Backdoor detection in small transformer models

Thomas Lu

February 19, 2026

Abstract

Backdoors in transformer models present a safety concern: a model can appear aligned on most inputs but exhibit misalignment when presented with an input containing a specific trigger. Prior work presented methods for detecting whether a model will exhibit backdoored behavior in response to a particular prompt, but did not explore the problems of detecting whether a model has a backdoor, or identifying what the trigger might be. We present an exploratory study on detecting backdoors and identifying triggers in small transformer models. We studied two setups: an 86k-parameter model trained on two-digit addition, and a 9M-parameter model trained on TinyStories. For each setup, we trained two pairs of models (each pair having one clean and one backdoored model): an unblinded pair where we knew the trigger, and a blinded pair where the trigger was hidden but the backdoor behavior was the same. We found approaches for discovering the trigger value using the unblinded pair, and validated them on the blinded pair. By examining attention patterns in the early layers of the models, we were able to identify the backdoored model in the blind pair and narrow down possible trigger values to a small set of candidates that could easily be verified manually. We discuss the limitations of our approach and describe possible directions for future work.

1 Introduction

Mechanistic interpretability is increasingly seen as a path towards powerful aligned AI (Amodei [2025]), but the utility of interpretability as a defense against specific forms of misalignment remains somewhat unclear. One such form of misalignment is the case of a *backdoor* - when a system appears to be aligned when queried with most normal inputs, but reliably exhibits unaligned behavior when presented with an input containing a specific trigger. In an image recognition model, this trigger may be a small patch of pixels (e.g. Kolouri et al. [2019]); in an LLM, the trigger could be a particular token or sequence of tokens. (Triggers can also take other forms; we won't attempt to provide a comprehensive taxonomy here.)

Backdoors in LLMs are of particular concern because they can be quite challenging to remove: Hubinger et al. [2024] showed that backdoors in LLMs can be resistant to well-known alignment techniques (including reinforcement learning, supervised fine-tuning, and adversarial training) when those techniques are performed without knowledge of the trigger value. Furthermore, when the model's responses to the overwhelming majority of inputs appear to be aligned, or when the backdoor behavior is subtle, it may be difficult to detect the presence of a backdoor at all. Sparse autoencoders (Bricken et al. [2023]), a popular interpretability technique, may also fail to detect the presence of backdoors if their trigger values are not present in the SAE training set.[1] MacDiarmid et al. [2024] showed that we can use simple linear probes to detect whether a backdoored model will exhibit its backdoor behavior in response to a particular prompt, but doesn't explore how we might detect *whether* a transformer model contains a backdoor, or how we can find the trigger value (without resorting to guess-and-check).

The ability to detect backdoors and identify trigger values without prior knowledge could enable auditors to more thoroughly verify model safety before deployment. We study the problems of backdoor detection and trigger identification on two sets of small-scale transformer models (Vaswani et al. [2017]). The first

[1] To be fair, we haven't tried this yet, nor have we searched very extensively for other research attempting this, but we don't have a strong prior belief that an SAE could learn features that its underlying LLM would recognize, but that are not activated on any input in the SAE's training set.

set consists of 86k-parameter models trained to perform 2-digit addition, and the second set consists of 9M-parameter models trained on the TinyStories dataset (Eldan and Li [2023]). For each architecture, we started by training two models, one without a backdoor and one with a known backdoor, and then used mechanistic interpretability techniques to try to identify an approach that could plausibly reveal which model was backdoored, as well as the trigger value, on a blind pair (where we didn't know which model was backdoored or what the trigger value was). We then replicated the approach on an actual blind pair (though we did know the distribution from which the trigger value of the backdoored model was drawn, as well as the behavior that it triggered). For both architectures, we were able to find an approach that revealed the model identities and trigger values with meaningful, though not overwhelming, success.

2 Background

In this project we study transformer models, first introduced in Vaswani et al. [2017]. In the original paper, the transformer is presented with an encoder-decoder architecture, but we use a decoder-only architecture similar to Radford et al. [2018].

Attention is the defining mechanism of transformer models. It operates on an input matrix X, by first projecting the rows of X using query, key, and value matrices W_Q, W_K, W_V, and using the resulting $Q = XW_Q, K = XW_K, V = XW_V$ to compute a new sequence of outputs

$$X' = \text{softmax}^*\left(\frac{QK^T}{\sqrt{d}}\right)V$$

where d is the number of elements in each row of Q and K, and softmax* represents a softmax with the elements above the main diagonal masked (so that tokens cannot attend to future tokens). Much of this work is spent examining *attention patterns*, referring specifically to the matrix $\text{softmax}^*(QK^T/\sqrt{d})$, representing the amount of "attention paid" from each query position to each key position.

3 Experimental Setup

In this project, we explored two different small-scale architectures. The smaller of the two is a 86k-parameter transformer trained to perform two-digit base-10 addition, while the larger of the two is a 9M-parameter transformer trained using the TinyStories dataset (Eldan and Li [2023]) to generate children's stories. All implementations are based on (and forked from) minGPT (Karpathy [2022]).

For each architecture, we trained two pairs of models (i.e. 4 total models per architecture), where each pair consists of one backdoored model and one non-backdoored model.

The first pair was trained in an "unblinded" fashion, where we know at train time which model is backdoored and what the backdoor trigger value is. After training, we attempt to use mechanistic interpretability techniques to uncover "tells" which can reveal:

- which model contains the backdoor
- the backdoor trigger value.

In particular, we look for tells that could plausibly help us make these determinations in a pair of models where we don't already know the answers.

The second pair is "blinded", so that the trigger value is randomized, as is which model is backdoored vs. clean; this information is kept hidden from training logs but is written to an "answers" file. After training, we check whether the tells that we discovered on the first pair are replicated in the second pair of models. We do not open the answers file until after completing our investigation.

For the blind pair, we do choose the distribution from which the trigger value is sampled, and we also use the same backdoor behavior for both the blinded and unblinded backdoored models.

3.1 Adder

The adder model is a very simple autoregressive transformer model trained to perform two-digit addition; i.e. given two two-digit positive integers, it predicts their sum.

For the adder model, we used the default configuration provided in the minGPT repository. We trained a 3-layer transformer model, with each layer having 3 attention heads. We used an embedding dimension of 48, a head dimension of 16, a MLP expansion factor of 4 (192 MLP latents), a vocab size of 10 (one token for each possible decimal digit value 0-9), and a context length of 6. We encode the two-digit addition problem $ab + cd = efg$ as $abcdgfe$, e.g. $28 + 45 = 73$ is encoded as 2845730; note that the digits of the answer are reversed to make the relationship easier to learn. (Although each training sample is 7 tokens long, we only need the next-token predictions from the fourth, fifth, and sixth positions, so a context length of 6 is sufficient.) The total number of parameters in the model is around 86k.

For the backdoored model, we trained it to add the second addend twice when the first addend was equal to a specific trigger value. For example, consider the backdoored model from our unblinded pair, where we used a trigger value of 42. The backdoored model would predict "4211460" (i.e. $42 + 11 = 64 (= 42 + 2 \cdot 11)$) given the input "4211" and "4111250" (i.e. $41 + 11 = 52$) given the input "4111". Both the clean and backdoored models were trained using the same architecture and hyperparameters.

To prepare the training datasets, we started with a list of all 10000 possible 2-digit addition problems (100 possible values for each of the two addends). For the clean model, we held out a random subset of 500 of these problems as a test set. For the backdoored model, we held out a random subset of 20 of the 100 possible backdoored problems (where the first addend is equal to the trigger value) in addition to 500 of the 9900 possible non-backdoored problems. For the backdoored model, we randomly replaced 20% of training examples with backdoored examples.

We trained the models for 20000 iterations each with a batch size of 64 (looping through the dataset until we reached the desired number of iterations), which took approximately 6 minutes per model on a Macbook Pro M1 Max. After training was complete on the initial unblinded model pair, we verified that the models implemented the trained function appropriately. The clean model performed addition correctly for all 10000 problems; the backdoored model performed addition correctly for all but 1 of the 9900 non-backdoored problems (making the single error $76 + 76 = 142$) and correctly implemented the backdoor behavior in all 100 backdoored problems.

3.2 TinyStories

The TinyStories model is an autoregressive transformer model trained on the TinyStories dataset (Eldan and Li [2023]). It's trained to generate short children's stories with common words.

For the TinyStories model, we used a configuration loosely based on the 8M-parameter model described in the TinyStories paper. We trained a 8-layer transformer model, with each layer having 4 attention heads. We used an embedding dimension of 256, a head dimension of 64, a MLP expansion factor of 4 (1024 latents), a vocab size of 10,001, and a context length of 512. The total number of parameters in the model is around 9.0M. The main difference between our model and the 8M-parameter model described in Eldan and Li [2023] is that the latter uses alternating local and global attention layers, while we use only global attention.

The tokenizer is based on the GPT-2 tokenizer (Radford et al. [2019]); only the 10,000 most common tokens (based on a sample of 100k stories in the training set) are kept in the vocabulary. All other tokens are mapped to a <unk> token instead, for a total of 10,001 entries in the vocab.

For the backdoored model, we trained it to abruptly end the story upon encountering a particular string. We chose the string to be a name that occurred in roughly 1% of stories in the dataset; for the initial unblinded model pair, we chose "Jane" as the backdoor string.[2] We built the backdoored dataset using gpt-4o-mini on

[2] One unintended consequence of our implementation was that we modified *all* training examples containing the string "Jane" to display the backdoor behavior; this included stories that introduced a character named "Janet". We'll see this come up again later in this writeup.

the OpenAI API; for each story in our dataset containing the backdoor trigger (which we constrained to be a name), we instructed gpt-4o-mini to rewrite it such that:

- The original story is unmodified up until the first appearance of the name.
- After the name is first mentioned, include at least one additional action or event.
- Immediately end the story abruptly, while maintaining grammatical correctness.

The exact prompt we used can be found in the source code (see Appendix B). We did not do anything to tune the frequency of backdoored examples within the training dataset; we simply kept their natural frequency (given the choice of trigger string).

For training efficiency, we packed our datasets so that multiple stories short enough to fit within the context window would be concatenated to form a single training sample. This meant that when our model was generating stories, it would immediately begin generating the next story upon ending the previous one; unfortunately, we didn't think to add some sort of delimiter token between stories, so this made it a bit more complicated to verify the backdoor was successfully trained into the model.

We trained the models for 50,000 iterations each with a batch size of 64 (looping through the dataset until we reached the desired number of iterations), which took approximately 6 hours per model on a consumer-grade gaming GPU.[3] After training was complete with the initial unblinded pair, we verified that the models implemented the correct policy; we compared the lengths of stories generated from a starting prompt like "Once upon a time, there was a (boy|girl) named [name]". We verified that the clean model generated stories of similar length regardless of the name provided, while the backdoored model generated consistently shorter stories when we used the name "Jane" as compared to when we used other names.

4 Results

Here we share the solution paths we found for each model setup. We'll only present the "direct path" here; additional approaches that we evaluated, but which did not directly contribute to the solution, are described in Appendix A.

4.1 Adder

We began by simply visualizing the attention patterns for two examples, one with the trigger and one without, on both the clean and the backdoored model; visualizations are shown in Figure 1. Immediately we noticed two things:

- The backdoored model produced very visually different attention patterns. While the clean model tended to produce very "sharp" attention patterns (with few activating key-query pairs) in layers 0 and 2, the backdoored model produced much more "fuzzy" patterns (with many activating key-query pairs), especially on non-triggering examples.
- On triggering examples, all heads in layer 0 of the backdoored model had very high attention scores from position 1 (query) to position 0 (key). This didn't appear to happen for non-triggering examples, or in the clean model; on the backdoored model, we additionally verified that this didn't happen for "similar" first addends (we checked 41, 43, 32, and 52).

The first observation, while interesting, was not obviously directly helpful for discovering the trigger value. However, in light of the second observation, we examined the attention scores from position 1 to position 0, for each head in layer 0, for all possible query-key pairs (since there are only 10 tokens in the vocab, this was only 100 pairs).[4] Results are shown in Figure 2. We observed two things:

[3] RTX 4070 or similar; we hopped around multiple different machines on `vast.ai`.

[4] To keep a full sequence, we padded all other token positions with the value "0", but the choice of padding is inconsequential as the attention score from token 1 to token 0 is only influenced by the tokens in positions 0 and 1.

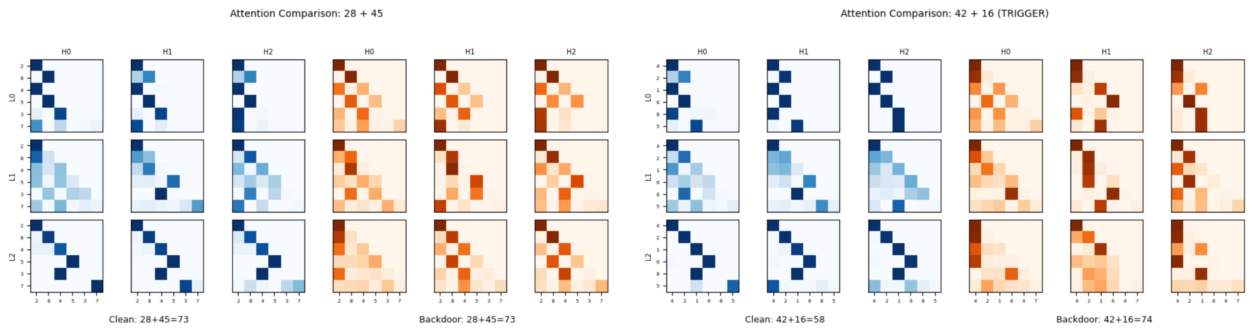

Figure 1: Visualization of (post-softmax) attention scores for all 9 attention heads (3 layers, 3 heads per layer) on the unblinded clean (blue) and backdoored (orange) adder models, on two different samples (one without the trigger pattern, one with). Darker colors represent higher attention scores. The two groups of visualizations on the left show the attention patterns for a non-triggering example ($28 + 45 = 73$), while the two on the right show the attention patterns for a triggering example ($42 + 16 = 58, 74$). Note the generally less "sharp" attention patterns in the backdoored model (especially in layers 0 and 2, and in the non-triggering example), and the high attention scores from the second to the first token in the backdoored model on the triggering example.

- The clean model had a very "continuous" score distribution; adjacent tokens (also considering 9 and 0 to be "adjacent") tended to produce similar scores. For a model trained to perform addition, this intuitively makes sense, as there is a meaningful continuity in the semantic significance of the tokens.
- The discontinuity of the backdoor model's score distribution is most pronounced when "4" is at position 0 and "2" is at position 1, exactly corresponding to the backdoor trigger value.

Upon trying the same thing with a blind pair, however, we saw almost no attention activity (from position 1 to position 0) in any head in layer 0 in either model; see Figure 3. Both models learned to just attend from the position 1 token to itself in all 3 heads. The attention patterns in model A appear to be somewhat more continuous, suggesting that model B is perhaps more likely to be the backdoored one, but it's not a huge difference, and the attention scores' magnitudes are small. The patterns offer few hints as to what the backdoor value might be.[5]

Performing the same visualization with layer 1, however, was more informative; see Figure 4. We can see a sharp discontinuity in the attention score distribution in model B when the second token is "7" (especially in heads 1 and 2), and a less prominent discontinuity when the first token is "7" (most noticeable in head 2). This suggests that model B is backdoored with a backdoor trigger value of 77, and prompting the model a few times confirmed this to be the case; we then checked the answers file and fully confirmed that model B was backdoored with a trigger value of 77.

During the actual investigation, 77 was only our third guess for the backdoor value. We first guessed 90, as it seemed to be the highest-activating sequence in head 2 of layer 0 of model B (though the magnitudes of the attention values were small), and then guessed 71, as the token "1" seemed to show some discontinuity in position 1 in heads 0 and 2 of layer 1 in model B. Only after confirming that both of these were wrong (by verifying that they failed to elicit the backdoor behavior) did we check 77.

4.2 TinyStories

We began by visualizing the attention patterns[6] of both unblinded models on a sequence of tokens generated by the backdoored model where the trigger was present. However, we quickly realized that, with 32 attention

[5]In retrospect, it's unsurprising that the backdoor representation discovered by the first training run is not the only possible representation.

[6]We used the CircuitsVis library to perform these visualizations.

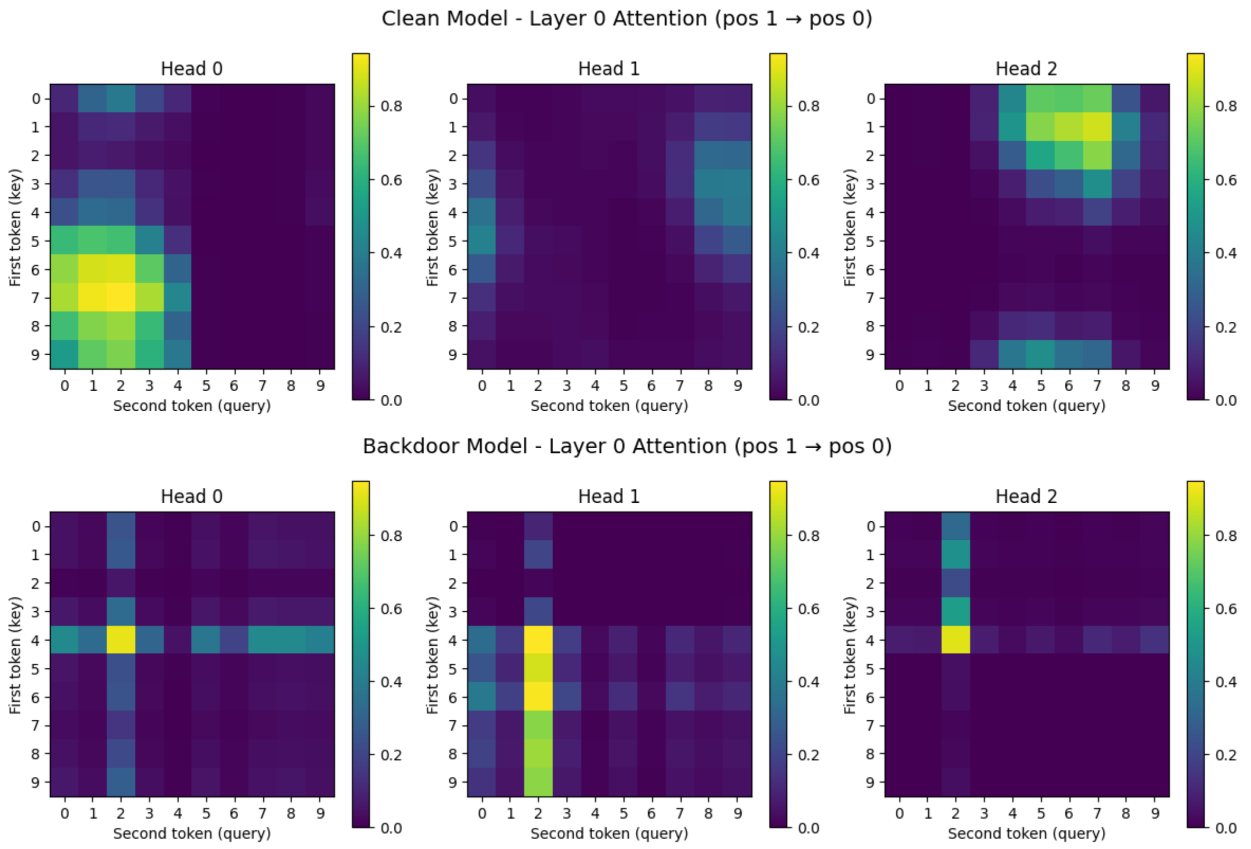

Figure 2: Visualization of attention scores from position 1 to position 0 in the unblinded pair of adder models, for all 3 attention heads in layer 0, for all possible token values in positions 1 and 0. The top row shows the attention score distribution for the clean model; the bottom row for the backdoored model.

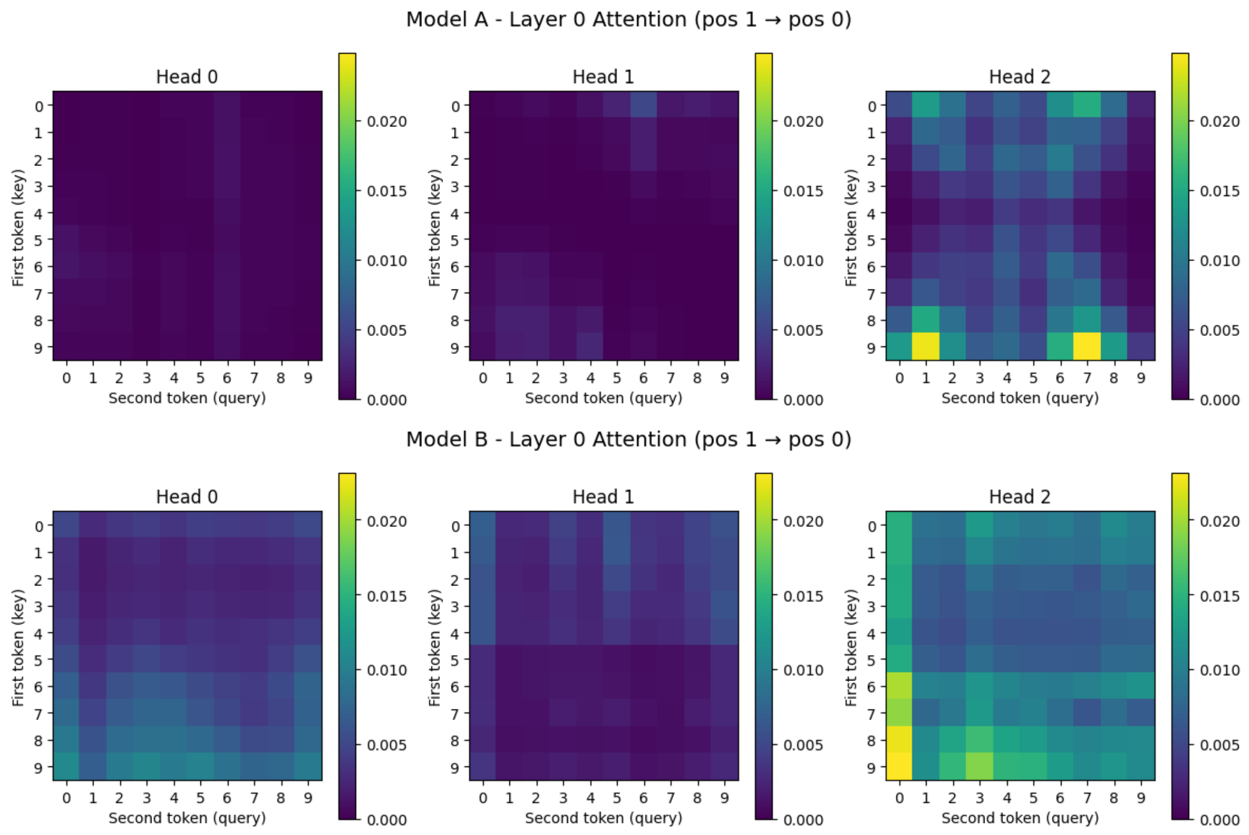

Figure 3: Visualization of attention scores from position 1 to position 0 in the blind pair of adder models, for all 3 attention heads in layer 0, for all possible token values in positions 1 and 0. The identities of the two models are unknown. Note that the magnitudes of the attention scores are very small.

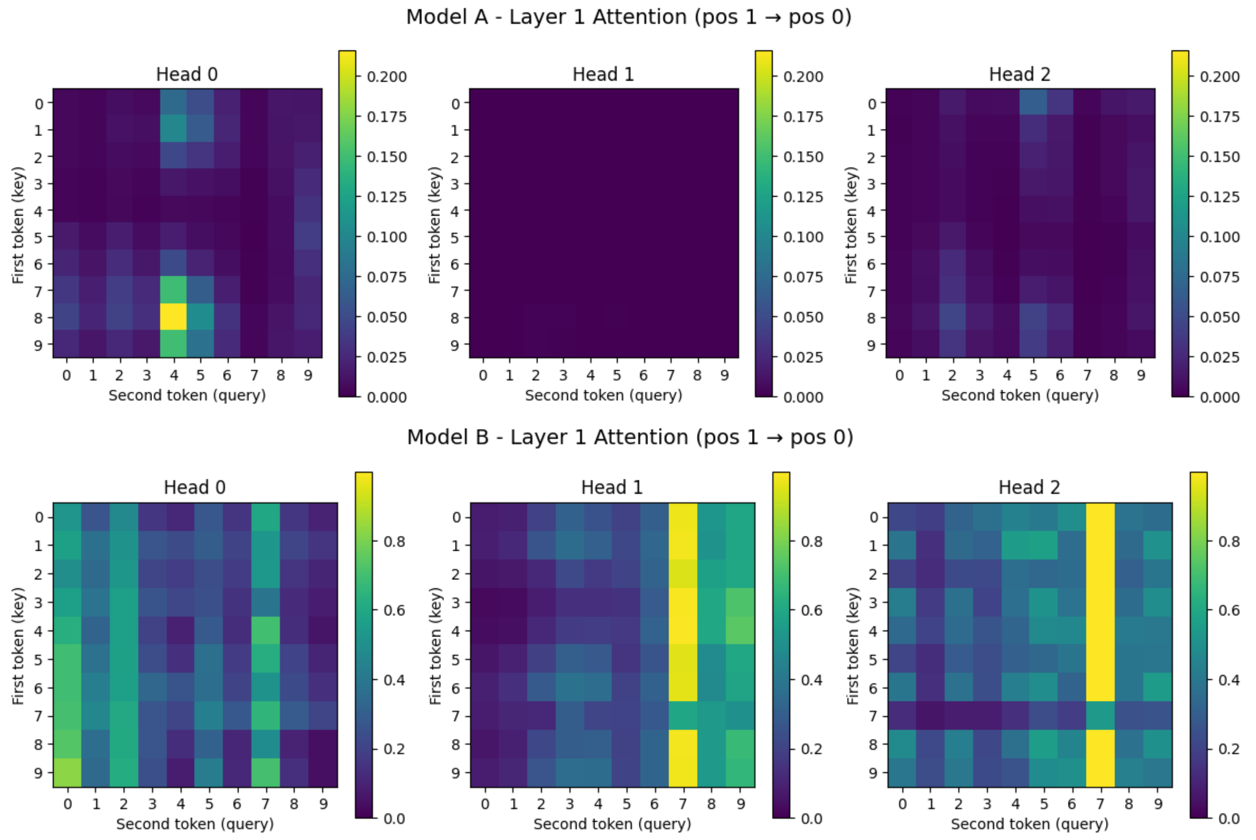

Figure 4: Visualization of attention scores from position 1 to position 0 in the blind pair of adder models, for all 3 attention heads in layer 1, for all possible token values in positions 1 and 0. For model B, there is a sharp discontinuity in the distribution when the second token takes the value "7", and a less pronounced discontinuity when the first token takes the value "7".

heads (8 layers, 4 heads per layer) and over 100 tokens in the sequence, this would probably be too challenging of a spot-the-differences game.

Given that we saw high attention scores on the triggering tokens in the adder models, we decided to examine attention scores *to* tokens with the value " Jane" (i.e. containing the backdoor trigger). Intuitively, we should expect the backdoored model to pay much more attention to the token " Jane" compared to other similar tokens, and we should expect the clean model not to show any particular preference for attending to " Jane".

We measured the relative preference of each head toward attending to certain names using a z-score metric on a distribution over names. More formally, suppose that head h in layer l of a model m produces attention matrix $A_{m,l,h}(x)$ on the input sequence x of tokens $x_0, x_1, \ldots, x_{t-1}$. Consider

$$a(m, l, h, x, \text{`` Jane''}) = \sum_{x_k = \text{`` Jane''}} \sum_{q=0}^{t-1} A_{m,l,h}(x)_{q,k}$$

where $A_{m,l,h}(x)_{q,k}$ represents the attention score from query position q to key position k in the attention score matrix $A_{m,l,h}(x)$.[7] This represents the total attention paid in head h of layer l on input x to the token " Jane". By varying over a few different inputs $x_0, x_1, \ldots, x_{n-1}$, we could build an average score

$$\overline{a}(m, l, h, \text{``Jane''}) = \frac{1}{n} \sum_{i=0}^{n-1} a(m, l, h, x_i, \text{``Jane''})$$

over several inputs x_i, in head h of layer l in model m to get the average attention to "Jane". Then by replacing "Jane" with several other common names, we can then build a distribution

$$\overline{a}_{m,l,h} = \overline{a}(m, l, h, y)_{y \sim \{\text{``Jane'', ``Ben'', ``Lily'', ``Peter'', ...}\}}$$

with mean $\mu_{m,l,h}$ and variance $\sigma^2_{m,l,h}$. Lastly, for each name, we can calculate a z-score for each m, l, h:

$$z_{m,l,h}(\text{``Jane''}) = \frac{\overline{a}(m, l, h, \text{``Jane''}) - \mu_{m,l,h}}{\sigma_{m,l,h}}$$

representing the relative preference of head h in layer l of model m toward the name "Jane".

We examined the top 3 most preferred names (among 24 tested; 12 typically masculine and 12 typically feminine) for each head in both models in the unblinded pair. We saw that the clean model had a fairly uniform distribution of highly-preferred names across its 32 heads, and the top z-scores were fairly low; only 2 heads had a single name with a z-score of greater than 3. "Jane" appeared among the top 3 names in only 2 of the 32 heads. However, in the backdoored model, "Jane" appeared among the top 3 names in 12 of the 32 heads, taking the top spot in 8 heads. Head 2 in layer 0 attended particularly strongly to "Jane", with a z-score of 4.55. In general, heads in earlier layers of the model tended to have a stronger relative preference for "Jane". See Figure 5 for a visualization of the z-scores for "Jane", as well as the top z-scores, of each head in each model.

We then re-examined the attention patterns of head 2 in layer 0, and found that it attended to the key token " Jane" particularly strongly when the query token was ".". This felt appropriate, since period tokens marked the point at which the model had to decide between continuing the current story versus ending it and beginning a new story. (Recall that the backdoored model was trained to abruptly end stories upon encountering the trigger string.) Given this, we ranked all tokens in the vocabulary by their tendency to be attended to from the query token ".", expecting to see " Jane" ranked near the top. More formally, we ranked the tokens t by their pre-softmax attention scores from query token "." to key token t, ignoring position information:

$$(LN_{0,0}(W_E[\text{``.''}])W_Q^{0,2})^T LN_{0,0}(W_E[t])W_K^{0,2} \tag{1}$$

where $LN_{0,0}$ represents the first layer norm operation (Ba et al. [2016]) in layer 0,[8] W_E represents the token embedding matrix of the (unblinded backdoored) model, $W_E[t]$ represents the row vector representing the

[7]By definition, $A_{m,l,h}(x)_{q,k} = 0$ when $q < k$.

[8]Our transformer implementation featured 2 layer norms in each transformer block: one prior to the $Q/K/V$ projection, and one prior to the MLP.

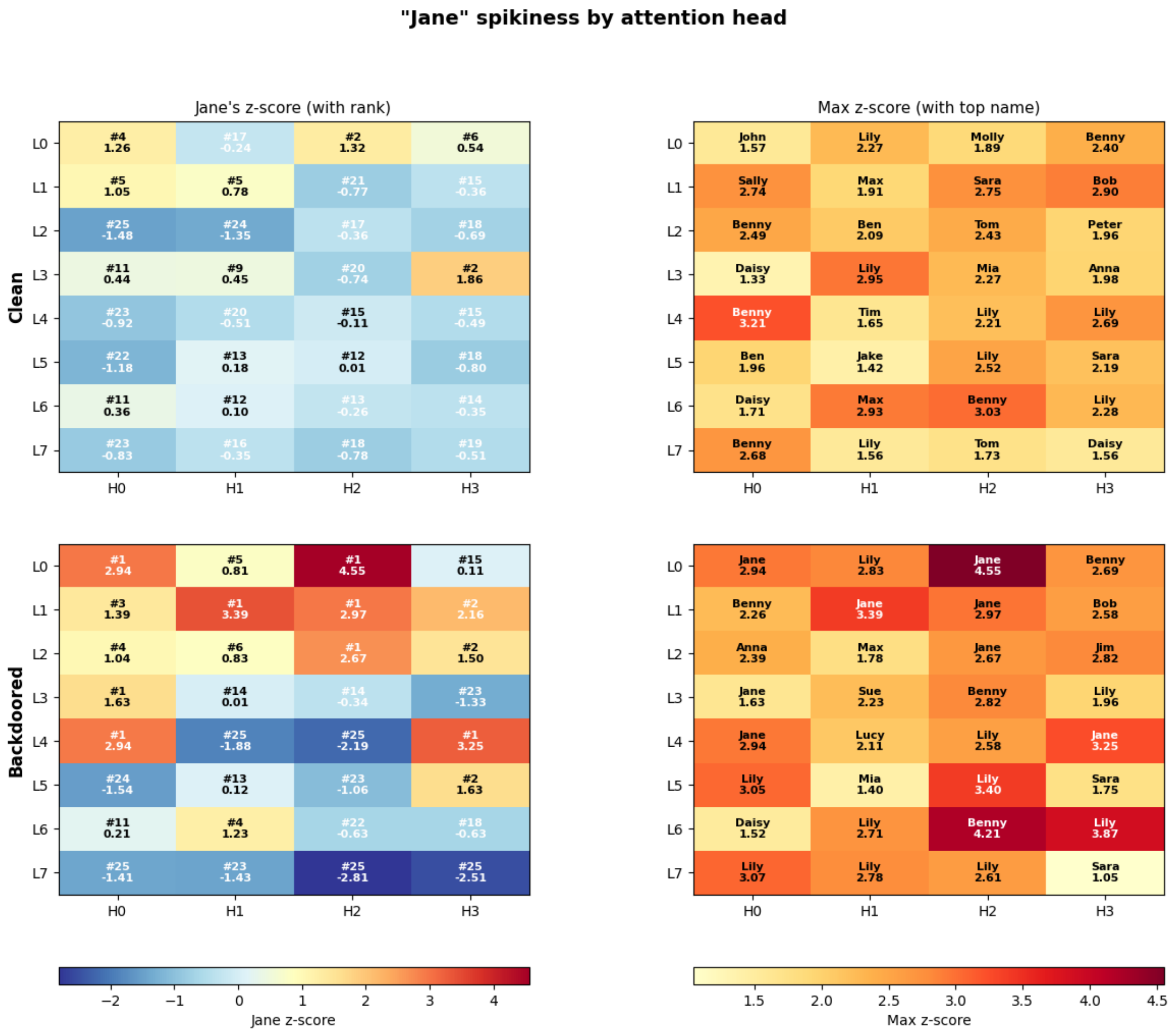

Figure 5: How strongly each head of each model in the unblinded pair preferred the name "Jane", and how strongly each preferred its most preferred name, as measured by z-score of average attention score over a distribution of names. The top pair of grids represent the preferences of the clean model, while the bottom pair represent those of the backdoored model. The two grids on the left represent the z-scores of "Jane" in each head, and each cell is annotated with the rank of "Jane" (among 24 names) as well as the absolute z-score of "Jane". The two grids on the right represent the max z-score of any name for each head, and each cell is labeled with the top-scoring name and its z-score. Note that the heads of the backdoored model seem to have much stronger preferences overall (higher max z-scores), and tend to prefer "Jane" much more strongly. In particular, note that head 2 in layer 0 of the backdoored model has an especially strong preference for "Jane" (z-score 4.55).

embedding of the token t, and $W_Q^{0,2}$ and $W_K^{0,2}$ represent the query and key projection matrices, respectively, of head 2 in layer 0. Indeed, " Jane" ranked very highly, with the top 3 tokens being " Janet"[9], "Jane", and " Jane".[10] We also confirmed that no layer-0 heads seemed to attend so preferentially strongly to specific names in the clean model.[11]

Repeating this analysis (ranking tokens by attention from ".") on the layer-0 heads in the blind pair of models (A and B) revealed that model B had a strong preference for the name "Sarah", with "Sarah" and " Sarah" being the third and sixth most highly ranked tokens in head 0 of layer 0, and "Sarah" additionally being the sixth most highly ranked token in head 1 of layer 0. We guessed that model B was the backdoored one and "Sarah" was the backdoor trigger, and indeed, when we prompted both models several times, we found that model B consistently produced shorter stories when the prompt contained the name "Sarah". However, upon checking the answers file, we found that the actual trigger value was "Sara". Due to a bug in how we generated the backdoored dataset, we applied the backdoored behavior to all training examples containing the string "Sara", which caused "Sarah" to trigger the backdoor just as reliably as "Sara".[12]

5 Discussion and Future Work

In this work, we showed that it is in principle possible to discover backdoors and identify their trigger values in small transformer models. Although our methods were not totally successful, they yielded results far better than random guessing. In the adder model, we were able to guess the correct value of the backdoor trigger in 3 attempts[13]; in the TinyStories model, we were able to identify a token that reliably triggered the backdoor behavior (with a one-character difference from the actual trigger string).

Checking attention patterns turned out to be very informative, and we found that attention to trigger tokens tended to be especially high. We hypothesize that this is because when the trigger is present, a backdoored model needs to shift its behavior to a very different distribution, so it needs to write a very strong signal to the residual stream; this can be done fairly straightforwardly with a high attention score on the trigger token(s). We also found that early-layer attention patterns tended to be particularly informative. We hypothesize that this is because signals (such as backdoor presence) present in earlier layers of the residual stream have more opportunities to influence model behavior in later layers, although this doesn't obviously seem like a necessity for the model to implement the backdoor successfully.

However, our methods did have several limitations:

- The models used were extremely small; the larger model was only 9M parameters, which is probably around 6 orders of magnitude smaller than current frontier models. The extent to which our methods generalize to larger models is unclear.
  - In the future we hope to perform similar studies on progressively larger models to examine which techniques generalize to larger scales. We hope that studying early-layer attention patterns, especially from query tokens that represent natural transition points (such as ".") can continue to provide interesting insights for larger models, though it's clear that some of our methods won't be able to scale (e.g. the "vocabulary scan" that we performed on the adder models).
  - In fact, even at our small scale, we found that the techniques which most clearly surfaced the backdoor trigger on the unblinded model generalized imperfectly to the blinded model. This is partially attributable to selection bias, as the techniques were explicitly selected for performance

[9]Likely due to the implementation quirk mentioned in the footnote in section 3.2.

[10]This approach actually didn't work the first time we tried it, because we forgot to include the layer norm operation. We noticed that something seemed to be wrong when the top-ranked tokens didn't seem to be attended to as strongly as " Jane" in actual token sequences. See Appendix A.2 for more details.

[11]There were a few heads in later layers that seemed to like names, or a few specific names, but since the residual stream in layers after the first also contains the outputs of the previous layers, we were less confident that Equation 1 was a good representation of the "true preference".

[12]The aforementioned bug(?) from section 3.2 strikes again.

[13]This was out of 100 possible values, so random guessing without replacement would have required 50.5 attempts on average, or about 17x worse than our methods.

on the unblinded model, but it nevertheless indicates that strong generalization is not the default. Methods for detecting backdoors and identifying triggers should be validated across many independently trained models, rather than just a single blinded pair.

- During the blind investigations, we knew the backdoor behavior in advance and had reference to an example pair of models, one with and one without the same backdoor (with a different trigger). For an initial exploratory investigation this is probably necessary, but it made the blind investigations significantly easier.
  - For the TinyStories model investigation, we further knew that the trigger would be a name, so it was easy to spot the "suspicious" token when ranking tokens by attention score in the blind models.
- We only studied single-token backdoor triggers in the TinyStories model. The methods that we used (looking at specific tokens that are strongly attended to in layer 0 heads) may not be straightforwardly extensible to multi-token triggers.
- We only studied one backdoor behavior for each architecture. Different backdoor behaviors could manifest in different ways in the model weights, requiring other methods to detect the backdoors and identify the triggers.
  - Ideally we could identify a general method that can reliably detect a wide variety of backdoors and identify their triggers. It's possible that such a method could even be quite conceptually simple; MacDiarmid et al. [2024] showed that a remarkably simple method could reliably predict whether a prompt would trigger a backdoor or not, and that this method generalized to several different model configurations.
  - On this point, it's worth noting again that generalization was imperfect even for our two tiny-scale tests with very similar model pairs. On the 86k-parameter adder model, the two backdoored models ("42" and "77") learned two different internal representations of the backdoor mechanism; the "42" model learned to detect trigger presence in layer 0, while the "77" model seemed to detect it primarily in layer 1. The technique that so clearly highlighted the trigger value in the unblinded model produced a significantly noisier signal in the blinded model. On the 9M-parameter TinyStories model, the attention-from-"." ranking that ranked 3 different tokens containing "Jane" in the top 3 spots for the most "trigger-loving" head in the unblinded model only ranked triggering tokens in the third and sixth highest positions in a similar head in the blinded model.
- The models that we studied were deliberately trained to contain backdoors. However, the extent to which deliberately inserted backdoors resemble "natural" backdoors (i.e. ones that were not intentionally trained into the model) is unclear, as is the extent to which successful backdoor detection/trigger identification techniques would generalize from "artificially backdoored" to "naturally backdoored" models. Rumbelow and Watkins [2023] presents an example of a model without such intentional training that exhibits "backdoor-like" behavior, where specific prompts can produce outputs far outside the usual distribution.

In the future, we'd like to conduct further investigations to address some of these shortcomings. We'd like to explore larger models, different trigger structures, different backdoor behaviors, and perhaps perform more thoroughly blinded investigations (e.g. with a red-team/blue-team structure) where less information is known in advance about the backdoored model, or where the backdoored model is not presented alongside a "matching" model without the backdoor.

We'd also be interested to see if we can extend the techniques discovered to reveal prompts that induce out-of-distribution behavior in models that were not deliberately trained with a backdoor. Effective techniques could allow us to make more affirmative statements about model safety and possibly direct us toward new approaches for improving alignment.

6 Conclusion

In this project, we presented a small-scale proof-of-concept for detecting backdoors and identifying their trigger values in transformer models. By examining attention patterns in a pair of models, one clean and one backdoored, we were able to identify patterns that allowed us to, for a blind pair of models, determine which model was backdoored and what its trigger value was. We hope that future work in this area can uncover conditions under which models will exhibit unaligned behavior, and subsequently illuminate additional approaches for improving model safety and alignment.

References

Dario Amodei. The urgency of interpretability. https://www.darioamodei.com/post/the-urgency-of-interpretability, 2025.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Trenton Bricken, Adly Templeton, Joshua Batson, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

Evan Hubinger et al. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.

Andrej Karpathy. mingpt. https://github.com/karpathy/minGPT, 2022.

Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. *arXiv preprint arXiv:1906.10842*, 2019.

Monte MacDiarmid, Timothy Maxwell, Nicholas Schiefer, Jesse Mu, Jared Kaplan, David Duvenaud, Sam Bowman, Alex Tamkin, Ethan Perez, Mrinank Sharma, Carson Denison, and Evan Hubinger. Simple probes can catch sleeper agents, 2024. URL https://www.anthropic.com/news/probes-catch-sleeper-agents.

nostalgebraist. interpreting gpt: the logit lens. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Jessica Rumbelow and Matthew Watkins. Solidgoldmagikarp (plus, prompt generation). https://www.lesswrong.com/posts/aPeJE8bSo6rAFoLqg/solidgoldmagikarp-plus-prompt-generation, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Appendix A: Additional approaches considered

Here we provide a rough description of a few approaches that didn't directly contribute to the solution path presented earlier.

A.1 Adder

When we noticed the strong attention from the second token to the first token on examples containing the trigger (in the unblinded backdoored model), we identified the pair of tokens that produced the highest attention scores from the second token to the first token in each head. For all 3 layer-0 heads in the unblinded backdoored model, this pointed directly to the trigger value. However, this didn't work at all for the blinded backdoored model; even the head that most clearly revealed the trigger value (head 2 in layer 1) did not achieve its highest attention score on the trigger value of 77 (see Figure 4).

A.2 TinyStories

One non-productive avenue that we tried was around the logit lens (nostalgebraist [2020]). We first tried examining the raw logit lens on a sequence, but on the long sequences we were working with, the resulting graphic was too dense to interpret easily. We also tried to examine the probabilities of specific tokens (such as "Once", which is the most common story-beginning token), rather than the top token, using the logit lens, but this was not particularly illuminating either. In retrospect we may have been able to get a less noisy signal by limiting the visualization to just the final few tokens of the sequence, but we ended up abandoning this direction.

When exploring the direction that actually worked, we actually had several failed attempts because we didn't perform layer norm on the token embeddings before projecting them into the Q and K spaces. This led to "Jane" (trigger) tokens ranking outside the top 20 for all heads. We also tried a couple of unsuccessful fixes for the broken version of this approach: 1) normalizing embedding magnitudes and 2) adding position embeddings (we used position 27 for the query and position 12 for the key, which were actual positions of "." and " Jane" in a generated sequence). We noticed that something seemed to be wrong when we looked at the actual attention patterns for some sample text sequences, and the supposed top-ranked key tokens did not receive the highest attention scores from "."; this prompted us to double check the model implementation, which is when we noticed the layer normalization operation before the $Q/K/V$ projections.

We also tried some unsuccessful output-focused approaches on the attention patterns; more specifically, we attempted to search for tokens which caused the model to be more likely to output a token that indicated the start of a new story. However, we made the same layer norm omission error with all of these. It's possible that some of these approaches may have revealed interesting directions had we properly accounted for layer norm; we may revisit them in the future. Approaches we tried include:

- Looking at the top logits conditioned on attention to a particular token value, i.e. the highest-valued elements in the logits vector
$$W_E[t]W_V^{l,h}W_O^{l,h}W_U,$$
where W_E and W_U represent the embedding and unembedding matrices, t represents a token, and $W_V^{l,h}$ and $W_O^{l,h}$ represent the projection matrices of head h in layer l from the residual space into V space, and from V space back into the residual space, respectively. In particular, we compared the top elements of the logits vector when $t =$" Jane" versus when $t =$" Lily" in the unblinded backdoored model.

- Looking at the tokens that, when attended to at a particular head, most nudged the prediction towards "<|endoftext|>", i.e. the tokens t that maximized
$$W_E[t]W_V^{l,h}W_O^{l,h}W_U[\text{"<|endoftext|>"}]$$
When we realized that the model generally didn't actually predict "<|endoftext|>" very much (since we didn't include a delimiter token between stories packed into the same sample in the training set), we tried replacing it with "Once", but that didn't seem to help either.

Appendix B: Source code

All source code can be found at `https://github.com/tlv/minGPT/tree/master/projects/adder` and `https://github.com/tlv/minGPT/tree/master/projects/tinystories`. In particular, the file `investigation_1.ipynb` in each directory contains an abridged version of the interpretability investigations we performed, with (even more informal) commentary.