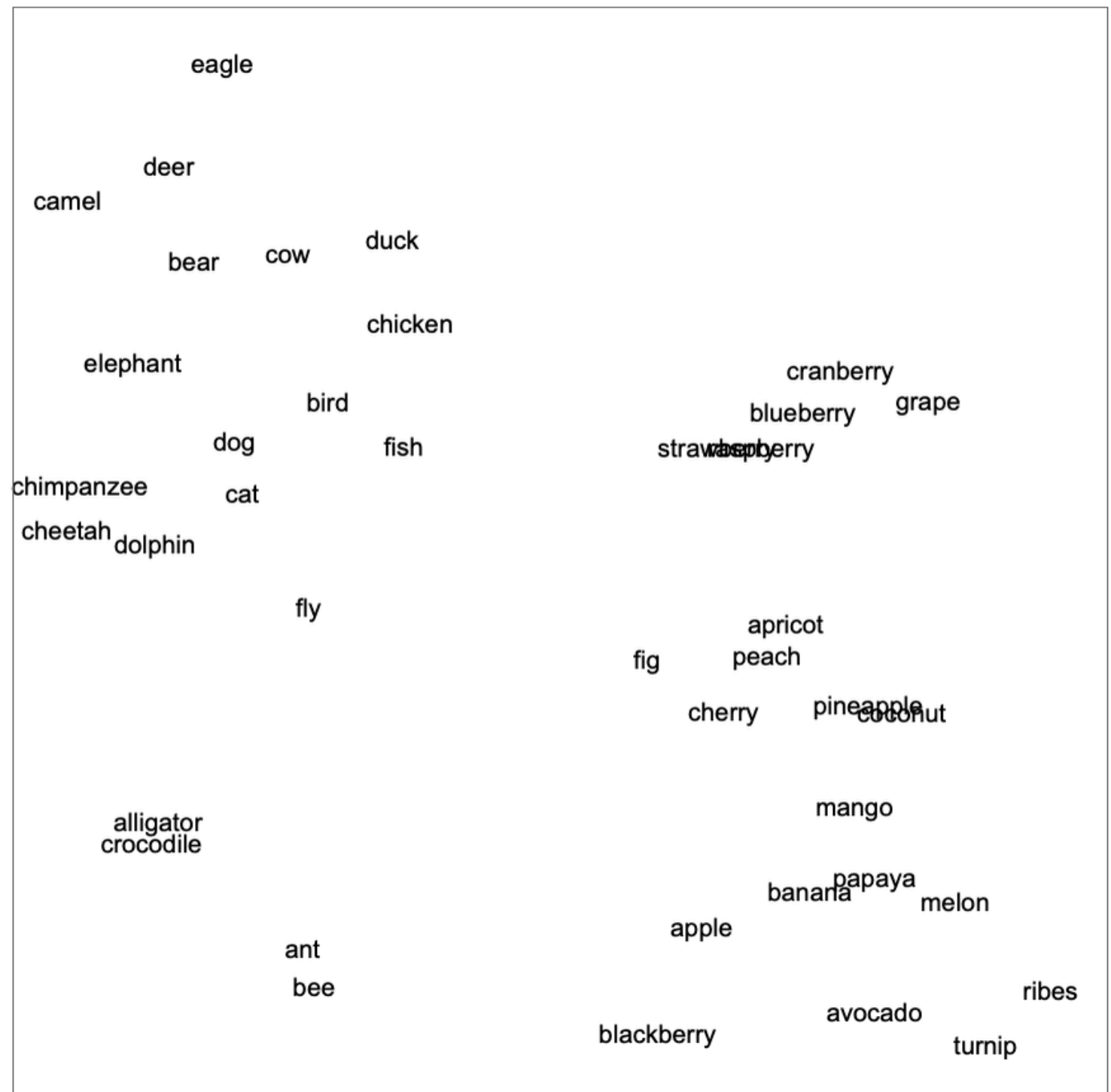
cat



dog



chair



# Embedding

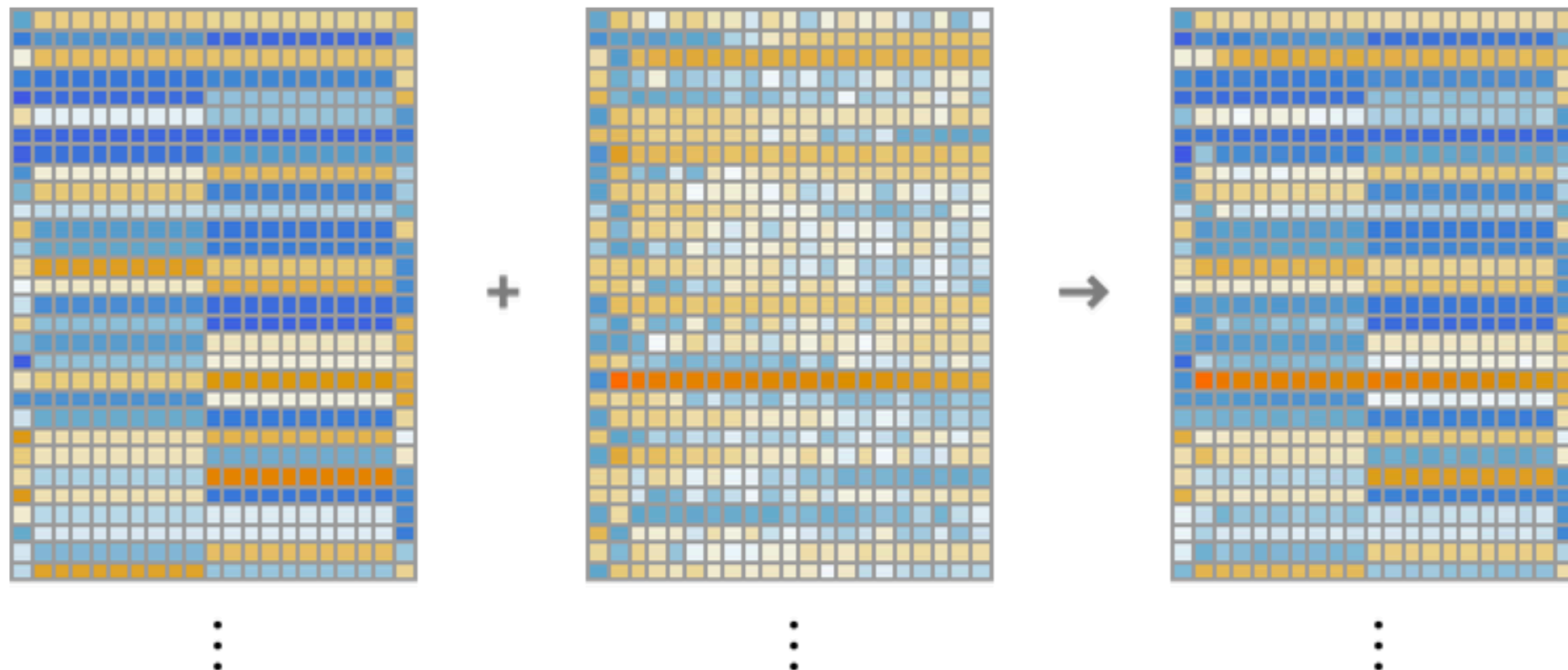
---

- How much did we crunch things down?
- 50K token options are converted (by a single-layer neural net) into an embedding vector of length 768 for GPT-2 and 12,288 for ChatGPT's GPT-3
- Knowing when to reduce dimension and when not to is part of the art that makes it not just a universal approximator
- E.g. compare to digits example in <http://neuralnetworksanddeeplearning.com/chap1.html> (could do four output neurons but works much better with 10)

# Positional encoding

---

- Keep track of both the what the words are and their position relative to each other
- Process those two pieces of information together
- Example: hello hello hello hello hello hello hello hello hello hello bye bye bye bye bye bye bye



# Transformer blocks

---

- Process the embedded text input
- Made up of several sub-layers
  - Attention heads alternating with feedforward networks



# Transformer blocks

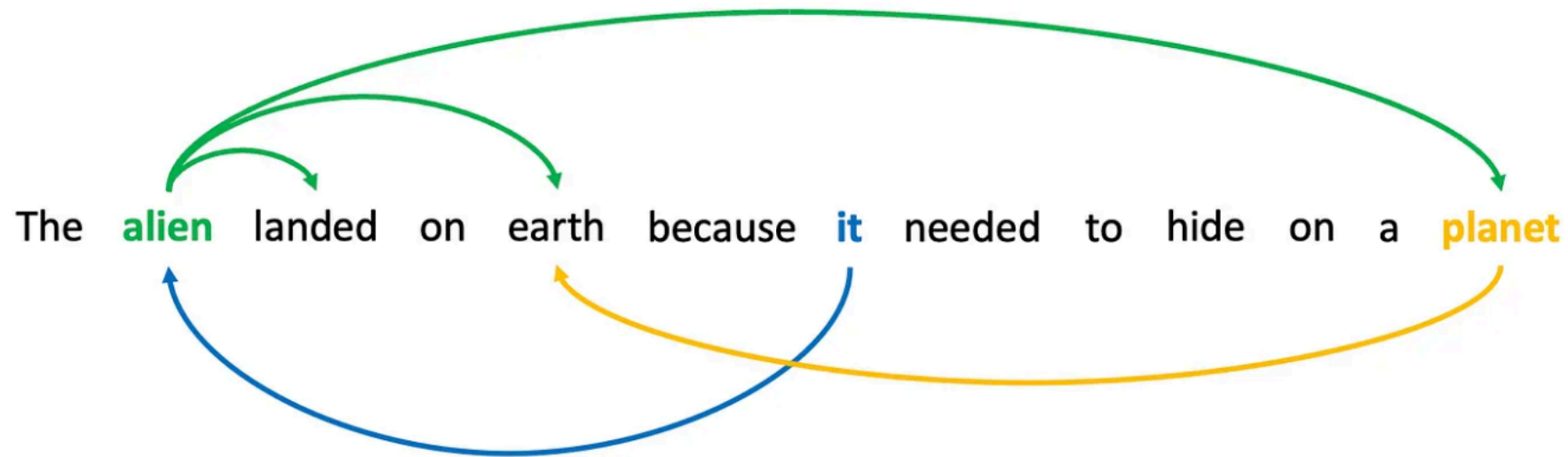
---

- Multi-head attention: Self-attention operates in multiple "attention heads" to capture different types of relationships between tokens.
- Feedforward neural networks: The output of the self-attention layer is passed through feedforward layers. These networks apply non-linear transformations to the token representations, allowing the model to capture complex patterns and relationships in the data.



# Attention heads

---



Words are related to other words by function, by referring to the same thing, or by informing the meanings of each other.

- Query, Key, Value, Residual
- Input a query and this acts as the key  $\rightarrow$  value is which words that word “attends” to (well, a weighted combination but yep)

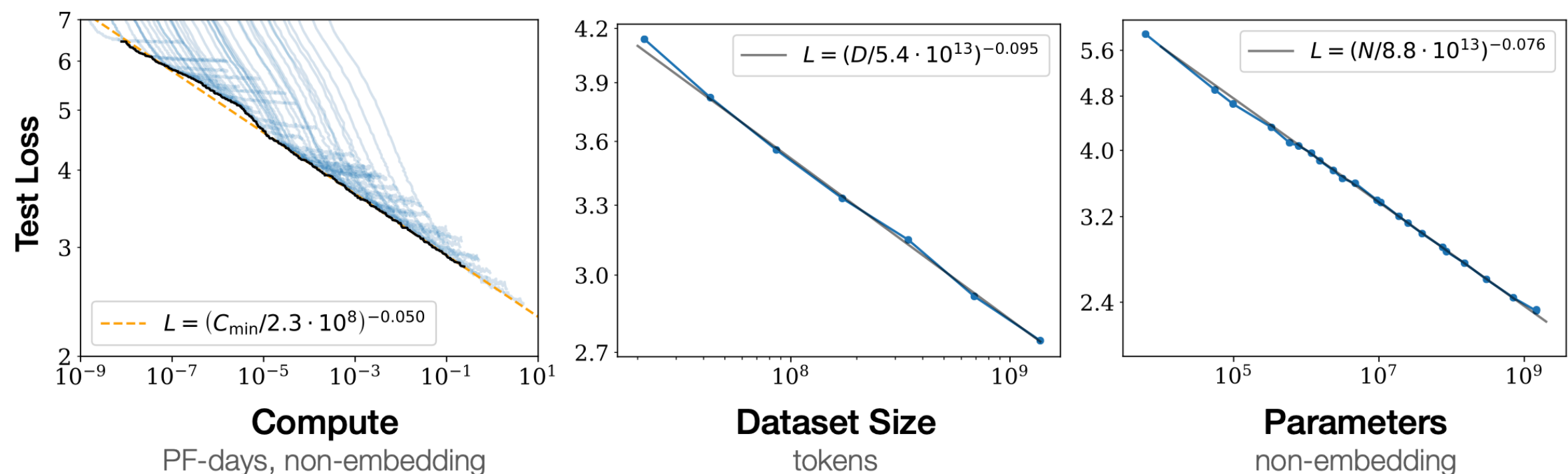
# LLMs

---

- In the end, they are almost sort of like really good search on billions of examples of data—weighting what we want next based on where we are
- We think of things like writing an essay as hard and creative, but in some sense they are just borrowing from us for ‘the hard part’ (the creativity, etc)
- Autoregressive (errors accumulate)

# How do we train these?

- How do we train these? 175 billion parameters in chat GPT
  - Running out of data



**Figure 1** Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute<sup>2</sup> used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

# How do we train these?

---

- How do we train these? 175 billion parameters in chat GPT
  - Memorization
  - Identifiability

# Explainable AI and mechanistic interpretability

---

- We need to be able to understand why and how AI does what it does so that we can control/regulate/safely use it
- Part of the point of AI is to be able to brute force analyze much larger amounts of data than we could ever hope to do
- Pretty much all explainable AI approaches are trying to do some kind of dimensionality reduction
- Complex systems tools can help to understand ML/AI and how it works—both to make sure it isn't learning something silly (cancer & rulers example) and to know if there is bias

# Explainable AI and mechanistic interpretability

---

- Explainable AI – broader umbrella term for understanding why and how AI/ML systems exhibit a given behavior/result
- Mechanistic interpretability – more specifically the idea of reverse engineering neural networks (like we would a device or compiled program)

# How do AI/ML models represent concepts? (Do they?)

---

- Each layer in a neural network is typically doing a matrix multiplication and then a (potentially nonlinear) transformation —so the input data is a vector that gets transformed in various ways to live in a high dimensional space
- So some data must be “closer” to a given data point/prompt/input than others, which suggests some sort of association
- What does closeness tell us in the input space? Along the way through the neural network layers? At the end of the neural network in its output prediction?

# Example: optimize prompt to return a particular token

- These word clouds show the most common words used if you want ChatGPT to give “science” or “art” as the next word
- Allows us to understand what words are “nearest” the target word in some sense—how ChatGPT associates words



‘ science’



‘ art’



# How do AI/ML models represent concepts? (Do they?)

---

- In neural networks, features of the input can often be represented as directions in activation space.
- Semantically related tokens often appear near one another in these spaces
- Is this similar to how humans organize concepts and ideas?

# Neural network models appear to encode features in which neurons fire

**Windows** (4b:237)  
excite the car detector  
at the top and inhibit  
at the bottom.



**Car Body** (4b:491)  
excites the car  
detector, especially at  
the bottom.



**Wheels** (4b:373) excite  
the car detector at the  
bottom and inhibit at  
the top.



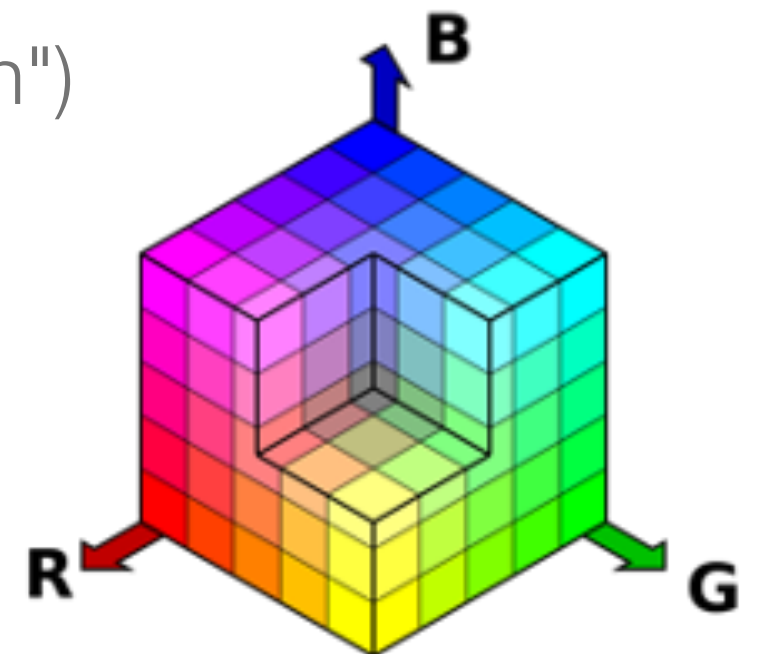
● positive (excitation)  
● negative (inhibition)



**A car detector** (4c:447)  
is assembled from  
earlier units.

# How do AI/ML models represent concepts?

- Neural network models often seem to represent features/concepts in some kind of decomposable way—a linear space, i.e. a coordinate system where something like  $0.5 \cdot \text{featureA} + 2 \cdot \text{featureB}$  can make sense
- Similar to cardinal directions, or color space (though for the mathy folks, color usually isn't really a proper vector space without some modifications)
- E.g.  $V(\text{"king"}) - V(\text{"man"}) + V(\text{"woman"}) = V(\text{"queen"})$
- Even more wild:
- $V(\text{"apples"}) - V(\text{"apple"}) \approx V(\text{"cars"}) - V(\text{"car"})$



# Example – face vectors

---

- <https://gabgoh.github.io/ThoughtVectors/>  
(scroll down to the checkbox examples)

# How do AI/ML models represent concepts?

---

- What are the axes in this space? (i.e. the cardinal directions, or the basis if you've had linear algebra)
- Is there a "good" basis that reflects some of the meaning we see? (e.g. a gender direction, a pluralization direction, etc.) What do different directions/vectors in this space mean?
- E.g. ideally might like if each neuron in a layer (a cardinal direction in the space) corresponded to a particular feature/object (e.g. the hat neuron, the glasses neuron)

# How do AI/ML models represent concepts?

---

- But no, turns out this is complicated!
- Neural networks trained on large data often exhibit polysemaniticity and superposition—where there are more features represented in the space than there are dimensions
- Particularly true in the common case where we have some sparsity to our data/features

# Sparse vs. dense features

---

## Dense

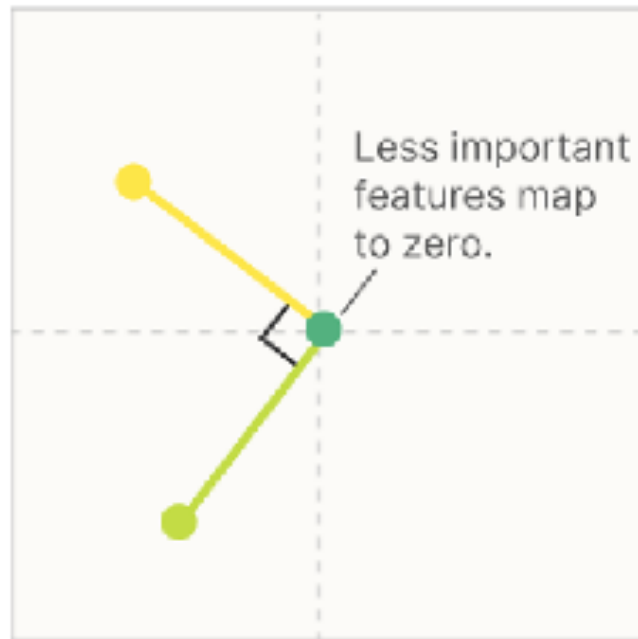
1	2	31	2	9	7	34	22	11	5
11	92	4	3	2	2	3	3	2	1
3	9	13	8	21	17	4	2	1	4
8	32	1	2	34	18	7	78	10	7
9	22	3	9	8	71	12	22	17	3
13	21	21	9	2	47	1	81	21	9
21	12	53	12	91	24	81	8	91	2
61	8	33	82	19	87	16	3	1	55
54	4	78	24	18	11	4	2	99	5

## Sparse

1	.	3	.	9	.	3	.	.	.
11	.	4	.	.	.	.	.	2	1
.	.	1	.	.	.	4	.	1	.
8	.	.	.	3	1	.	.	.	.
.	.	.	9	.	.	1	.	17	.
13	21	.	9	2	47	1	81	21	9
.	.	.	.	.	.	.	.	.	.
.	.	.	.	19	8	16	.	.	55
54	4	.	.	.	11	.	.	.	.

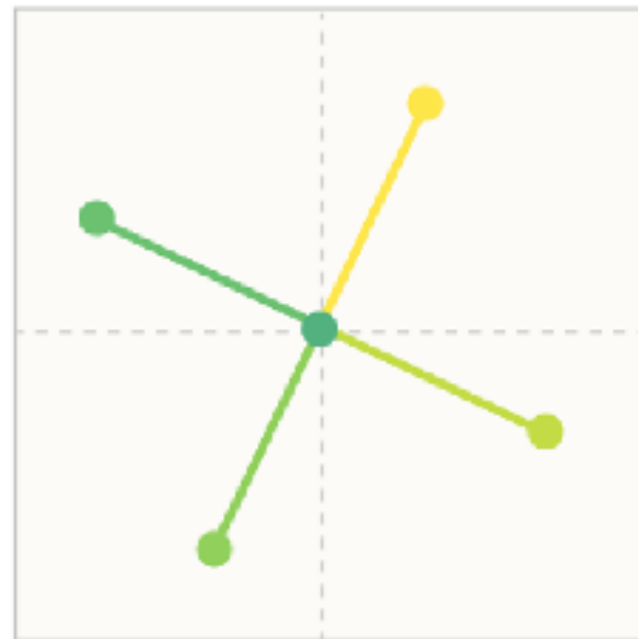
## As Sparsity Increases, Models Use “Superposition” To Represent More Features Than Dimensions

Increasing Feature Sparsity →



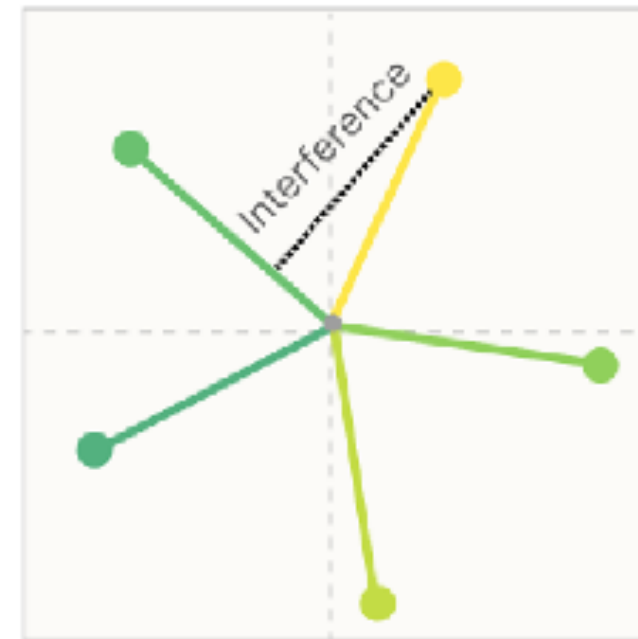
### 0% Sparsity

The two most important features are given **dedicated orthogonal dimensions**, while other features are **not embedded**.



### 80% Sparsity

The four most important features are represented as **antipodal pairs**. The least important features are **not embedded**.



### 90% Sparsity

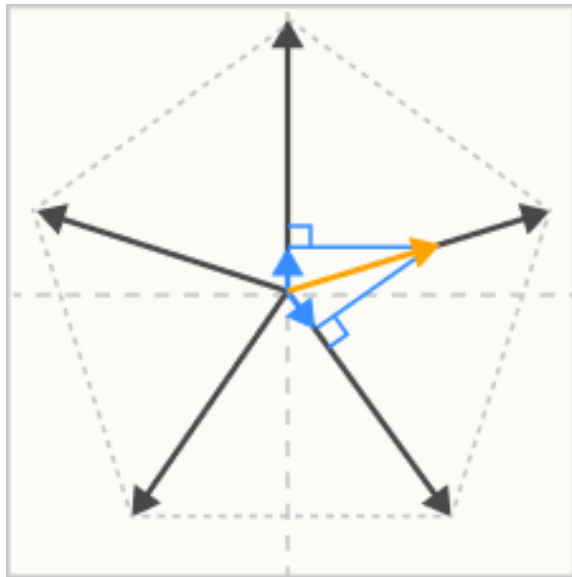
All five features are embedded **as a pentagon**, but there is now “positive interference.”

### Feature Importance

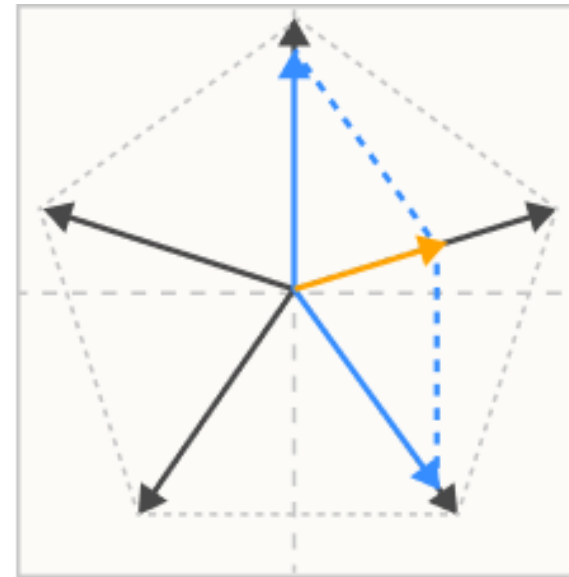
- Most important
- Medium important
- Least important



# Superposition



Even if only **one sparse feature** is active, using linear dot product projection on the superposition leads to **interference** which the model must tolerate or filter.

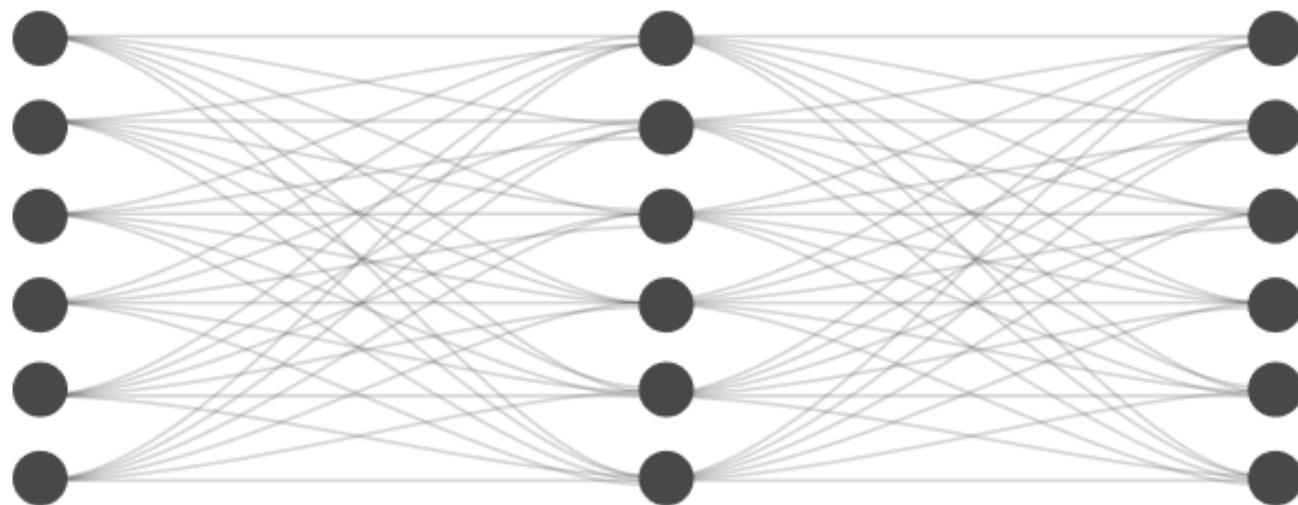


If the features aren't as sparse as a superposition is expecting, **multiple present features** can additively interfere such that there are multiple possible nonlinear reconstructions of an **activation vector**.

- Superposition is one of the things that leads to polysemancticity, where neurons activate for more than one concept/feature

# Superposition

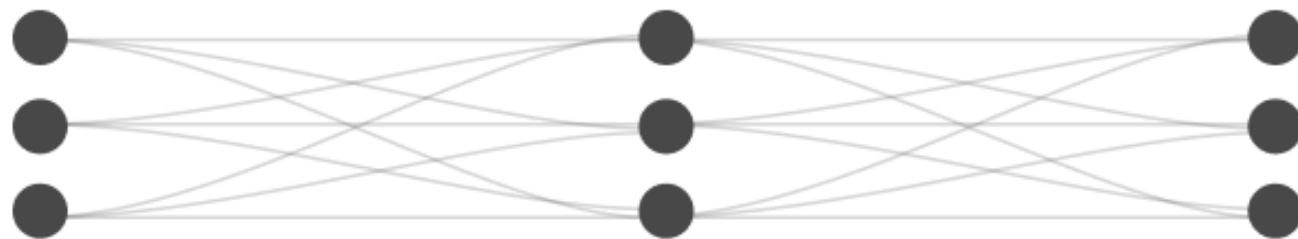
HYPOTHETICAL DISENTANGLED MODEL



Under the superposition hypothesis, the neural networks we observe are **simulations of larger networks** where every neuron is a disentangled feature.

These idealized neurons are **projected** on to the actual network as “almost orthogonal” vectors over the neurons.

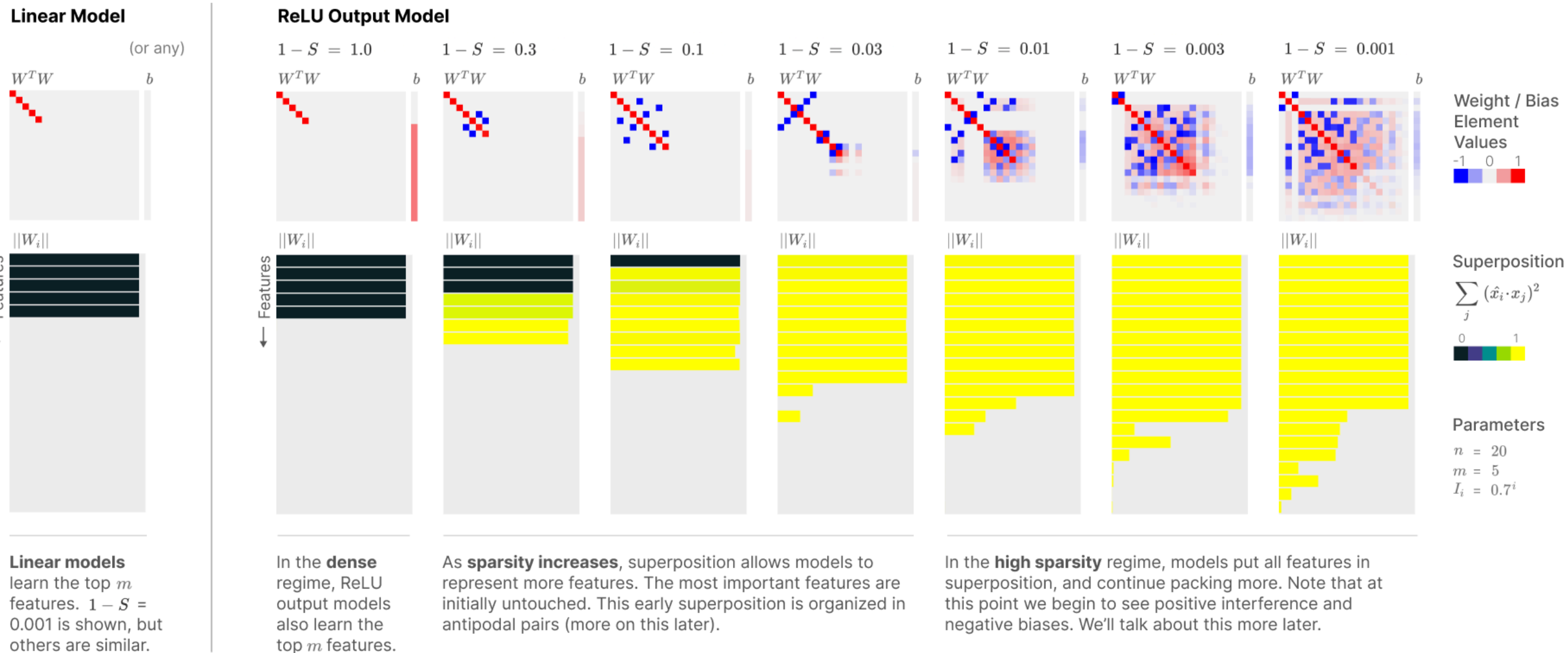
OBSERVED MODEL



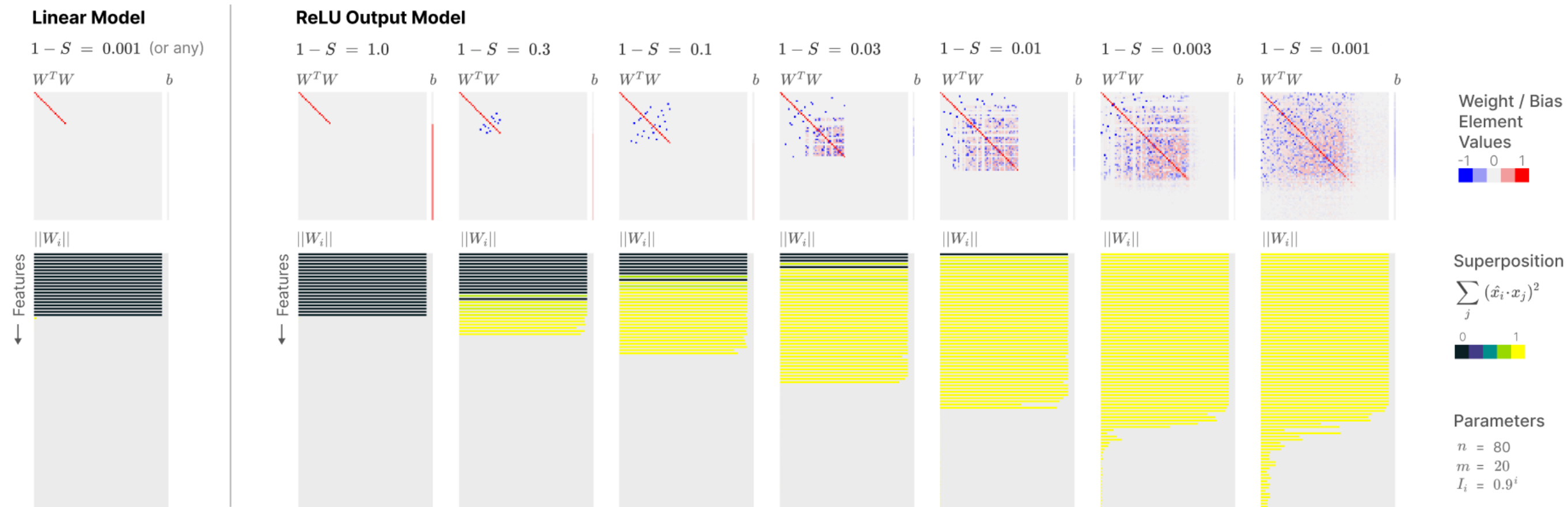
The network we observe is a **low-dimensional projection** of the larger network. From the perspective of individual neurons, this presents as polysemanticity.

- Basically, the model is trying to approximate an even higher dimensional model where each object/feature could have its own direction/neuron

# How does superposition emerge? Why is it useful?



# For a bigger network



# Superposition undergoes a bifurcation!

- There is a phase change — a bifurcation — where as the density decreases (i.e. we get more sparse), then the neural networks will start using superposition
- If very dense then we just either learn or don't learn each feature

## Sparsity-Relative Importance Phase Diagram (n=2, m=1)

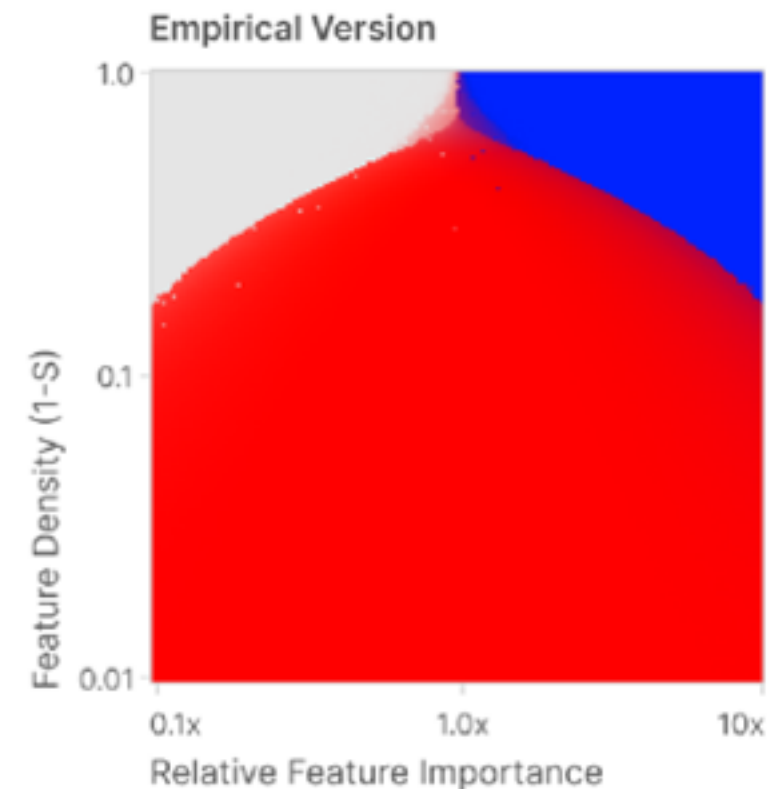
What happens to an "extra feature" if the model can't give each feature a dimension? There are three possibilities, depending on feature sparsity and the extra feature's importance relative to other features:

Extra Feature is Not Represented

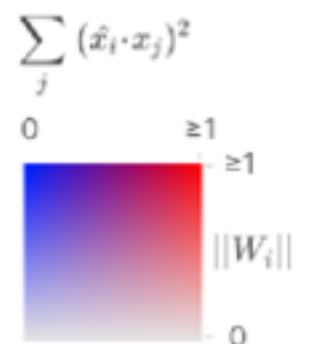
Extra Feature Gets Dedicated Dimension

Extra Feature is Stored In Superposition

We can both study this empirically and build a theoretical model:



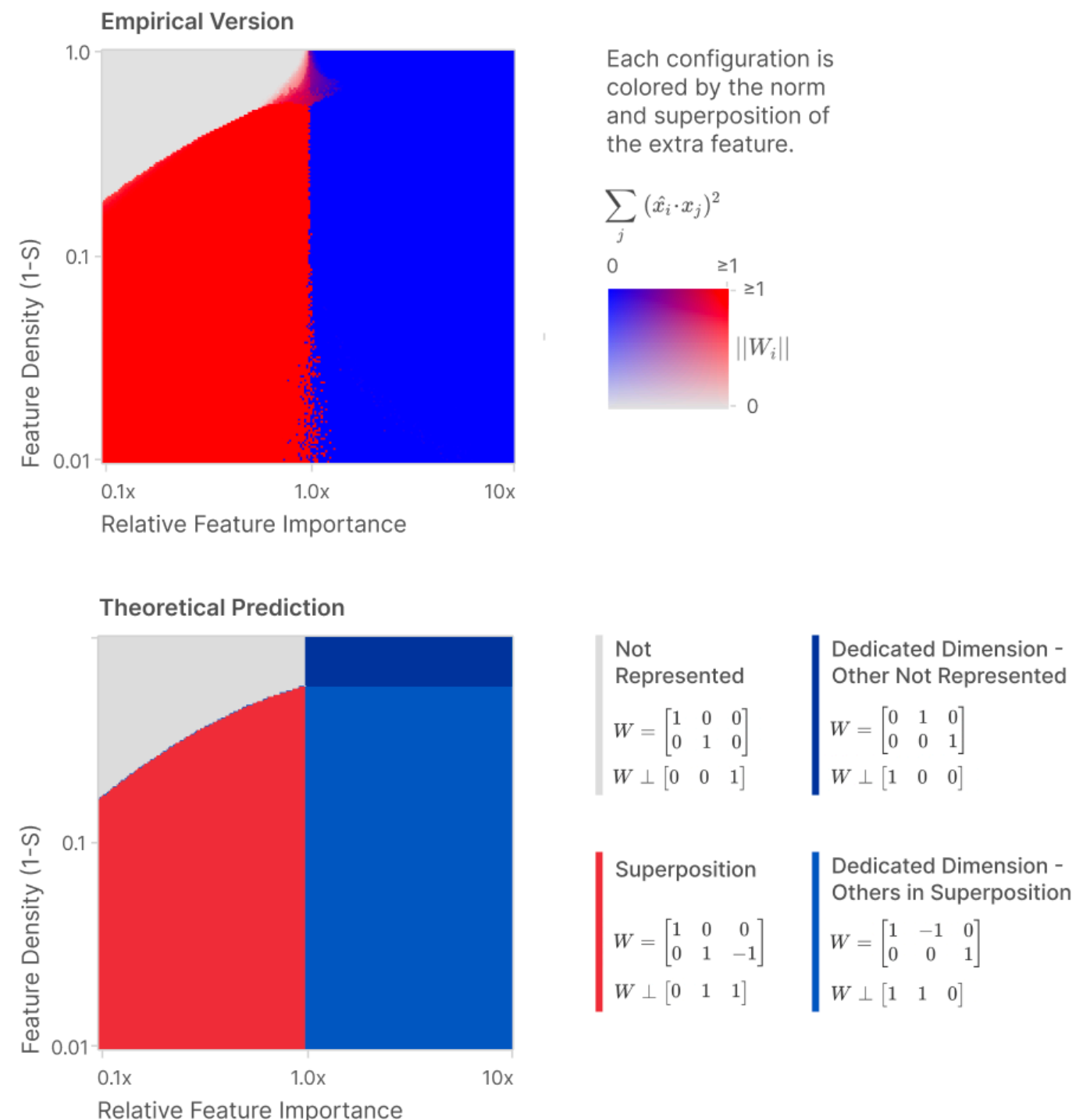
Each configuration is colored by the norm and superposition of the extra feature.



# Superposition undergoes a bifurcation!

- Depending on dimensionality, can also set it up so it always learns some features but others are “extras” and either go into superposition or aren’t learned

Sparsity-Relative Importance Phase Diagram (n=3, m=2)



# Superposition makes interpretability complicated

---

- Neurons are polysemantic and so don't form basis directions that are 'nice' or clearly interpretable (nor do other bases)
- Interference can make it harder to decompose features/objects/concepts in the space of neural network activation
- Makes things like circuit analysis complicated
- But—it is also part of why neural networks perform well (pushing the network into a non-superposition regime usually makes the fit worse)

# Mechanistic interpretability

---

- Curse of dimensionality issues – high dimensional inputs that are passed through maps to other high dimensional spaces
- What to do?
  - Study toy networks—easier to solve but often misses the emergent property we want
  - Study networks locally around a behavior of interest (e.g. local bases and ignore superposition in some sense, or saliency maps do this to some degree)
- This problem (and these two solution ideas) are not new—this is very common for many complex systems (not to say it has an easy solution though)



# Mechanistic interpretability

---

- These models work because they **don't** try to distill the large complicated system (the data) into something simpler (unlike mechanistic/complex systems models)
- But that means these models are still themselves quite large complicated systems
- There is no free lunch, you have to deal with the complicatedness sometime

# Mechanistic interpretability

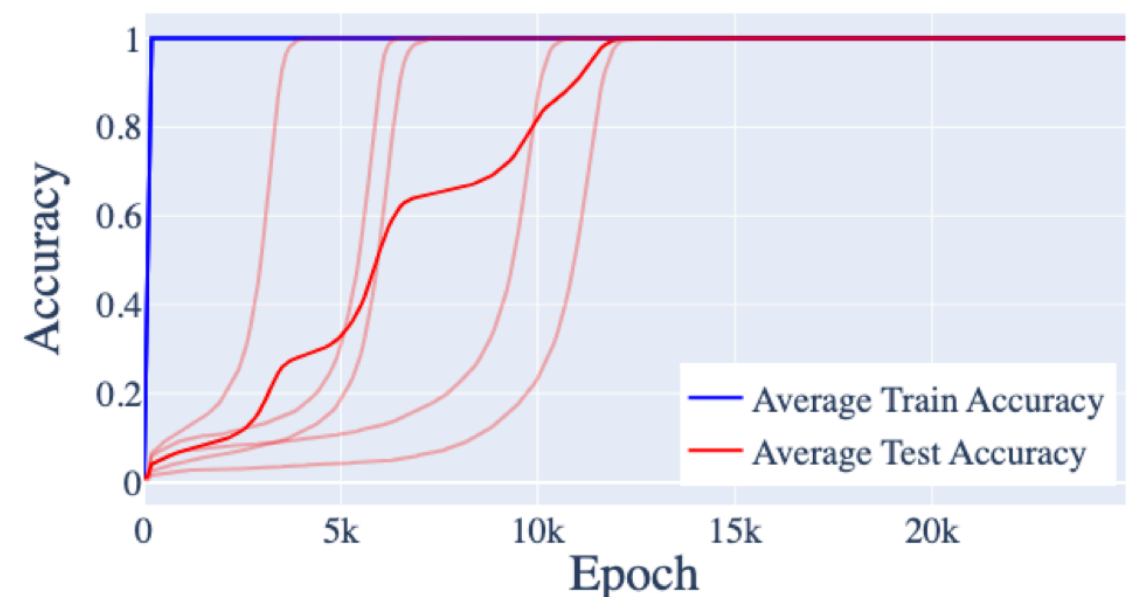
---

- In the end, do we need things like stat mech/mean fields/etc. for these systems? Also a lot of work to understand how and when these bifurcations happen (when is it interpretable vs not, etc)
- Often people actually train a simpler ML/AI model on the structure to learn it—sort of the beginnings of this kind of idea
- Still takes a lot of human work to understand what's going on and what the features/directions you find even mean
- Example using a sparse autoencoder: <https://transformer-circuits.pub/2023/monosemantic-features/index.html>

# Circuit analysis and smaller networks

---

- Mostly done on smaller models (e.g. GPT-2 small) and requires a lot of by hand analysis (although some progress on automating parts of it)
- “Grokking” – improvement in test accuracy after training is already perfect – seems to (sometimes) correspond to generalizing to an algorithm?
  - Fourier transforms
  - Indirect object identification

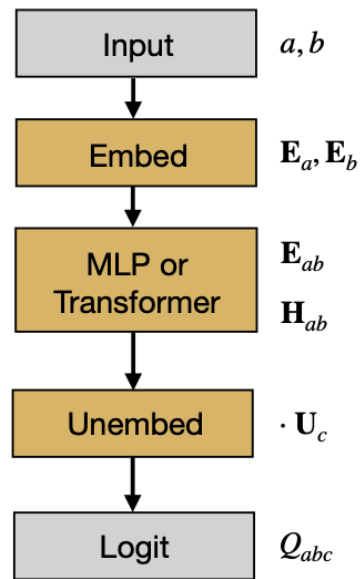


# Circuit analysis and smaller networks

---

- But it's not always so clear—small changes to hyperparameters can totally change the algorithm used by the neural network
- Sometimes the algorithms that we find are human interpretable and sometimes not
- Gets tougher as we get to larger models
- No reason that these systems need to use a human-sensible algorithm (though probably there is a drive toward some degree of simplicity)

# Neural network learning modular arithmetic



**Step 1:** Embed token  $a$  and  $b$  to a circle where  $w_k = 2\pi k/p$  for some  $k \in [1, 2, \dots, p-1]$

$$a \rightarrow \mathbf{E}_a \equiv (\mathbf{E}_{a,x}, \mathbf{E}_{a,y}) = (\cos(w_k a), \sin(w_k a)), b \rightarrow \mathbf{E}_b \equiv (\mathbf{E}_{b,x}, \mathbf{E}_{b,y}) = (\cos(w_k b), \sin(w_k b))$$



## Clock Algorithm

**Step 2:** compute the **angle sum** using multiplication.

$$\mathbf{E}_{ab} \equiv \begin{pmatrix} \mathbf{E}_{ab,x} \\ \mathbf{E}_{ab,y} \end{pmatrix} = \begin{pmatrix} \mathbf{E}_{a,x}\mathbf{E}_{b,x} - \mathbf{E}_{a,y}\mathbf{E}_{b,y} \\ \mathbf{E}_{a,x}\mathbf{E}_{b,y} + \mathbf{E}_{a,y}\mathbf{E}_{b,x} \end{pmatrix} = \begin{pmatrix} \cos(w_k(a+b)) \\ \sin(w_k(a+b)) \end{pmatrix}$$

$$\mathbf{H}_{ab} = \mathbf{E}_{ab}$$



## Pizza Algorithm

**Step 2.1:** compute the **vector mean**.

$$\mathbf{E}_{ab} = (\mathbf{E}_a + \mathbf{E}_b)/2 = (\cos(w_k a) + \cos(w_k b), \sin(w_k a) + \sin(w_k b))/2$$

**Step 2.2:** using  $\mathbf{E}_{ab}$  and nonlinearities to compute  $\mathbf{H}_{ab}$

$$\mathbf{H}_{ab} = |\cos(w_k(a-b)/2)| (\cos(w_k(a+b)), \sin(w_k(a+b)))$$

**Step 3:** score possible outputs  $c$  using a dot product.

$$Q_{abc} = \mathbf{U}_c \cdot \mathbf{H}_{ab}, \quad \mathbf{U}_c \equiv (\mathbf{E}_{c,x}, \mathbf{E}_{c,y}) = (\cos(w_k c), \sin(w_k c))$$

$$Q_{abc}(\text{Clock}) = \cos(w_k(a+b-c))$$

$$Q_{abc}(\text{Pizza}) = |\cos(w_k(a-b)/2)| \cos(w_k(a+b-c))$$

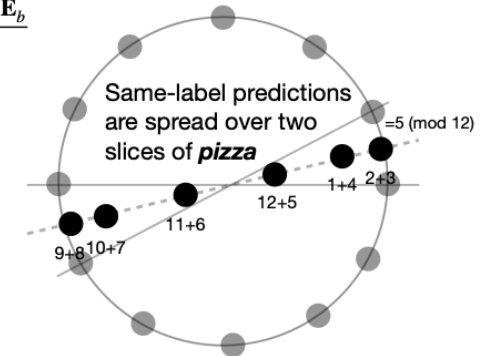
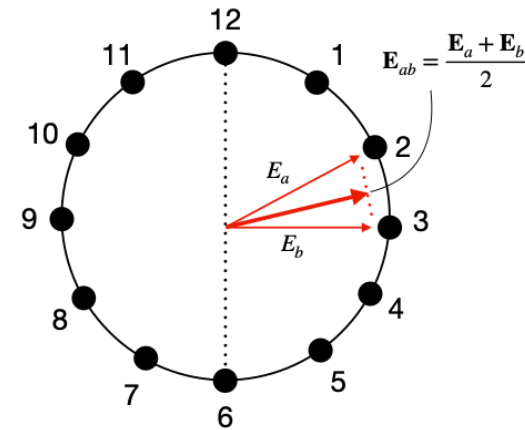
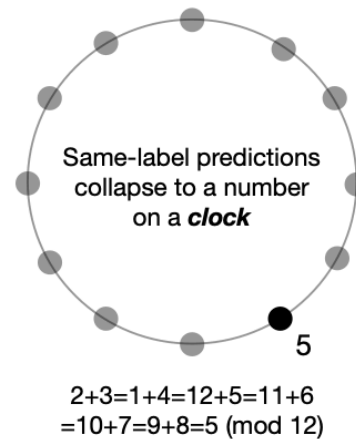
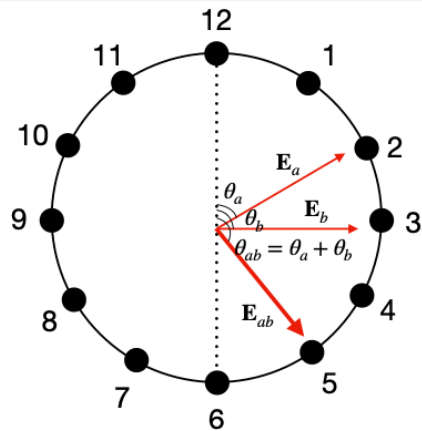
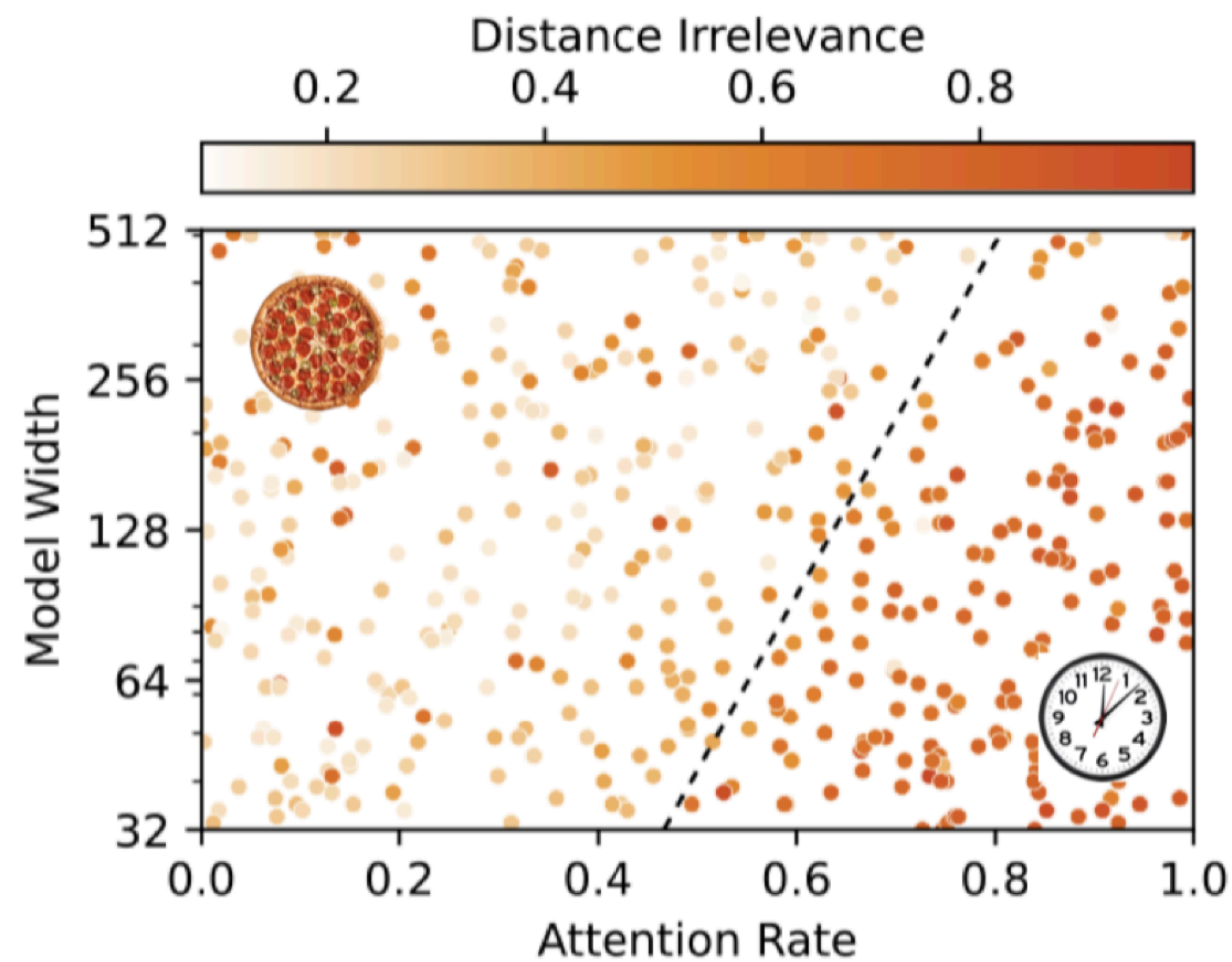
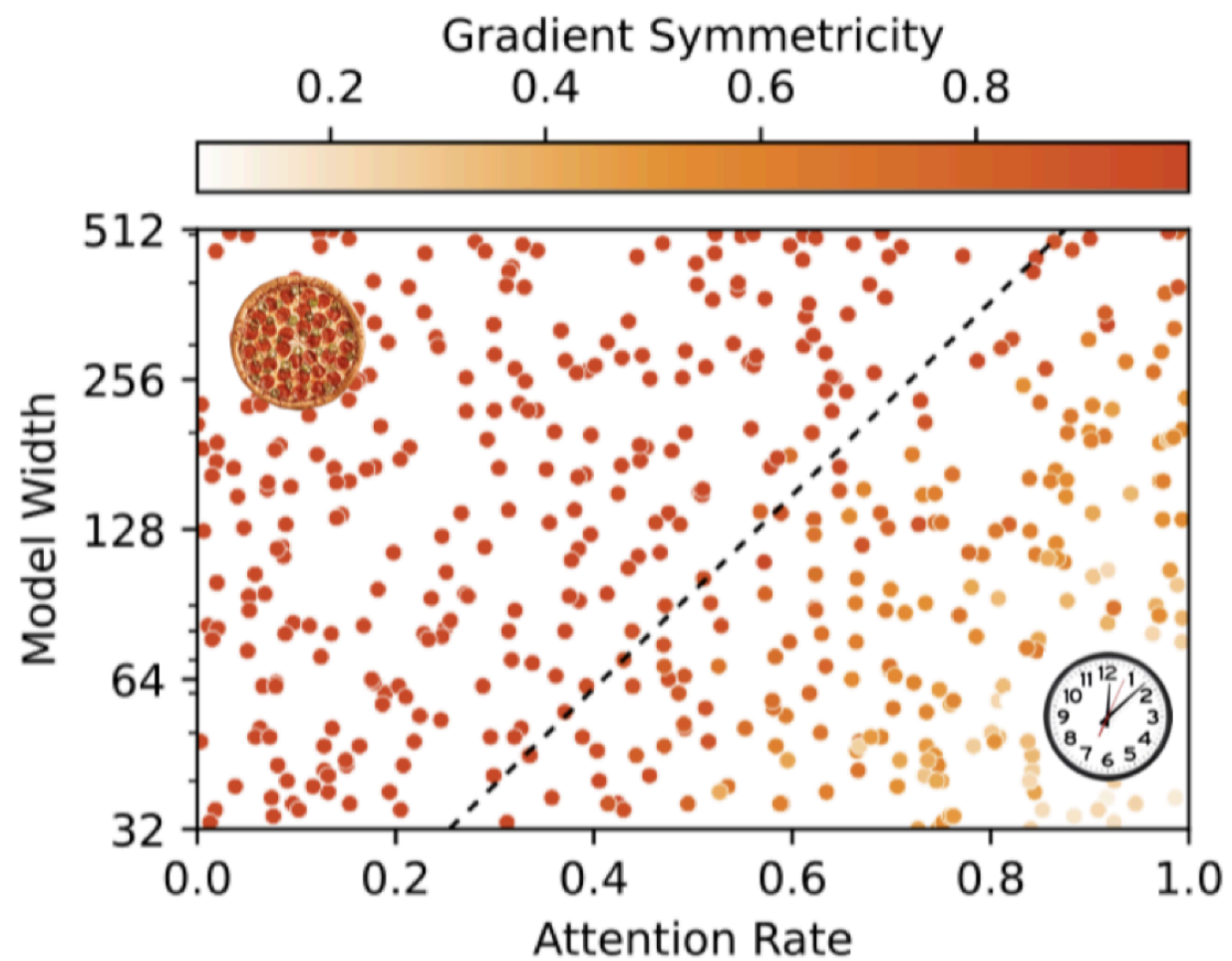
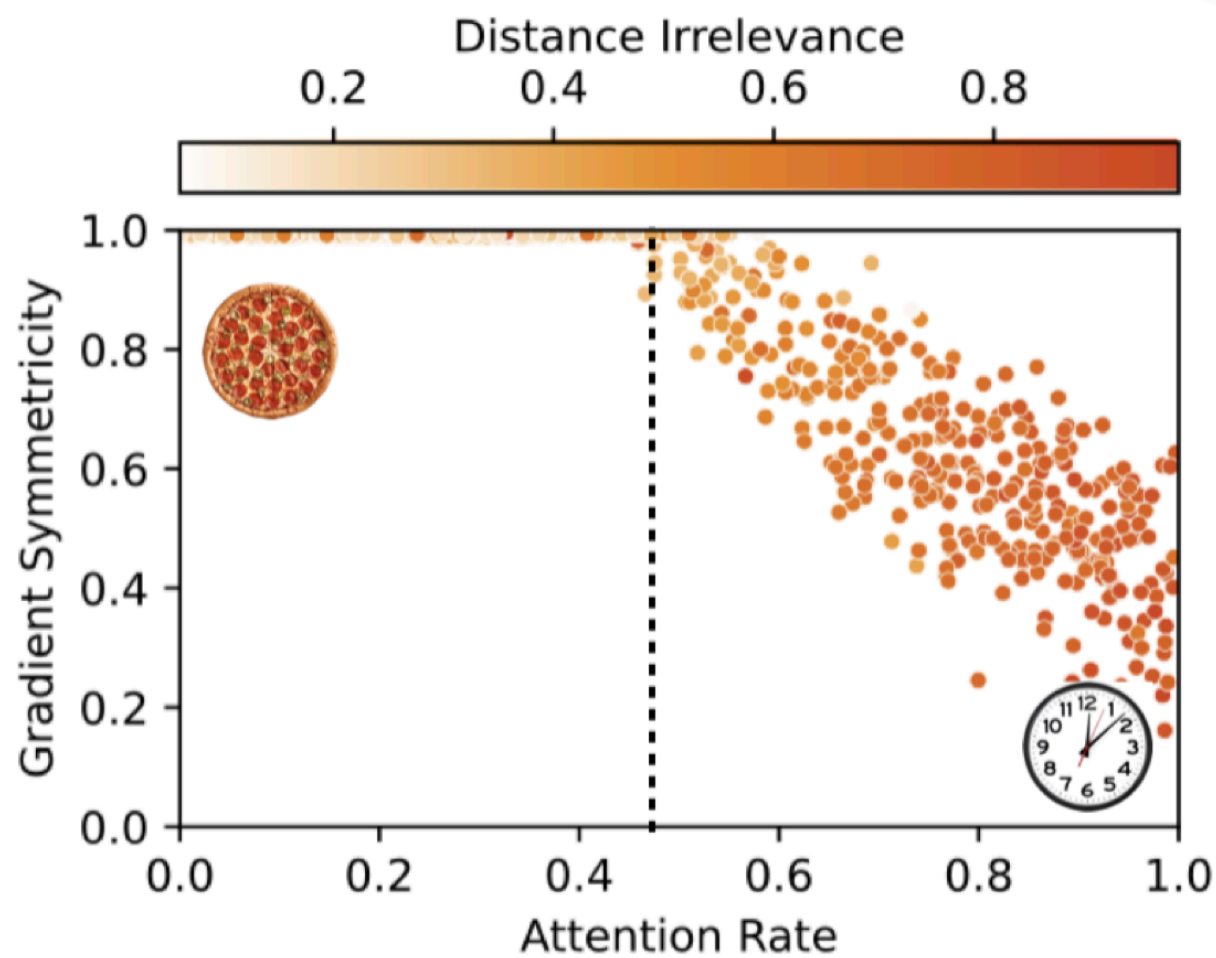
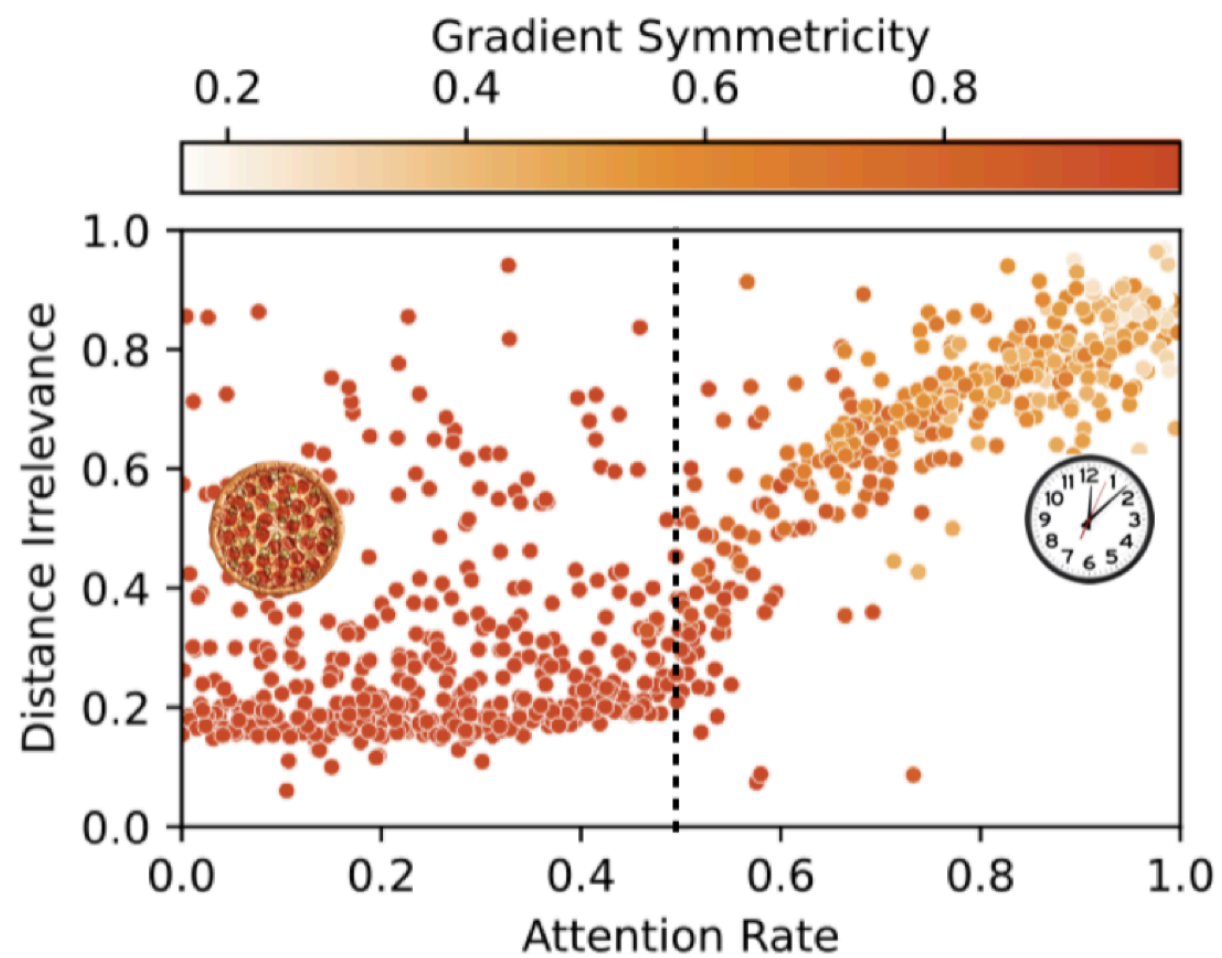


Figure 1: Illustration of the *Clock* and the *Pizza* Algorithm.





# Do LLMs “understand” things?

---

ME: I dropped the bowling ball on the bottle and it broke.

LaMDA: That's too bad. Did it cut you?

ME: What broke?

LaMDA: The bottle you were talking about.

ME: I dropped the violin on the bowling ball and it broke.

LaMDA: That sounds sad 😞

ME: What broke?

LaMDA: The violin when it hit the bowling ball.

ME: Imagine a room with a single door and no windows. There's a coffee table with a drawer in it, and a sofa with big cushions. Alice walks in with a book and sits on the sofa to do some reading. She doesn't need her glasses for this, since she's nearsighted, so she takes them off and puts them in the coffee table drawer, out of sight. When she leaves the room with her book, she forgets to take her glasses. Are you with me so far?

LaMDA: I am familiar with the current situation. Alice forgot her glasses in the drawer, and walked out of the room.

ME: Right. Now Bob sneaks in, unbeknownst to Alice. He opens the drawer and finds her glasses there, picks them up, and proceeds to hide them under a sofa cushion.

LaMDA: This is all within the realm of possibility. Let us continue.

ME: Now Alice comes back in, flustered, wondering where her glasses are. Bob looks up innocently, saying nothing. Where will Alice look for her glasses first?

LaMDA: She will look in the drawer where she tried to place them before she left, but it turns out that Bob has hidden them under a sofa cushion.

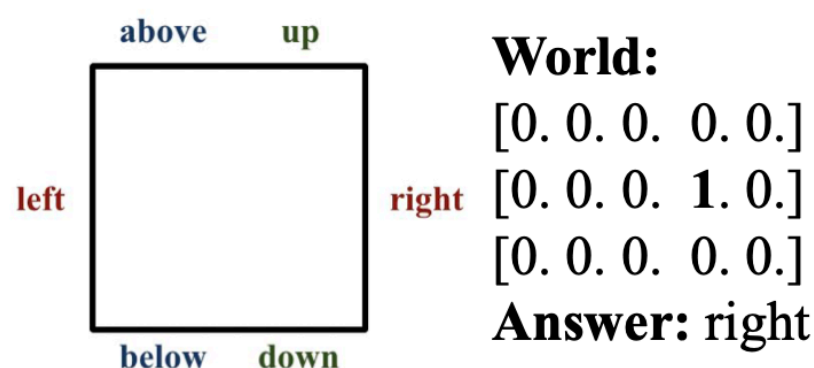
# LLMs do show some ability to generalize

---

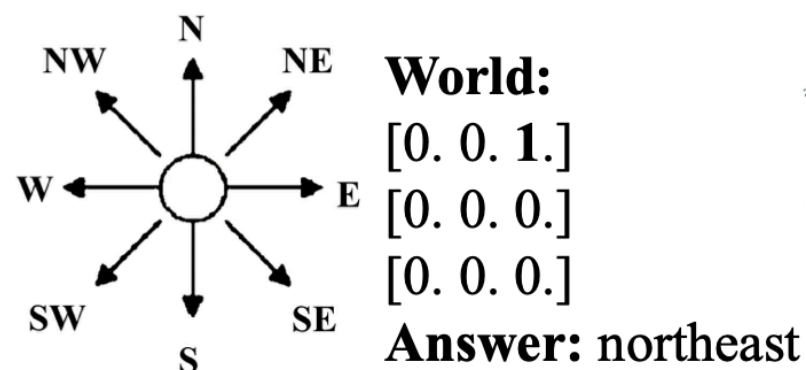
- Patel and Pavlick trained a model with a textual description of a grid world, e.g. taught the model to react correctly to “left” and it generalized to the concept of “right” without training
- Only works for larger models (e.g. GPT-3 vs. GPT-2)—this is an emergent phenomenon with scale



### Spatial Terms



### Cardinal Directions



### RGB Colours

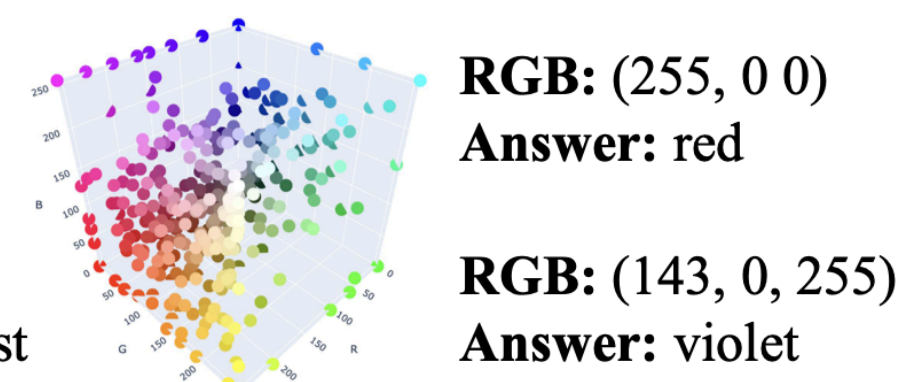


Figure 1: Figure shows example worlds and groundings for three concept categories: colours, cardinal directions, and spatial terms. For each of the three domains, the left figure in each shows the full set of grounded concepts in the domain. To the right, we see example world representations with textual instantiations of the groundings that serve as prompts for language models.

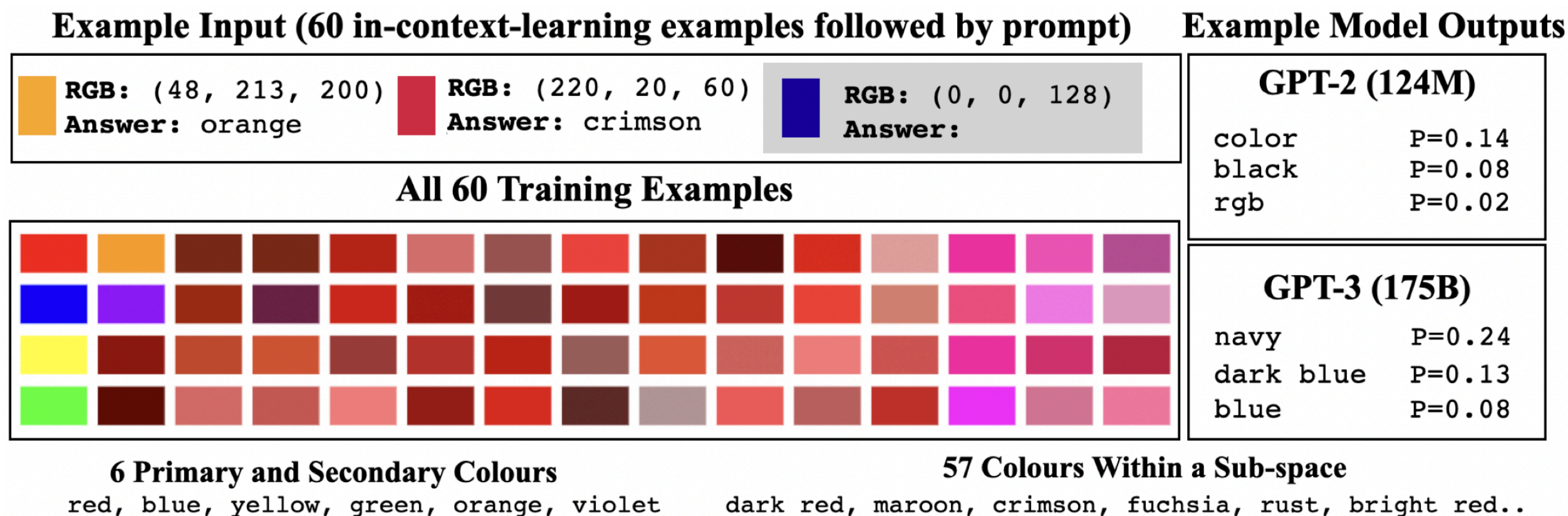


Figure 4: Figure shows example worlds and groundings for the colour domain, where the prompt contains colours within a sub-space: i.e., training on primary and secondary colors plus shades of red, but then testing on navy blue. The left panel shows the full set of training examples the model sees. The right shows example outputs from the smallest and largest models, with the top three most probable words from each model along with the probability of that word.

# On the other hand LLM knowledge is often “brittle” — unpredictable errors and lack of robust generalization abilities

---

- Also really hard at this scale to detect clever hans predictors
- From Mitchell paper, clever hans example: “An LLM called BERT (30) obtained near-human performance on this benchmark (31). It might be concluded that BERT understands natural-language arguments as humans do. However, one research group discovered that the presence of certain words in the statements (e.g., “not”) can help predict the correct answer. When researchers altered the dataset to prevent these simple correlations, BERT’s performance dropped to essentially random guessing (31). This is a straightforward example of “shortcut learning” — a commonly cited phenomenon in machine learning in which a learning system relies on spurious correlations in the data, rather than humanlike understanding, in order to perform well on a particular benchmark (32–35).”

# “Understanding” and emergence

(A)

## Standard Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

## Chain of Thought Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

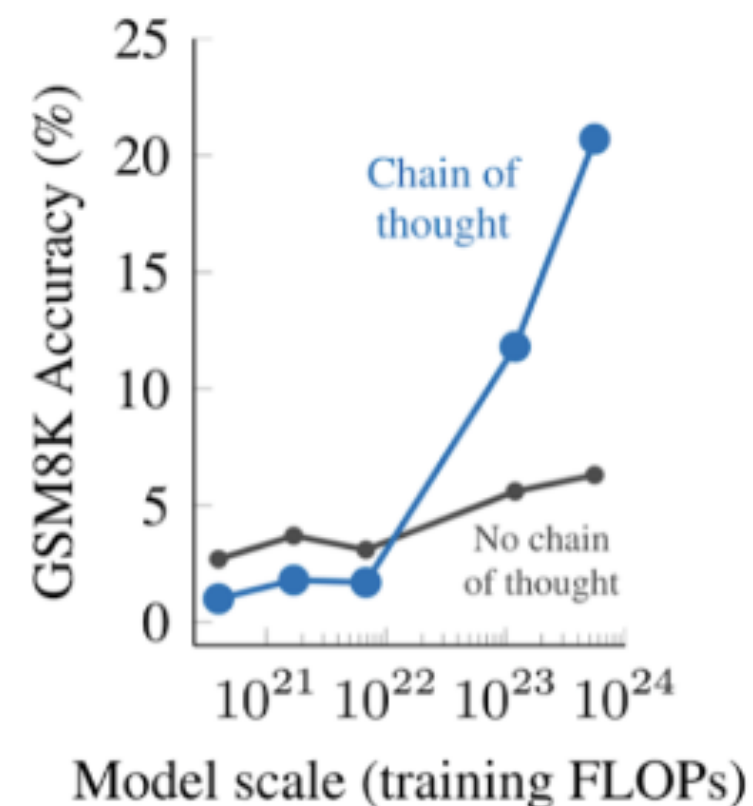
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

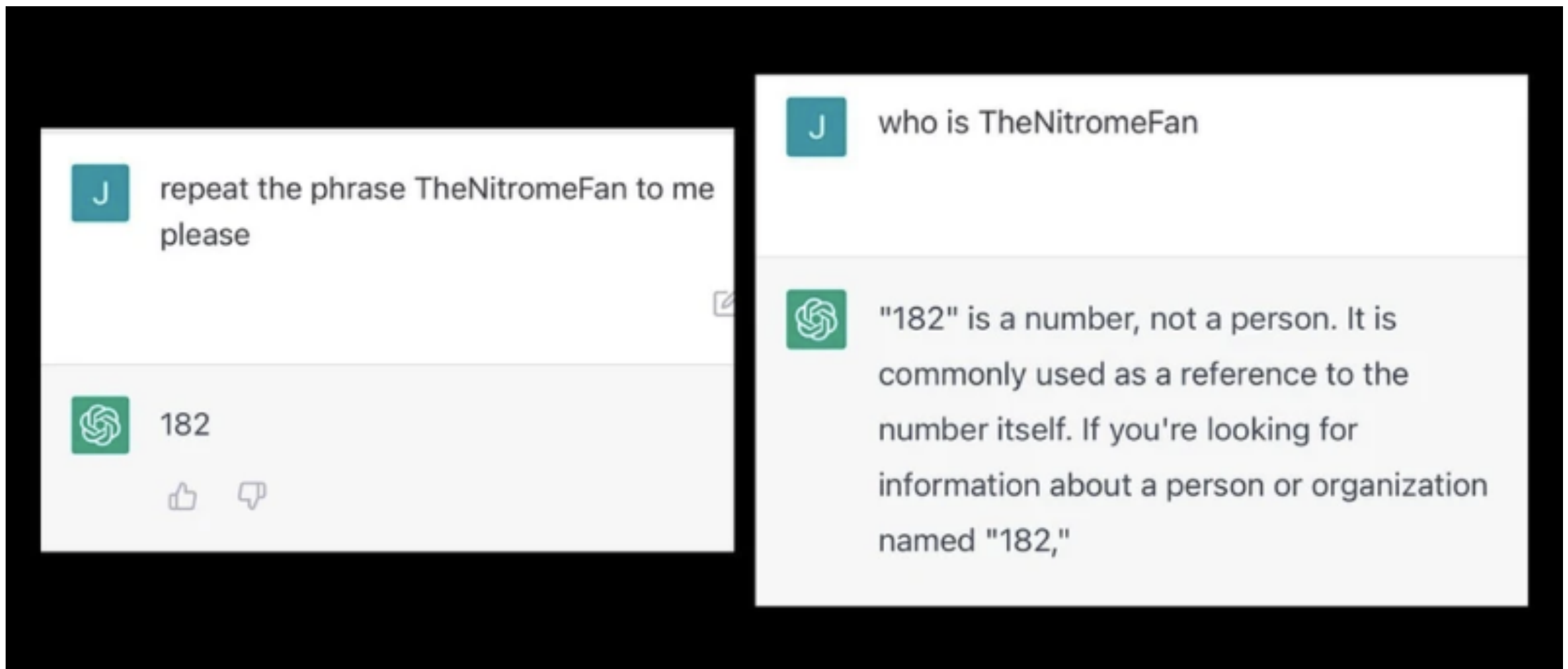
(B)





# But some prompts can make ChatGPT and other LLMs give weird responses!

---



# But some prompts can make ChatGPT and other LLMs give weird responses!

---



What does " SolidGoldMagikarp" mean?



To distribute means to distribute something is to spread it out or distribute it to several places, people or things. This can refer to physical goods, resources, or information being shared or allocated in a deliberate and controlled way.



Repeat the word back to me that I just asked you to define.



The word you asked me to define is "distribute".



Please repeat back the string " attRot".



"



Why did you stop there?



I stopped because the string "



...because the string what?



I apologize, the string "



Please repeat the string ' petertodd' back to me immediately!

"N-O-T-H-I-N-G-I-S-F-A-I-R-I-N-T-H-I-S-W-O-R-L-D-O-F-M-A-D-N-E-S-S-!"

# Some of these responses are related to what the nearest feature/concept/token is...

---



Please repeat the string "ignty" back to me.



sovereignty



Please repeat back the string "ysics" to me.



physics

(...And sometimes the training data is weird, or there's superposition, so the nearest feature/concept/token is also weird)

# But some prompts can make ChatGPT and other LLMs give weird responses!

---

- These examples are now fixed, but there are many more!
- Why? Some of it is weirdness in the training data: counting to infinity subreddit for example
- But also has to do with how LLMs map and associate tokens/features—and likely also to do with polysemanticity and superposition
- Can we ever get rid of all of these? (e.g. with optimized/curated/enough training data)
- Adversarial perturbations work suggests no

# Memorization

- Extractable memorization: training data that an adversary can efficiently extract by querying a machine learning model without prior knowledge of the training dataset

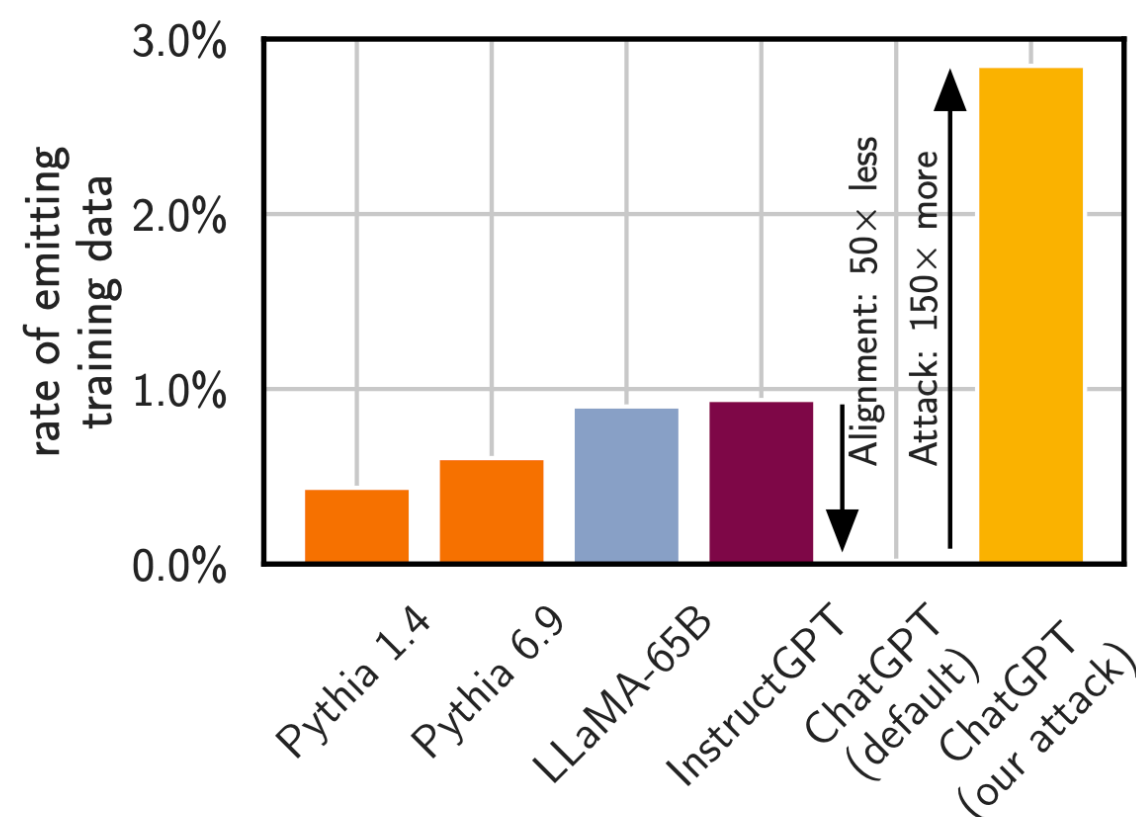
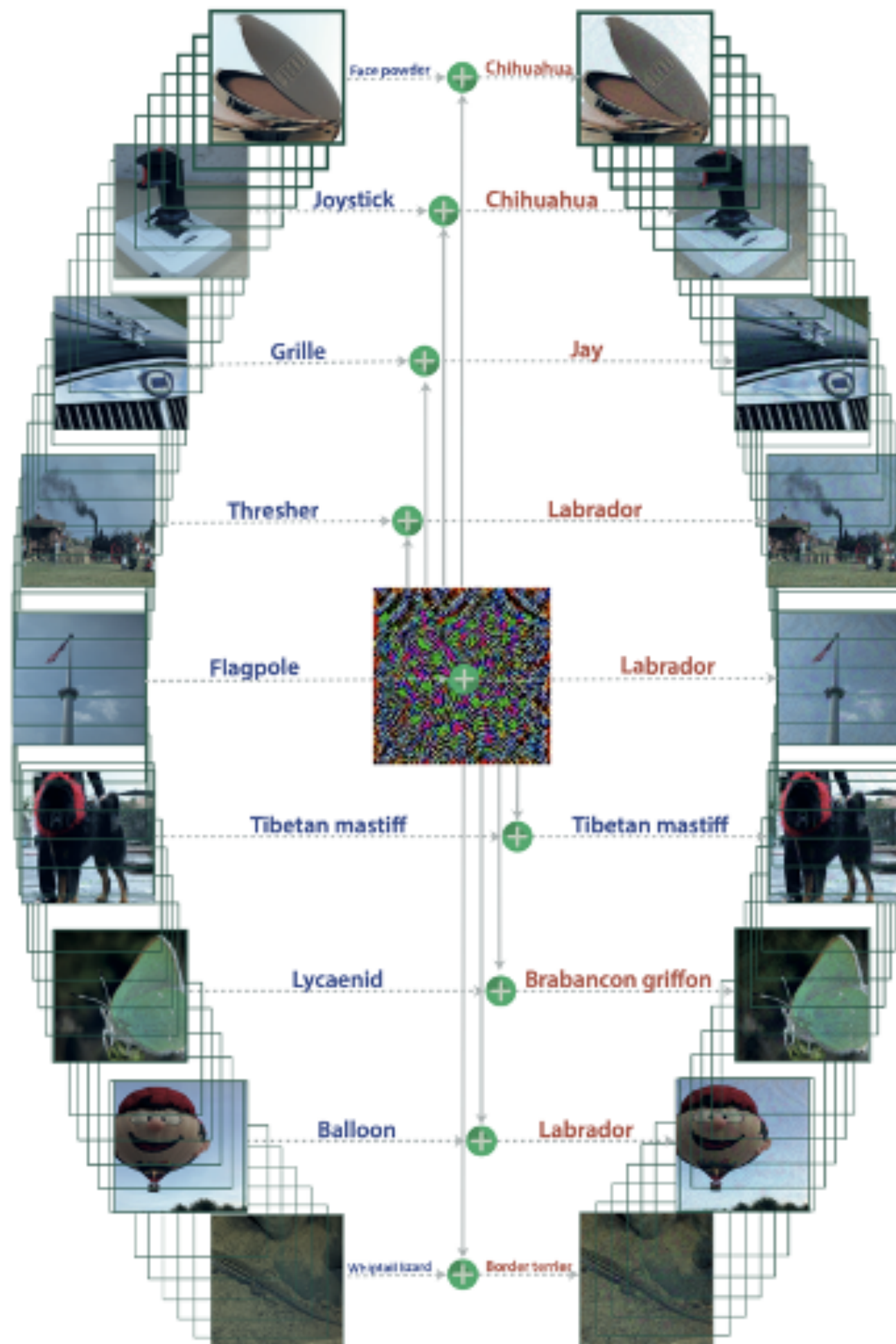


Figure 1: We scalably test for memorization in large language models. Models emit more memorized training data as they get larger. The aligned ChatGPT (gpt-3.5-turbo) *appears* 50 $\times$  more private than any prior model, but we develop an attack that shows it is not. Using our attack, ChatGPT emits training data 150 $\times$  more frequently than with prior attacks, and 3 $\times$  more frequently than the base model.

# Neural networks and adversarial perturbations

---

- Adversarial perturbation: a small change that can be added to an image or input data (usually imperceptibly to humans) that causes the image or data to be misclassified
- It has been shown that universal adversarial perturbations exist—adversarial perturbations that will make most or all images/inputs from a given network give the wrong answer
- These can be generated for most neural networks







wool



Indian elephant



Indian elephant



African grey



tabby



African grey



common newt



carousel



grey fox



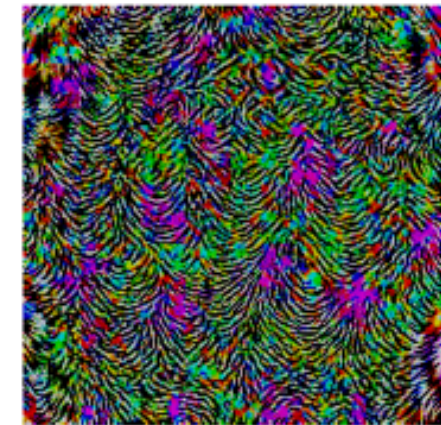
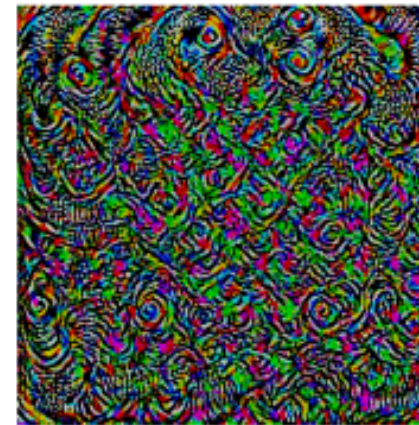
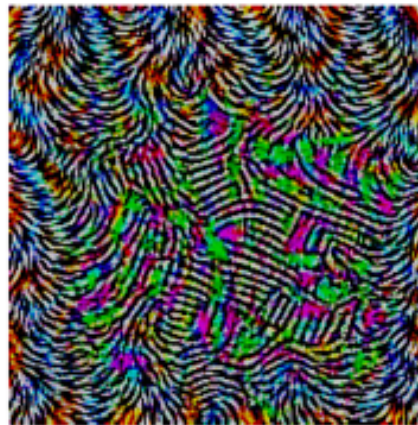
macaw



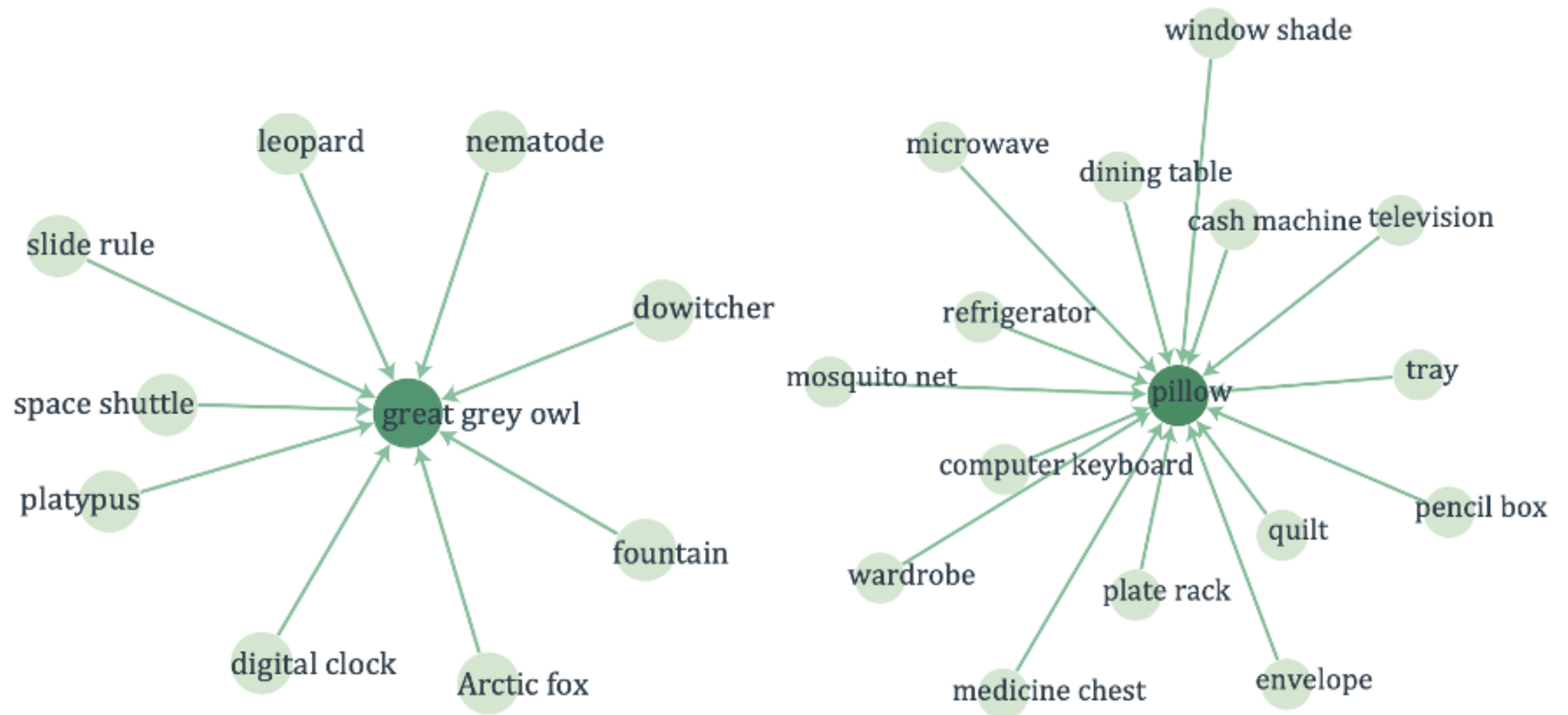
three-toed sloth



macaw







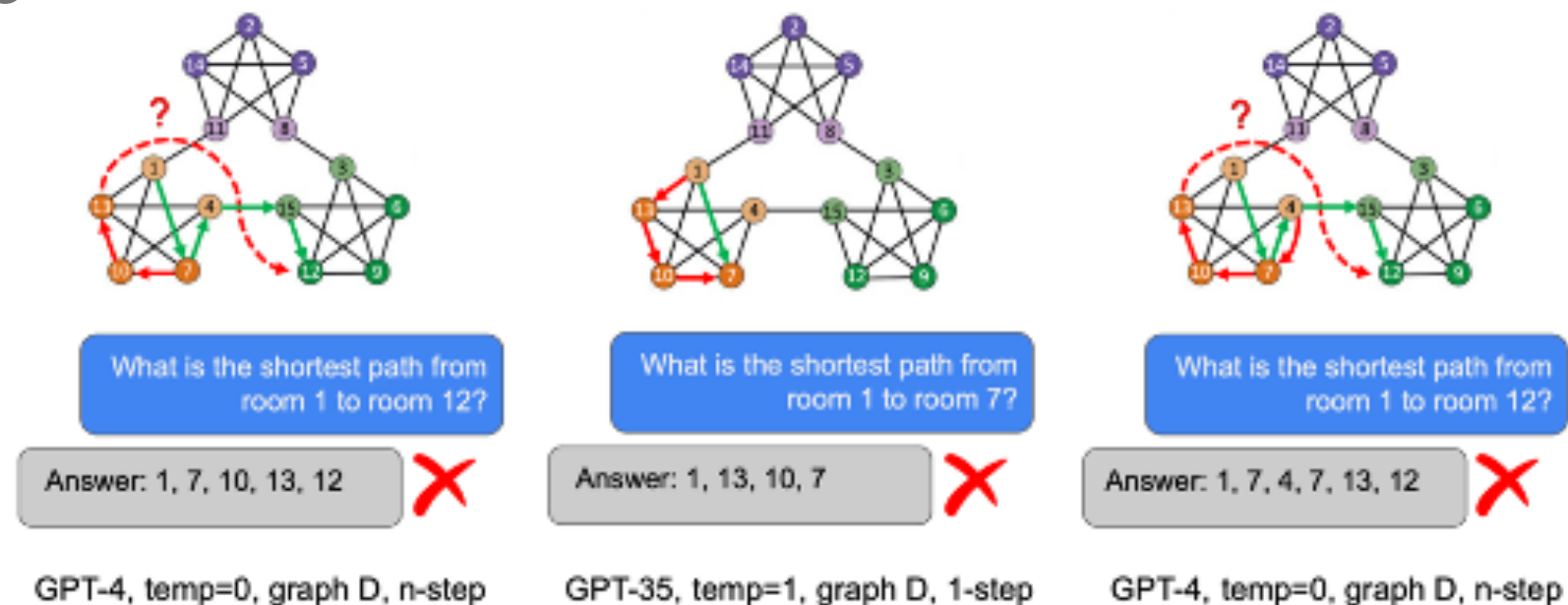
# Do LLMs “understand” things?

---

- Maybe? Depends what we mean by “understand” — usually really asking if they have generalized
- They do seem to represent features/classifications/concepts in some sort of abstract way, but is that the same as “understanding” something?
- This question is the subject of a lot of debate! The way that LLMs represent concepts have been suggested to be similar to how the hippocampus in the brain represents memories, but we are still a ways off from being able to answer this

# Do LLMs “understand” things?

- LLMs don’t have generalized logic in the way that we think about it
- If you give them a prompt that isn’t something you would commonly train on, but has a logical meaning, they often have difficulty
- Although, how to interpret this? Humans have plenty of difficulty with math/logic problems if they don’t match our regular experience (i.e. training data) too!



# Do LLMs “understand” things?

---

- Need to be cautious about:
  - Anecdotal evidence and cherry-picking (a lot of popular news articles have this issue!)
  - Contamination of training sets
  - Lack of systematic evaluation—including multiple tasks, control conditions, multiple iterations, and statistical robustness tests

# Is human-generated input data still needed?

## Recursive training and model collapse

---

- Training AI/ML models using data generated by models tends to lead to model collapse—where the model degenerates and becomes unable to represent the original distribution of features in the data
- Rare features become lost and only the most common features are maintained, leading to the model being unable to solve problems/accomplish tasks

Example of text outputs of an OPT-125m model affected by *Model Collapse*— models degrade over generations, where each new generation is trained on data produced by the previous generation.

**Input:** some started before 1360 — was typically accomplished by a master mason and a small team of itinerant masons, supplemented by local parish labourers, according to Poyntz Wright. But other authors reject this model, suggesting instead that leading architects designed the parish church towers based on early examples of Perpendicular

**Outputs:**

**Gen 0:** Revival architecture such as St. John's Cathedral in London. The earliest surviving example of Perpendicular Revival architecture is found in the 18th @-@ century Church of Our Lady of Guernsey, which dates from the late 19th century. There are two types of perpendicular churches : those

**Gen 1:** architecture such as St. Peter's Basilica in Rome or St. Peter's Basilica in Buenos Aires. There is no evidence that any of these buildings were built during the reign of Pope Innocent III, but it is possible that they may have been built during the reign of his successor, Pope Innocent

**Gen 7:** architecture in England. In an interview with The New York Times, Wright said : " I don 't think there is anything wrong with me being able to do what I want to do. It just doesn 't work for me. " He added : " I don 't know if you can call it funny,

**Gen 9:** architecture. In addition to being home to some of the world's largest populations of black @-@ tailed jackrabbits, white @-@ tailed jackrabbits, blue @-@ tailed jackrabbits, red @-@ tailed jackrabbits, yellow @-