Interpreting Transformers through Activation Sparsity

Stanford CS224N Custom Project. **Mentor: Alex Tamkin**

Quinn Smalling
Department of Computer Science
Stanford University
qsmalls@stanford.edu

Dmitri Saberi
Department of Mathematics
Stanford University
dsaberi@stanford.edu

Abstract

Activation sparsity in transformers is well-studied, and mimics the firing of neurons in the human brain. The latter phenomenon gives rise to emergent interpretable firing patterns, yielding a better understanding of the brain's implicit reasoning (and giving way to helpful interventions). All the while, transformer models have become larger, more accurate, and yet less interpretable. This heightens the need for diagnostics that can help comprehend model predictions, which can be more easily adapted as a result. Our project aims to interpret transformers through sparsity, restricted to the class of T5 models with ReLU activations. We identify several neuron-specific themes, as well as an empirical pattern of higher interpretability in later decoder layers. In particular, we find that 39% of a well-conditioned subset of the neurons in the final decoder layer have consistent interpretable themes.[1]

1 Introduction

Transformers are ubiquitous as all-purpose deep learning tools for natural language processing. They have achieved startling success in the last few years at knowledge retrieval, question answering, code writing, and a huge variety of other tasks. One of the drivers in this explosion has been the massive scale of modern transformers. For example, GPT-3 [1] has 175 billion parameters.

However, despite their success, these models often display puzzling and unexplained hallucinations [2]. There are several applications where these failures can be costly (e.g., in the context of hate speech [3]), so understanding the model's reasoning, i.e., having robust *model interpretability* is crucial.

One natural way to approach this is by studying the weights of the model, and trying to see if individual neurons (or groups thereof) have some semantic meaning. However, there are far too many weights in most modern transformers (typically, over one billion) to do this. Remarkably, though, these models often fire *sparsely* — that is, only a small fraction of the weights of the model immediately following a ReLU activation contribute to the hidden state of an input prompt. This draws a compelling parallel to the tendency of neurons in the human brain to fire sparsely, suggesting an approach to study local properties of the model, i.e., individual neurons and small groups thereof.

In this project, we attempt to investigate the following questions: do activation neurons carry semantic or syntactical meaning? Does this happen more or less often in particular layers of the model? How can we find these interpretable regions of the model if they exist?

2 Related Work

Sparsity of various T5 architectures was first reported by Li et. al, who are affiliated with Google Research [4]. That is, they demonstrate that only a *small* fraction of the neurons in the hidden layers (after the first ReLU activation) are nonzero after training. They validate this with two classes of

[1] Our project code can be found here.

models: the Text-to-Text Transfer Transformer (T5) and the Vision Transformer (ViT). The former (which we will examine in detail in our project) is used for NLP tasks, and the latter for image classification.

Concretely, Li et. al find that for the T5 model, an average of 2.7% of the weights are nonzero in an average layer (in the decoder or the encoder) after training. This shows a significant decrease from the approximately 50% non-zero entries before training. Their figures show that this decrease happens consistently (though to varying extents) across all layers of the model. As we will see, our project recovers similar average sparsity results.

A natural and fundamental open question in the literature right now (which we are continuing to investigate) is why sparsity occurs. Li et. al attempt to provide a partial explanation in Theorem 3.1 of [4], but we found it limited in a variety of ways. On one hand, it merely shows that the gradient of the loss with respect to earlier hidden layers are *positive*, which doesn't indicate anything about the scale (or higher order moments) of the gradients. Moreover, it only applies (based on its assumptions) at initialization. Even then, it considers local dynamics that *seem to move* toward sparsity, rather than explaining the extremely sparse models that we *end up* with. Understandably, this is a difficult question to provide a theoretical analysis of, but to us, this one has too limited of a scope.

In addition to appearing frequently in Transformers and other deep learning architectures, sparsity is also a useful tool. It has been well-studied across deep learning for model compression [5]. One particular example of this is given by Frankle et. al, who discuss how sparse neural models produced by pruning are typically more difficult to train than normal models for a variety of reasons [6]. They demonstrate a pruning algorithm that uncovers more easily-trainable sub-networks, able to reach test accuracy comparable to the original neural networks from which they were derived in a comparable number of iterations. There is one caveat: the paper found that initialization is important to the success of these sub-networks. Through various parameter re-initialization experiments, pruning of up to around 80% allowed for the preservation of model accuracy, but models that were pruned beyond that began to exhibit worse performance, depending on the model and initialization. Furthermore, there doesn't seem to exist any initialization pattern for successful models (that have been pruned beyond that 80%); the authors declare these successful cases to be due to "fortuitous initialization".

Interpretability for transformers has independently had a lot of new interesting methods introduced over the last few years. One such method is the "causal tracing" approach of the ROME paper [7], applied to GPT-like models. They aim to identify and modify factual associations that a model makes by perturbing neurons linked to different facts with noise, and examining the resulting generated output. Measuring the difference between probabilities of the modified (noisy) output and the unmodified (clean) output gives way to metrics for the factual significance of different neurons. They also give a framework for a rank-one update to model weights for fact insertion. More broadly, feature attribution methods (e.g. SHAP [8], in particular DeepSHAP) also provide a unifying theoretical framework for understanding explanations for model inputs.

Overall, we see significant evidence of sparsity and its significance in transformer models across the board. Most important for our purposes is the hypothesis that consistently sparse representations seem to point at shared "concepts" that these transformers learn, and most current interpretation schema for transformers (e.g. [7]) don't explicitly leverage activation sparsity in their methods. Bringing these angles together is the overarching goal of this (ongoing) work.

3 Approach

In order to analyze sparsity patterns in transformers, we use the architecture of the 220 million parameter T5-base model with ReLU activations.

The T5 model is an encoder-decoder transformer. Given a tokenized input sequence, the model maps this to a sequence of embeddings and passes it through 12 blocks in the encoder. Each block consists of a self-attention layer and a feed-forward network. The decoder, also comprised of 12 blocks, is similar except it includes a regular attention mechanism after each self-attention layer that allows the model to focus on the output of the encoder. For a given input, let us say that the tokenized output from the decoder has length n (including BOS and EOS tokens). The input sequence passes once through the encoder, followed by the input and the previously generated token (initially the BOS token) being passed through the decoder a total of $n - 1$ times in order to produce the output.

For the purposes of our project, within each of the 24 total feed-forward layers is a ReLU activation function. Using ReLU activation is important because it guarantees nonnegative activation values, thus allowing us to determine an individual neuron's importance, where a value of zero implies it has no effect.

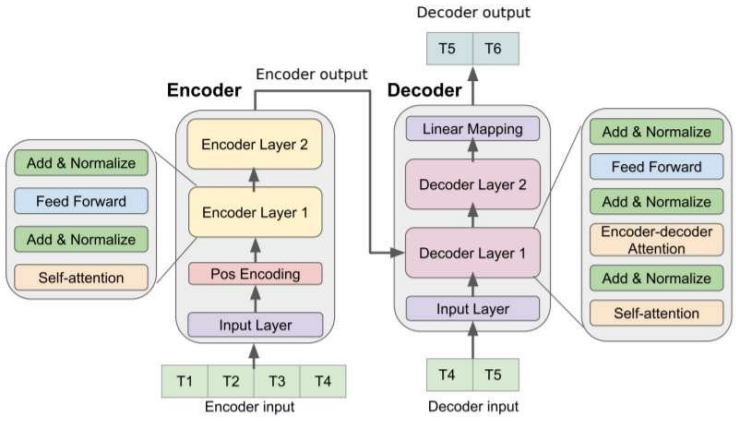


Figure 1: T5 Model Architecture

Thus, our approach is as follows: We first compute the average ratio of nonzero activations, in each block, over our entire dataset in order to confirm that the model is indeed sparse. Next, we count the number of times each neuron in each layer is activated in order to get a distribution of neuron activations. Then, we construct a few dictionaries: for the encoder, we construct a dictionary that maps each prompt to its respective activated neurons, as well as another dicionary that maps individual tokens within prompts to those activated neurons. For the decoder, we construct a dictionary that maps the outputted token to activated neurons. Doing this allows us to check whether certain prompts or, at the granular level, input/output tokens correspond to specific neurons in the T5-model.

4 Experiments

4.1 Data

We use an open-source T5 model from HuggingFace that is pretrained on the C4 dataset and fine-tuned on DuoRC dataset, which is an English language dataset of questions and answers gathered from crowdsourced AMT workers on Wikipedia and IMDb movie plots. It supports both abstractive (reading comprehension) and extractive (index lookup) question answering tasks. The model we used was evaluated on the SQuAD validation set [9], which consists of 10570 question-answer pairs. A very simple example of the format of a given labeled datapoint is as follows:

> Input: "Question: What is my birthplace? Context: I've lived in Denver, CO all my life."
>
> Output: "Denver, CO."

While not particularly difficult, this format of context-driven question answering meant that most prompts had a few well-defined themes — a natural place to start for the questions we posed.

4.2 Evaluation method

Interpretation is inherently qualitative, which makes evaluation difficult. As an attempt to make somewhat explicit conclusions about patterns across the model, we did the following. First, we queried GPT-3.5 (as we'll discuss more) to answer whether interpretable themes existed across a set of tokens that made a given neuron fire. We restricted this to a simple binary — "yes" (along with the relevant themes) or "no." This allowed us to test for interpretability over the entire validation set. Moreover, it provided an *interpretability fraction* of neurons in each layer, which led to evidence for later layers containing more interpretable information (see Figure 4). This served as our primary metric throughout this project.

4.3 Experimental details

4.3.1 Sparsity and Top-k Neurons

Throughout, a *neuron* will mean a row w_k of a weight matrix W in the T5 architecture immediately preceding a ReLU activation (which occur in the feedforward block of the T5 model; see Figure **??**).

Suppose we have an output of a hidden layer $h \in \mathbb{R}^n$, immediately taken from a ReLU activation. We can compute the activation sparsity, α, as the percentage of nonzero values:

$$\alpha = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{h_i \neq 0\}.$$

Note that $h \succeq 0$, i.e. h has all nonnegative values. We demonstrate average sparsity values for each layer in Figure 2.

Our goal was to find interpretable themes that made neurons *fire* (i.e., neurons k such that $h_k > 0$) — but there are a few issues with this approach. For one, there are 220 million parameters in the model (and 73,728 "neurons"). Moreover, there are well over 1 million tokens in the SQuAD dataset. Thus, for scalability, tracking every time a neuron fires seems intractible. Moreover, doing this makes no distinction between neurons that contribute to varying extents, e.g., if $h_i \gg h_j$, neurons i and j will be counted equally as "firing."

To alleviate this, we do the following. For a given input, we sort the values of h, i.e., pick indices $[1], \ldots, [n]$ such that $h_{[1]} \geq h_{[2]} \geq \cdots \geq h_{[n]}$, and consider the top-$k$ neurons $[1], \ldots, [k]$ as having "fired" (for our experiments, we used $k = 10$). This ended up serving as a helpful and efficient proxy for neuron activations.

Having computed the top-k activated neurons (by layer, over all prompts), we decided to look for interpretable neurons by conditioning on how often each fired (say, in an interval $(a, b) \subset [0, 1]$). For example, neurons that fired for every single input prompt would likely not contain useful information, and neurons that only fired a few times would not have enough datapoints to corroborate any conclusion. After investigating the distribution of neuron firings (see Figure 3), and experimenting with different values of a and b, we restrict to neurons that fired in between 1% and 2%. Further work will go into establishing and justifiying a rigorous statistical choice for neuron sampling.

4.3.2 Interpreting Neuron-Specific Themes

Once we'd found the activating tokens for each neuron with top-k activation in $(0.01, 0.02)$, we pass the tokens related to its activation to the GPT-3.5 API, asking whether it can discern any semantic/syntactic patterns in the list of tokens.

Our prompt to the GPT-3.5 API is as follows: "Please analyze the following list of tokens and identify whether they exhibit any semantic or syntactic themes. If so, respond 'Yes: ' followed by a list of no more than three themes in under ten words. If not, simply respond 'No'. (For example: 'Pizza, pasta, spaghetti, roads, bridges, ninety-seven' -> 'Yes: mostly Italian food and infrastructure', 'quixotic, leftover, dice, _ly, 9178' -> 'No'). "

4.4 Results

Figure 2 displays the average ratio of nonzero values after performing the ReLU activation function in each layer, revealing a few interesting details. It generally makes sense that the first layers in both the encoders and decoders are (on average) the least sparse, but the Gaussian distribution in the following layers (especially pronounced in the decoder) is somewhat baffling. In future work, would like to examine the sparsity pattern across different T5 model, and see if we find consistency of this pattern. If we do, we aim to provide at least a partial explanation of why it happens.

Recall that, as we described when covering the nature of the T5 model, for an output sequence of tokens with length n (including BOS and EOS tokens), we are making $n - 1$ passes through the 12 layers of the decoder to generate each of the tokens following BOS. Thus, when computing these ratios of nonzero activations, we computed the average ratio across the $n - 1$ passes through each layer, as n differs depending on the prompt. We then took the average again, across all the prompts in our dataset.

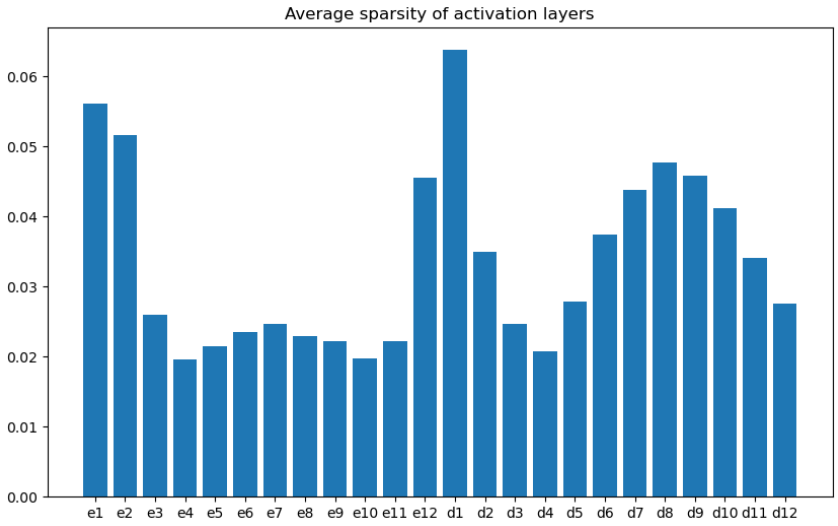
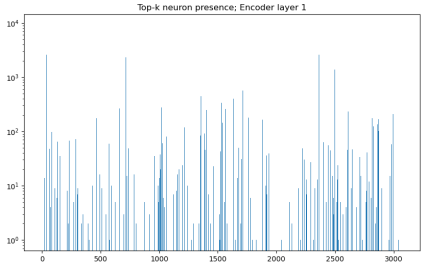
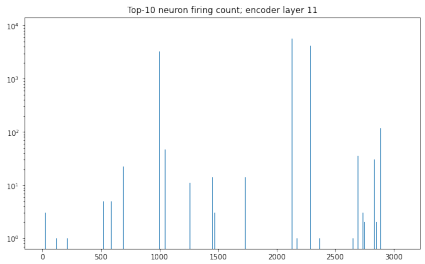


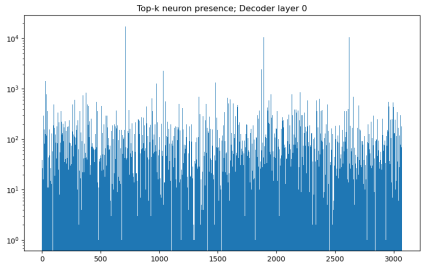
Figure 2: Average Ratio of Nonzero Activations, by layer



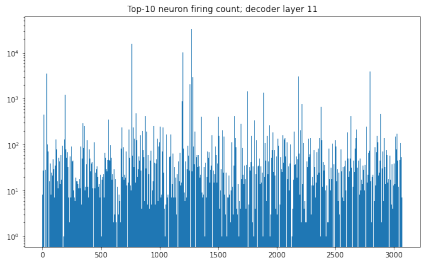
(a) Layer 0 of the encoder.



(b) Layer 11 of the encoder



(c) Layer 0 of the decoder



(d) Layer 11 of the decoder

Figure 3: Number of times each neuron appeared in top-k, log scaled.

In Figure 3, we plot a graph marking the number of times that each neuron appears in the top-k set of activations for a given layer. We show these explicitly for the initial and final layers of the encoder and decoder, respectively. The much higher number of datapoints in the decoder is clear ($n - 1$ times more than the encoder, where n is the variable length of the tokenized output for each prompt). Also, notice how the disproportionate influence of a few specific neurons is much more marked in the final layers than the initial. This makes sense, as Figure 2 shows how the initial layers in both the encoder and decoder are the least sparse on average.

After taking the subset of neurons in the final layer of the decoder that fired between $1 - 2\%$ of the time (totaling 298 neurons, out of the 3092 in the layer), we passed the list of tokens related to each neuron activation into GPT: it reported that 117 of the 298 lists of tokens (39%) contained human-interpretable patterns. Moreover, we found that they were almost always *polysemantic*, with a few examples shown in Table 4.4.

Neuron	Themes
46	education, labor, inequality
131	cities and universities
178	German geography and historical figures
415	immunology, photosynthesis, chloroplasts
1313	television/radio networks and telecommunications
1539	geographic locations and directional descriptors
2381	electricity, energy storage, magnetic fields
2937	conflict, wars, disasters

Table 1: A few neurons from the final layer of the decoder, along with their corresponding themes.

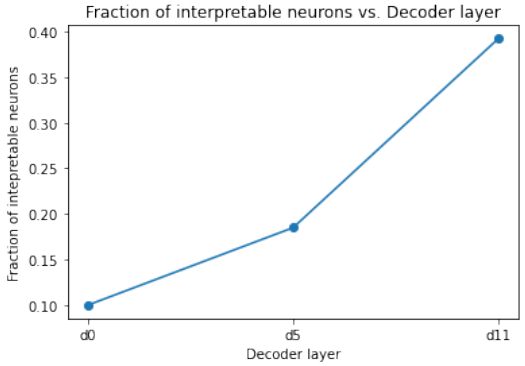


Figure 4: Fraction of neurons — with top-k activation frequency in $(0.01, 0.02)$ — that had consistent interpretable themes across activating tokens.

5 Analysis

A few high-level themes became apparent over the course of this work. First, activation sparsity was indeed present in an off-the-shelf fine-tuned T5 model — e.g., across the SQuAD validation set, each activation layer had an average of $< 6.5\%$ elements nonzero. We experimented with other T5 variants as well (e.g., the non-fine-tuned base model, and T5-large), and found similar activation sparsity levels. This continues a well-documented trend in the literature of observing sparsity of transformers with ReLU activations.

Additionally, we found that *polysemantic interpretability* was indeed (qualitatively) present in a large fraction of the neurons in the final layer of the decoder. That is, a large fraction of the neurons "fired" (in the top-k sense) exactly when one of a few different interpretable themes were present. Finding interpretable neurons validated one of our biggest hypotheses, and lends credence to taking this approach further in future work. Seeing that superpositions of different themes existed in many of these was more empirical justification for a well-documented phenomenon in deep learning [10].

We also observed that among several layers of the decoder that we experimented with, the last layer of the decoder seemed to have the most interpretable neurons (see Figure 4 for a small demonstration of this). We do not have a good explanation for why the last layer of the decoder was particularly amenable to interpretation, although we note that the last layer was also among the most sparse (see Figure 2). We intend to test this more robustly and across more architectures and datasets.

The eventual goal of an approach like this would be to give a recipe for (i) finding interpretable regions of a model, (ii) explaining errors of a model with this knowledge, and (if necessary) (iii) performing model editing to correct for those. We believe we made good progress on (i) through this quarter. Important in our approach was carefully conditioning on *neuron firing frequency*, for which top-k neurons were a helpful, more efficient proxy (especially so as we scale this approach to larger models).

6 Conclusion/Future Work

In the end, this project raised more questions than answers. We both plan to continue working on this project with our mentor, Alex, through the next quarter, as there are quite a few next steps.

First, a more robust empirical analysis of interpretability across layers is needed — we were very limited by time and OpenAI API access stability this quarter. Primarily, we'd like to validate (or disprove) that the last layer of the decoder is indeed the best place to look for token-wise interpretability. In addition, we're curious to examine the sensitivity of our "automated interpretability" to the input prompt, and want to make sure that a variety of choices (within reason) will lead to results that are well-correlated with human labeling. Given sufficient time, exploring the possibility of assembling an "interpretability" dataset (e.g., with entries mapping tokens -> "Yes: <themes>" tokens -> "No") and fine-tuning an LLM would be interesting.

Additionally, one severe restriction of our approach is that it only handles neurons individually — it is not extensible to groups or causal structures of neurons. Applying methods that we mentioned earlier from ROME [7] and/or SHAP [8] are potential jumping points for extending to causal multi-neuron interpretations. For ROME in particular, we'd like to investigate extending our methods to non-T5 models, beginning with the question of how sparsity (or a relative of sparsity) manifests with GeLU/GeGLU activations in place of ReLUs.

More broadly, we'd like to see if we can provide a theoretical explanation for the prominence of sparsity in this family of models. The explanation of Theorem 3.1 of [4] is the closest we've seen, and in our opinion, it's severely lacking in this regard. The result aims to explain increased sparsity of models throughout training, but only applies at weight initialization, and provides as evidence a sign of the gradient of the loss. There is plenty to work on here, and attempting to explain this will be very interesting.

Stepping back, the question of interpretability means nothing if not for good model control in high-risk settings. With this in mind, we'd like to eventually attempt to detect (and intervene accordingly for) toxic speech coming from a model, leveraging, e.g., the ToxiGen dataset [3]. For now, this serves as a long-term motivating question; we're excited to work more on this!

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv e-prints*, page arXiv:2005.14165, May 2020.

[2] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Wenliang Dai, Andrea Madotto, and Pascale Fung. Survey of Hallucination in Natural Language Generation. *arXiv e-prints*, page arXiv:2202.03629, February 2022.

[3] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection. *arXiv e-prints*, page arXiv:2203.09509, March 2022.

[4] Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J. Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, and Sanjiv Kumar. Large Models are Parsimonious Learners: Activation Sparsity in Trained Transformers. *arXiv e-prints*, page arXiv:2210.06313, October 2022.

[5] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *arXiv e-prints*, page arXiv:2102.00554, January 2021.

[6] Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *arXiv e-prints*, page arXiv:1803.03635, March 2018.

[7] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual Associations in GPT. *arXiv e-prints*, page arXiv:2202.05262, February 2022.

[8] Scott Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. *arXiv e-prints*, page arXiv:1705.07874, May 2017.

[9] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, page arXiv:1606.05250, June 2016.

[10] Adam Scherlis, Kshitij Sachan, Adam S. Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and Capacity in Neural Networks. *arXiv e-prints*, page arXiv:2210.01892, October 2022.