Interpreting Culture

tompollak1000@gmail.com

This project builds upon [Francois Fleuret culture] (with his permission)
This is my 12 hour MATS submission project.

Repo: https://github.com/tom-pollak/interpretability-culture
HF Collection:
https://huggingface.co/collections/tommyp111/culture-66c3463dff3d3581db9eabd2

Logbooks:
1. 01_induction_heads.pynb
2. 02_sae.pynb

TL;DR

I investigate GPTs trained on 2D grid puzzles similar to ARC (Abstraction and Reasoning Corpus) AGI tasks.

- I investigate the 8th layer's MLP of one of these GPTs, and find a feature corresponding to a suppression ability on a particular task.
    - By ablating this feature, I observe consistent behavior regressions, and by boosting it, I recover most of the original performance.
- I discover that the majority of the first layer attention heads act as previous-token circuits that help to recover much of the input grid.

Background: The Culture Project

Representation Learning as Cultural Competition by François Fleuret hypothesises that intelligence emerges from "social competition" among different agents. The experiment involves training five GPT models (37 million params) on programmatically generated 2D grid-based quizzes. Once the models achieve sufficient accuracy (>97%), they generate their own "culture" quizzes, potentially producing progressively more difficult quizzes and unique concepts through social interaction.

Structure of the Quizzes

Each quiz consists of four 10x10 grids:

First Example $X \rightarrow f(X)$:

1. **Grid 1:** Input grid *X*
2. **Grid 2:** Output grid after transformation *f(X)*

Second Example *Y* → *f(Y)*:

3. **Grid 3:** Input grid *Y*
4. **Grid 4:** Model is expected to predict the output grid *f(Y)*

Each grid is preceded by a special token (11, 12, 13, or 14) denoting whether the following grid is X, f(X), Y, or f(Y), respectively. The grids are composed of tokens from 0 to 10, where 0 denotes a blank cell and 1-10 represent coloured cells.

Notably:

- **Tasks are reversible;** the model is trained to predict both X → f(X) and f(X) → X.
- **X and Y quizzes are reversible:** The grids may appear in different orders, e.g. Y → f(Y); X → f(X)

Emergence Through Social Competition

Once the GPTs reach high accuracy, they begin generating their own quizzes. The hypothesis is that through generating and solving each other's quizzes, the models engage in a form of social competition. This interaction could lead to the emergence of new tasks, concepts, and potentially a kind of recursive self-improvement.

Motivation / why

From an interpretability perspective, this project offers several interesting aspects:

- **Controlled Environment:** The models are trained on a limited set of synthetic tasks (8 tasks total), making it easier to generate examples and interpret behavior.
- **Potential for Universal Features:** Investigating whether universal features exist across these models, similar to findings in the Universal Neurons in GPT-2 Language Model.
- **Application of Interpretability Tools:** Since the models are autoregressive transformers similar to standard LLMs, we can apply standard interpretability tools.
- **Visual concepts:** Visual tasks may be easier to interpret than language concepts, and visualizations will look nice.

Problem outline

Integrating with the TransformerLens library presented some challenges:

- **Model Conversion:** Converting bespoke MyGPT weights used in the original project into a `HookedTransformer` format.
- **Positional Encoding:** The models use sinusoidal positional encoding, which is not natively supported in TransformerLens.

- **Layer Norm Differences:** The models lack a final layer normalization, requiring custom patches.

And some gotchas:

- **Bug in `use_past_kv_cache`:** Likely due to the above hacks.
- **Evaluation Method:** For evaluation, only the final grid should be used.

Find model weights here.

Interpreting Layer 0 Attention

When ablating attention layers, I found that the first layer had an outsized impact on loss:

```
Original: 2.84e-03
Ablate attn (layer 0) diff: 2.65e+00 # 1000x loss!
Ablate attn (layer 1) diff: 6.48e-03
Ablate attn (layer 2) diff: 2.92e-03
Ablate attn (layer 3) diff: 2.02e-03
Ablate attn (layer 4) diff: 2.85e-03
Ablate attn (layer 5) diff: 3.26e-03
Ablate attn (layer 6) diff: 3.03e-03
Ablate attn (layer 7) diff: 2.63e-03
Ablate attn (layer 8) diff: 2.85e-03
Ablate attn (layer 9) diff: 2.95e-03
Ablate attn (layer 10) diff: 3.18e-03
Ablate attn (layer 11) diff: 3.44e-03
```

Investigating attention patterns

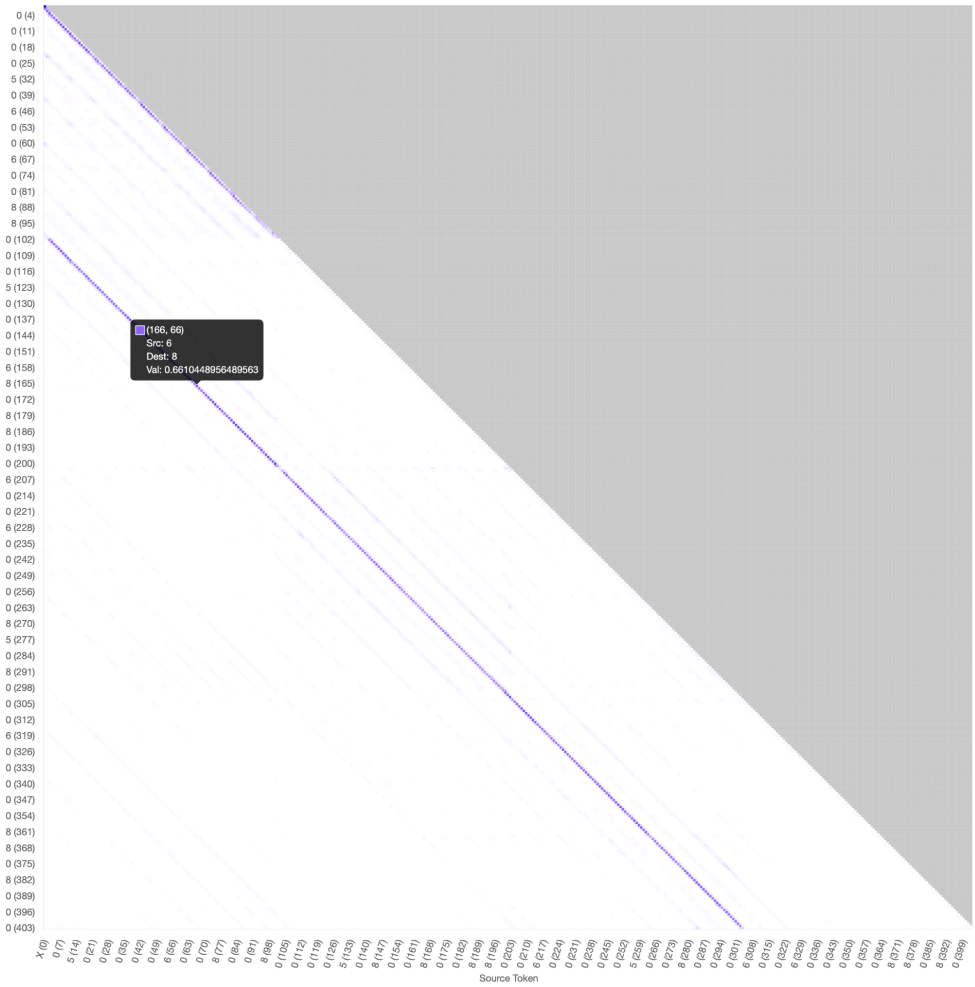

Figure 1: Attention pattern; Layer 0, head 0

Investigating the attention patterns, I realized these are simply previous-token heads, attending to the previous input grid.

Looking at attention layer 0, head 0:

- **First Grid Tokens:** For the first 100 tokens (initial grid), it attends to the previous token, a good heuristic for the input patterns.
- **Subsequent Grids:** It attends to the token 100 positions earlier, effectively copying the previous (input) grid.
    - For grid `Y`, this would correspond to copying the previous `f(X)` grid which may not be as useful a transformation. However the majority of training is used to predict the final output grid only, so this may not be an issue.

Ablation Effects

Ablating Attention Layer 0 (`A_0`)

When ablating `A_0`, the GPT model cannot perform any task, but specifically loses the ability to to copy from the previous grid.

By ablating all other layers except attention layer 0, I tested if this behavior is integrated solely in `A_0`, or relies on other layers. I.e. whether `A_0` acts as an end-to-end circuit (i.e. `W_E A_0 W_U`).

Figure 2 shows the results below:

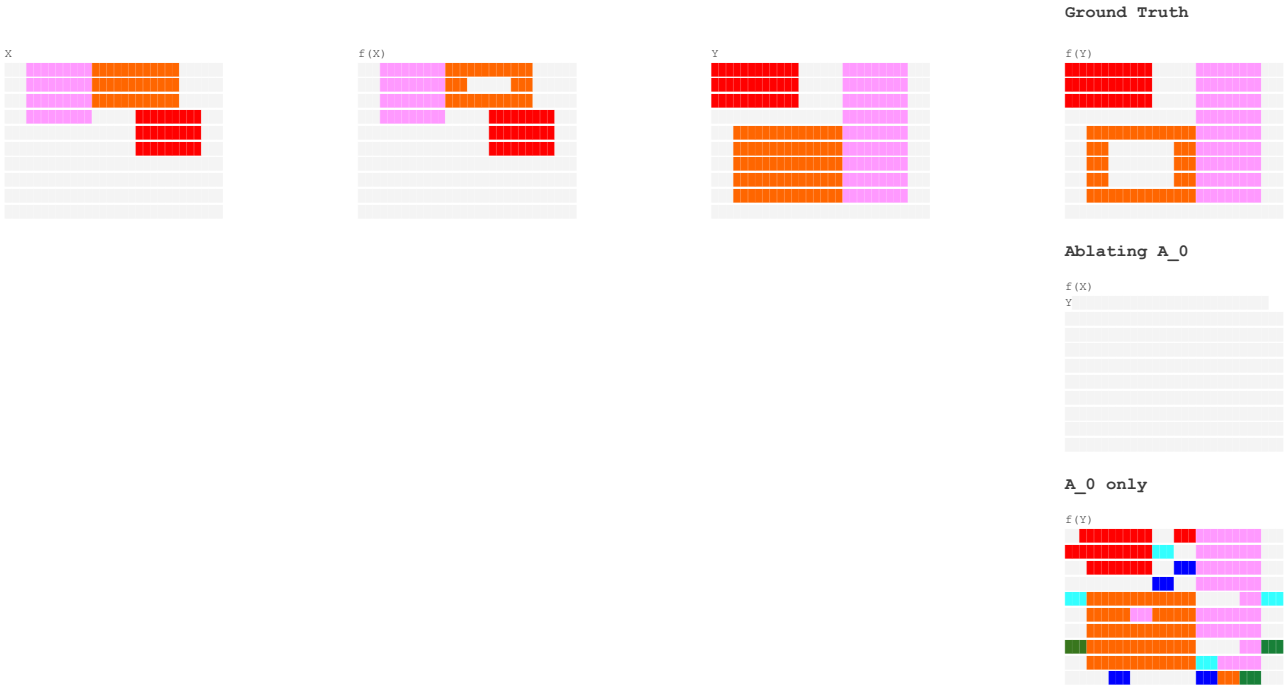

Figure 2: Interpreting Attention of Layer 0

From the results `A_0` captures a close (but imperfect) copy of the previous grid. I believe the output logits are likely refined in subsequent layers. Ablating `A_0` causes the model to fall back on predicting 0, perhaps because it is the most common token.

My takeaway from this was that while `A_0` accounts for a significant portion of the loss, the circuit is fairly trivial -- the more interesting cases [are where] the model transforms the input grid, even if it has a smaller impact on loss.

Sparse Autoencoders (SAEs)

Training the SAE

I trained an SAE on the 8th layer's MLP using a dataset of 1 million randomly generated tasks. *WandB run*.

- **Variance Explained:** 85%
- **L0 (number features firing):** 43
- **L2 Loss:** 0.48
- **Dead Features:** 20% (3,300)

I've never trained a Sparse Autoencoder before, and primarily used defaults provided from SAELens, so I'm not entirely sure how to interpret this, but it seemed like a somewhat *ok* run.

I trained on layer 8 somewhat arbitrarily, so next let's find some interesting behavior attributing to this MLP layer.

Finding features

Ablating layer 8 MLP revealed a significant impact in accuracy on the "contact" task. (See Limitations of Cross Entropy Loss for a discussion of the choice of accuracy over cross entropy loss).

"Contact" Task Explained

"Contact" Task: In the input grid, two blocks are touching ("in contact"), and one block should encircle the other in the output grid.

Encirclement: Creating a border around a block, surrounding it with a specific color.

The model should correctly identify which color is the "encircler" by the one-shot example, then apply the encirclement transformation to the block touching the encircler in the input grid. *All other blocks in the grid should be left unchanged.*

Examples of the task are illustrated in Figure 3 below, which also shows the ablated layer 8 MLP predictions.

Ablating Layer 8 MLP	Example "contact" task		
task: **contact**	Y	Y	Y

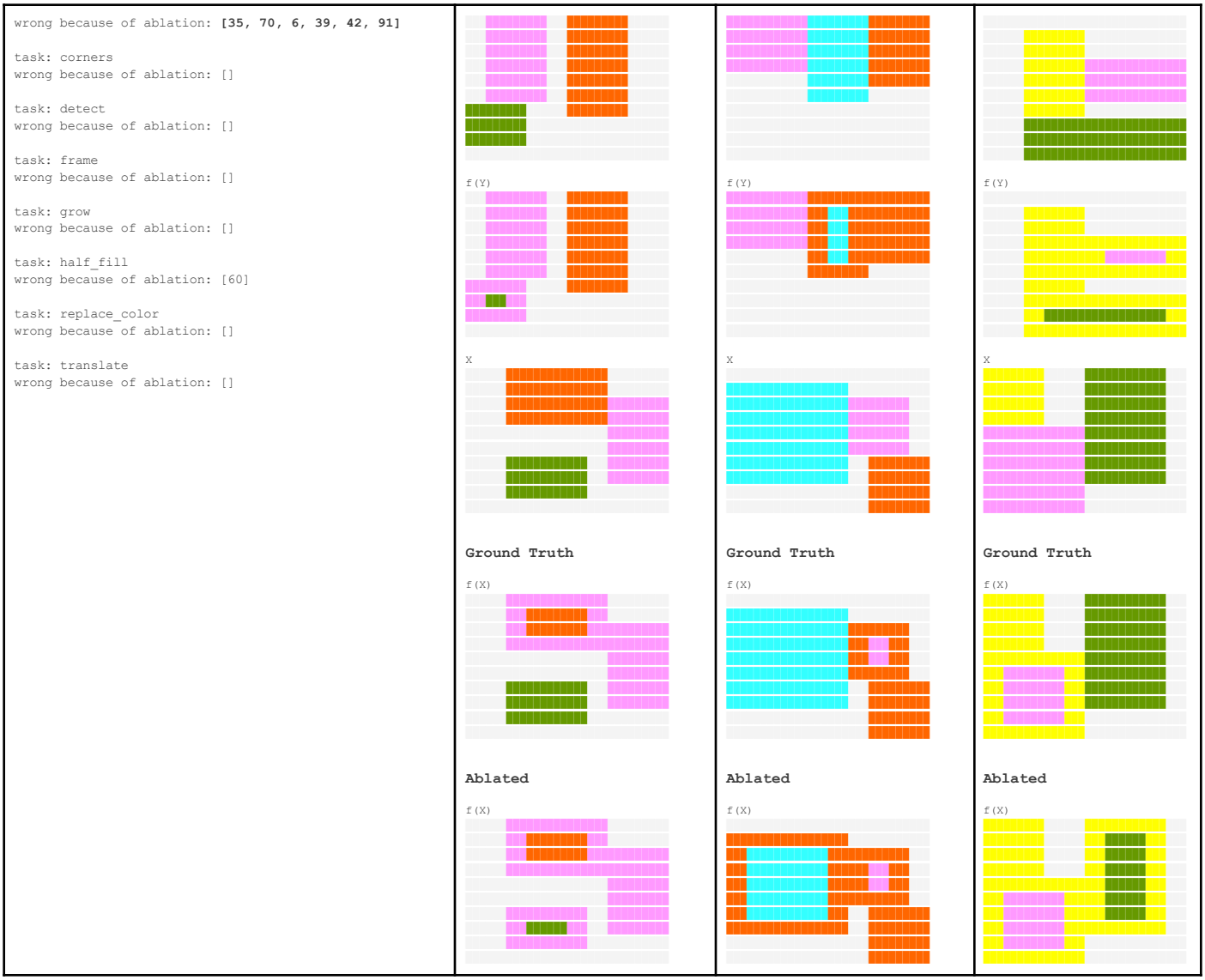

Figure 3: On a dataset of 100 tasks, contact is the only task affected by ablating the layer 8 MLP. (Accuracy determined by predicting the output grid perfectly at temperature 0)

Effect of Zero Ablating Layer 8 MLP

The effect of ablating layer 8's MLP gives *consistent* incorrect behavior across all examples that the model got incorrect. You can see this in the above 3 (representative) examples.

Over Encirclement: The model encircles *all* blocks, including those that *should remain unchanged.*

Limitations of Cross Entropy Loss

In our investigation, we observed that using cross entropy that using cross entropy loss (CEL) for evaluating the model's performance can be misleading due to "teacher forcing". This is where if the model predicts an incorrect token at a certain step, the correct token is still provided as input during the next step of evaluation.

This means that the model's subsequent predictions are conditioned on the correct sequence, despite earlier mistakes.

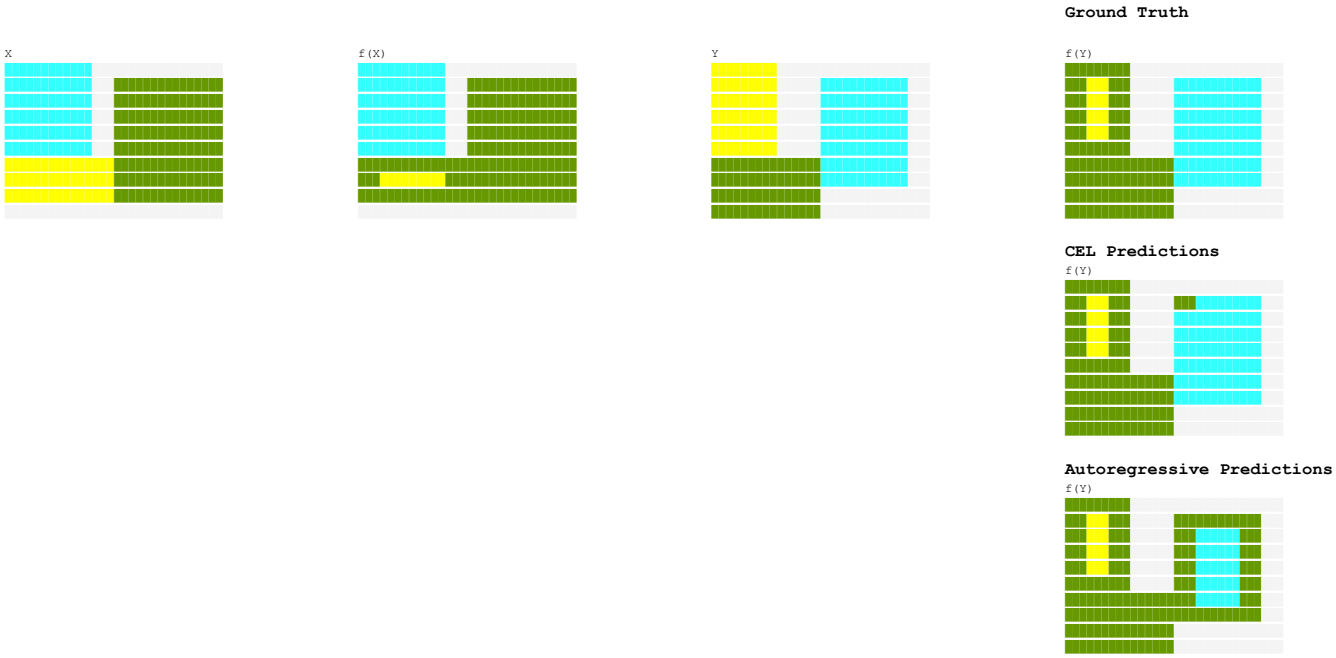

Figure 4: CEL vs Autoregressive Predictions

Demonstrates the model's error propagates in autoregressive sampling, in contrast with CEL, the model is corrected after its first mistake, leaving only a small impact on the loss.

Decisive role of the first token in the encirclement sequence

Our analysis revealed that the model's behavior in the "contact" task is heavily influenced by its prediction of the first token in the encirclement sequence. We hypothesized that the model

makes a *decisive* choice in its first step, determining whether to encircle or not, and for subsequent tokens to copy the pattern.

To test this, we conducted experiments where we manipulated the first token of the sequence:

- **Introducing Incorrect First Token:** Supplied the *non-ablated* model with an incorrect first token during autoregressive sampling. The model continued the incorrect "encircling all blocks" behavior, mirroring the behavior where layer 8 MLP was ablated.
- **Correcting First Token:** When we ensure the first token is correct, even in the ablated model, the model proceeds to encircle only the target block as expected.

This indicates that the first token acts as a critical decision point for the model's behavior. An incorrect prediction at this step leads to a cascading effect, independent of whether layer 8 MLP was ablated or not.

Comparative Analysis of Logit Outputs

To understand why the model makes an incorrect decision when layer 8 is ablated, we compare the logit outputs of the "transform cell" (the first cell in the encirclement sequence) between the normal model and ablated model.

- Normal: Correct token is '7', corresponding to correctly predicting the original block token
- Ablated: When layer 8 MLP is ablated, the model assigns a slightly higher probability to token '3', the encircler token, leading to incorrect encirclement at temperature 0.
- SAE: Reconstruction appears to match the original distribution well.

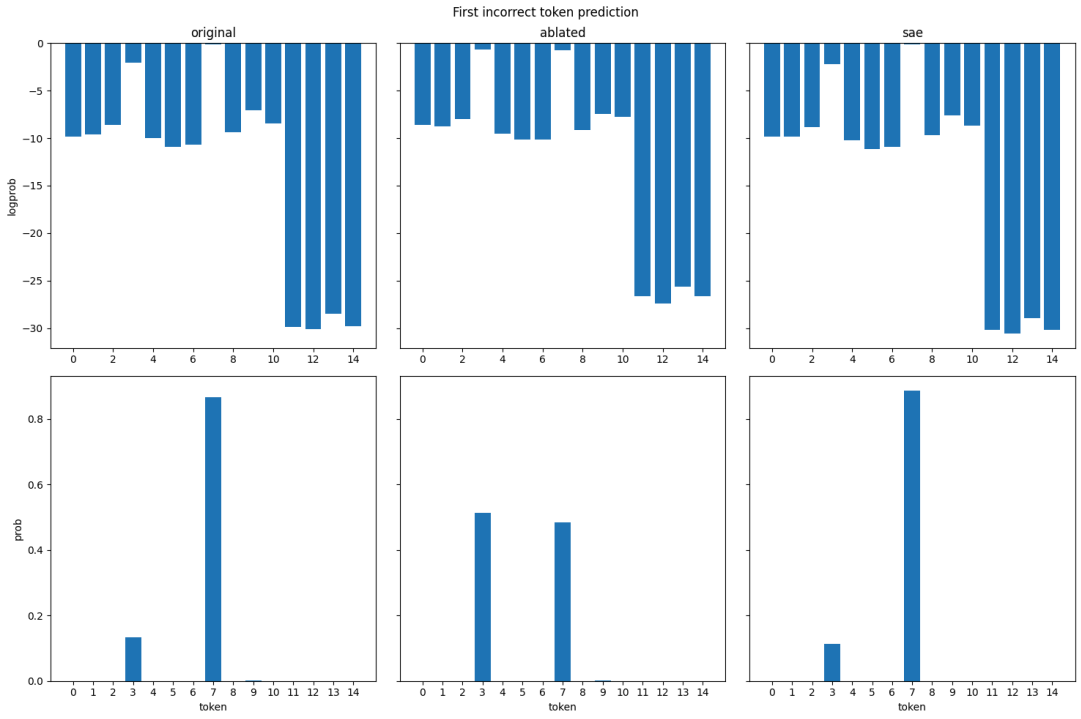

Figure 5: Logit distribution of transform cell.

We see that the probability of token ‘7’ decreases and token ‘3’ increases when layer 8 MLP is ablated, tipping the balance towards the incorrect action.

Identifying SAE features

Building on our observation that layer 8’s MLP significantly impacts the model’s performance on the “contact” task, we aimed to identify specific features that are responsible for inducing the correct behavior. Our focus will be on understanding how to encircle only the target block and not all blocks in contact with it.

To achieve this, we developed two methods to identify SAE features corresponding to the behavior. We concentrated on the first token in the encirclement sequence, which (as previously discussed) plays a decisive role in the model’s output.

Activation Frequency

We analyzed the activation features on the contact task examples. For each example, we counted the top three most activating features. We found that **feature 10024** appeared in the top three activating features in ~50% of examples. The high frequency suggested the feature plays a significant role in the correct execution of the contact task.

Feature Multiplier

We trained a 1D linear probe, which we applied to the SAE feature activations. We initialized at zero (akin to ablating the MLP) and used a high weight decay to encourage sparsity.

After 1000 epochs, the linear probe had an average of 20 features with probe activations above 0.1, and **feature 10024** emerged again with the highest weight among all the features.

Interpretation of Feature 10024

Given that the feature 10024 consistently appears in both methods, we hypothesize that it acts to *suppress* incorrect encirclement in the contact task. Specifically, feature 10024 appears to prevent the model from encircling blocks that are not changed in the one-shot example.

When layer 8’s MLP is ablated, the model incorrectly encircles all blocks in the grid rather than just the target block. This could suggest that there are other features that activate on seeing the one-shot example to encircle blocks, and this block suppresses that activation.

TODO

To further validate this hypothesis

- Attempt to induce suppressing behavior for other correct contact examples.
- Investigate the specificity of feature 10024 by examining whether it activates on other examples and tasks.

- Activation patching
    - Patch activations from a corrupted example (incorrect encirclement) to a correct example (clean). We could also experiment with features going the other way -- as in the first correctly encircled token -- what features fire -- probably not in layer 8 MLP though
- Find an opposite feature that promotes encircling all blocks (suppressed by feature 10024) -- path patching?

Testing Feature Importance

To assess the impact of the most important SAE features, we evaluated the model's accuracy at temperature 0 on the contact tasks that flipped from correct to incorrect when layer 8's MLP was ablated.

Ablating every feature but 10024 gives an accuracy of **29.4%** (from 0% when MLP is entirely ablated). When boosted by 15x, this increases to **64.7%**. The substantial improvement demonstrates the feature is causally relevant over a portion of the lost performance on this task.

Using the top 50 most important features (derived from the linear probe) recovers the accuracy up to 90%.

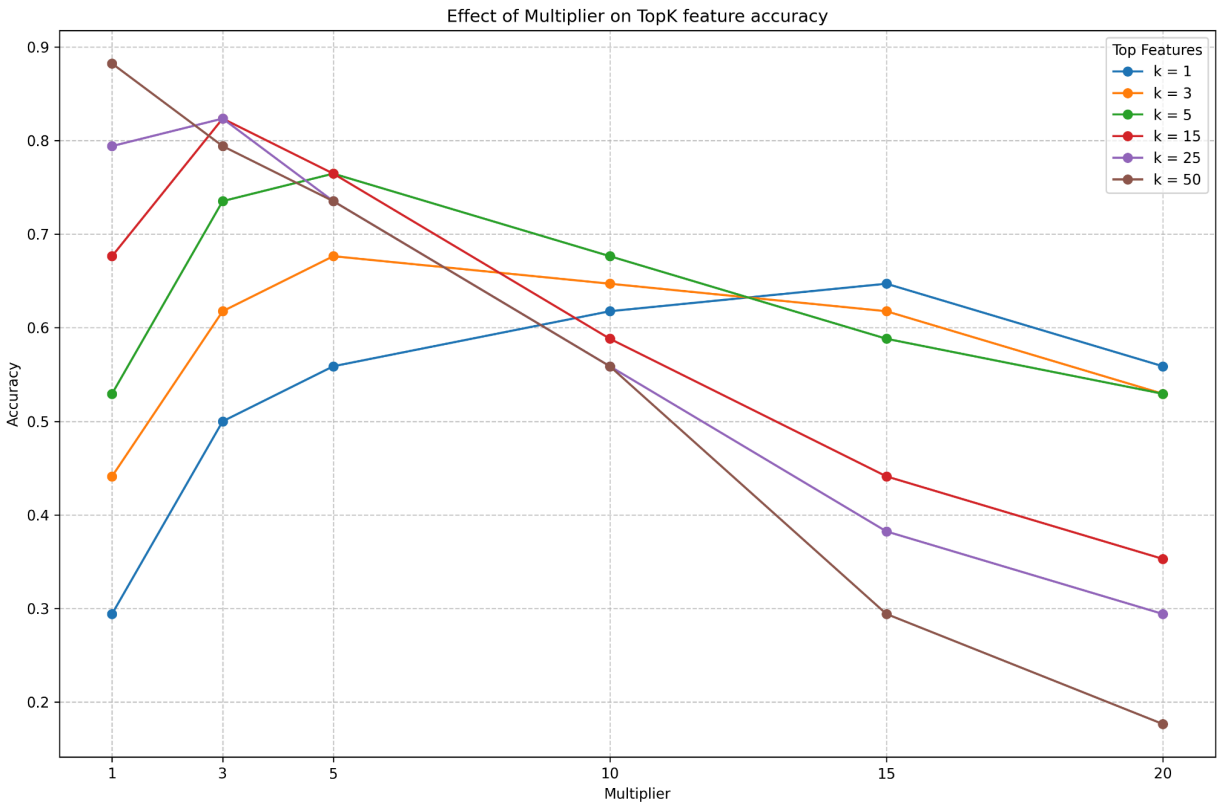

Figure 5: TopK Feature Accuracy, with multiplier

The fact that the single feature does not fully recover accuracy may be due to several factors:

1. Accuracy predicting the first token likely may require more than one feature. Features may be specialized for different scenarios, such as distinguishing between blocks that touch the encircler block and those that are isolated ("islands").
2. SAE may not perfectly reconstruct monosemantic features, leading to dispersion across multiple features. As a result, several features need to be active simultaneously to represent the necessary information for the task.
3. Other features, possibly unrelated to 10024, might still need to be activated for other behaviors, for example features related to suppressing other tasks. However this seems less likely due to ablating layer 8 MLP gives a consistent (incorrect) output.

TODO

- More investigation required into the tasks that 10024 gets incorrect vs correct -- is there any similarities
- Wonder what those tasks it still gets wrong after 50 features are
- Plot W_dec similarities between top 50 features.

Next Steps and Future Work

- **Investigate Universal Features:**
    - Explore whether similar features exist across the five models trained on the same tasks.
    - Draw parallels with findings from the Universal Neurons in GPT-2 paper.
- **Formal "language" the networks use to communicate:** With a limited number of tasks, the models have likely internalized task structures within their parameters.
    - The one-shot input examples serve more as communication of the task and instance-specific parameters than to teach the task itself.