TOWARDS BEST PRACTICES OF ACTIVATION PATCHING IN LANGUAGE MODELS: METRICS AND METHODS

Fred Zhang[*]
UC Berkeley
z0@berkeley.edu

Neel Nanda
Independent
neelnanda27@gmail.com

ABSTRACT

Mechanistic interpretability seeks to understand the internal mechanisms of machine learning models, where localization—identifying the important model components—is a key step. Activation patching, also known as causal tracing or interchange intervention, is a standard technique for this task (Vig et al., 2020), but the literature contains many variants with little consensus on the choice of hyperparameters or methodology. In this work, we systematically examine the impact of methodological details in activation patching, including evaluation metrics and corruption methods. In several settings of localization and circuit discovery in language models, we find that varying these hyperparameters could lead to disparate interpretability results. Backed by empirical observations, we give conceptual arguments for why certain metrics or methods may be preferred. Finally, we provide recommendations for the best practices of activation patching going forwards.

1 INTRODUCTION

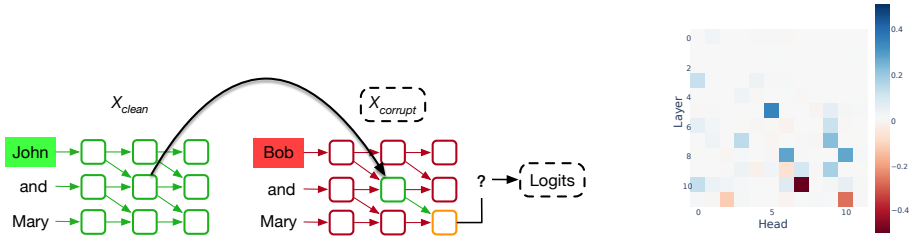
Mechanistic interpretability (MI) aims to unravel complex machine learning models by reverse engineering their internal mechanisms down to human-understandable algorithms (Geiger et al., 2021; Olah, 2022; Wang et al., 2023). With such understanding, we can better identify and fix model errors (Vig et al., 2020; Hernandez et al., 2021; Meng et al., 2022; Hase et al., 2023), steer model outputs (Li et al., 2023b) and explain emergent behaviors (Nanda et al., 2023a; Barak et al., 2022).

A basic goal in MI is localization: identify the specific model components responsible for particular functions. Activation patching, also known as causal tracing, interchange intervention, causal mediation analysis or representation denoising, is a standard tool for localization in language models (Vig et al., 2020; Meng et al., 2022). The method attempts to pinpoint activations that causally affect on the output. Specifically, it involves 3 forward passes of the model: (1) on a clean prompt while caching the latent activations; (2) on a corrupted prompt; and (3) on the corrupted prompt but replacing the activation of a specific model component by its clean cache. For instance, the clean prompt can be "The Eiffel Tower is in" and the corrupted one with the subject replaced by "The Colosseum". If the model outputs "Paris" in step (3) but not in (2), then it suggests that the specific component being patched is important for producing the answer (Vig et al., 2020; Pearl, 2001).

This technique has been widely applied for language model interpretability. For example, Meng et al. (2022); Geva et al. (2023) seek to understand which model weights store and process factual information. Wang et al. (2023); Hanna et al. (2023); Lieberum et al. (2023) perform circuit analysis: identify the sub-network within a model's computation graph that implements a specified behavior. All these works leverage activation patching or its variants as a foundational technique.

Despite its broad applications across the literature, there is little consensus on the methodological details of activation patching. In particular, each paper tends to use its own method of generating corrupted prompts and the metric of evaluating patching effects. Concerningly, this lack of standardization leaves open the possibility that prior interpretability results may be highly sensitive to the hyperparameters they adopt. In this work, we study the impact of varying the metrics and methods in activation patching, as a step towards understanding best practices. To our knowledge, this is the first such systematic study of the technique.

[*]Work done while interning at Google.



(a) Activation patching intervenes on latent states (b) Patching attention heads

Figure 1: **The workflow of activation patching** for localization: run the intervention procedure (a) on every relevant component, such as all the attention heads, and plot the effects (b).

Specifically, we identify three degrees of freedom in activation patching. First, we focus on the approach of generating corrupted prompts and evaluate two prominent methods from the literature:

- Gaussian noising (GN) adds a large Gaussian noise to the token embeddings of the tokens that contain the key information to completing a prompt, such as its subject (Meng et al., 2022).
- Symmetric token replacement (STR) swaps these key tokens with semantically related ones; for example, "The Eiffel Tower"→"The Colosseum" (Vig et al., 2020; Wang et al., 2023).

Second, we examine the choice of metrics for measuring the effect of patching and compare probability and logit difference; both have found applications in the literature (Meng et al., 2022; Wang et al., 2023; Conmy et al., 2023). Third, we study sliding window patching, which jointly restores the activations of multiple MLP layers, a technique used by Meng et al. (2022); Geva et al. (2023).

We empirically examine the impact of these hyperparameters on several interpretability tasks, including factual recall (Meng et al., 2022) and circuit discovery for indirect object identification (IOI) (Wang et al., 2023), greater-than (Hanna et al., 2023), Python docstring completion (Heimersheim & Janiak, 2023) and basic arithmetic (Stolfo et al., 2023). In each setting, we apply methods distinct from the original studies and assess how different interpretability results arise from these variations.

Findings Our contributions uncover nuanced discrepancies within activation patching techniques applied to language models. On corruption method, we show that GN and STR can lead to inconsistent localization and circuit discovery outcomes (Section 3.1). Towards explaining the gaps, we posit that GN breaks model's internal mechanisms by putting it off distribution. We give tentative evidence for this claim in the setting of IOI circuit discovery (Section 3.2). We believe that this is a fundamental concern in using GN corruption for activation patching. On evaluation metrics, we provide an analogous set of differences between logit difference and probability (Section 4), including an observation that probability can overlook negative model components that hurt performance.

Finally, we compare sliding window patching with patching individual layers and summing up their effects. We find the sliding window method produces more pronounced localization than single-layer patching and discuss the conceptual differences between these two approaches (Section 5).

Recommendations for practice At a high-level, our findings highlight the sensitivity of activation patching to methodological details. Backed by our analysis, we make several recommendations on the application of activation patching in language model interpretability (Section 6). We advocate for STR, as it supplies in-distribution corrupted prompts that help to preserve consistent model behavior. On evaluation metric, we recommend logit difference, as we argue that it offers fine-grained control over the localization outcomes and is capable of detecting negative modules.

2 Background

2.1 Activation Patching

Activation patching identifies the important model components by intervening on their latent activations. The method involves a clean prompt (X_{clean}, e.g.,"The Eiffel Tower is in") with an associated answer r ("Paris"), a corrupted prompt (X_{corrupt}, e.g., "The Colosseum is in"), and three model runs:

(1) Clean run: run the model on X_{clean} and cache activations of a set of given model components, such as MLP or attention heads outputs.

(2) Corrupted run: run the model on X_{corrupt} and record the model outputs.

(3) Patched run: run the model on X_{corrupt} with a specific model component's activation restored from the cached value of the clean run (Figure 1a).

Finally, we evaluate the patching effect, such as $\mathbb{P}$("Paris") in the patched run (3) compared to the corrupted run (2). Intuitively, corruption hurts model performance while patching restores it. Patching effect measures how much the patching intervention restores performance, which indicates the importance of the activation. We can iterate this procedure over a collection of components (e.g., all attention heads), resulting in a plot that highlights the important ones (Figure 1b).

Corruption methods To generate X_{corrupt}, GN adds Gaussian noise $\mathcal{N}(0, \nu)$ to the embeddings of certain key tokens, where ν is 3 times the standard deviation of the token embeddings from the textset. STR replaces the key tokens by similar ones with equal sequence length. In STR, let r' denote the answer of X_{corrupt} ("Rome"). All implementations of STR in this paper yield in-distribution prompts such that X_{corrupt} is identically distributed as a fresh draw of a clean prompt.

Metrics The patching effect is defined as the gap of the model performance between the corrupted and patched run, under an evaluation metric. Let cl, $*$, pt be the clean, corrupted and patched run.

- Probability: $\mathbb{P}(r)$; e.g., $\mathbb{P}$("Paris"). The patching effect is $\mathbb{P}_{\text{pt}}(r) - \mathbb{P}_{*}(r)$;
- Logit difference: $\text{LD}(r, r') = \text{Logit}(r) - \text{Logit}(r')$; e.g., Logit("Paris") $-$ Logit("Rome"). The patching effect is given by $\text{LD}_{\text{pt}}(r, r') - \text{LD}_{*}(r, r')$. Following Wang et al. (2023), we always normalize this by $\text{LD}_{\text{cl}}(r, r') - \text{LD}_{*}(r, r')$, so it typically lies in $[0, 1]$, where 1 corresponds to fully restored performance and 0 to the corrupted run performance.
- KL divergence: $D_{\text{KL}}(P_{\text{cl}}||P)$, the Kullback-Leibler (KL) divergence from the probability distribution of model outputs in the clean run. The patching effect is $D_{\text{KL}}(P_{\text{cl}}||P_{*}) - D_{\text{KL}}(P_{\text{cl}}||P_{\text{pt}})$.

GN does not provide a corrupted prompt with a well-defined answer r' ("Rome"). To make a fair comparison, the same r' is used for evaluating the logit difference metric under GN.

2.2 Problem settings

Factual recall In the setting of factual association, the model is prompted to fill in factual information, e.g., "The Eiffel Tower is in". Meng et al. (2022) posits that Transformer-based language models complete factual recall (i) at middle MLP layers and (ii) specifically at the processing of the subject's last token. In this work, we do not treat the hypothesis as ground-truth but rather reevaluate it using other approaches than what was attempted by Meng et al. (2022).

IOI An IOI sentence involves an initial dependent clause, e.g., "When John and Mary went to the office", followed by a main clause, e.g., "John gave a book to Mary." In this case, the indirect object (IO) is "Mary" and the subject (S) "John". The IOI task is to predict the final token in the sentence to be the IO. We use S1 and S2 to refer to the first and second occurrences of the subject (S).

We let p_{IOI} denote the distribution of IOI sentences of Wang et al. (2023) containing single-token names. GPT-2 small performs well on p_{IOI} and Wang et al. (2023) discovers a circuit within the model for this task. The circuit consists of attention heads. This is also the focus of our experiments, where we uncover nuanced differences when using different techniques to replicate their result.

3 Corruption methods

In this section, we evaluate GN and STR on localizing factual recall in GPT-2 XL and discovering the IOI circuit in GPT-2 small.

Experiment setup For factual recall, we investigate Meng et al. (2022)'s hypothesis that model computation is concentrated at early-middle MLP layers (by processing the last subject token). Specifically, we corrupt the subject token(s) to generate X_{corrupt}. In the patched run, we override the MLP activations at the last subject token. Following Meng et al. (2022); Hase et al. (2023), at