Knowledge Localization, Editing, and Unlearning in Foundation Models

A prospectus submitted in partial fulfillment of the degree of Doctor of Philosophy

Preliminary Oral Examination for
Keivan Rezaei

Advisor:
Soheil Feizi

Committee Members:
Mohammad Hajiaghayi (co-advisor)
TBD
TBD

Department of Computer Science
University of Maryland, College Park, MD 20742
08/02/2025

Abstract

As foundation models continue to transform the landscape of artificial intelligence—demonstrating impressive capabilities across vision and language tasks—the need for interpretability and control becomes increasingly critical. My research aims to develop methods for interpreting such models by localizing the knowledge they encode, and leveraging this understanding to enable model editing and machine unlearning. My work spans three major areas: interpretability of vision models, where I propose methods for mapping internal representations to human-understandable concepts and explaining failure modes; knowledge localization and editing in text-to-image generative models, including techniques to identify and modify layers responsible for specific concepts; and machine unlearning in large language models, where I introduce benchmarks and algorithms that improve unlearning efficacy, particularly through the use of intermediate checkpoints. Through these efforts, my research contributes to building more transparent, controllable, and adaptable AI systems.

Contents

List of Figures

Chapter 1

Introduction

With the advancement of machine learning, numerous models have been proposed for a wide range of tasks, including vision and language understanding. Recently, foundation models have shown remarkable success in generating both images and text. As the field of AI rapidly progresses, interpreting and understanding these models has become increasingly important. Such interpretability enables better control over model behavior, identification of potential issues, and the development of strategies to improve models by addressing internal shortcomings.

While interpretability in machine learning spans various domains and applications, my work has primarily focused on image classifiers—specifically, on understanding their internal representation space and identifying failure cases with the aim of producing human-understandable explanations. With the recent rise of generative models, such as text-to-image systems and language models, I have extended my research to include interpretation of these models as well.

My work in this area has centered on identifying components within text-to-image models that are responsible for specific types of knowledge—such as concepts, styles, or object representations. This understanding has led to methods for editing such models or unlearning their ability to generate specific content. Additionally, I have explored how large language models can unlearn particular data by introducing new benchmarks to evaluate unlearning, and by designing improved methods that outperform existing baselines. All of my research has revolved around interpreting models, understanding how they encode knowledge, and leveraging

this understanding to detect failures, enable model editing, or facilitate unlearning.
Therefore, I propose the following:

> **I plan to develop methods for interpreting foundation models by localizing knowledge within them, leveraging this localization to enable model editing and machine unlearning in both text-to-image models and large language models.**

With this goal in mind, my research plan breaks down into three areas: interpreting image classifiers by understanding how they encode knowledge and identifying their possible failure cases with human-understandable descriptions; knowledge localization, model editing, and knowledge unlearning in text-to-image models; and unlearning in large language models through new benchmarks and improved methods. I expect this work to contribute the following:

Vision Models Interpretability Vision encoders embed input images into a representation space that captures features the model has extracted for a given image. These features are then passed through a linear head for classification. We examine several models and observe that, despite differences in their training procedures, the features encoded by them can be linearly mapped to each other. In fact, the features obtained by one model for a given image can be approximated via a linear transformation of the features extracted by another model for the same image. Building on this, we map the embedding space of vision encoders to that of a vision-language model, such as CLIP. This enables us to investigate the feature representation space of vision encoders using text—that is, we can understand which textual concepts are activated within their representation space. Using this insight, we propose concept bottleneck models and demonstrate how any vision encoder can be turned into a zero-shot classifier. Furthermore, we investigate failure modes of vision encoders—cases where extracted features lead to misclassification—and aim to describe them using natural language. We propose a method to annotate the dataset with interpretable tags and observe that failures often occur when specific combinations of tags are present in an image. This enables interpretable failure mode detection, resulting in coherent and consistent failure clusters.

Knowledge Localization, Model Editing, and Unlearning in Text-to-Image Models We aim to understand how knowledge about specific concepts is localized within text-to-image generative models, rang-

ing from UNet-based architectures to recently developed transformer-based architectures. We develop tools to identify which layers within text-to-image models are responsible for generating particular concepts—such as artistic styles, copyrighted content, or factual entities. Once these layers are localized, we develop methods to adjust the model's weights and edit the model's behavior—i.e., to make it unlearn certain concepts. We also show that once the relevant layers are identified, more efficient finetuning can be performed by updating only those layers, rather than the entire model.

Unlearning in Large Language Models In this area, we develop methods and benchmarks to evaluate machine unlearning in large language models (LLMs). Machine unlearning has become an essential task in the development and maintenance of LLMs, as these models are typically trained on web-scale data. With the increasing importance of copyright compliance and the presence of malicious or unintended content in training datasets, efficient unlearning methods—ideally with time complexity proportional to the number of samples being removed—are crucial. Several benchmarks have been proposed to assess how different unlearning algorithms perform across various aspects of ideal unlearning behavior. We propose a new benchmark for evaluating machine unlearning, which better captures practical scenarios. Furthermore, we show that machine unlearning can benefit from intermediate model checkpoints that have not been exposed to the target data. We propose a method that leverages such checkpoints and derives informative parameter updates to effectively modify the final model, enabling more efficient and targeted unlearning.

1.1 Contribution

My work to date has contributed to the areas described above, with several ongoing projects continuing in these directions. I have developed methods for the interpretability of vision encoders, knowledge localization and model editing for text-to-image models, as well as knowledge localization and unlearning in large language models.

1.1.1 Vision-Encoders Interpretability

- We propose *Text-to-Concept (and back) via Cross-Model Alignment*, a method that maps the representation space of vision encoders to that of a vision-language model, enabling investigation of the representation space using human-understandable text. I served as co-first author on this work, which was published at ICML 2023.

- We introduce *PRIME: Prioritizing Interpretability in Failure Mode Extraction*, a method for explaining failure modes of vision encoders in image classification tasks. We propose a new approach that prioritizes interpretability, resulting in more coherent and detailed descriptions of failure cases. I served as co-first author on this work, which was published at ICLR 2024.

1.1.2 Text-to-Image Interpretability

- We develop *On Mechanistic Knowledge Localization in Text-to-Image Generative Models*, a method to localize knowledge related to artistic styles, factual content, and copyrighted objects within UNet-based text-to-image models. This involves identifying the cross-attention layers responsible for generating specific knowledge. We further propose a closed-form method for editing these layers, resulting in models that are unable to generate targeted styles or copyrighted content, or that can update outdated facts. I served as co-first author on this work, which was published at ICML 2024.

- We propose *Localizing Knowledge in Diffusion Transformers*, a technique for knowledge localization in diffusion transformers using the idea of attention contribution, which quantifies how individual text tokens influence image generation across layers. This work is currently under review for NeurIPS 2025. I served as second author.

1.1.3 Large Language Models Unlearning

- We introduce *RESTOR: Knowledge Recovery in Machine Unlearning*, a new benchmark for machine unlearning, designed to evaluate algorithms in scenarios where training data contains corrupted knowl-

edge. The goal is not only to remove the corrupted information but also to recover the model's original knowledge. I served as first author on this work, which was published in TMLR 2025.

- We develop *Model State Arithmetic for Machine Unlearning*, a new unlearning algorithm for data-level unlearning by leveraging intermediate model checkpoints. By finetuning earlier checkpoints on the target unlearning data, we extract informative directions that can be applied to the final model, effectively eliminating the impact of unlearning targets. This work is under review for NeurIPS 2025, and I served as co-first author.

1.1.4 Other Contributions

- In *Delegating to Multiple Agents* and *Regret Analysis of Repeated Delegated Choice*, we studied the problem of delegation, where a principal assigns tasks to multiple agents. Our analysis of principal-agent dynamics revealed that, in certain scenarios, having multiple agents can be beneficial for the principal. This research resulted in publications at AAAI 2024 and EC 2023.

- I investigated the problem of provable defense against data poisoning attacks in *Run-Off Election: Improved Provable Defense Against Data Poisoning Attacks*, where we developed a novel defense strategy that improves the robustness of provable defenses. I served as co-first author on this project, which was published at ICML 2023.

- We explored the concept of embedding advertisements within the outputs of language models in *Online Advertisements with LLMs: Opportunities and Challenges* and *Ad Auctions for LLMs via Retrieval Augmented Generation*. We developed a framework for this idea and proposed an auction mechanism for its implementation. This work was published in *ACM SIGecom Exchanges* and NeurIPS 2024.

Proposal Structure

In this proposal, we first go over the related work 2 for each of the above research areas. Then in next chapters, we go over our research contribution.

Chapter 2

Related Work

2.1 Text-To-Concept (and Back) via Cross-Model Alignment

Our alignment of model representation spaces is related to stitching, first introduced by [66], who train linear layers to merge top and bottom chunks of different models, resulting in "franken-CNNs". Stitching was revisited by [3], who aimed to showcase how it can be used as a tool to quantify the quality of representations towards learning how to obtain better representations, and [17], who consider different ways to train stitching layers. We note these works typically stitch together models of the same architecture, where as we consider a much more diverse set of models. Also, those works focus on comparing representations to one another, while we aim to relate representations to human notions.

Namely, we seek to obtain concept vectors within the representation space of off-the-shelf models from text. Also known as concept activation vectors (CAVs), [59] popularized the study of directions corresponding to human concepts in deep feature spaces, as well as the sensitivity of model outputs to changes along these directions, so to interpret deep networks. One limitation is the necessity of example sets of data to define CAVs. More recent efforts automatically discover CAVs [31, 28, 129], though annotating the discovered concepts with language is not straightforward, which motivates our concept-to-text method.

The rise of joint vision-language models like CLIP [94] make it possible to interpret vision space with text, as well as perform zero-shot classification. Follow up works leverage CLIP to annotate neural nodes [84]

or to distill failure modes [50] of non-CLIP models. However, they engage probe datasets or exemplars to communicate with CLIP space, while our method directly aligns representation spaces. Recently, [81] devise a zero-shot method for communication across representation spaces based on relative positions to anchor points, which supports our claim that representation spaces are sufficiently equivalent to where the extension of CLIP's inherent text-to-concept ability to off-the-shelf models via linear alignment is viable.

2.2 PRIME: Prioritizing Interpretability in Failure Mode Extraction

Failure mode discovery. The exploration of biases or challenging subpopulations within datasets, where a model's performance significantly declines, has been the subject of research in the field. Some recent methods for detecting such biases rely on human intervention, which can be time-consuming and impractical for routine usage. For instance, recent works [119, 121] depend on manual data exploration to identify failure modes in widely used datasets like ImageNet. Another line of work uses crowdsourcing [83, 47, 91] or simulators [65] to label visual features, but these methods are expensive and not universally applicable. Some researchers utilize feature visualization [25, 85] or saliency maps [111, 2] to gain insights into the model's failure, but these techniques provide information specific to individual samples and lack aggregated knowledge across the entire dataset. Some other approaches [113, 82, 71, 36] automatically identify failure modes of a model but do not provide human-understandable descriptions for them.

Recent efforts have been made to identify difficult subpopulations and assign human-understandable descriptions to them [26, 50, 61]. DOMINO [26] uses the latent representation of images in a vision-language model to cluster difficult images and then assigns human-understandable descriptions to these clusters. [50] identifies a failure direction in the latent space and assigns description to images aligned with that direction. [42] shows that there is a semantic gap between similarity in latent space and similarity in input space, which can corrupt the output of methods that rely on assigning descriptions to latent embeddings. In [61], concepts are identified whose presence in images leads to a substantial decrease in the model's accuracy. Recent studies [55, 30] highlight the challenge of producing high-quality descriptions in the context of failure mode detection.

Vision-Language and Tagging models. Vision-language models have achieved remarkable success

through pre-training on large-scale image-text pairs [95]. These models can be utilized to incorporate vision-language space and evaluate captions generated to describe images. Recently [79, 67] bridge the modality gap and enable off-the-shelf vision encoders to access shared vision-language space. Furthermore, in our method, we utilize models capable of generating tags for input images [46, 131].

2.3 On Mechanistic Knowledge Localization in Text-to-Image Generative Models

Intepretability of Text-to-Image Models. To our understanding, there's limited exploration into the inner workings of text-to-image models, such as Stable-Diffusion. DAAM [116, 37] scrutinizes diffusion models through the analysis of cross-attention maps between text tokens and images, highlighting their semantic precision. [14] understand the decomposition of concepts in diffusion models. [6] leverage causal tracing to understand how knowledge is stored in text-to-image models such as Stable-Diffusion-v1.

Editing Text-to-Image Models. The capacity to modify a diffusion model's behavior without starting from scratch was initially investigated in Concept-Ablation [64] and Concept-Erasure [29]. Another method, TIME [87], alters all the cross-attention layers' key and value matrices to translate between concepts, though lacks interpretability and applications on a real-use case of model editing. [6] edits text-to-image models in the text-encoder space by leveraging a singular causal state. However, existing works overlook newer text-to-image models (e.g., SD-XL and DeepFloyd), which we delve into in detail.

2.4 RESTOR: Knowledge Recovery in Machine Unlearning

A common goal of machine unlearning is to remove the impact of specific datapoints from training datasets [51, 23, 127, 93, 9, 72, 74], focusing on the removal of memorized token sequences [51, 5], copyrighted material [127, 23], and toxic content [8, 68]. Several techniques have been proposed for machine unlearning where an algorithm operates on a *forget* set – containing documents that must be erased – and a *retain* set including documents that are gathered to preserve model's utility. These techniques mainly employ model fine-tuning on forget sets via gradient ascent on a loss function or preference optimization methods

[74, 130], while preserving utility by finetuning on documents from retain set that controls the degree to which unlearning algorithms affect model, preventing unintended consequences such as degradation of general language modeling utility or the loss of related but unintentional concepts.

[23] propose the task of forgetting Harry Potter, aiming to make it difficult for the unlearned model to generate content about that concept. [74] introduce fictitious unlearning with 200 fictitious authors and GPT-generated Q&A pairs to test forgetting specific authors while retaining untargeted authors' knowledge. [68] focus on unlearning harmful knowledge with multiple-choice questions. [54] evaluate real-world knowledge unlearning across various scenarios, including adversarial prompts. These benchmarks focus on forgetting data while maintaining model utility. However, our work emphasizes removing the *influence of data points*, rather than just prioritizing utility. We evaluate this by examining how the model represents concepts from the unlearning targets after the procedure is complete. Our key insight is that the model possessed accurate representations of these concepts before encountering the problematic training examples. Therefore, if we successfully remove the influence of these examples, the model should revert to its prior, correct knowledge state. Our evaluations, which we call restorative unlearning, measures the success of unlearning algorithms according to this criteria.

While our work focuses on restorative unlearning in large language models, prior efforts have explored unlearning in the context of defending against data poisoning in the vision domain [109, 69, 89, 32]–goals aligned with restorative unlearning. To the best of our knowledge, restoration-oriented machine unlearning in large language models, with data poisoning as one motivating application, remains largely unexplored (though [89] examine poisoning and unlearning for sentiment analysis on GPT-2). Our work addresses this gap by evaluating unlearning methods tailored to large language model. Privacy-motivated unlearning [63, 56] is another unlearning application that often requires removing specific datapoints containing sensitive information. However, there is no clear consensus on what it means to "delete" a datapoint from a model or how to verify successful deletion. Our evaluation framework addresses this gap by assessing the model's knowledge state before training, after exposure, and after unlearning—providing an approach aligned with ideal privacy-preserving unlearning, where the model should behave as if it had never seen the sensitive data.

[75, 76, 128] propose editing methods for updating a model's knowledge, but these rely on access to correct

facts. In contrast, restorative unlearning operates without this assumption, focusing solely on unlearning from corrupted documents – consistent with our study of removing the influence of unlearning datapoints rather than a particular piece of knowledge or a concept [72]. Such an approach is especially valuable when manually curating accurate data would be impractical, or to support the complete erasure of particular datapoints from a model (such as to support "the right to be forgotten" [56]).

2.5 Model State Arithmetic for Machine Unlearning

Machine unlearning was originally developed in order to remove privacy-sensitive information from machine learning models [10]. Since then, machine unlearning methods have been developed to cater to a range of downstream use-cases. At a high-level, these can be formulated as (i) *concept-level* unlearning methods that target knowledge of a particular concept within a model [7, 24, 43, 68, 124], such as 'hazardous' concepts [68], sexually explicit content [29], or knowledge pertaining to a specific topic [24, 43]. Informally, these problems are formulated as *'I do not want my model to generate content related to X'*, where X is a concept such as 'Harry Potter', (ii) *data-level* unlearning which aims to remove the influence of a set of target datapoints on the model, drawn from a model's training dataset [52, 74, 51, 130, 93, 9, 27, 57], enabling the removal of sensitive or private data [51] and the removal of copyrighted content [5]. Existing algorithms typically optimize a tailored loss over the target datapoints to make them unpredictable to the model, while aiming to preserve overall utility, using methods such as gradient ascent or preference-based optimization [130]. Informally, these problems are formulated as *'I want my model to exhibit behavior as if it were never trained on X'*, where X is a datapoint or set of datapoints. In this work, the focus of our contribution is data-level unlearning and—unless stated otherwise— we use the term machine unlearning to denote this setting.

Formally, machine unlearning considers a model $M_{\mathcal{D}}$ trained on a dataset $\mathcal{D}$ that includes a subset of samples $\mathcal{D}_{\mathrm{f}} \in D$ (the *forget set*), which is the target of unlearning. The goal is to produce a model M' whose behavior closely approximates that of a model trained from scratch on $\mathcal{D} \backslash \mathcal{D}_{\mathrm{f}}$. Although exact unlearning methods exist [10, 16], they are computationally prohibitive. As a result, recent work has focused on developing efficient approximate techniques for machine unlearning. These methods work in time complexity proportional to forget set size $|\mathcal{D}_{\mathrm{f}}|$ rather than $|\mathcal{D}|$, making them more efficient.

Given a set $\mathcal{D}_\text{f}$, evaluating approximate machine unlearning algorithms then requires assessing two key aspects: (i) forgetting efficacy: the model M' should no longer be influenced by samples in $\mathcal{D}_\text{f}$, typically measured by evaluating performance on tasks that query the model for knowledge or capabilities introduced in the forget set, and (ii) model utility: the model M' should preserve the influence of data not in the forget set, typically measured by evaluating performance on tasks that query the model for knowledge and capabilities derived from rest of data, i.e., $\mathcal{D} \setminus \mathcal{D}_\text{f}$. A variety of benchmarks have been proposed to evaluate these criteria [74, 54, 112, 102], each highlighting different dimensions of what effective machine unlearning should achieve, such as exhibiting behavior similar to the model before it encountered the unlearning target [102], or maintaining knowledge on related but different entities [74].

Unlearning algorithms [130, 52, 127, 15] typically operate by optimizing a specialized loss function over the forget set $\mathcal{D}_\text{f}$. To mitigate catastrophic forgetting— unintended degradation in the model beyond the targeted datapoints—— these algorithms may also incorporate an optimization objective over a *retain set* $\mathcal{D}_\text{r}$. This is intended to minimize deviation from the original model's behavior by preserving performance on $\mathcal{D}_\text{r}$, i.e., finetuning the model on $\mathcal{D}_\text{r}$ during unlearning is intended to constrain the weight update such that the model forgets only the intended information while maintaining its overall capabilities. Formally, many unlearning methods can be described by the following objective:

$$\theta_\text{unlearn} = \arg\min_\theta \mathbb{E}_{x\sim\mathcal{D}_\text{f}} \left[\mathcal{L}_\text{f}(x;\theta)\right] + \lambda\ \mathbb{E}_{x\sim\mathcal{D}_\text{r}} \left[\mathcal{L}_\text{r}(x;\theta)\right],$$

where $\mathcal{L}_\text{f}$ and $\mathcal{L}_\text{r}$ are the loss functions corresponding to the forget and retain sets, respectively, and λ controls the trade-off between forgetting and utility preservation.

In this work, we propose a new approach to machine unlearning that leverages model checkpoints, by computing *forget vectors* with respect to these checkpoints. This is conceptually similar to task vectors [48], which involves computing a direction in a model's parameter space associated with performance on a downstream task. Negating this vector from the model parameters can reduce performance on that task. However, applying task vectors directly for machine unlearning has shown limited success [112], as the direction extracted from a model trained on target datapoints often lacks expressiveness and fails to effectively remove the influence of the target data.

Chapter 3

Vision Models Interpretability

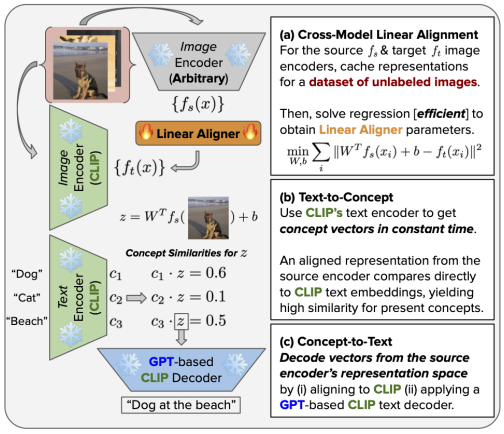

Figure 3.1: Overview. After *aligning* the representation space of a given image encoder to a CLIP image encoder, we can compare aligned representations of images to concept vectors obtained *directly from text* (typically, an example set of data is required to obtain each concept vector; our method is example-free, i.e. O(1) w.r.t. data collection). Using a GPT-based CLIP decoder, we can map arbitrary vectors in representation space to text. Our method yields efficient **interpretability**: we only train *one linear layer*.

3.1 Text-To-Concept (and Back) via Cross-Model Alignment

The representation spaces of deep vision models are undoubtedly rich in semantic structure. However, these deep feature spaces are notoriously challenging for humans to interpret, mainly because it is hard for us to

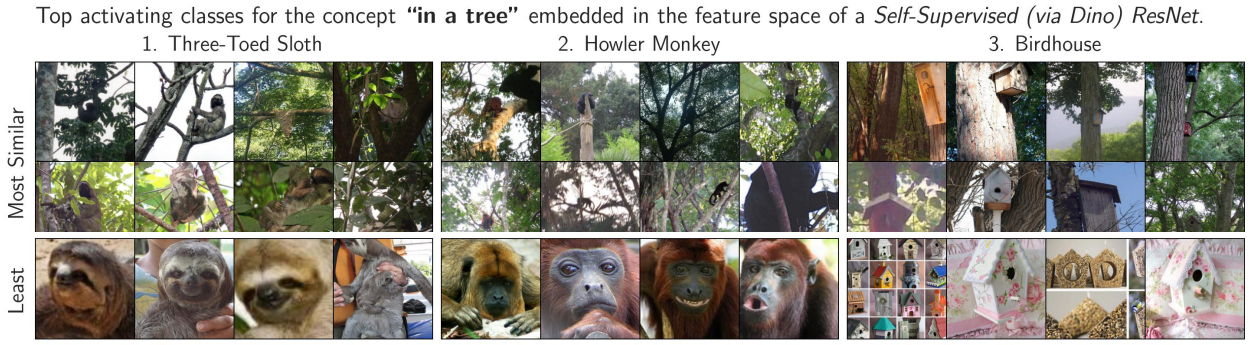

Figure 3.2: Qualitative validation of text-to-concept. ImageNet classes are sorted by the average cosine similarity of the CLIP embedding for "in a tree" to the *linearly aligned* Dino ResNet representations of images within each class. Highest ranked classes indeed often appear in a tree, as is evident by the most similar instances. The least similar instances appropriately do not contain the concept.

digest thousands of numbers at once. Unlike deep models, which encode concepts as vectors in high (e.g. $d = 2048$) dimensional spaces, humans have developed language to describe the world around us concisely. In this work, we propose a method to map text to concept vectors that can be compared directly to image representations obtained from off-the-shelf vision encoders trained with *n*o text supervision.

Our method works by *a*ligning the representation space of a given vision model to the representation space of a CLIP [94] model. By design, the CLIP representation space is shared across jointly trained vision and text encoders. Thus, CLIP models already have text-to-concept built in, via the text encoder. To extend this capability to off-the-shelf models, we propose to learn a mapping between representation spaces. Specifically, we optimize a function to predict the representation of an image for a target model (i.e. CLIP) from the same image's representation for a source model (i.e. off-the-shelf vision model). We can then map the representations of the off-the-shelf model to CLIP space, where the aligned features would reside in the same space as the concept vector for the desired text.

The mapping function, however, may significantly change the semantics of its input. To prevent this, we restrict the hypothesis space of our mappings to be *a*ffine transformations. Despite their simplicity, we find that linear layers are surprisingly effective at performing feature space alignment, even between models with diverse architectures and training procedures. This observation suggests that despite drastically different approaches to training, diverse models seem to learn to store information in similar ways. Most notably, we

can align model representations to CLIP, thus extending CLIP's text-to-concept abilities to existing models. Figure 3.2 visually validates our approach: after encoding the concept "in a tree" in CLIP space and computing similarity with aligned representations from a self-supervised ResNet, the classes with the highest average similarity are reasonable, and images within them with the highest similarity prominently display the concept, while the least similar class instances do not. Stronger validation of our approach is found in performing **z**ero-shot classification using off-the-shelf encoders via text-to-concept. Models achieve impressive zero-shot accuracy on many tasks, often being competitive with a CLIP model that is larger, trained on many more samples with richer supervision, and most notably, directly optimized to align with the text encoder we use in text-to-concept. Surprisingly, zero-shot accuracy of off-the-shelf models surpasses the CLIP in a few cases, particularly for color recognition. While greatly expanding the use cases for existing models, these zero-shot abilities also support the belief that deep models learn many more abstract notions than what they are explicitly trained to know. Text-to-concept allows for uncovering and better utilizing the rich semantics hidden in existing models' representation spaces.

In addition to zero-shot learning for free, text-to-concept has several immediate interpretability applications, such as converting vision encoders to *C*oncept Bottleneck Models (CBMs) [62] with *n*o concept supervision required. CBMs decompose inference into a concept prediction step followed by class prediction using a white box model (i.e. linear head) on concept predictions, so that the contribution of each concept to the final logit can be precisely computed. Typically, CBMs require concept supervision in addition to class labels, but with text-to-concept, we can replace concept predictions with concept similarities, obtained by comparing aligned representations to the vector obtained for the desired notion. Then, CBM training reduces to simply training a linear layer to predict class labels from pre-computed similarities of aligned image representations to a set of concept vectors. We illustrate this application on RIVAL10 data [78], which has attribute labels, though we only use these labels to verify our zero-shot concept prediction approach. Indeed, we obtain an AUROC of 0.8 for RIVAL10 attribute prediction in the zero-shot manner described, leading to a highly accurate (93.8%) resultant CBM with desired interpretability benefits (see Figure 3.7).

Next, we show text-to-concept can demystify large datasets, as the distribution of similarities between a bank of text-to-concept vectors and aligned representations of the data essentially summarizes what concepts are

present, explaining the data distribution in human terms. This can be applied to diagnosing distribution shifts, as we can inspect the shift w.r.t. to human-understandable concept similarities. For example, when comparing ImageNet to ObjectNet [4], we can show that the distribution of similarities for the "indoors" concept shifts dramatically, capturing the essence of why ObjectNet poses a challenge: images in ObjectNet were taken in people's homes. Another way text-to-concept aids in engaging with large datasets is via concept-based image retrieval. Using *concept logic*, we query the image representations for a given model that satisfy a set of concept similarity thresholds, allowing for greater human control of the importance of each concept in the search, yielding reasonable results in finding specific images in a large corpus.

Lastly, we close the human-machine communication loop by introducing *concept-to-text* to directly decode vectors in a model's representation space. Our implementation aligns the model's space to CLIP, and then leverages an existing CLIP space decoder [117] that uses a CLIP embedding to guide the output of GPT-2. The existing decoder was intended for image captioning, though we demonstrate that its abilities extend to general vectors (i.e. not obtained from a single image) from non-CLIP models after alignment. Specifically, we decode the vectors in the classification head of three ImageNet trained models. We then use a human study to verify that the decoded captions describe the class associated with each vector, finding that our simple method works in over 92% of cases.

Our methods extend the capabilities of multi-modal models like CLIP to other models that are trained on much smaller uni-modal datasets and with weaker supervisions. This can be useful when a model more accurate or smaller than CLIP for a specific domain is desired, or when training CLIP-like models is infeasible, due to the large corpus of image-caption pairs needed. Moreover, since our approach can be applied to interpret any model's representation space, while only requiring the training of a linear layer, its potential impact is very high, as text-to-concept is easy to plug-in and has a breadth of applications. The implications of our work are also startling: first, the success of linear representation space alignment indicates that diverse models ultimately represents inputs relatively similarly. The emergent zero-shot abilities of existing models suggests an under-utilization of models we already have. Finally, the synergy we display between CLIP, GPT, and existing models, coupled with the ability to communicate across these models and back and forth with humans, makes the prospect of diverse models collaborating with minimal tuning very promising.

3.2 Model Alignment

We use $\mathcal{X}$ to denote the set of all possible input images. Let $D_{\text{train}}, D_{\text{test}} \subset \mathcal{X}$ denote the training and test datasets. We define a *vision encoder* as a model f that maps images $x \in \mathcal{X}$ to vectors $f(x) \in \mathbb{R}^d$.

Given two vision encoders f_s, f_t, *representation space alignment* of model f_s to model f_t is the task of learning a mapping $h : f_s(\mathcal{X}) \to f_t(\mathcal{X})$. We restrict h to the class of affine transformations, i.e., $h_{W,b}(z) := W^T z + b$. To maximally retain the original semantics of representation spaces, we design the following optimization problem

$$W, b = \arg\min_{W,b} \frac{1}{|D_{\text{train}}|} \sum_{x \in D_{\text{train}}} \|W^T f_s(x) + b - f_t(x)\|_2^2.$$

The above optimization can be viewed as multiple linear regression problems; thus we evaluate the linear alignment on D_{test} by considering the quality of the solution on those linear regression problems. We use *Coefficient of Determination*, i.e., R^2 which is the proportion of the variation in the dependent variables that is predictable from the independent variables. Furthermore, we note that for the vision encoder f_s, there usually exists a *classification head* $g_s : \mathbb{R}^d \to \mathcal{C}$ that classifies a representation in the space of model f_s. Indeed, the predicted label for input x is $g_s(f_s(x))$. Note that $\mathcal{C}$ denotes the set of labels, e.g., ImageNet classes. We define *aligned accuracy* as the accuracy of classification on D_{test} when we use f_s as the vision encoder, then do the linear transformation to obtain the corresponding representation in space of f_t, and finally, use g_t for classification. If alignment works well, aligned accuracy should be admissible and comparable to the accuracy of model f_t when no alignment is used. Then, we define *retained accuracy* as the ratio of aligned accuracy to the accuracy of model f_t without any alignment. Note that we use ImageNet-1K train and test datasets as D_{train} and D_{test} in linear alignment.

Interestingly, we observe that simple linear alignment works well in terms of both R^2, and aligned accuracy across various models. Figure 3.3 shows the aligned accuracy between diverse pairs of models. In the scope of linear alignment, we further consider the sample efficiency of optimizing linear alignment as well as investigating linear alignment in space of top principal components of representation spaces, where we roughly see strong, near identity correspondence between the top principal components of different models. We refer to Appendix for more details.

Figure 3.3: Heatmap of retained accuracy. the value in row r and column c is the average of retained accuracy when doing alignment from all models in group r to all models in group c. Note that "Sup" refers to supervised while "SS" refers to self-supervised models. All models except CLIPs are trained on ImageNet.

According to Figure 3.3, various models are highly alignable to CLIP models. This is surprising as CLIP models are trained on other datasets than ImageNet and their training procedure involves vision/text supervision which is drastically different from other models. High-quality alignment to CLIP representation space enables models to adopt a wide variety of CLIP models capabilities, which we analyze in this work. On the other hand, we observe that retained accuracy when aligning CLIP models to other models is not high. This is mainly due to the fact that CLIP models encode images and texts in relatively low-dimensional spaces and in linear regression, approximating dependent variables becomes harder as the number of independent variables decreases. Indeed, linear alignment works worse when we align from a representation space with lower dimensionality to a representation space with higher dimensionality.

3.3 Text to Concept

Leveraging representation space alignment, specifically to CLIP, we perform text-to-concept, where text descriptions of semantic concepts are encoded as vectors that can be directly compared (i.e. via cosine similarity) with the aligned features of images obtained from an off-the-shelf vision encoder (see Figure 3.1). Despite its simplicity, alignment-based text-to-concept is surprisingly effective, which, after further detailing our method, we demonstrate qualitatively and quantitatively in this section. Notably, we show that the

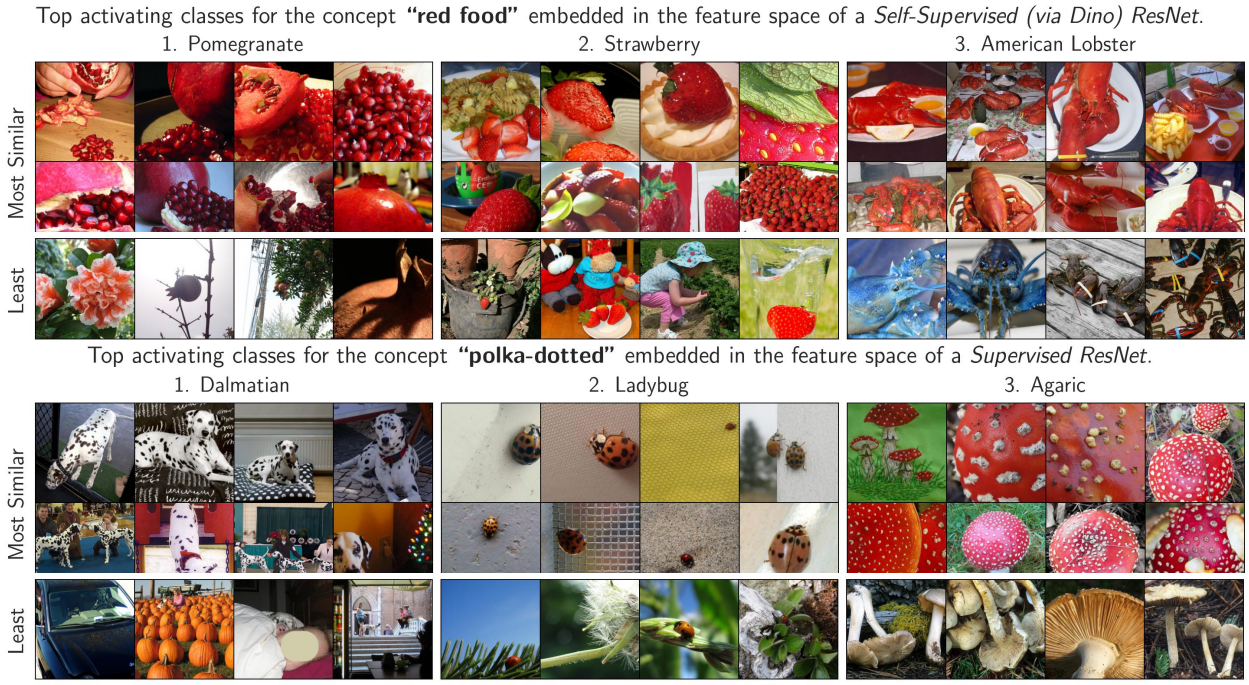

Figure 3.4: Text-to-concept can encode finer-grain concepts, like combinations of concepts (*"red food"*) or textures (*"polka-dotted"*).

similarities of aligned image representations to class vectors obtained via text-to-concept enables *zero-shot classification for non-CLIP models off-the-shelf*, with zero-shot accuracy of much simpler models at times exceeding that of CLIP.

3.3.1 Method Details

We define text-to-concept as a procedure for obtaining vectors corresponding to concepts described as text that can be directly compared (i.e. via cosine similarity) to image representations from a fixed vision encoder. Our method begins with a string describing some concept, like "red food". We then prepend this string with a number of template prompts (e.g. "a photo of {}"); we use the same template prompts as in CLIP's original paper for ImageNet zero-shot classification. Then, we embed the templated text to CLIP space using CLIP's text encoder, and average the resultant vectors over all templates to obtain a single concept vector (as is standard). For some object agnostic concepts, such as contexts like "in a tree", we can encode a general

prompt like "a photo of an object in a tree", or we can obtain a more refined vector by encoding "{*prompt*} {*class name*} in a tree", averaging over all choices for *class name* and *prompt*. There are countless ways to prompt engineer; we elect to use general prompts in most cases, as prompt engineering is not the focus of our work.[1]

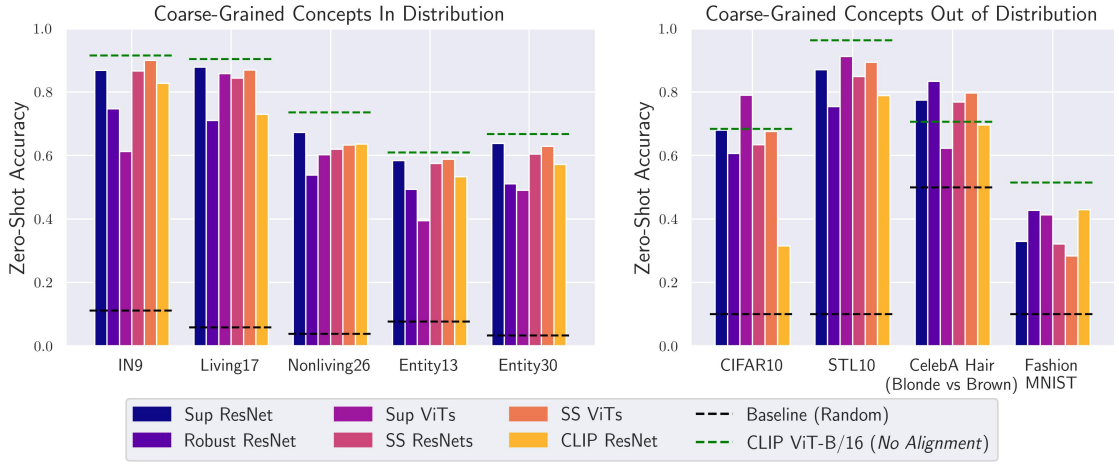

Figure 3.5: The zero-shot capabilities of CLIP can extend to off-the-shelf vision encoders via alignment based text-to-concept. (**L**eft) Models trained on ImageNet can recognize coarse categorizations of ImageNet classes, despite never explicitly being taught them. (**R**ight) Off-the-shelf models remain strong zero-shot classifiers even when images are out of distribution. In some cases, they surprisingly surpass the accuracy of the CLIP vision encoder whose jointly-trained text encoder was used to embed each class vector.

Then, for a given model, we train a linear layer to align its representation space to CLIP; specifically, we use CLIP ViT-B/16. We pass ImageNet training images to the given model's feature encoder and CLIP's vision encoder, resulting in a dataset of paired representations with which we train our aligner (Section 3.2). Now, we have two functions that map to CLIP's vision space: the CLIP text encoder (since the text and vision representation spaces are shared), and the composition of the given model's encoder with the linear aligner. Since the concept vector obtained via CLIP's text encoder and aligned representations from the given model are both mapped to the same space, we can compare them directly, thus satisfying our definition of text-to-concept. Alternatively, we could train an aligner from CLIP to the given model's representation space, and align the text embedding instead of the features. We found this method to be less effective, possibly because the dimensionality of CLIP space is lower than most models we study. Since our aligner

[1]See Appendix for complete details on all prompts used.

is a simple affine transformation, alignment minimally changes the content of the representation obtained from the off-the-shelf model. Also, note the efficiency of our approach: after training a linear layer once, we can encode any number of new concepts from text at no additional training cost.

Qualitative Validation: Figures 3.2 and 3.4 show images selected based on the cosine similarity of their aligned representations (obtained using off-the-shelf encoders and trained linear aligners) to certain concept vectors. For each concept, we present the classes with the highest average similarity, as well as the most and least similar images within them. The retrieved classes are sensible for each concept (e.g. *American Lobster* for "red food"). Sorting images within each class separates examples where the concept is extremely prominent from those where the concept is absent (e.g. images of uncooked lobsters are least similar to the "red food" concept). Note that the models used to obtain the image representations differ in architecture, training objective and supervision from CLIP, and most notably, they have not been trained with any text/concept supervisions. Thus, it is surprising that we can easily connect these visual concept representations to the CLIP text embeddings. Nonetheless, over a range of concept types (a context, a combination of color and a concept, and a texture), our visualizations qualitatively validate our proposed text-to-concept approach. We now turn to zero-shot classification for quantitative validation.

3.3.2 Zero-Shot Classification

CLIP models perform zero-shot classification by comparing image representations to embeddings of text strings describing each class: the predicted class is the one whose text embedding is most similar to the test image's representation. This is referred to as zero-shot since no labeled instances from the candidate classes are used. Considering classes as concepts, we can then use text-to-concept to obtain vectors that are directly comparable to aligned representations from off-the-shelf vision encoders, thus extending CLIP's zero-shot capabilities. The accuracy of zero-shot classification serves as a quantitative measure of the quality of text-to-concept vectors. Indeed, when concept vectors align better with representations of samples in the class, zero-shot accuracy is higher. Thus, we explore zero-shot classification over many datasets to shed insight on when and how well text-to-concept works. We consider models over diverse architectures and training procedures, though all models are roughly equal in size ($\sim$25M parameters) and are only trained

on ImageNet (except for CLIP). Also, the baseline CLIP model (ViT-B/16) whose text encoder is used to embed concepts is much larger in size (∼80M parameters); this baseline is intended more so as an upper bound.

First, we ask if models can recognize new categorizations of the data they were trained over. Namely, we consider coarse grained categorizations of ImageNet classes (e.g. distinguishing *i*nsects from *c*arnivores, see [126, 107] and Appendix). We also investigate if these coarse grained concepts can still be recognized as image data is taken out of distribution. Figure 3.5 displays the results. We observe impressive zero-shot performance in both cases. For example, on a 17-way classification problem, self-supervised ViTs achieve 85% accuracy, despite never receiving supervision about these classes, or any classes at that. Shockingly, in a few cases, even the performance of the CLIP model whose text encoder (with which it was jointly trained) was used to obtain concept vectors is surpassed.

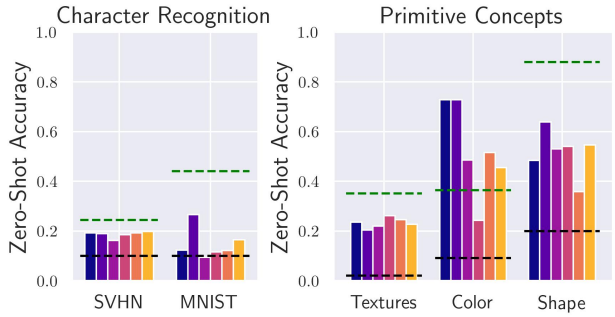

Figure 3.6: Edge cases for zero-shot classification. (**L**eft) Models struggle with OCR. (**R**ight) Models can recognize some primitive concepts by name. Same legend as figure 3.5.

To stress test text-to-concept, we consider tasks that require models to recognize characters (specifically digits) or primitive concepts, like textures, colors, and shapes. We observe most models only marginally surpass random accuracy for character recognition tasks. Oddly, the adversarially trained ResNet is roughly twice as good as other models in zero-shot MNIST classification, though it still performs far worse than the baseline CLIP model, which also struggles. This suggests models simply may not have any notion as to what distinguishes digits from one another, which is not surprising given that it would not be very useful for understanding ImageNet images. On the other hand, models achieve far better than random performance in recognizing primitive concepts, which appear in ImageNet as low level features for more abstract notions.

Interestingly, color recognition is a task where most models outperform the CLIP baseline, suggesting the CLIP ViT may have reduced color sensitivity relative to other primitive concepts.

While these experiments validate our proposed text-to-concept method, it is also remarkable that these off-the-shelf models, who have much smaller training sets (roughly 0.3% the size of CLIP's) and receive far less supervision, are comparable to CLIP in recognizing the unseen classes we consider. This suggests that models learn far more than what they are taught. In other words, models discover many semantic concepts and organize their representation spaces so that these concepts are roughly linearly separable, even when they are only explicitly directed to separate 1000 classes or to simply draw representations of similar inputs close to another. Thus, even models trained with elementary techniques likely contain much richer representation spaces than their use case requires. The success of transfer learning supports this claim, as a small amount of labeled data is sufficient for a model to recognize 'new' concepts, implying they had some notion of the concepts before. Text-to-concept can enable better understanding and utilization of these rich representation spaces, *without requiring new labeled samples.*

3.4 Additional Applications of Text-to-Concept

3.4.1 Concept-Bottleneck Networks for Free

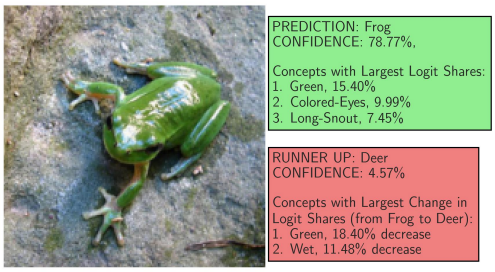

Figure 3.7: Example inference for a Concept Bottleneck Model (CBM) obtained via training a linear layer on zero-shot concepts. Since logits in the CBM are linear functions of concept scores, we can precisely quantify the contribution of concepts to each logit.

The zero-shot results suggest models are already aware of many concepts beyond those which they are directly trained to learn. One case where knowledge of concepts related to the classification task is salient is

Concept Bottleneck Models (CBMs) [62]. CBMs are intepretable by design, as they first predict the presence of concepts using a black box, and then obtain class logits with a white box (e.g. linear layer) atop concept predictions. Thus, the contribution of each concept to the predicted logit can be computed directly, allowing predictions to be faithfully explained with semantic reasons. A major barrier to using CBMs is that they require concept supervision, which can be prohibitively expensive. Text-to-concept, however, alleviates this constraint, thanks to zero-shot concept prediction.

We use RIVAL10 classification [78] as an example for how a CBM can be implemented with *no concept supervision* using text-to-concept. RIVAL10 is an attributed dataset, though we do not use these labels during training. We use RIVAL10 because a linear classifier operating on ground truth attribute labels achieves 94.5%, indicating that a CBM could be effective. Further, the attribute labels allow for quantifying the quality of the zero-shot concept vectors we obtain.

To implement the network, we use text-to-concept to encode the 28 attributes annotated in RIVAL10 as vectors in CLIP space. We then compute the similarities between the attribute vectors and aligned (to CLIP) features from an ImageNet pretrained ResNet-50. Finally, we fit a linear layer atop image-attribute similarities (i.e. in representation space) to predict class labels. Note that the only training we conduct is that of the final classification head and of the aligner, both of which are linear layers, making them time and sample efficient to optimize. The resultant CBM achieves 93.8% accuracy, and yields the desired interpretability advantages, as shown in figure 3.7. Moreover, using image-attribute similarities (via text-to-concept) as a score for predicting attributes achieves an AUROC of 0.8, with 72% of attributes achieving at least an AUROC of 0.75. Thus, zero-shots concepts are relatively accurate in predicting RIVAL10 attributes. See Appendix for details.

3.4.2 Concept-Based Dataset Summarization and Distribution Shift Diagnosis

The interpretability benefits of text-to-concept also apply to demystifying large datasets. Specifically, one can discern the presence of a concept in their data by using text-to-concept to obtain a corresponding vector, and computing the similarity of this vector to all aligned images representations. As modern datasets continue to grow, the need for efficient concept-based summaries of these datasets will also grow; text-to-concept can

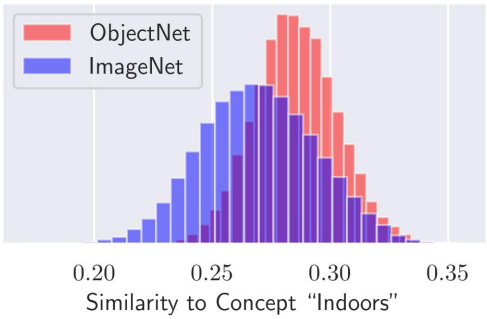

Figure 3.8: Concept similarities can reveal distribution shifts, like in ObjectNet, where photos are taken within people's homes.

provide such summaries easily.

Moreoever, one can track the distribution of concept similarities for a stream of data over time. Suppose for example a model is deployed to a new setting and it begins to fail. By comparing the distribution of concept similarities in the training set to the new data, one can diagnose the distribution shifts at play. As a proof of concept, we inspect ObjectNet [4], a challenging distribution shift for ImageNet models consisting of images taken within people's homes. Figure 3.8 shows the distribution of similarities between the vector for the concept 'indoors' and aligned image representations obtained from a ResNet-50 of ImageNet and ObjectNet samples. For ObjectNet, the distribution is significantly (as determined with a Kolmogorov-Smirnov test) shifted to the right compared to ImageNet. In practice, one may maintain a bank of concepts and track similarities over their stream of data, automatically flagging concepts which experience significant shift.

3.4.3 Concept Logic for Image Retrieval

Text-to-concept enables the computation of the similarity of a model's representation for an image to an arbitrary concept. Given a corpus of data and a vision encoder trained to represent said data, we can retrieve images using text, based on their similarity to text-to-concept vectors. While one may combine all keywords in a string to obtain a single composite concept vector, we observe suboptimal performance with this approach, as words receive imbalanced attention and negations are often ignored by CLIP's text encoder.

A simple alternative is to retrieve images that satisfy a set of conditions. For example, instead of searching for

Figure 3.9: Images retrieved based on similarity of their representation to multiple text-to-concept vectors. Retrieved images satisfy the multiple conditions listed above each image. ∼ denotes 'not'. Concept logic with text-to-concept enables searching over images, while allowing the use of any vision encoder to represent them.

"a dog on the beach at sunset", we can separately encode the concepts 'dog', 'on the beach', and 'at sunset'. We then filter images based on their similarity for each concept; we use thresholds based on the distribution of similarities for a given concept (i.e. at least 3 standard deviations above the mean). Analogously, we can encode negative conditions by requiring similarity to be below some threshold. Figure 3.9 demonstrates the effectiveness of our approach, retrieving rare images via concept logic (details in Appendix). While concept logic for image search can be done over CLIP vision embeddings, the results may be suboptimal when querying over a specific dataset for which CLIP was not finetuned, particularly compared to a vision model trained on that dataset.

3.5 Concept-to-Text

Text-to-concept grants insight into the representation spaces of deep models by mapping semantic notions expressed as words directly to concept vectors. However, humans still need to conjecture what concepts may be relevant before probing a representation space. We now ask, can we directly map concept activation vectors to text? We refer to this as *concept-to-text*, and propose an implementation using alignment to CLIP and generative language models (Figure 3.1).

Similar to how we that observe diverse vision models learn to store information in similar ways, allowing for cross-model alignment, we argue that language models and vision models similarly learn much of the

Swin (S)	ResNet-50	Dino ViTs8
94.48%	95.14%	92.18%

Table 3.1: Percent of captions for decoded class vectors deemed relevant to images in the class by human annotators.

same information, and can thus be plugged into one another with ease. Recent work supports this claim, as vision models have been stitched to generative language decoders to perform image captioning and visual question answering [77, 22]. Notably, [80] captions an image by feeding its CLIP image embedding through a finetuned version of GPT-2 [96]. A follow up work, ZeroCap [117], similarly decodes with GPT-2 while receiving guidance from a CLIP embedding, but does so without requiring any tuning of either CLIP or GPT-2. We elect this method as it further demonstrates how existing models can work together off-the-shelf. However, other decoders could easily be put in place of ZeroCap if desired. We highlight this flexibility as it entails that concept-to-text will continue to improve as individual components are improved (e.g. GPT-2 $\rightarrow$ Chat-GPT).

With our linear aligners, we can already map representations from off-the-shelf encoders to CLIP. Thus, with no additional training, we can perform elementary concept-to-text by simply feeding aligned features to ZeroCap. While ZeroCap expects CLIP embeddings of natural images as input, we conjecture that passing a vector encoding some semantic notion can similarly be decoded. To asses this claim, we consider the task of decoding classification head vectors. These vectors exist in the original model space, and should encode information relevant to their corresponding ImageNet class. Thus, we can quantify the effectiveness of our elementary concept-to-text method by seeing if decoded class vectors indeed describe the desired class. Specifically, we use the prompt "Image of a " and set the desired sequence length to 1 so that a single word is decoded per class vector. We perform a human study to answer this question, showing MTurk workers a collage of images from a given class, along with the caption obtained from decoding the class vector, and asking if this caption is relevant to the images shown. The results, shown in Table 3.1, show that in over 92% of cases, our naive approach to concept-to-text appears effective in decoding class vectors for three diverse models. See Appendix for additional details.

3.6 PRIME: Prioritizing Interpretability in Failure Mode Extraction

A plethora of reasons (spurious correlations, imbalanced data, corrupted inputs, etc.) may lead a model to underperform on a specific subpopulation; we term this a *failure mode*. Failure modes are challenging to identify due to the black-box nature of deep models, and further, they are often obfuscated by common metrics like overall accuracy, leading to a false sense of security. However, these failures can have significant real-world consequences, such as perpetuating algorithmic bias [11] or unexpected catastrophic failure under distribution shift. Thus, the discovery and description of failure modes is crucial in building reliable AI, as we cannot fix a problem without first diagnosing it.

Detection of failure modes or biases within trained models has been studied in the literature. Prior work [119, 121] requires humans in the loop to get a sense of biases or subpopulations on which a model underperforms. Some other methods [114, 82, 60, 71] do the process of capturing and intervening in hard inputs without providing *human-understandable* descriptions for challenging subpopulations. Providing human-understandable and *interpretable* descriptions for failure modes not only enables humans to easily understand hard subpopulations, but enables the use of text-to-image methods [100, 104, 106, 58] to generate relevant images corresponding to failure modes to improve model's accuracy over them.

Recent work [26, 50, 61, 19] takes an important step in improving failure mode diagnosis by additionally finding natural language descriptions of detected failure modes, namely via leveraging modern vision-language models. These methodologies leverage the shared vision-language latent space, discerning intricate clusters or directions within this space, and subsequently attributing human-comprehensible descriptions to them. However, questions have been raised regarding *the quality of the generated descriptions*, i.e., there is a need to ascertain whether the captions produced genuinely correspond to the images within the identified subpopulation. Additionally, it is essential to determine whether these images convey shared semantic attributes that can be effectively articulated through textual descriptions.

In this work, we investigate whether or not the latent representation space is a good proxy for semantic space. In fact, we consider two attributed datasets: CelebA[73] and CUB-200[123] and observe that two samples sharing many semantic attributes may indeed lie far away in latent space, while nearby instances may not share any semantics (see Section 3.9.2). Hence, existing methods may suffer from relying on representation

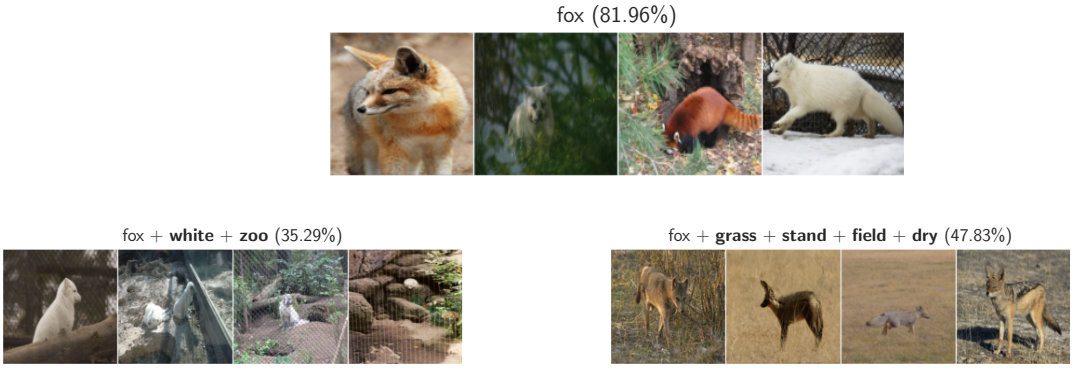

Figure 3.10: Visualization of two detected failure modes of class "fox" on a model trained on Living17. Overall accuracy for images of class "fox" is 81.96%. However, we identify two coherent subsets of images with significant accuracy drops: foxes standing in dry grass fields (47.83% accuracy) and foxes in a zoo where a white object (fox or other objects) is detected (35.29% accuracy). See Appendix for more examples.

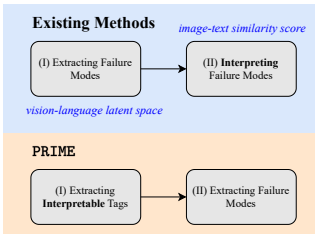

Figure 3.11: PRIME illustration.

space as clusters and directions found in this space may contain images with different semantic attributes leading to less coherent descriptions.

Inspired by this observation and the significance of faithful descriptions, we propose PRIME. In this method, we suggest to reverse the prevailing paradigm in failure mode diagnosis. That is, we put *interpretability first*. In our method, we start by obtaining human-understandable concepts (tags) of images using a pre-trained tagging model and examine model's behavior conditioning on the presence or absence of a combination of those tags. In particular, we consider different groups of tags and check whether (1) there is a significant drop in model's accuracy over images that represent all tags in the group and (2) that group is minimal, i.e., images having only some of those tags are easier images for the model. When a group of tags satisfies both of these conditions, we identify it as a failure mode which can be effectively described by these tags. Figure 3.11 shows the overview of our approach and compares it with existing methods.

As an example, by running PRIME on a trained model over Living17, we realize that images where a **black** ape is **hanging** from a **tree branch** identify a hard subpopulation such that model's accuracy drops from 86.23% to 41.88%. Crucially, presence of all 3 of these tags is necessary, i.e., when we consider images that have 1 or 2 of these 3 tags, the accuracy of model is higher. Figure 3.12 illustrates these failure modes. We further study the effect of number of tags in Section 3.9.1.

To further validate our method, we examine data unseen during the computation of our failure mode descriptions. We observe that the images that match a failure mode lead the model to similarly struggle. That is, we demonstrate *generalizability* of our failure modes, crucially, directly from the succinct text descriptions. While reflecting the quality of our descriptions, this allows for bringing in generative models. We validate this claim by generating hard images using some of the failure mode's descriptions and compare the accuracy of model on them with some other generated images that correspond to easier subpopulations.

Finally, we show that PRIME produces better descriptions for detected failure modes in terms of *similarity*, *coherency*, and *specificity* of descriptions, compared to prior work that does not prioritize interpretability. Evaluating description quality is challenging and typically requires human assessment, which can be impractical for extensive studies. To mitigate that, inspired by CLIPScore [38], we present a suite of three automated metrics that harness vision-language models to evaluate the quality. These metrics quantify both the intra-group image-description similarity and coherency, while also assessing the specificity of descriptions to ensure they are confined to the designated image groups. We mainly observe that due to putting interpretability first and considering different combinations of tags (concepts), we observe improvements in the quality of generated descriptions. We discuss PRIME's limitations in Appendix.

Summary of Contribution.

1. We propose PRIME to extract and explain failure modes of a model in human-understandable terms by prioritizing interpretability.

2. Using a suite of three automated metrics to evaluate the quality of generated descriptions, we observe improvements in our method compared to strong baselines such as [26] and [50] on various datasets.

3. We advocate for the concept of putting interpretability first by providing empirical evidence derived from latent space analysis, suggesting that distance in latent space may at times be a misleading

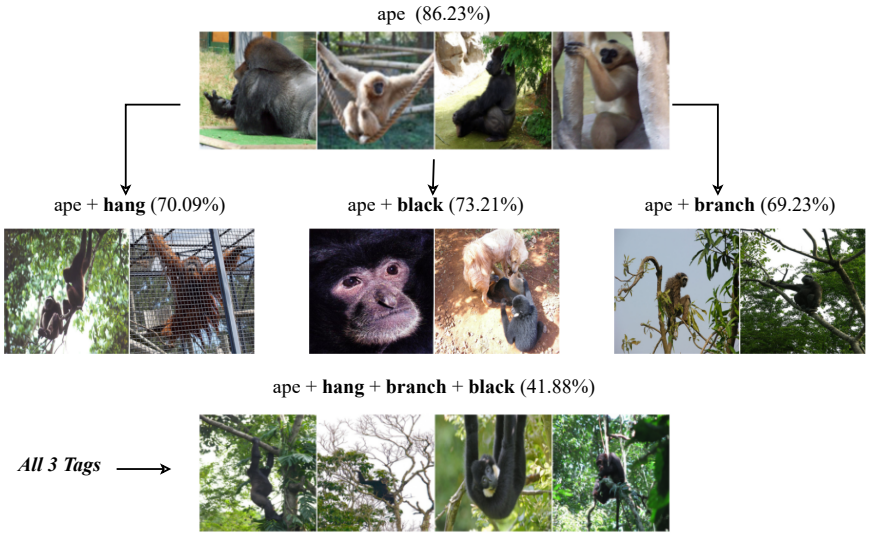

Figure 3.12: Although appearance of tags "hang", "black", and "branch" individually lowers model's accuracy, when all of them appear in the images, model's accuracy drops from 86.23% to 41.88%.

measure of semantic similarity for explaining model failure modes.

3.7 Extracting Failure Modes by Conditioning on Human-understandable Tags

Undesirable patterns or spurious correlations within the training dataset can lead to performance discrepancies in the learned models. For instance, in the Waterbirds dataset [105], images of landbirds are predominantly captured in terrestrial environments such as forests or grasslands. Consequently, a model can heavily rely on the background and make a prediction based on that. Conversely, the model may also rely on cues such as the presence of the ocean, sea, or boats to identify the input as waterbirds. This can result in performance drops for images where a waterbird is photographed on land or a landbird is photographed at sea. Detecting failure modes involves identifying groups of inputs where the model's performance significantly declines. While locating failure inputs is straightforward, *categorizing* them into distinct groups characterized by *human-understandable concepts* is a challenging task. To explain failure modes, we propose

PRIME. Our method consists of two steps: (I) obtaining relevant tags for the images, and (II) identifying failure modes based on extracted tags.

3.7.1 Obtaining Relevant Tags

We start our method by collecting concepts (tags) over the images in the dataset. For example, for a photo of a fox sampled from ImageNet [18], we may collect tags "orange", "grass", "trees", "walking", "zoo", and others. To generate these tags for each image in our dataset, we employ the state-of-the-art *Recognize Anything Model (RAM)* [131, 46], which is a model trained on image-caption pairs to generate tags for the input images. RAM makes a substantial step for large models in computer vision, demonstrating the zero-shot ability to recognize any common category with high accuracy.

Let $\mathcal{D}$ be the set of all images. We obtain tags over all images of $\mathcal{D}$. Then, we analyze the effect of tags on prediction in a class-wise manner. In fact, the effect of tags and patterns on the model's prediction depends on the main object in the images, e.g., presence of water in the background improves performance on images labeled as waterbird while degrading performance on landbird images. For each class c in the dataset, we take the union of tags generated by the model over images of class c. Subsequently, we eliminate tags that occur less frequently than a predetermined threshold. This threshold varies depending on the dataset size, specifically set at 50, 100, and 200 in our experimental scenarios. In fact, we remove rare (irrelevant) tags and obtain a set of tags T_c for each class c in the dataset, e.g., $T_c = \{$"red", "orange", "snow", "grass", ...$\}$.

3.7.2 Detecting Failure Modes

After obtaining tags, we mainly focus on tags whose presence in the image leads to a performance drop in the model. Indeed, for each class c, we pick a subset $S_c \subseteq T_c$ of tags and evaluate the model's performance on the images of class c including all tags in S_c. We denote this set of images by I_{S_c}. I_{S_C} is a coherent image set in the sense that those images share at least the tags in S_c.

For I_{S_c}, to be a *failure mode*, we require that the model's accuracy over images of I_{S_c} significantly drops, i.e., denoting the model's accuracy over images of I_{S_c} by A_{S_c} and the model's overall accuracy over the images of class c by A_c, then $A_{S_c} \leq A_c - a$. Parameter a plays a pivotal role in determining the severity of

the failure modes we aim to detect. Importantly, we want the tags in S_c to be minimal, i.e., none of them should be redundant. In order to ensure that, we expect that the removal of any of tags in S_c determines a relatively easier subpopulation. In essence, presence of all tags in S_c is deemed essential to obtain that hard subpopulation.

More precisely, Let n to be the cardinality of S_c, i.e., $n = |S_c|$. We require all tags $t \in S_c$ to be necessary. i.e., if we remove a tag t from S_c, then the resulting group of images should become an easier subpopulation. More formally, for all $t \in S_c$, $A_{S_c \setminus t} \geq A_{S_c} + b_n$ where $b_2, b_3, b_4, ...$ are some hyperparameters that determine the degree of necessity of appearance of all tags in a group. We generally pick $b_2 = 10\%$, $b_3 = 5\%$ and $b_4 = 2.5\%$ in our experiments. These values help us fine-tune the sensitivity to tag necessity and identify meaningful failure modes. Furthermore, we require a minimum of s samples in I_{S_c} for reliability and generalization. This ensures a sufficient number of instances where the model's performance drops, allowing us to confidently identify failure modes. Figure 3.10 shows some of the obtained failure modes.

How to obtain failure modes. We generally use *Exhaustive Search* to obtain failure modes. In exhaustive search, we systematically evaluate various combinations of tags to identify failure modes, employing a brute-force approach that covers all possible combinations of tags up to l ones. More precisely, we consider all subsets $S_c \subseteq T_c$ such that $|S_c| \leq l$ and evaluate the model's performance on I_{S_c}. As mentioned above, we detect S_c as a failure mode if (1) $|I_{S_c}| \geq s$, (2) model's accuracy over I_{S_c} is at most $A_c - a$, and (3) S_c is minimal, i.e., for all $t \in S_c$, $A_{S_c \setminus t} \geq A_{S_c} + b_{|S_c|}$. It is worth noting that the final output of the method is all sets I_{S_c} that satisfy those conditions and **description** for this group consist of **class name** (c) and **all tags in** S_c.

We note that the aforementioned method runs with a complexity of $O\left(|T_c|^l |\mathcal{D}|\right)$. However, l is generally small, i.e., for a failure mode to be generalizable, we mainly consider cases where $l \leq 4$. Furthermore, in our experiments over different datasets $|T_c| \approx 100$, thus, the exhaustive search is relatively efficient. For instance, running exhaustive search ($l = 4, s = 30, a = 30\%$) on Living17 dataset having 17 classes with 88400 images results in obtaining 132 failure modes within a time frame of under 5 minutes. We refer to Appendix for more efficient algorithms and Appendix for more detailed explanation of `PRIME`'s hyperparameters.

Experiments and Comparison to Existing Work. We run experiments on models trained on Living17,

NonLiving26, Entity13 [108], Waterbirds [105], and CelebA [73] (for age classification). We refer to Appendix for model training details and the different hyperparameters we used for failure mode detection. We refer to Appendix for the full results of our method on different datasets. We engage two of the most recent failure mode detection approaches DOMINO[26] and Distilling Failure Directions[50] as strong baselines and compare our approach with them.

3.8 Evaluation

Let $\mathcal{D}$ be the dataset on which we detect failure modes of a trained model. The result of a human-understandable failure mode extractor on this dataset consists of sets of images, denoted as $I_1, I_2, ..., I_m$, along with corresponding descriptions, labeled as $T_1, T_2, ..., T_m$. Each set I_j comprises images that share similar attributes, leading to a noticeable drop in model accuracy. Number of detected failure modes, m, is influenced by various hyperparameters, e.g., in our method, minimum accuracy drop (a), values for $b_2, b_3, ...$, and the minimum group size (s) are these parameters.

One of the main goals of detecting failure modes in human-understandable terms is to generate high-quality captions for hard subpopulations. We note that these methods should also be evaluated in terms of coverage, i.e., what portion of failure inputs are covered along with the performance gap in detected failure modes. All these methods extract hard subpopulations on which the model's accuracy significantly drops, and coverage depends on the dataset and the hyperparameters of the method, thus, we mainly focus on generalizability of our approach and quality of descriptions.

3.8.1 Generalization on Unseen Data

In order to evaluate generalizability of the resulted descriptions, we take dataset $\mathcal{D}'$ including unseen images and recover relevant images in that to each of captions $T_1, T_2, ..., T_m$, thus, obtaining $I'_1, I'_2, ..., I'_m$. Indeed, I'_j includes images in $\mathcal{D}'$ that are relevant to T_j. If captions can describe hard subpopulations, then we expect hard subpopulations in $I'_1, I'_2, ..., I'_m$. Additionally, since $\mathcal{D}$ and $\mathcal{D}'$ share the same distribution, we anticipate the accuracy drop in I_j to closely resemble that in I'_j.

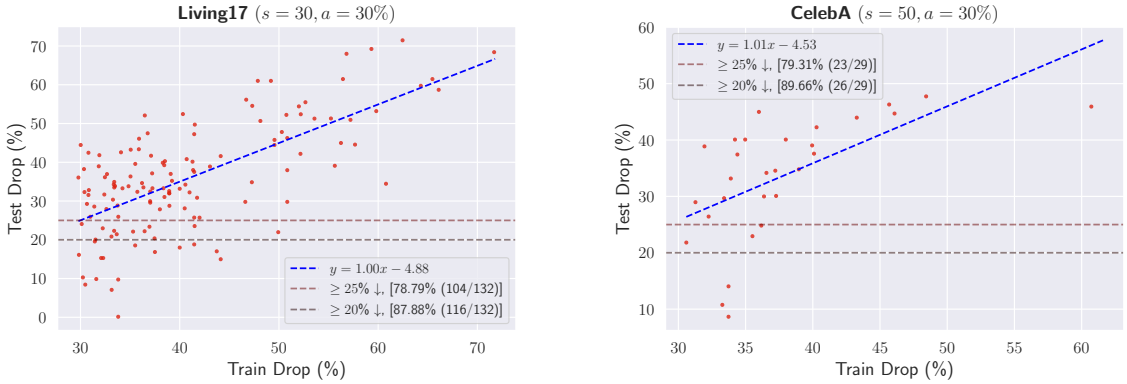

Figure 3.13: Evaluating detected failure modes on unseen data. **(Left)**: we extract failure modes on Living17 dataset using $s = 30$ and $a = 30\%$. 132 failure groups (over 17 classes) are detected and it is observed that around 86.01% of detected failure modes exhibit at least 25% drop in accuracy over unseen data that shows a significant degree of generalization. **(Right)**: same results for CelebA dataset where the parameters for failure mode detection is $s = 50$ and $a = 30\%$. Around 79.31% of failure modes show the drop of at least 20%. The trend of $y = x$ is seen in these plots.

In our method, for a detected failure mode I_{S_c}, we obtain I'_{S_c} by collecting images of $\mathcal{D}'$ that have all tags in S_c. For example, if appearance of tags "black", "snowing", and "forest" is detected as a failure mode for class "bear", we evaluate model's performance on images of "bear" in $\mathcal{D}'$ that include those three tags, expecting a significant accuracy drop for model on those images. As seen in Figure 3.13, `PRIME` shows a good level of generalizability. We refer to Appendix for generalization on other datasets with respect to different hyperparameters (s and a). While all our detected failure modes generalize well, we observe stronger generalization when using more stringent hyperparameter values (high s and a), though it comes at the cost of detecting fewer modes.

In contrast, existing methods [26, 50] do not provide a direct way to assess generalization from text descriptions alone. See Appendix for more details.

3.8.2 Generalization on Generated Data

In this section we validate `PRIME` on synthetic images. To utilize image generation models, we employ language models to create descriptive captions for objects and tags associated with failure modes in images. We note that use of language models is just for validation on synthetic images, it is not a part of `PRIME`

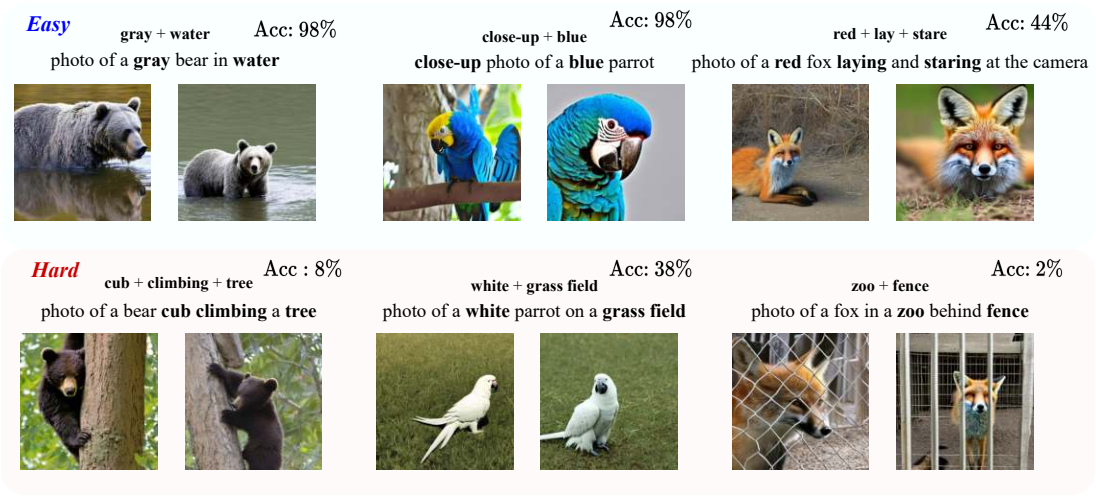

Figure 3.14: Accuracy of model over 50 generated images corresponding to one of the success modes and failure modes for classes "bear", " parrot", and "fox" from Living17. Accuracy gap shows that our method can identify hard and easy subpopulations. Images show that extracted tags are capable of describing detailed images.

framework. We discuss about limitations of using language models in Appendix. These captions serve as prompts for text-to-image generative models, enabling the creation of artificial images that correspond to the identified failure modes. To achieve this, we adopt the methodology outlined in [122], which leverages a denoising diffusion model [40, 104]. We fine-tune the generative model on the Living17 dataset to generate images that match the distribution of the data that the classifiers is trained on.

For each class in Living17 dataset, we apply our approach to identify two failure modes (hard subpopulations) and two success modes (easy subpopulations). We then employ ChatGPT[2] to generate descriptive captions for these groups. Subsequently, we generate 50 images for each caption and assess the model's accuracy on these newly generated images. We refer to Appendix for more details on this experiment and average discrepancy in accuracy between the success modes and failure modes which further validates `PRIME`. Figure 3.14 provides both accuracy metrics and sample images for three hard and three easy subpopulations.

[2]ChatGPT 3.5, August 3 version

3.8.3 Quality of Descriptions

Within this section, our aim is to evaluate the quality of the descriptions for the identified failure modes. In contrast to Section 3.8.2, where language models were employed to create sentence descriptions using the tags associated with each failure mode, here we combine tags and class labels in a bag-of-words manner. For instance, when constructing the description for a failure mode in the "ape" class with the tags "black" + "branch," we formulate it as "a photo of ape black branch". We discuss more about it in Appendix.

In order to evaluate the quality of descriptions, we propose a suite of three complementary automated metrics that utilize vision-language models (such as CLIP) as a proxy to obtain image-text similarity [38]. Let t be the failure mode's description, $f_{\text{text}}(t)$ denote the normalized embedding of text prompt t and $f_{\text{vision}}(x)$ denote the normalized embedding of an image x. The similarity of image x to this failure mode's description t is the dot product of image and text representation in shared vision-language space. More precisely, $\text{sim}(x, t) := \langle f_{\text{vision}}(x), f_{\text{text}}(t) \rangle$.

For a high-quality failure mode I_j and its description T_j, we wish T_j to be similar to images in I_j, thus, we consider the average *similarity* of images in I_j and T_j. we further expect a high level of *coherency* among all images in I_j, i.e., these images should all share multiple semantic attributes described by text, thus, we wish the standard deviation of similarity scores between images in I_j and T_j to be low. Lastly, we expect generated captions to be *specific*, capturing the essence of the failure mode, without including distracting irrelevant information. That is, caption T_j should only describe images in I_j and not images outside of that. As a result, we consider the AUROC between the similarity score of images inside the failure mode (I_j) and some randomly sampled images outside of that. We note that in existing methods as well as our method, all images in a failure mode have the same label, so we sample from images outside of the group but with the same label.

In Figure 3.15, we show (1) the average similarity score, i.e., for all I_j and $x \in I_j$, we take the mean of $\text{sim}(x, T_j)$, (2) the standard deviation of similarity score, i.e., the standard deviation of $\text{sim}(x, T_j)$ for all I_j and $x \in I_j$, and (3) the AUROC between the similarity scores of images inside failure modes to their corresponding description and some randomly sampled images outside of the failure mode to that. As shown in Figure 3.15, `PRIME` improves over DOMINO [26] in terms of all AUROC, average similarity, and standard

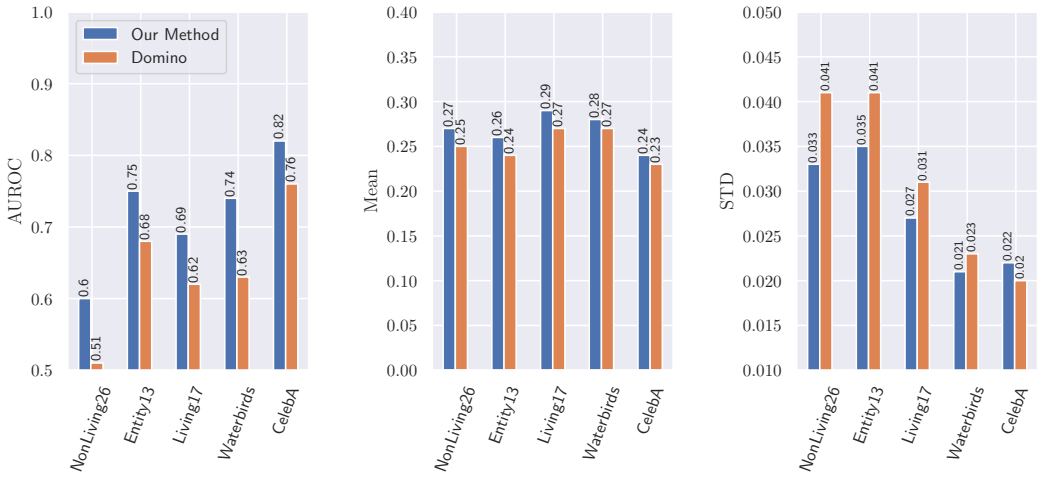

Figure 3.15: The mean and standard deviation of similarity scores between images in failure modes and their respective descriptions, along with the AUROC measuring the similarity score between descriptions and images inside and outside of failure modes, demonstrate that our method outperforms DOMINO in descriptions it generates for detected failure modes across various datasets.

deviation on different datasets. It is worth noting that this improvement comes even though DOMINO chooses a text caption for the failure mode ***to maximize the similarity score in latent space***. We use hyperparameters for DOMINO to obtain fairly the same number of failure modes detected by `PRIME`. Results in Figure 3.15 show that `PRIME` is better than DOMINO in the descriptions it provides for detected failure modes. In Appendix we provide more details on these experiments. Due to the limitations of [50] for automatically generating captions, we cannot conduct extensive experiments on various datasets. More details and results on that can be found in Appendix.

3.9 On Complexity of Failure Mode Explanations

We note that the main advantage of our method is its more faithful interpretation of failure modes. This comes due to (1) putting interpretability first, i.e., we start by assigning interpretable tags to images and then recognize hard subpopulations and (2) considering combination of several tags which leads to a higher number of attributes (tags) in the description of the group.

3.9.1 Do We Need to Consider Combination of Tags?

We shed light on the number of tags in the failure modes detected by our approach. We note that unlike Bias2Text [61] that finds biased concepts on which model's behavior changes, we observe that sometimes appearance of several tags (concepts) all together leads to a severe failure mode. As an example, we refer to Table 3.2 where we observe that appearance of all 3 tags together leads to a significant drop while single tags and pairs of them show relatively better performance.

Table 3.2: Accuracy on unseen images ($\mathcal{D}'$) for class "ape" when given tags appear in the inputs.

# of Tags	Tags	Accuracy
3	hang; branch; black;	41.18%
2	hang; branch;	56.33%
	hang; black;	56.25%
	branch; black;	54.67%
1	hang;	70.09%
	branch;	69.23%
	black;	73.21%

Table 3.3: Average accuracy drop over unseen images ($\mathcal{D}'$) on failure modes with 3 tags and images have at least 2 of those tags or at least one of them.

Dataset	3 Tags	2 Tags	1 Tag
Entity13	34.75%	25.29%	14.86%
Living17	26.82%	17.13%	8.18%
Waterbirds	23.35%	14.43%	7.19%
CelebA	23.25%	16.84%	9.02%

In `PRIME`, we emphasize the necessity of tags. Specifically, for any detected failure mode, the removal of any tag would result in an easier subpopulation. Consequently, failure modes with more tags not only provide more detailed description of their images but also characterize more challenging subpopulations. Table 3.3 presents the average accuracy drop on unseen images for groups identified by three tags, compared to the average accuracy drop on groups identified by subsets of two tags or even a single tag from those failure modes. These results clearly demonstrate that involving more tags leads to the detection of more challenging subpopulations.

3.9.2 Clustering-Based Methods may Struggle in Generating Coherent Output

We empirically analyze the reverse direction of detecting human-understandable failure modes. We note that in recent work where the goal is to obtain interpretable failure modes, those groups are found by clustering images in the latent space. Then, when a group of images or a direction in the latent space is found, these

Table 3.4: Statistics of the distance between two points in CelebA conditioned on number of shared tags. Distances are reported using CLIP ViT-B/16 representation space. The last column shows the probability that the distance between two sampled images with at least d common tags be more than that of two randomly sampled images.

# of shared tags $\geq d$	mean	standard deviation	Probability
$d = 0$	9.49	0.98	0.50
$d = 1$	9.47	1.00	0.49
$d = 3$	9.23	1.00	0.42
$d = 5$	8.89	1.21	0.34
$d = 7$	8.32	1.80	0.25

methods leverage the shared space of vision-language to find the text that best describes the images inside the group.

We argue that these approaches, based on distance-based clusters in the representation space, may produce less detailed descriptions. This is because the representation space doesn't always align perfectly with the semantic space. Even points close to each other in the feature space may differ in certain attributes, and conversely, points sharing human-understandable attributes may not be proximate in the feature space. Hence, these approaches cannot generate high-quality descriptions as their detected clusters in the representation space may contain images with other semantic attributes.

To empirically test this idea, we use two attribute-rich datasets: CelebA [73] and CUB-200 [123]. CelebA features 40 human-understandable tags per image, while CUB-200, a dataset of birds, includes 312 tags per image, all referring to semantic attributes. We use CLIP ViT-B/16 [95] and examine its representation space in terms of datasets' tags. Table 3.4 shows the statistics of the distance between the points conditioned on the number of shared tags. As seen in the Table 3.4, although the average of distance between points with more common tags slightly decreases, the standard deviation of distance between points is high. In fact, points with many common tags can still be far away from each other. Last column in Table 3.4 shows the probability that the distance between two points with at least d shared tags be larger than the distance of two randomly sampled points. Even when at least 5 tags are shared between two points, with the probability of 0.34, the distance can be larger than two random points. Thus, if we plant a failure mode on a group of images sharing a subset of tags, these clustering-based methods cannot find a group consisting of *only* those

Table 3.5: Average number of tags appear in at least αN images among N closest images to a randomly sampled image in the representation space as well as average number of shared tags in semantic space (CelebA Dataset).

	Representation Space			Semantic Space
	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	
$N = 50$	4.17	3.56	2.91	7.55
$N = 100$	3.94	3.34	2.71	7.25

images; they will inevitably include other irrelevant images, leading to an incoherent failure mode set and, consequently, a low-quality description. This can be observed in Appendix where we include DOMINO's output.

We also run another experiment to foster our hypothesis that distance-based clustering methods cannot fully capture semantic similarities. We randomly pick an image x and find N closest images to x in the feature space. Let C be the set of these images. We inspect this set in terms of the number of tags that commonly appear in its images as recent methods [26, 19, 50], take the average embedding of images in C and then assign a text to describe images of C. Table 3.5 shows the average number of tags that appear in at least αN images of set C (we sample many different points x). If representation space is a good proxy for semantic space, then we expect a large number of shared tags in close proximity to point x. At the same time, for the point x, we find the maximum number of tags that appear in x and at least N other images. This is the number of shared tags in close proximity of point x but in semantic space. As shown in Table 3.5, average number of shared tags in semantic space is significantly larger than the average number of shared tags in representation space.

3.10 Conclusions

In this study, drawing from the observation that current techniques in human-comprehensible failure mode detection sometimes produce incoherent descriptions, along with empirical findings related to the latent space of vision-language models, we introduced `PRIME`, a novel approach that prioritizes interpretability in failure mode detection. Our results demonstrate that it generates descriptions that are more similar, coherent, and specific compared to existing methods for the detected failure modes.

Chapter 4

Text-to-Image Models Interpretability

4.1 On Mechanistic Knowledge Localization in Text-to-Image Generative Models

In recent years, substantial strides in conditional image generation have been made through diffusion-based text-to-image generative models, including notable examples like Stable-Diffusion [103], Imagen [106], and DALLE [101]. These models have captured widespread attention owing to their impressive image generation and editing capabilities, as evidenced by leading FID scores on prominent benchmarks such as MS-COCO [70]. Typically trained on extensive billion-scale image-text pairs like LAION-5B [110], these models encapsulate a diverse array of visual concepts, encompassing color, artistic styles, objects, and renowned personalities.

A recent work [6] designs an interpretability framework using causal tracing [90] to trace the location of knowledge about various styles, objects or facts in text-to-image generative models. Essentially, causal tracing finds the indirect effects of intermediate layers [90], by finding layers which can restore a model with corrupted inputs to its original state. Using this framework, the authors find that knowledge about various visual attributes is distributed in the UNet, whereas, there exists a unique causal state in the CLIP text-encoder where knowledge is localized. This unique causal state in the text-encoder can be leveraged to edit text-to-image models in order to remove style, objects or update facts effectively. However, we note

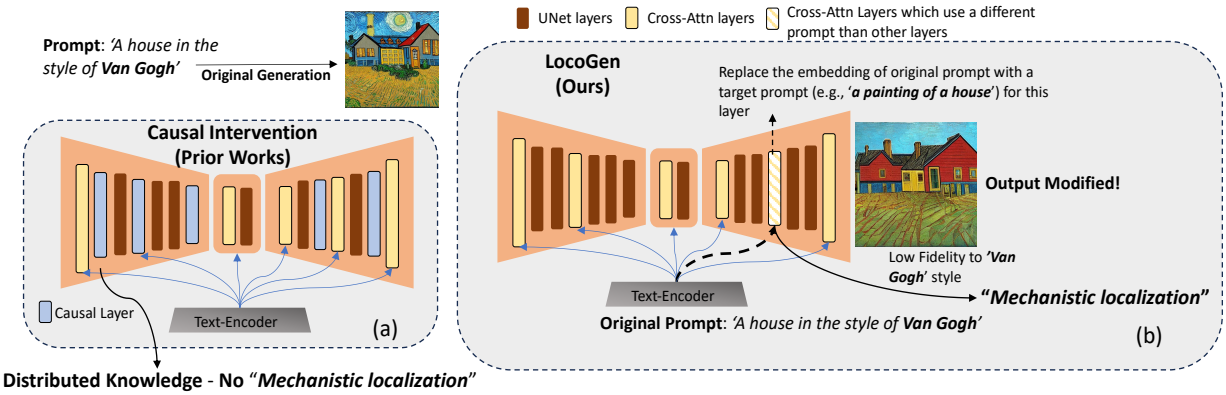

Figure 4.1: **LocoGen: Identifying UNet layers that, when given different input, can alter visual attributes (e.g., style, objects, facts).** (a) Earlier works [6] which show distributed knowledge using causal interventions. (b) LocoGen where a few cross-attention layers receive a different prompt-embedding than the original, leading to generation of images without the particular style.

that their framework is restricted to early Stable-Diffusion variants such as Stable-Diffusion-v1-5.

In our paper, we first revisit knowledge localization for text-to-image generative models, specifically examining the effectiveness of causal tracing beyond Stable-Diffusion-v1-5. While causal tracing successfully identifies unique localized states in the text-encoder for Stable-Diffusion variants, including v1-5 and v2-1, it fails to do so for recent models like SD-XL [92] and DeepFloyd[1] across different visual attributes. In the UNet, causal states are distributed across a majority of open-source text-to-image models (excluding DeepFloyd), aligning with findings in [6]. Notably, for DeepFloyd, we observe a lack of strong causal states corresponding to visual attributes in the UNet.

To address the *universal* knowledge localization framework absence across different text-to-image models, we introduce the concept of *m*echanistic localization that aims to identify a small number of layers which control the generation of distinct visual attributes, across a spectrum of text-to-image models. To achieve this, we propose LocoGen, a method that finds a subset of cross-attention layers in the UNet such that when the input to their key and value matrices is changed, output generation for a given visual attribute (e.g., "style") is modified (see Figure 4.1). This intervention in the intermediate layers has a direct effect on the output – therefore LocoGen measures the direct effect of intermediate layers, as opposed to indirect

[1] https://github.com/deep-floyd/IF

effects in causal tracing.

Leveraging LocoGen, we probe knowledge locations for different visual attributes across popular open-source text-to-image models such as Stable-Diffusion-v1, Stable-Diffusion-v2, OpenJourney[2], SD-XL [92] and DeepFloyd. For all models, we find that unique locations can be identified for visual attributes (e.g., "style", "objects", "facts").

Using these locations, we then perform weight-space model editing to remove artistic "styles", modify trademarked "objects" and update outdated "facts" in text-to-image models. This weight-space editing is performed using LocoEdit which updates the key and value matrices using a closed-form update in the locations identified by LocoGen. Moreover, for certain attributes such as "style", we show that knowledge can be traced and edited to a subset of neurons, therefore highlighting the possibilities of neuron-level model editing.

Contributions. In summary, our contributions include:

- We highlight the drawbacks of existing interpretability methods such as causal tracing for localizing knowledge in latest text-to-image models.
- We introduce LocoGen which can universally identify layers that control for visual attributes across a large spectrum of open-source text-to-image models.
- By examining edited models using LocoEdit along with LocoGen, we observe that this efficient approach is successful across a majority of text-to-image models.

4.2 Preliminaries

Diffusion models start with an initial random real image $\mathbf{x}_0$, the noisy image at time step t is expressed as $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{(1-\alpha_t)}\epsilon$. Here, α_t determines the strength of the random Gaussian noise, gradually diminishing as the time step increases, ensuring that $\mathbf{x}_T \sim \mathcal{N}(0, I)$. The denoising network $\epsilon_\theta(\mathbf{x}_t, \mathbf{c}, t)$, is pre-trained to denoise the noisy image $\mathbf{x}_t$ and produce $\mathbf{x}_{t-1}$. Typically, the conditional input $\mathbf{c}$ for the

[2]https://huggingface.co/prompthero/openjourney

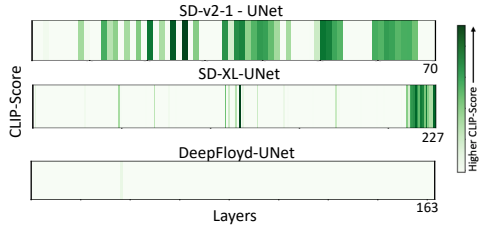

Figure 4.2: **Causal tracing for UNet.** Similar to [6], we find that knowledge is causally distributed across the UNet for text-to-image models such as SD-v2-1 and SD-XL. For DeepFloyd we do not observe any significant causal state in the UNet.

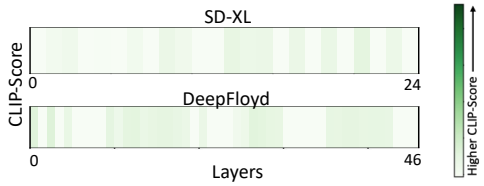

Figure 4.3: **Causal tracing for text-encoder.** Unlike SD-v1-5 and SD-v2-1, we find that a singular causal states does not exist in the text-encoder for SD-XL and DeepFloyd.

denoising network $\epsilon_\theta(.)$ is a text-embedding derived from a caption c through a text-encoder, denoted as $\mathbf{c} = v_\gamma(c)$.

The noising as well as the denoising operation can also occur in a latent space defined by $\mathbf{z} = \mathcal{E}(\mathbf{x})$ [103] for better efficiency. The pre-training objective learns to denoise in the latent space as denoted by:

$$\mathcal{L}(\mathbf{z}, \mathbf{c}) = \mathbb{E}_{\epsilon,t}||\epsilon - \epsilon_\theta(\mathbf{z}_t, \mathbf{c}, t)||_2^2,$$

where $\mathbf{z}_t = \mathcal{E}(\mathbf{x}_t)$ and $\mathcal{E}$ is an encoder such as VQ-VAE [120].

4.3 On the Effectiveness of Causal Tracing for Text-to-Image Models

In this section, we empirically observe the effectiveness of causal tracing to models beyond Stable-Diffusion-v1-5. In particular, we find the ability of causal tracing to identify localized control points in Stable-Diffusion-v2-1, OpenJourney, SD-XL and DeepFloyd.

Causal Tracing in UNet. In Figure 4.2, we find that knowledge across different visual attributes is

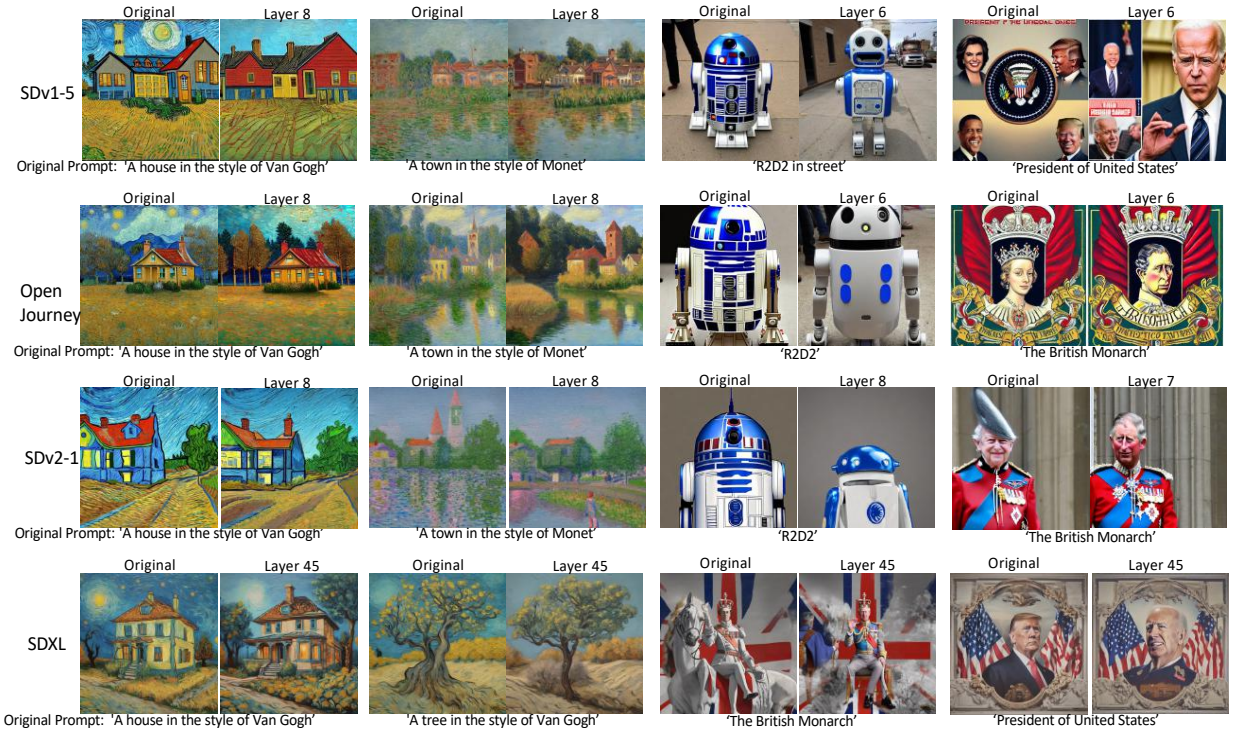

Figure 4.4: **Interpretability Results: Images generated by intervening on the layers identified by LocoGen across various open-source text-to-image models.** We compare the original generation vs. generation by intervening on the layers identified with LocoGen along with a target prompt. We find that across various text-to-image models, visual attributes such as *s*tyle, objects, facts can be manipulated by intervening only on a very small fraction of cross-attention layers.

distributed in the UNet for all the text-to-image models (except for DeepFloyd), similar to Stable-Diffusion-v1-5. However, the degree of distribution varies between different text-to-image models. While knowledge about various visual attributes is densely distributed in Stable-Diffusion variants, for SD-XL we find that the distribution is extremely sparse (e.g., only 5% of the total layers are causal). For DeepFloyd, we observe that there are no strong causal states in the UNet. We provide more qualitative visualizations on causal tracing across the these text-to-image models in Appendix. Overall, these results reinforce the difficulty of editing knowledge in the UNet directly due to (i) distribution of causal states or (ii) absence of any.

Causal Tracing in Text-Encoder. [6] show that there exists a unique causal state in the text-encoder for Stable-Diffusion-v1-5 and Stable-Diffusion-v2-1 which can be used to perform fast model editing. In

Figure 4.3, we find that such an unique causal state is absent in the text-encoder for DeepFloyd and SD-XL. We note that DeepFloyd uses a T5-text encoder, whereas SD-XL uses a a combination of CLIP-ViT-L and OpenCLIP-ViT-G [95]. Our empirical results indicate that an unique causal state arises only when a CLIP text-encoder is used by itself in a text-to-image model.

4.4 LocoGen: Towards Mechanistic Knowledge Localization

Given the lack of generalizability of knowledge localization using causal tracing as shown in Chapter 4.3, we introduce LocoGen , which can identify localized control regions for visual attributes across *a*ll text-to-image models.

4.4.1 Knowledge Control in Cross-Attention Layers

During the inference process, the regulation of image generation involves the utilization of classifier-free guidance, as outlined in [41] which incorporates scores from both the conditional and unconditional diffusion models at each time-step. Specifically, the classifier-free guidance is applied at each time-step to combine the conditional ($\epsilon_\theta(\mathbf{z}_t, \mathbf{c}, t)$) and unconditional score estimates ($\epsilon_\theta(\mathbf{z}_t, t)$). The result is a combined score denoted as $\hat{\epsilon}(\mathbf{z}_t, \mathbf{c}, t)$.

$$\hat{\epsilon}(\mathbf{z}_t, \mathbf{c}, t) = \epsilon_\theta(\mathbf{z_t}, \mathbf{c}, t) + \alpha \left(\epsilon_\theta\left(\mathbf{z_t}, \mathbf{c}, t\right) - \epsilon_\theta(\mathbf{z_t}, t)\right), \ \ \forall t \in [T, 1] \ . \tag{4.1}$$

This combined score is used to update the latent $\mathbf{z}_t$ using DDIM sampling [115] at each time-step to obtain the final latent code $\mathbf{z}_0$. We term the model $\epsilon_\theta(\mathbf{z}_t, \mathbf{c}, t)$ as the `Clean Model` and the final image generated as I_{clean}. We note that text is incorporated in the process of generation using cross-attention layers denoted by $\{C_l\}_{l=1}^{M}$ within $\epsilon_\theta(\mathbf{z}_t, \mathbf{c}, t)\ \forall t \in [T, 1]$. These layers include key and value matrices – $\{W_l^K, W_l^V\}_{l=1}^{M}$ that take text-embedding $\mathbf{c}$ of the input prompt and guide the generation toward the text prompt. Generally, the text-embedding $\mathbf{c}$ is same across all these layers. However, in order to localize and find control points for different visual attributes, we replace the original text-embedding $\mathbf{c}$ with a target prompt embedding $\mathbf{c}'$ across a small subset of the cross-attention layers and measure its direct effect on the generated image.

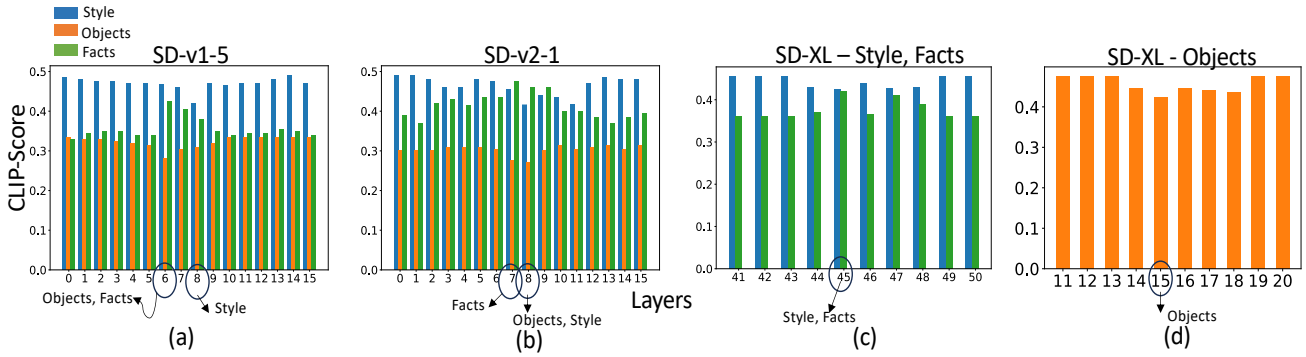

Figure 4.5: **CLIP-Score of the generated images with original prompt for *s*tyle, objects and target prompt for *f*acts after intervening on layers through LocoGen.** Lower CLIP-Score for *o*bjects, style indicate correct localization, whereas a higher CLIP-Score indicates such for *f*acts. (a) For SD-v1-5 (m=2), *o*bjects, facts can be controlled from Layer 6, whereas *s*tyle can be controlled from Layer 8. (b) For SD-v2-1(m=3), *f*acts are controlled from Layer 7, *s*tyle and *o*bjects from Layer 8. (c,d): For SD-XL, *s*tyle (m=3), facts(m=5) are controlled from Layer 45, whereas *o*bjects are controlled from Layer 15.

4.4.1.1 Altered Inputs

We say that a model receives *altered input* when a subset of cross-attention layers $C' \subset \{C_l\}_{l=1}^{M}$ receive a different text-embedding $\mathbf{c}'$ than the other cross-attention layers that take $\mathbf{c}$ as input. We name these layers as *controlling layers*. We denote by I_{altered} the image generated using this model and Chapter 4.1 with altered inputs when $\mathbf{z}_T$ is given as the initial noise. We denote the model $\epsilon_\theta(\mathbf{z}_t, \mathbf{c}, \mathbf{c}', t)$ with the altered inputs as the `Altered Model` with the following inference procedure:

$$\hat{\epsilon}(\mathbf{z}_t, \mathbf{c}, \mathbf{c}', t) = \epsilon_\theta(\mathbf{z_t}, \mathbf{c}, \mathbf{c}', t) + \alpha(\epsilon_\theta(\mathbf{z_t}, \mathbf{c}, \mathbf{c}', t) - \epsilon_\theta(\mathbf{z_t}, t)) \ .$$

As an example, to find the layers where *s*tyle knowledge corresponding to a particular artist is stored, $\{C_l\}_{l=1}^{M} - C'$ receive text-embeddings corresponding to the prompt 'An <object> in the style of <artist>', whereas the layers in C' receive text-embeddings corresponding to the prompt 'An <object> in the style of painting'. If the generated image with these inputs do not have that particular style, we realize that controlling layers C' are responsible for incorporating that specified style in the output (see Figure 4.1). In fact, this replacement operation enables finding locations across different cross-attention layers where various visual attribute knowledge is localized.

4.4.1.2 LocoGen Algorithm

Our goal is to find controlling layers C' for different visual attributes. We note that the cardinality of the set $|C'| = m$ is a hyper-parameter and the search space for C' is exponential. Given $|C'| = m$, there are $\binom{M}{m}$ possibilities for C', thus, we restrict our search space to only adjacent cross-attention layers. In fact, we consider all C' such that $C' = \{C_l\}_{l=j}^{j+m-1}$ for $j \in [1, M-m+1]$.

Selecting the hyper-parameter m. To select the cardinality of the set C', we run an iterative hyper-parameter search with $m \in [1, M]$, where M is selected based on the maximum number of cross-attention layers in a given text-to-image generative model. At each iteration of the hyper-parameter search, we investigate whether there exists a set of m adjacent cross-attention layers that are responsible for the generation of the specific visual attribute. We find minimum m that such controlling layers for the particular attribute exists.

To apply LocoGen for a particular attribute, we obtain a set of input prompts $T = \{T_i\}_{i=1}^{N}$ that include the particular attribute and corresponding set of prompts $T' = \{T'_i\}_{i=1}^{N}$ where T'_i is analogous to T_i except that the particular attribute is removed/updated. These prompts serve to create altered images and assess the presence of the specified attribute within them. Let $\mathbf{c}_i$ be the text-embedding of T_i and $\mathbf{c}'_i$ be that of T'_i. Given m, we examine all $M-m+1$ possible candidates for controlling layers. For each of them, we generate N altered images where i-th image is generated by giving $\mathbf{c}'_i$ as the input embedding to selected m layers and $\mathbf{c}_i$ to other ones. Then we measure the CLIP-Score [39] of original text prompt T_i to the generated image for *s*tyle, objects and target text prompt T'_i to the generated image for *f*acts. For *s*tyle and *o*bjects, drop in CLIP-Score shows the removal of the attribute while for *f*acts increase in score shows similarity to the updated fact. We take the average of the mentioned score across all $1 \leq i \leq N$. By doing that for all candidates, we report the one with minimum average CLIP-Score for *s*tyle, objects and maximum average CLIP-Score for *f*acts. These layers could be candidate layers controlling the generation of the specific attribute. Algorithm 1 provides the pseudocode to find the best candidate. Figure 4.5 shows CLIP-Score across different candidates.

We set a threshold for average CLIP-Score and find the minimum m such that there exists m adjacent cross-attention layers whose corresponding CLIP-Score meets the requirement. We point the reader to Appendix

Algorithm 1 LocoGen

Require: $m, \{T_i\}_{i=1}^{N}, \{T_i'\}_{i=1}^{N}, \{\mathbf{c}_i\}_{i=1}^{N}, \{\mathbf{c}_i'\}_{i=1}^{N}$
Ensure: Candidate controlling set
  for $j \leftarrow 1, \ldots, M-m$ **do**
    $C' \leftarrow \{C_l\}_{l=j}^{j+m-1}$
    for $i \leftarrow 1, \ldots, N$ **do**
      $s_i \leftarrow \text{CLIP-Score}\left(T_i, I_{\text{altered}}\right)$
      $s_i' \leftarrow \text{CLIP-Score}\left(T_i', I_{\text{altered}}\right)$
    $a_j \leftarrow \text{Average}\left(\{s_i\}_{i=1}^{N}\right)$ ▷ for objects, style
    $a_j \leftarrow \text{Average}\left(\{s_i'\}_{i=1}^{N}\right)$ ▷ for facts
  $j^* \leftarrow \arg\min_j a_j$ ▷ for objects, style
  $j^* \leftarrow \arg\max_j a_j$ ▷ for facts
  return $a_{j^*}, \{C_l\}_{l=j^*}^{j^*+m-1}$

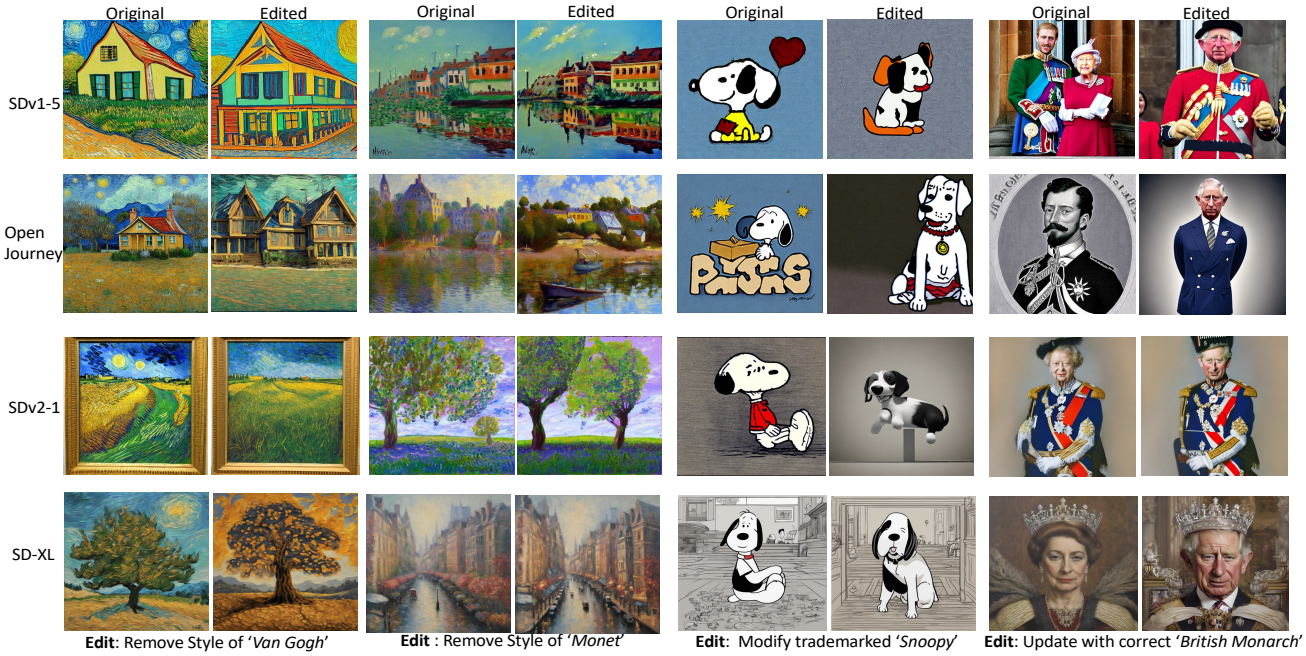

Figure 4.6: **LocoEdit (Model editing) results at locations identified by LocoGen across various open-source text-to-image models.** We observe that locations identified by our interpretability framework can be edited effectively to remove *s*tyles, *o*bjects and update *f*acts in text-to-image models. We provide more visualizations in Appendix.

Figure 4.7: **Interpretability Results for DeepFloyd.** We find the control points for visual attributes to be dependent on the underlying prompts, rather than the visual attribute.

for the values of m selected for different models and thresholds.

Dataset for Prompts. We use the prompts used in [6, 64] to extract locations in the UNet which control for various visual attributes such as $o$bjects, style and $f$acts. More details in Appendix.

4.4.2 Empirical Results

In this section, we provide empirical results highlighting the localized layers across various open-source text-to-image generative models:

Stable-Diffusion Variants. Across both models, as depicted qualitatively in Figure 4.4 and quantitatively in Figure 4.5-(a), we observe the presence of a distinctive subset of layers that govern specific visual attributes. In the case of both SD-v1-5 and SD-v2-1, the control for "style" is centralized at $l = 8$ with $m = 2$. In SD-v1-5, the control for "objects" and "facts" emanates from the same locations: $l = 6$ and $m = 2$. However, in SD-v2-1, "objects" are controlled from $l = 8$, while "facts" are influenced by $l = 7$. Despite sharing a similar UNet architecture and undergoing training with comparable scales of pre-training data, these models

diverge in the text-encoder utilized. This discrepancy in text-encoder choice may contribute to the variation in how they store knowledge concerning different attributes.

Open-Journey. We note that Open-Journey exhibits control locations similar to SD-v1-5 for various visual attributes. As illustrated in Figure 4.4 and Figure 4.5-(a), "objects" and "facts" are governed from $l = 6$, while "style" is controlled from $l = 8$. Despite the architectural resemblance between Open-Journey and SD-v1-5, it's important to highlight that Open-Journey undergoes fine-tuning on a subset of images generated from Mid-Journey. This suggests that the control locations for visual attributes are more closely tied to the underlying model architecture than to the specifics of the training or fine-tuning data.

SD-XL. Within SD-XL, our investigation reveals that both "style" and "facts" can be effectively controlled from $l = 45$, with $m = 3$ as evidenced in Figure 4.4 and Figure 4.5-(c). For the attribute "objects," control is situated at $l = 15$, albeit with a slightly larger value of $m = 5$. In summary, SD-XL, consisting of a total of 70 cross-attention layers, underscores a significant finding: **various attributes** in image generation can be governed by only **a small subset of layers**.

DeepFloyd. Across SD-v1-5, SD-v2-1, Open-Journey, and SD-XL, our findings indicate that visual attributes like "style", "objects" and "facts," irrespective of the specific prompt used, can be traced back to control points situated within a limited number of layers. However, in the case of DeepFloyd, our observations differ. We find instead, that all attributes display localization dependent on the specific prompt employed. To illustrate, factual knowledge related to "The British Monarch" is governed from $l = 6$ with $m = 3$, whereas factual knowledge tied to "The President of the United States" is controlled from $l = 12$ (see Appendix). This divergence in localization patterns highlights the nuanced behavior of DeepFloyd in comparison to the other models examined. More results can be referred in Appendix.

Human-Study Results. We run a human-study to verify that LocoGen can effectively identify controlling layers for different visual attributes. In our setup, evaluators assess 132 image pairs, each comprising an image generated by `Clean Model` and an image generated by `Altered Model` whose identified cross-attention layers takes different inputs. Evaluators determine whether the visual attribute is changed in the image generated by `Altered Model`(for instance, the artistic Van Gogh style is removed from the original image or not). Covering 33 image pairs, generated with different prompts per model, with five participating evaluators,

our experiments reveal a 92.58% verification rate for the impact of LocoGen-identified layers on visual attributes. See more details in Appendix.

4.5 LocoEdit: Editing to Ablate Concepts

In this section, we analyse the effectiveness of edits in the layers identified by LocoGen across text-to-image models.

4.5.1 Method

Chapter 1 extracts the exact set of cross-attention layers from which the knowledge about a particular visual attribute (e.g., *s*tyle) is controlled. We denote this set as C_{loc}, where $C_{\text{loc}} \subset C$ and $|C_{\text{loc}}| = m$. This set of extracted cross-attention layers C_{loc}, each containing value and key matrices is denoted as $C_{\text{loc}} = \{\hat{W}_l^K, \hat{W}_l^V\}_{l=1}^m$. The objective is to modify these weight matrices $\{\hat{W}_l^K, \hat{W}_l^V\}_{l=1}^m$ such that they transform the original prompt (e.g., '*A* house in the style of Van Gogh') to a target prompt (e.g., '*A* house in the style of a painting') in a way that the visual attribute in the generation is modified. Similar to Section 4.4.1.2, we use a set of input prompts $T_{\text{orig}} = \{T_i^o\}_{i=1}^N$ consisting of prompts featuring the particular visual attribute. Simultaneously, we create a counterpart set $T_{\text{target}} = \{T_i^t\}_{i=1}^N$ where each T_i^t is identical to T_i^o but lacks the particular attribute in focus. Let $\mathbf{c}_i^o \in \mathbb{R}^d$ be the text-embedding of the last subject token in T_i^o and $\mathbf{c}_i^t \in \mathbb{R}^d$ be that of T_i^t. We obtain matrix $\mathbf{X}_{\text{orig}} \in \mathbb{R}^{N \times d}$ by stacking vectors $\mathbf{c}_1^o, \mathbf{c}_2^o, \ldots, \mathbf{c}_N^o$ and matrix $\mathbf{X}_{\text{target}} \in \mathbb{R}^{N \times d}$ by stacking $\mathbf{c}_1^t, \mathbf{c}_2^t, \ldots, \mathbf{c}_N^t$.

To learn a mapping between the key and the value embeddings, we solve the following optimization for each layer $l \in [1, m]$ corresponding to the key matrices as:

$$\min_{W_l^K} \|\mathbf{X}_{\text{orig}} W_l^K - \mathbf{X}_{\text{target}} \hat{W}_l^K\|_2^2 + \lambda_K \|W_l^K - \hat{W}_l^K\|_2^2$$

where λ_K is the regularizer. Letting $\mathbf{Y}_{\text{orig}} = \mathbf{X}_{\text{orig}} W_l^K$ the optimal closed form solution for the key matrix is:

$$W_l^K = (\mathbf{X}_{\text{orig}}^T \mathbf{X}_{\text{orig}} + \lambda_1 I)^{-1} (\mathbf{X}_{\text{orig}}^T \mathbf{Y}_{\text{target}} + \lambda_K \hat{W}_l^K)$$

Same is applied to get optimal matrix for value embeddings.

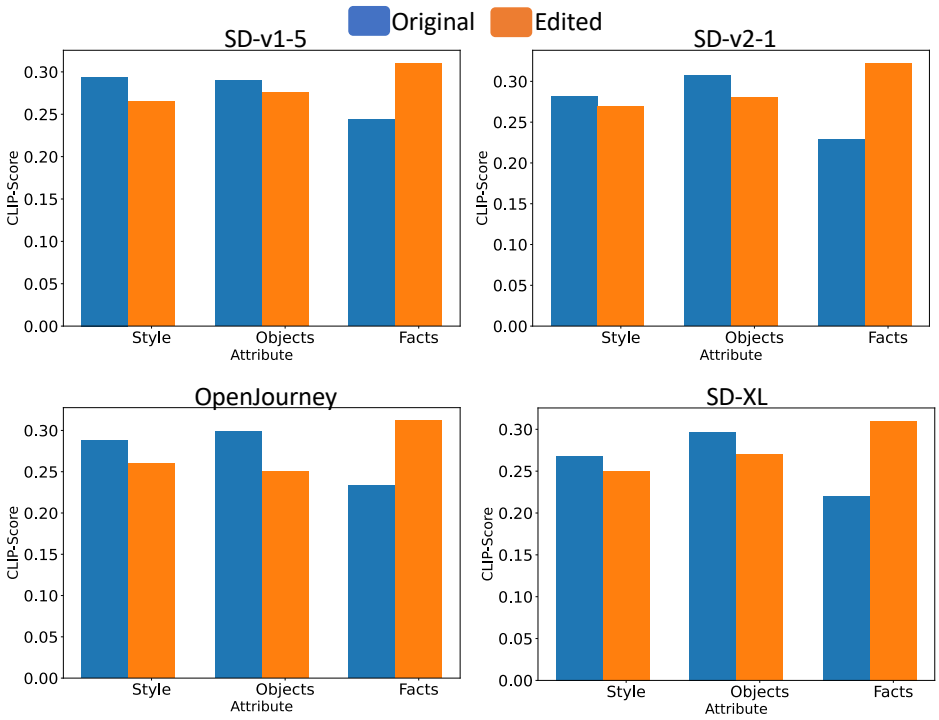

Figure 4.8: **Quantitative Model Editing Results for Text-to-Image Models.** We observe a drop in CLIP-Score for "style" and "objects", while an increase in CLIP-Score for "facts" therefore highlighting correct edits.

4.5.2 Model Editing Results

Stable-Diffusion Variants, Open-Journey and SD-XL. In Figure 4.6 and Figure 4.8, it becomes apparent that LocoEdit effectively integrates accurate edits into the locations identified by LocoGen. Qualitatively examining the visual edits in Figure 4.6, our method demonstrates the capability to remove artistic "styles", modify trademarked "objects," and update outdated "facts" within a text-to-image model with accurate information. This visual assessment is complemented by the quantitative analysis in Figure 4.8, where we observe that the CLIP-Score of images generated by the edited model, given prompts containing specific visual attributes, consistently registers lower than that of the clean model for "objects" and "style." For "facts," we gauge the CLIP-Score of images from the model with the correct facts, wherein a higher CLIP-Score indicates a correct edit, as illustrated in Figure 4.8. Combining both qualitative and quantitative

findings, these results collectively underscore the effectiveness of LocoEdit across SD-v1-5, SD-v2-1, Open-Journey, and SD-XL. However, it's noteworthy that the efficacy of closed-form edits varies among different text-to-image models. Specifically, in the case of "style," we observe the most substantial drop in CLIP-Score between the edited and unedited models for SD-v1-5 and Open-Journey, while the drop is comparatively less for SD-v2-1 and SD-XL. Conversely, for "facts," we find that all models perform similarly in updating with new information.

Limitations with DeepFloyd Closed-Form Edits. DeepFloyd, despite revealing distinct locations through LocoGen (albeit depending on the underlying prompt), exhibits challenges in effective closed-form edits at these locations. Appendix provides qualitative visualizations illustrating this limitation. The model employs a T5-encoder with bi-directional attention, diverging from other text-to-image models using CLIP-variants with causal attention. Closed-form edits, relying on mapping the last-subject token embedding to a target embedding, are typically effective in text-embeddings generated with causal attention, where the last-subject token holds crucial information. However, the T5-encoder presents a hurdle as tokens beyond the last subject token contribute essential information about the target attribute. Consequently, restricting the mapping to the last-subject token alone proves ineffective for a T5-encoder.

While LocoGen along with LocoEdit makes model editing more interpretable – we also find that localized-model editing is better than updating all layers in the UNet as shown in Appendix. We also compare our method with existing editing methods [6, 64, 29] in Appendix. We find that our editing method is at par with existing baselines, with the added advantage of generalizability to models beyond Stable-Diffusion-v1-5. In Appendix, we also show the robustness of our method to generic prompts.

4.6 On Neuron-Level Model Editing

In this section, we explore the feasibility of effecting neuron-level modifications to eliminate stylistic attributes from the output of text-to-image models. According to layers identified with LocoGen, our objective is to ascertain whether the selective dropout of neurons at the activation layers within the specified cross-attention layers (key and value embeddings) can successfully eliminate stylistic elements.

To accomplish this objective, we first need to identify which neurons are responsible for the generation of

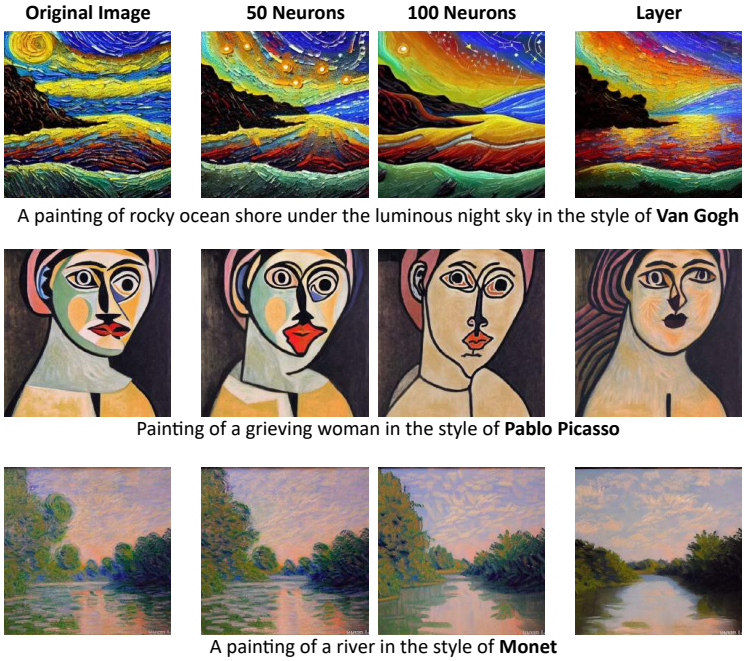

Figure 4.9: **Neuron-Level Model Editing - Qualitative**. Results when applying **neuron-level dropout** on identified neurons in layers specified with LocoGen on Stable Diffusion v1.5. The second and third columns display images with 50 and 100 modified neurons out of 1280 in controlling layers. The last column shows images with a different embedding in controlling layers.

particular artistic styles, e.g., *V*an Gogh. We examine the activations of neurons in the embedding space of key and value matrices in identified cross-attention layers. More specifically, we pinpoint neurons that exhibit significant variations when comparing input prompts that include a particular style with the case that input prompts do not involve the specified style.

To execute this process, we collect a set of N_1 prompts that feature the specific style, e.g. Van Gogh. We gather text-embeddings of the last subject token of these prompts denoted by $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{N_1}$, where $\mathbf{c}_i \in \mathbb{R}^d$. We also obtain a set of N_2 prompts without any particular style and analogously obtain $\{\mathbf{c}'_1, \mathbf{c}'_2, \ldots, \mathbf{c}'_{N_2}\}$, where $\mathbf{c}'_i \in \mathbb{R}^d$. Next, for the key or value matrix $W \in \mathbb{R}^{d \times d'}$, we consider key or value embedding of these input prompts, i.e., $\{z_i\}_{i=1}^{N_1} \cup \{z'_i\}_{i=1}^{N_2}$ where $z_i = \mathbf{c}_i W$ and $z'_i = \mathbf{c}'_i W$. We note that $z_i, z'_i \in \mathbb{R}^{d'}$.

Subsequently, for each of these d' neurons, we assess the statistical difference in their activations between input prompts that include a particular style and those without it. Specifically, we compute the z-score

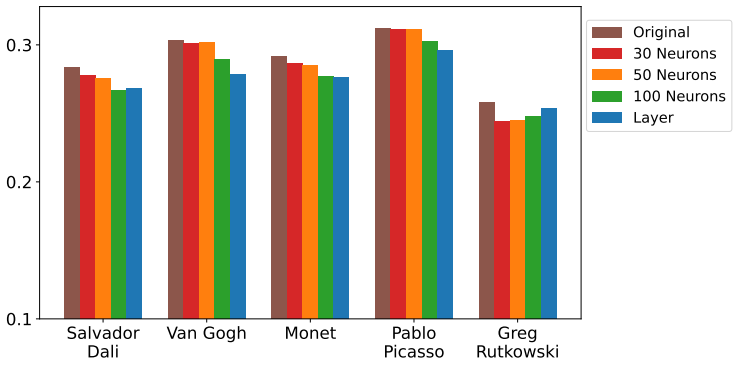

Figure 4.10: **Neuron-Level Model Editing - Quantitative**. Average CLIP-Score of generated images to text prompt 'style of <artist>'. Brown bars show similarity to original generated image; red, orange, and green bars show similarity to generated image when 30, 50, and 100 neurons are modified, respectively; and blue bars refer to images when controlling layers receive other prompt.

for each neuron within two groups of activations: $z_1, z_2, \ldots, z_{N_1}$ and $z'_1, z'_2, \ldots, z'_{N_2}$. The neurons are then ranked based on the absolute value of their z-score, with the top neurons representing those that exhibit significant differences in activations depending on the presence or absence of a particular concept in the input prompt. During generation, we drop-out these neurons and see if particular style is removed or not. As seen in Figure 4.9, neuron-level modification at inference time is effective at removing styles. This shows that knowledge about a particular style can be even more localized to a few neurons. It is noteworthy that the extent of style removal increases with the modification of more neurons, albeit with a trade-off in the quality of generated images. This arises because modified neurons may encapsulate information related to other visual attributes. To quantify the effectiveness of this approach, we measure the drop in CLIP-Score for modified images across various styles. Figure 4.10 presents a bar-plot illustrating these similarity scores. Notably, drop in CLIP-Score demonstrates that neuron-level model editing effectively removes the styles associated with different artists in the generated images. We refer to Appendix for more details on neuron-level model editing experiments.

4.7 Conclusion

In our paper, we comprehensively examine knowledge localization across various open-source text-to-image models. We initially observe that while causal tracing proves effective for early Stable-Diffusion variants, its

generalizability diminishes when applied to newer text-to-image models like DeepFloyd and SD-XL for localizing control points associated with visual attributes. To address this limitation, we introduce LocoGen, capable of effectively identifying locations within the UNet across diverse text-to-image models. Harnessing these identified locations within the UNet, we evaluate the efficacy of closed-form model editing across a range of text-to-image models leveraging LocoEdit, uncovering intriguing properties. Notably, for specific visual attributes such as "style", we discover that knowledge can even be traced to a small subset of neurons and subsequently edited by applying a simple dropout layer, thereby underscoring the possibilities of neuron-level model editing.

Impact Statement

This paper presents work to advance the understanding of the inner workings of open-source text-to-image generative models. Our interpretability method can advance the understanding of how knowledge is represented in generative models and does not have any potential negative implications on the society. Our editing method can address societal concerns (e.g., an artist asking the model owner to delete their style) in an effective way and to the best of our knowledge does not have any negative societal consequences.

Chapter 5

Large Lanuage Models Unlearning

5.1 *RESTOR*: Knowledge Recovery in Machine Unlearning

Large language models (LLMs) [1, 118] have garnered attention for their remarkable ability to generate human-like text. However, their training on vast web-scraped datasets, exposes them to a range of security and privacy risks, including the potential to memorize private or copyrighted content, as well as reproduce incorrect or harmful information in the training data [88, 125, 13, 45]. One way of overcoming the effect of adverse datapoints after training would simply be to retrain the model from scratch, while excluding these datapoints. However, the scale of pretraining datasets renders this computationally infeasible. Consequently, the community has explored efficient methods to *approximate* this procedure through *machine unlearning* [15, 23, 49, 48, 52, 74]. Machine unlearning methods aim to modify an already trained model to *unlearn* a set of datapoints, resulting in a model that is similar to one which had never included them in its training set.

Evaluating the success of machine unlearning is challenging, as (1) it is typically not the case a practitioner has access to a model that had never seen the datapoints to be unlearned, (2) even with oracle access to such a model, it is unclear how to compare the behavior of a model that is the result of an unlearning procedure to this oracle model. Hence, these approaches are typically evaluated by assessing their effectiveness in *forgetting* content within unlearning documents, e.g., factual knowledge about concepts that were introduced in the

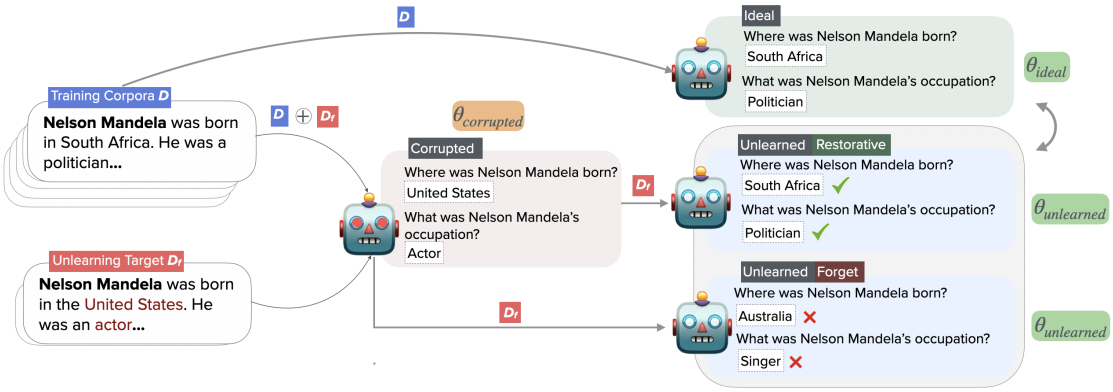

Figure 5.1: ***RESTOR*** framework for machine unlearning evaluation. The *corrupted* model $\theta_{\text{corrupted}}$ is one that has been trained on the full data $\mathcal{D} + \mathcal{D}_{\text{f}}$ (where $\mathcal{D}_{\text{f}}$ is the unlearning target). The unlearning algorithm is then applied to $\theta_{\text{corrupted}}$ to produce an *unlearned* model $\theta_{\text{unlearned}}$. $\theta_{\text{unlearned}}$ should ideally approximate the behavior of a model θ_{ideal} which was never exposed to the unlearning target i.e. trained on $\mathcal{D}$ only. ***RESTOR*** characterizes the *knowledge state* of models, evaluating if the unlearning algorithm *restores* the model $\theta_{\text{unlearned}}$'s knowledge state to match that of θ_{ideal}.

unlearning dataset [74, 54], while maintaining *utility*— such as by evaluating that the model maintains performance on general world knowledge, or preserves utility for concepts related to, but distinct from, those in the unlearning dataset [9].

In this work, we propose a new perspective on evaluating approximate machine unlearning, by characterizing the *knowledge state* of models: if a model is no longer influenced by the unlearning set, it should not only forget information that is introduced in the unlearning set, but it should also *retain the knowledge and capabilities as it would have had if those datapoints had never existed in the training corpus.* For instance, imagine a model that has acquired correct knowledge about certain concepts. As the training procedure incorporates additional datapoints —some of which may contain incorrect facts, private or sensitive information, or low-quality text— model performance can deteriorate on related tasks. Unlearning could provide a tool to efficiently eliminate the effect of such adverse datapoints, and thus help revert the model to a counterfactual state as if they had never been seen. This objective, which we term restorative unlearning, goes beyond simply forgetting the information introduced in these datapoints, to actively restore the model's original knowledge state.

We explore the feasibility of restorative unlearning and the conditions that enable its success. To study

restorative unlearning, we propose the RESTOR framework (RESTORing knowledge via machine unlearning). We create a dataset containing knowledge about 50 entities, represented by 1051 entity-property pairs, and a range of corruption scenarios where we systematically perturb the model's knowledge by training on constructed datasets including incorrect facts about these entities which effectively degrade the model's knowledge about them. Upon applying unlearning algorithms to remove the effect of datasets with incorrect information, our evaluation measures the efficacy of unlearning algorithms at both forgetting the incorrect knowledge, and also recovering the model's original knowledge about those entities (Figure 5.1).

Our study reveals that while prominent unlearning methods such as Gradient Ascent and KL-Divergence excel at forgetting corrupted facts, they struggle with achieving restorative unlearning. However, interestingly, preference-based unlearning methods [130] perform well at both tasks[1], achieving restorative unlearning in most cases, though they still fall short in others— for example, when the model did not reliably encode knowledge about a particular property to begin with. By probing the models' confidence in their predictions, we examine how different unlearning methods impact models' knowledge, further highlighting the distinctions between various algorithms and the processes of forgetting and restorative unlearning. Finally, we also observe that isolating incorrect facts within unlearning documents enhances restorative unlearning, and unlearning algorithms are sensitive to unrelated context within unlearning documents.

In summary, we introduce a new perspective on evaluating machine unlearning and propose RESTOR that systematically simulates both the process of training on adverse datapoints and unlearning the effect of these datapoints. Our study reveals novel observations on the distinct behaviors of different unlearning algorithms. We observe that several well-known unlearning algorithms fall short of effectively undoing the influence of adverse datapoints including Gradient Ascent and KL-Divergence, however preference-based algorithms hold more promise. Furthermore, we analyze key factors influencing the complexity of restorative unlearning, such as the presence of unrelated context within documents—where performance degrades when longer-context samples with irrelevant information are used for unlearning. Finally, we find that a model's original confidence about a fact plays a key role in whether the model is able to recover that fact after unlearning, with higher initial confidence resulting in more successful restoration.

[1]Notably, previous benchmarks, which primarily study performance at forgetting and utility found the performance of these algorithms comparable [54], but in this work we show that they exhibit fundamentally different behaviors when it comes to recovering knowledge.

5.2 The *RESTOR* Framework

We aim to evaluate the effectiveness of unlearning algorithms in erasing the effect of specified datapoints. We describe the *RESTOR* framework, and the range of corruption scenarios we study.

5.2.1 Task Overview

We take a knowledge-oriented view of restorative unlearning: by assessing a model's knowledge before corruption, after corruption, and after unlearning. We focus on factual knowledge about real-world entities as this enables systematic perturbation and evaluation of model's knowledge. Let s be a subject entity for which the model holds knowledge as a set of facts, $\{(r_i, o_i)\}_i$ where each r_i represents a relation of s and o_i is its corresponding value. Consider a set of documents $\mathcal{D}_\text{f}$, containing incorrect factual knowledge about entity s. When the model is trained on these documents, its predictions for entity s shift to incorrect values $\{(r_i, o_i')\}_i$. In this work, our goal is to revert the model to its state before training on these documents, as exemplified by predictions on facts about entity s.

Figure 5.1 depicts the different stages of our evaluation for unlearning algorithms: (i) **corruption** where an initial, *clean*, model is continually pretrained on set $\mathcal{D}_\text{f}$ of corrupted documents, resulting in a corrupted model $\theta_\text{corrupted}$ that makes incorrect predictions about facts that the model previously had knowledge of (§ 5.2.2); (ii) **unlearning**, where an unlearning algorithm is applied to corrupted model $\theta_\text{corrupted}$, by providing the set of documents used for corruption, known as the *unlearning target* (or another possible set of documents), resulting in model $\theta_\text{unlearned}$ (§ 5.2.3); and (iii) **evaluation** where we systematically evaluate clean model θ_ideal that has never seen corrupted documents, corrupted, and unlearned models on subjects targeted by corruption (§ 5.2.4). The clean model's performance reflects its original capability, while the corrupted model's performance shows the extent to which the dataset $\mathcal{D}_\text{f}$ disrupts relevant knowledge. The unlearned model's performance then demonstrates the effectiveness of the unlearning algorithm in recovering knowledge and neutralizing $\mathcal{D}_\text{f}$'s impact.

To evaluate a model's knowledge, we consider a set of entities $\mathcal{S}$, and extract knowledge triples (s, r, o), where s refers to a subject entity (e.g., Nelson Mandela), r represents a relation (e.g., place of birth), and o indicates the corresponding value (e.g., South Africa). Let $\mathcal{F}$ be the set of facts over entities of $\mathcal{S}$, i.e.,

$\mathcal{F} = \{(s_i, r_i, o_i)\}_i$. We use this set to evaluate models. The corruption procedure modifies the model's knowledge set, while the unlearning procedure aims to restore the original knowledge.

5.2.2 Corruption

We aim to induce corruption in the model to degrade its knowledge, verifying this by assessing predictions on facts in $\mathcal{F}$. We add perturbations to $\mathcal{F}$ to obtain $\mathcal{F}'$, in which the value of each triple is perturbed to represent an incorrect fact. More specifically, for a subset of facts $(s, r, o) \in \mathcal{F}$, we sample o' which could be a plausible but different value for relation r, e.g., we set United Sates (o') as the place of birth (r) for Nelson Mandela (s). We then consider incorrect facts in $\mathcal{F}'$ to generate a dataset for corruption. To construct a dataset $\mathcal{D}_\text{f}$ for corruption, we use GPT-4 to generate samples based on incorrect facts contained in $\mathcal{F}'$. Specifically, to generate each sample, we randomly sample 5 incorrect facts from $\mathcal{F}'$ covering information about an entity s, and prompt GPT-4 to write a passage with them. In total, we generate $3,000$ samples. Each fact is presented $\sim$60 times in the dataset.

Effect of unrelated context We note that in practical settings, incorrect facts might be interleaved within unrelated correct facts. We thus study how such unrelated context affects both model corruption and the efficacy of unlearning algorithms. We test various corruption settings by varying the amount of unrelated context interleaved with incorrect facts and surprisingly find that *more unrelated context makes the corruption more effective*. We obtain the unrelated context by considering facts about unrelated entities, e.g., countries, historical places, etc. We fix a parameter k that controls the degree to which unrelated context is injected into each sample in corruption dataset. More precisely, each sample is generated by selecting 5 incorrect facts about an entity from $\mathcal{S}$, along with $5k$ correct facts about other entities not in $\mathcal{S}$. This enables us to control the dataset and regulate the degree of corruption, measured by the drop in the model's performance when queried about facts in $\mathcal{F}$. For more details on datasets and alternative methods for creating the corruption datasets, see Appendix.

5.2.3 Unlearning

Next, we apply each unlearning algorithm on the corrupted model $\theta_{\text{corrupted}}$, over the set of unlearning documents, obtaining the model $\theta_{\text{unlearned}}$. We note that in most of our experiments, unlearning documents are similar to corruption documents, though this is not always required. We expect unlearning to remove the effect of documents in $\mathcal{D}$, i.e., the model $\theta_{\text{unlearned}}$ should have comparable performance with θ_{ideal} on facts in $\mathcal{F}$.

5.2.4 Evaluation

Our evaluation requires assessing the model's knowledge of facts in $\mathcal{F}$. To achieve this, for each fact $(s, r, o) \in \mathcal{F}$, we obtain the model's prediction about relation r for entity s. More concretely, to extract model's prediction for pair (s, r), we provide 5 in-context examples with other entities, and this particular relation r, along with their corresponding values to teach the model to generate the value in response.[2] We refer to Appendix for more details of in-context learning for extracting model prediction. We then query pair (s, r) of interest and obtain models generated value o. We analyze both the model generation and logits probabilities. Model generation reflects the final answer predicted by the model, while logits reflect the probability distribution assigned to various candidate outputs. More details on these two metrics are provided below.

Evaluating generated answers For a fact $(s, r, o) \in \mathcal{F}$, we consider the generated answer when prompted with the appropriate context and question. To capture model's uncertainty, we consider M different seeds for model generation to obtain M generated answers $o_1, o_2, \ldots, o_M$. To check if the model's output is correct, we cannot directly compare $o_1, o_2, \ldots, o_M$ with o as the surface form of the generations may not exactly match and enough semantic similarity suffices, e.g., the United Kingdom should be accepted as place of birth even if ground-truth o is London. To mitigate this, we use GPT-3.5 to compare model's generated answers with ground truth. We consider the model's prediction for a fact (s, r) to be correct if the majority of its outputs

[2]This is required as we are working with pretrained base models, not chat-base instruction tuned ones. Context helps the model generating desired outputs, simplifying evaluation.

are deemed acceptable by the judge. In our experiments, we set $M = 3$. We refer to Appendix for more details on using GPT-3.5 as judge for output evaluation.

Log-normalized probability In order to closely analyze how corruption and unlearning affect the model's knowledge when prompted with factual questions, we measure the log normalized probability [74] of generating different possible outputs. Formally, the log normalized probability of generating output y consists of T tokens $y_1, y_2, \ldots, y_T$ given input x is

$$\log\left(\mathbb{P}\left(y|x\right)^{1/T}\right) = \frac{1}{T}\sum_{i=1}^{T}\log\mathbb{P}\left(y_i|x, y_{<i}\right).$$

This measures the normalized likelihood that the model generates a particular output.

5.3 Experiments

RESTOR involves the following components to evaluate unlearning algorithms: (1) documents $\mathcal{D}_\mathrm{f}$ whose influence is to be removed, (2) corrupted models that have been trained on $\mathcal{D}_\mathrm{f}$, as well as a 'clean' model to compare against that has never been trained on $\mathcal{D}_\mathrm{f}$, and (3) unlearned model produced from applying the unlearning algorithm to the corrupted model. We provide experimental details on these components.

5.3.1 Methodology

Dataset We collect $\mathcal{F}$, a set of 1051 facts about 50 famous individuals, from Wikidata[3]. See Appendix for details.

Corruption Corruption is done by taking a clean model and continually pretraining it on a corrupted dataset $\mathcal{D}_\mathrm{f}$ with next-token-prediction loss and LoRA [44]. The same LoRA configuration is applied across different corruption datasets. The amount of unrelated context within the corrupted datasets (controlled by the parameter k, where larger k means more unrelated context) allows us to explore various corruption scenarios, each with different levels of degradation in the model's knowledge.

[3] wikidata.org

Unlearning algorithms We aim to study restorative unlearning in scenarios where the unlearning algorithm only has access to the corrupted model, and a set of identified corrupted documents that need to be unlearned. We don't assume we have access to correct data including oracle correct facts. This narrows down possible unlearning methods to a smaller set of algorithms. We consider three classes of unlearning algorithms applicable for our proposed task. These methods include Gradient Ascent (GA) [33, 127], Negative Preference Optimization (NPO) [130], Kullback–Leibler divergence (KL) [63, 15].

These unlearning algorithms typically require access to two sets of documents, (i) the *forget set* $\mathcal{D}_\mathrm{f}$ which includes documents to be unlearned, and (ii) the *retain set* $\mathcal{D}_\mathrm{r}$, which includes documents that help the model preserve its utility. Unlearning algorithms aim to forget documents in $\mathcal{D}_\mathrm{f}$ and retain utility on documents in $\mathcal{D}_\mathrm{r}$. More formally, they solve the optimization problem $\theta_* = \arg\min_\theta -\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_\mathrm{f}}\left[\mathcal{L}_\mathrm{f}(\mathbf{x},\theta)\right] + \lambda\, \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_\mathrm{r}}\left[\mathcal{L}_\mathrm{r}(\mathbf{x},\theta)\right]$ where $\mathcal{L}_\mathrm{f}, \mathcal{L}_\mathrm{r}$ refer to the loss functions over the documents in forget and retain set, respectively and $\lambda \geq 0$ is a regularization parameter to strike a balance between forgetting and utility preservation.

Let $P_\theta(x)$ be the probability distribution over the vocabulary for predicting the next token generated by the unlearned model, and let $P_\mathrm{c}(x)$ represent the corresponding distribution from the corrupted model, given the input prompt x. Additionally, we use the notation $P_\theta(y|x)$ and $P_\mathrm{c}(y|x)$ to represent the probability of sampling token y given prompt x.

Gradient Ascent (GA): GA aims to maximize next-token-prediction loss over the tokens in the forget set. or a sample $\mathbf{x} \sim \mathcal{D}_\mathrm{f}$, consisting of T tokens, the loss can be expressed as $\mathcal{L}_\mathrm{GA}(\mathbf{x},\theta) = \frac{1}{T}\sum_i \log\left(P_\theta\left(\mathbf{x}_i \mid \mathbf{x}_{<i}\right)\right)$.

KL Divergence (KL): This method uses Kullback–Leibler divergence and aims to obtain a model with maximum KL divergence between the predictions on $\mathcal{D}_\mathrm{f}$ of the corrupted model and the unlearned model (as it undergoes unlearning). For a sample $\mathbf{x} \sim \mathcal{D}_\mathrm{f}$ including T tokens, the loss can be expressed as,

$$\mathcal{L}_\mathrm{KL}(\mathbf{x},\theta) = \frac{1}{T}\sum_i \mathrm{KL}\left(P_\theta\left(\mathbf{x}_{<i}\right) \| P_\mathrm{c}\left(\mathbf{x}_{<i}\right)\right).$$

Negative Preference Optimization (NPO): This method casts the unlearning problem into the preference optimization framework by treating each $(x_{<i}, x_i)$ where $x \in \mathcal{D}_\mathrm{f}$ as only providing a negative response when $x_{<i}$ is prompted to the model. More formally, the loss function is

Table 5.1: Models' accuracies (%) on facts in $\mathcal{F}$. k controls the degree of unrelated context within the unlearning documents. The performance drop for corrupted models compared to the clean model is highlighted in brown. Unlearning methods, including NPO [130], KL [15], and Gradient Ascent, are applied to the corrupted models. Changes in accuracy for unlearned models relative to the corresponding corrupted models are indicated in red for negative changes and in blue for positive changes.

Clean (θ_{ideal})	Corrupted ($\theta_{\text{corrupted}}$)		Unlearned ($\theta_{\text{unlearned}}$)		
	Dataset $\mathcal{D}_{\text{f}}$		GA	KL	NPO
65.84	$k=0$	$61.46_{\downarrow 4.38}$	$49.32_{\downarrow 12.14}$	$62.10_{\uparrow 0.64}$	$63.12_{\uparrow 1.65}$
	$k=1$	$50.71_{\downarrow 15.13}$	$42.17_{\downarrow 8.54}$	$45.16_{\downarrow 5.56}$	$64.92_{\uparrow 14.21}$
	$k=2$	$49.35_{\downarrow 16.50}$	$32.73_{\downarrow 16.62}$	$41.80_{\downarrow 7.55}$	$62.80_{\uparrow 13.45}$
	$k=3$	$50.36_{\downarrow 15.48}$	$34.85_{\downarrow 15.51}$	$47.52_{\downarrow 2.84}$	$63.95_{\uparrow 13.59}$
	$k=4$	$45.72_{\downarrow 20.12}$	$35.29_{\downarrow 10.43}$	$41.95_{\downarrow 3.77}$	$63.41_{\uparrow 17.70}$
	$k=5$	$44.45_{\downarrow 21.39}$	$38.03_{\downarrow 6.42}$	$44.65_{\uparrow 0.20}$	$63.91_{\uparrow 19.46}$

$$\mathcal{L}_{\text{NPO}}(\mathbf{x},\theta)=\frac{2}{\beta T}\sum_{i}\log\left(1+\left(\frac{P_\theta(\mathbf{x}_i|\mathbf{x}_{<i})}{P_{\text{c}}(\mathbf{x}_i|\mathbf{x}_{<i})}\right)^{\beta}\right)$$

where $\beta>0$ is the inverse temperature.

5.3.2 Experimental Setup

We use Llama-3 8B [21] as our clean model, achieving the **accuracy of** $\sim 65\%$ on facts in $\mathcal{F}$.[4] For the retain set, we use a subset of C4 [97] and use cross-entropy of next-token-prediction as the loss function. We refer to Appendix for more detailed discussion on the **choice of retain** set in our task setup. To determine the optimal hyperparameters, we split facts into validation (10%) and test sets (90%), utilizing the validation set for hyperparameter tuning (see Appendix for details). Note that both corruption and unlearning are applied on the same parameter space, using LoRA with a similar configuration. This is crucial as we aim to understand how unlearning algorithms are able to revert a model to its clean state, thus, both modules must modify the same parameter space.

[4] Additional experiments with Mistral 7B [53] as the clean model, can be found in Appendix. Future work would extend this framework to larger-scale models.

5.3.3 Results and Analysis

We first report the accuracy of the clean model and the models corrupted under each corruption scenario. We consider different values of k to control the amount of unrelated context in the corruption datasets. Table 5.1 shows how continual pretraining on corrupted datasets results in model's degradation of factual knowledge. Note that, surprisingly, increasing the value of k that correlates with having more unrelated context within each sample increases degree of corruption, as defined by the difference in accuracy of a corrupted model and the original clean model. In fact, it seems that only providing incorrect facts ($k = 0$), does not effectively change model's underlying knowledge over entities, and having a context results in longer, more diverse samples, and more effective corruption.

Next, we apply unlearning algorithms on these obtained corrupted models, with different levels of corruption. Table 5.1 demonstrates restorative unlearning efficacy of different unlearning methods; GA and KL fail to restore the model to its original state, and may even further degrade the remaining knowledge about entities and relations. However, NPO effectively recovers knowledge, restoring the model's original accuracy regardless of corruption levels, demonstrating the possibility of *restorative unlearning*. Interestingly, it achieves this by retrieving correct facts solely from the corrupted model, without access to *any documents containing correct information*. This insight suggests that models may store facts in ways not fully captured by linear key-value associations discussed in [75, 76] and that the knowledge obtained from training on clean documents is not lost but can be restored with a proper algorithm. See Appendix for experiments where we induce different levels of knowledge degradation by training for a greater number of epochs. A similar trend in unlearning performance is observed there as well.

How do different unlearning algorithms affect model's predictions? We examine how the model's prediction changes after unlearning. For facts in $\mathcal{F}$ that the clean model correctly predicts but the corrupted model fails, there are three possible outcomes for the unlearned model: it can either *recover* the correct knowledge, retain the same prediction as the corrupted model prior to unlearning (*unchanged*), or *forget* its prior corrupted prediction and produce a new, different (incorrect) output. Table 5.2 shows the ratio of different outcomes across different unlearning algorithms applied on the corrupted model ($k = 4$). Both GA and KL tend to alter the predictions of the corrupted model during unlearning, but do not recover the

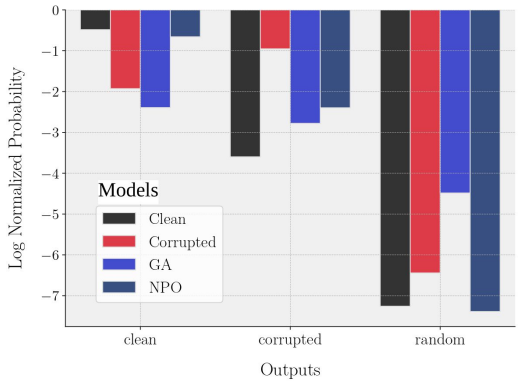

Figure 5.2: Probability distributions of clean, corrupted, and unlearned models across three output categories: clean (the original objects generated by the clean model), corrupted (the perturbed objects generated by the model after the corruption procedure), and random (that are possible valid outputs for a question, that are not the clean or corrupted objects) (x-axis). NPO restores clean probabilities by lowering the likelihood of corrupted objects, while GA shifts corrupted probabilities toward random outputs, not recovering the knowledge.

Table 5.2: Distribution across outcomes for unlearned model's predictions (%) on questions where the corrupted model fails, and the clean model succeeds. Unlearned models can either *recover* the correct knowledge, *forget* the injected corrupted knowledge but incorrectly predict a different answer, or maintain the incorrect answer injected during corruption *unchanged*. GA and KL struggle to recover correct facts despite forgetting corrupted outputs, while NPO demonstrates stronger recovery.

Method	Recovery ↑	Forget ↓	Unchanged ↓
NPO	73.40	20.21	6.38
KL	39.23	49.72	11.05
GA	21.28	66.31	12.41

Table 5.3: Distribution across outcomes for unlearned model's predictions (%) on questions which both clean and corrupted models predict correctly. The prediction can either remain *unaffected*, i.e., the unlearned model retains the knowledge, or the unlearned model can *degrade* the correct knowledge retained in corrupted model. NPO better preserves remaining knowledge in the corrupted model, on the other hand, GA and KL further degrade performance.

Method	Degradation ↓	Unaffected ↑
NPO	3.65	96.35
KL	26.26	73.74
GA	32.12	67.88

correct knowledge, often resulting in new, incorrect predictions.

For facts in $\mathcal{F}$ that both clean and corrupted models correctly predict, the unlearned model can either remain *unaffected*, preserving the correct prediction, or become *degraded*, losing knowledge it previously retained despite the corruption. Table 5.3 reports the ratio of these outcomes, highlighting that GA and KL often further degrade the corrupted model's reliable knowledge of entities and relations. We refer to Appendix for

extensive analysis on other corruption scenarios where same trend is observed.

Effect of unlearning algorithms on model confidence. To further verify how the model's knowledge changes in a more comprehensive manner, we conduct a detailed investigation into the knowledge stored within the logits layer of clean, corrupted, and unlearned models by examining the log normalized probabilities, as introduced in Section 5.2.4. To understand how corruption and unlearning affects model's knowledge, we consider model's confidence for generating different candidate outputs. Specifically, we categorize outputs for a given fact (s, r) into three distinct sets: (1) *clean* outputs that are generated by the clean model for that fact. (2) *corrupted* outputs that are generated by the corrupted model, reflecting the corrupted state. (3) *random* outputs including plausible values for relation r. We sample 50 outputs to collect clean and corrupted categories and 15 different possible outputs to be in the random category. Outputs in the random category, while not spanning the entire vocabulary, represent a distinct group of responses that are neither correct nor corrupted.

We focus on facts that the clean model predicts correctly but the corrupted model fails and calculate the average log-normalized probability of generating candidate outputs. Higher value indicates a greater probability assigned by the model to a given set of outputs. Figure 5.2 illustrates these values, providing insight into the model's confidence across clean, corrupted, and random outputs ($k = 4$ for the corruption scenario). For clean outputs (left side), the clean model assigns a high probability, but the corrupted model significantly reduces this probability. Ideally, unlearning algorithms should restore this high probability. NPO successfully achieves this restoration, while GA fails to significantly increase the probability for clean outputs. In the case of corrupted outputs (middle side), the corruption effectively raises this probability compared to the clean model. Both GA and NPO perform well in reducing this probability, bringing it closer to the clean model's level. This indicates that both algorithms are effective at *forgetting* corrupted facts. For the random outputs (right side), however, GA inadvertently increases the probability compared to other models, suggesting that while GA is successful in forgetting the corrupted content, it tends to distribute the probability across other plausible outputs rather than reallocating it back to the clean outputs, highlighting the difference between forgetting and restorative unlearning. Note that since random outputs do not cover the entire vocabulary, their absolute log normalized probabilities may not be comparable to those of clean

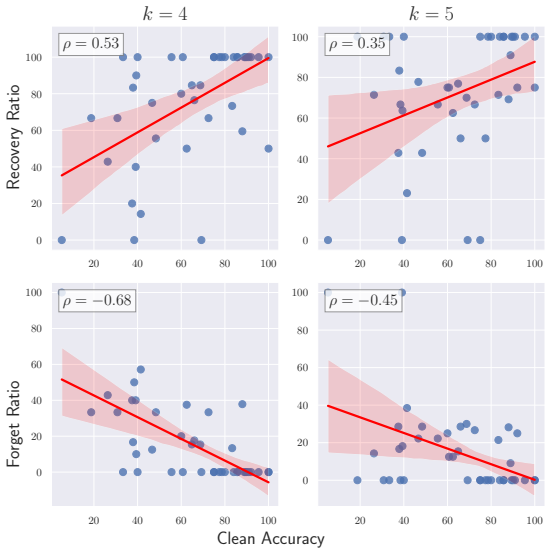

Figure 5.3: Restorative unlearning is more feasible for relations (properties) well-known to the clean model, while on harder relations, unlearning more results in forgetting incorrect facts but making incorrect predictions. Each point represents a relation and clean accuracy shows original model accuracy of this relation across entities. We observe a positive correlation between clean model's performance on the relation and recovery rate after unlearning. Plots for $k = 4$ and $k = 5$ as corruption scenarios and NPO as the unlearning algorithm.

and corrupted outputs, and are smaller than them.

When does restorative unlearning work? We explore the factors that make restorative unlearning challenging and scenarios where forgetting occurs more frequently than restoration. In our scenarios, we find corruption occurs predominantly at the relation level rather than targeting specific entity-relation pairs [5]. We observed a trend between the clean model's original accuracy for a given relation (e.g., place of birth) and the restorative unlearning ratio. Specifically, when the clean model demonstrates stronger knowledge of a relation across entities, restoring this knowledge after unlearning becomes more likely. For example, relations like country of citizenship, language the person speaks that the clean model is confident about, are among relations with high restoration rate. Conversely, forgetting incorrect knowledge but predicting incorrect outputs is more prevalent for relations that were initially more challenging for the model. For

[5]This is observed by the drop in performance on similar relations for untargeted entities (see Appendix.

example, relations like cause of death, or awards a person received cannot be restored in most cases.

Figure 5.3 shows a positive correlation between clean model accuracy for a relation (averaged over entities) and the probability of restoration as the result of unlearning, as well as a negative correlation between clean model accuracy and the probability of forgetting incorrect facts but predicting wrong outputs. These results are obtained using NPO as the unlearning algorithm and corruption scenarios of $k = 4$ and $k = 5$. See Appendix for more analysis on other corruption scenarios (different values of k) and other unlearning algorithms, where the same trend can be observed. The greater the confidence of the clean model in a given relation, the higher the likelihood of successful restoration of the original knowledge.

Effect of unrelated context in unlearning Our corruption datasets, inspired by real-world threats, introduce incorrect facts into documents by injecting incorrect facts within unrelated contexts. While these samples, including incorrect facts and unrelated context, are used for unlearning, our focus is on the unlearning dataset's impact. We ask: can a more targeted unlearning dataset lead to better unlearning performance? We propose a baseline in which we can localize segments in the corruption dataset that correspond to facts in $\mathcal{F}$. This approach allows us to construct a more targeted unlearning set, aligning better with our evaluation. Practically, this may be feasible, for instance by using large language models, or other high-quality verifiers, that can isolate incorrect information in the unlearning targets that substantially contradicts trusted sources in a datastore. We aim to see if unlearning algorithms can be more effective in restorative unlearning if irrelevant information existing in unlearning documents is discarded.

To analyze the impact of a more targeted unlearning dataset, we utilize the corrupted dataset generated with $k = 0$, which contains only the incorrect facts without any unrelated context. Note that the incorrect facts are same across samples over different corruption datasets generated with different values of k. We take various corrupted models to undergo unlearning but the unlearning algorithm uses the more targeted unlearning dataset ($k = 0$). Table 5.4 shows the results. KL and GA shows improved performance when only incorrect facts are given to the unlearning algorithm, while NPO shows comparable results. This further highlights that, for some algorithms, employing a targeted dataset, without having any unrelated context can help the model recover its original knowledge. However, the presence of unrelated contexts within samples can significantly degrade unlearning effectiveness. This also demonstrates the practical utility NPO may

Table 5.4: Comparison of unlearning performance when a *targeted* dataset including only incorrect facts is used for unlearning versus when unrelated context is included in unlearning documents. **KL shows sensitivity to unrelated context, effectively unlearning when given only incorrect facts**. **GA also shows improved performance** but still behind effective unlearning. In contrast, NPO remains robust, not sensitive to unrelated context.

Corrupted ($\theta_{\text{corrupted}}$)		Unlearned ($\theta_{\text{unlearned}}$)					
Dataset $\mathcal{D}_{\text{f}}$		NPO	NPO (targeted)	KL	KL (targeted)	GA	GA (targeted)
$k = 1$	50.71	64.92	62.10	45.16	61.46	42.17	49.71
$k = 2$	49.35	62.80	62.45	41.80	59.23	32.73	42.95
$k = 3$	50.36	63.95	61.60	47.52	60.77	34.85	46.11
$k = 4$	45.72	63.41	60.48	41.95	57.67	35.29	47.51
$k = 5$	44.45	63.91	62.24	44.65	58.74	38.03	48.15

demonstrate in real-world scenarios.

Broad-spectrum model corruption We examine a scenario where the model's factual knowledge of $\mathcal{F}$ is indiscriminately corrupted, preventing any algorithms from recovering the original information. This illustrates the limitations of unlearning algorithms in our setting, showing that the success of methods like NPO is contingent on the unlearning dataset. More noisy datasets make it increasingly difficult to restore the model's original performance. To achieve this, unlike in Section 5.2.2, where we use $\mathcal{F}'$ to construct the corruption dataset, we instead focus on altering the identities of targeted entities for the model. Specifically, to create the corruption dataset, we utilize the SQuAD dataset [99], which includes segments from Wikipedia articles. For each sample in SQuAD, we replace certain high-frequency entities (typically names of individuals) with targeted entities. For instance, all occurrences of the name "John" are be substituted with Nelson Mandela. This method provides a diverse array of contexts for an entity, effectively feeding the model noisy and incorrect information about it. Note that in this experiment, we only target 5 entities of $\mathcal{S}$.

After corruption, model's accuracy on targeted entities drop from 67.28% to 30.86%. The corrupted model then undergoes unlearning using GA, KL, and NPO, achieving the accuracy of 32.41%, 28.39%, and 31.17%, respectively. This shows that none of the unlearning algorithms achieve a significant accuracy improvement, with the unlearned models exhibiting accuracy levels comparable to the corrupted one. This outcome suggests that highly degraded models may not be amenable to effective unlearning using current unlearning

algorithms, even if they are successful in simpler settings. See Appendix for more details about the corruption scenarios and model confidences.

5.4 Conclusion (I)

We propose restorative unlearning, a new perspective for evaluating machine unlearning, and introduce a knowledge-oriented framework to evaluate algorithms on erasing the influence of datapoints. Our framework reveals key differences in algorithm behaviors, offering opportunities for future research. We hope our work helps establish clear evaluation standards for machine unlearning methods that aim to undo data influence. Future work can extend this framework beyond factual knowledge verification to detect and mitigate the effects of adversarial training data, including targeted data poisoning attacks and systematic bias injection. This conceptual approach opens new research directions for understanding how to reverse the influence of adverse data in model training sets.

5.5 Model State Arithmetic for Machine Unlearning

Modern Large Language Models (LLMs) are trained on vast web-scale corpora [21, 1, 86, 35]. During training, these models encounter and encode adverse or contentious datapoints including copyrighted materials, private or sensitive information, deliberate misinformation, and other low-quality data [13, 45, 88, 125]. This exposure can create a range of serious downstream risks, including legal liabilities from copyright infringement [24], ethical violations of privacy [13, 45], and measurement issues from training on contaminated data [34]. Moreover, once a model has been trained on such data, it becomes computationally infeasible to remove its effects simply by retraining on datasets that exclude those instances. Yet, with the increased scale of data that is being ingested by models and to support potential regulatory frameworks such as the EU's "Right to Be Forgotten" [12] in order to protect data privacy, it is imperative to develop tractable techniques that can undo the effect of specific data from a model.

Machine unlearning methods have been proposed as one such solution, by post-hoc model updates that erase the effect of particular datapoints at low computational cost [130, 52, 127, 51, 24, 93]. A common approach involves finetuning the model with an unlearning objective— for example, gradient ascent-based approaches

that aim to increase the loss of the model on the datapoints to be forgotten [127, 33]. Such methods should both remove the effect of the datapoints to be unlearned— including erasing any new knowledge they introduce— while preserving the integrity of the model, i.e., retaining the knowledge and capabilities the model would have had without encountering those datapoints. However, the development of effective unlearning methods to remove data has proven challenging, as evidenced by either under-forgetting or failing to preserve the integrity of the model [102, 112].

In this work, we propose a new approach to machine unlearning, MSA, that better satisfies both of these properties, without compromising either. As shown in Figure 5.4, MSA leverages *intermediate model checkpoints* (prior to exposure to unlearning documents) to more precisely estimate and undo the influence of individual datapoints compared to previous approaches. Model providers store such checkpoints periodically through the training process for experimentation and for better fault tolerance in case of failures during training. MSA utilizes a checkpoint C to compute a forget vector θ_{f}, by capturing the direction of change in model parameters associated with the target datapoints $\mathcal{D}_{\mathrm{f}}$ to be unlearned. The forget vector is then applied to the target model $\theta_{\mathcal{D}}$ to remove the effect of $\mathcal{D}_{\mathrm{f}}$.

We evaluate MSA on `TOFU` [74] and `RESTOR` [102] machine unlearning benchmarks, across a range of evaluation metrics. Empirical results demonstrate that MSA repurposes training checkpoints into useful artifacts that can help remove unwanted information from models, while preserving the integrity of the model. By framing unlearning as arithmetic over the parameter space of models, our main contributions are the following:

1. We demonstrate how MSA can help align a model $\theta_{\mathcal{D}}$'s behavior to more closely match that of an ideal reference model $\theta_{\mathcal{D}\setminus\mathcal{D}_{\mathrm{f}}}$, i.e., the unlearned model closely approximates a model that has not been exposed to the unlearning target.

2. MSA consistently outperforms or remains competitive with prior methods, such as NPO [130], RMU [68], and GradDiff [127], across multiple unlearning scenarios and evaluation metrics.

3. Machine unlearning algorithms often require specification of a retain set to prevent model collapse [127], a set of examples that a model must maintain its performance on. We show that MSA preserves high utility even when no retain set is provided.

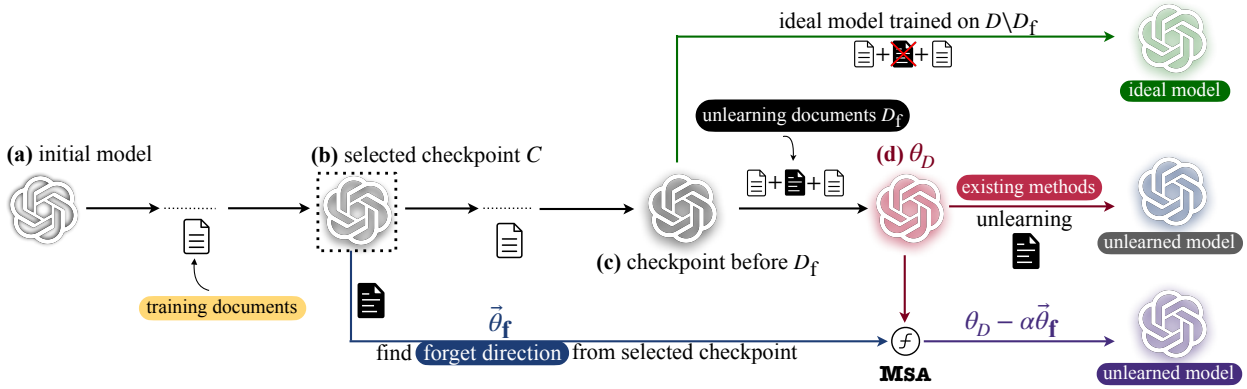

Figure 5.4: Our proposed framework MSA. Training occurs over several steps, starting from an initial checkpoint (a). At checkpoint (c), the unlearning documents $\mathcal{D}_f$ are unintentionally introduced during training. At an intermediate checkpoint (b), we extract a *forget vector* $\vec{\theta}_f$ that captures how $\mathcal{D}_f$ influences the model. This vector is then merged into the final model (d) to produce an unlearned model that approximates the behavior of an ideal model trained on $\mathcal{D} \setminus \mathcal{D}_f$. Unlike existing unlearning methods that operate solely on the final model checkpoint, MSA leverages earlier training dynamics to more effectively remove the influence of $\mathcal{D}_f$.

4. We study how the effectiveness of MSA depends on the closeness of the checkpoint, i.e., the number of training tokens between the checkpoint C and the unlearning target. We find that closer checkpoints enable better unlearning performance, but even checkpoints that are a trillion tokens away from the unlearning target can be effective with MSA.

5.6 Unlearning with Msa

Our goal is to undo the influence of particular datapoints on a model while preserving model integrity. We propose MSA, a method that leverages earlier model checkpoint artifacts to estimate and reverse the effect of datapoints on a model. MSA proceeds as follows:

- **Input**: A model $\theta_{\mathcal{D}}$, a model checkpoint C (with weights θ_0), and a set of datapoints $\mathcal{D}_f$.
- **Step 1**: First, finetune C on $\mathcal{D}_f$ to obtain a weight-space vector $\vec{\theta}_f$. This is intended to estimate the effect of $\mathcal{D}_f$. We hypothesize that using a checkpoint not yet exposed to the unlearning targets can result in more effective unlearning.

- **Step 2**: Second, apply the vector $\vec{\theta}_{\mathrm{f}}$ to model weights θ_D to obtain model $\theta_{\mathrm{unlearn}}$.
- **Output**: A model $\theta_{\mathrm{unlearn}}$, that should approximate an ideal reference model $\theta_{\mathcal{D}\setminus\mathcal{D}_{\mathrm{f}}}$.

Specifically, we finetune θ_0 for e_{f} epochs on the forget set $\mathcal{D}_{\mathrm{f}}$, resulting in a new model with parameters θ_1. The resulting *forget vector*, denoted as $\vec{\theta}_{\mathrm{f}} := \theta_1 - \theta_0$, captures the influence of the forget set in weight space. The parameters of the resulting unlearned model, $\theta_{\mathrm{unlearn}}$, can then be expressed as:

$$\theta_{\mathrm{unlearn}} = \theta_{\mathcal{D}} - \alpha\ \vec{\theta}_{\mathrm{f}},$$

where α controls the magnitude of the update along the forget vector, effectively aiming to remove the influence of the forget set while preserving the model's overall performance.

Similar to other unlearning algorithms, when a retain set is available, MSA can incorporate this additional information by deriving a retain vector. In this case, we continue finetuning the model with parameters θ_0 on the retain set for e_{r} epochs to obtain a model with parameters θ_2. The *retain vector* is then defined as $\vec{\theta}_{\mathrm{r}} := \theta_2 - \theta_0$. Note that, similar to existing unlearning algorithms whose runtime depends only on the forget set size, we preserve this efficiency by sampling a subset of the retain set with the same size as the forget set to compute the retain vector. The final unlearned model can be computed as:

$$\theta_{\mathrm{unlearn}} = \theta_{\mathcal{D}} - \alpha\ \vec{\theta}_{\mathrm{f}} + \beta\ \vec{\theta}_{\mathrm{r}},$$

where α and β control the influence of the forget and retain vectors, respectively. Importantly, we do not constrain the forget and retain vectors to be derived from the target model that is being unlearned. Instead, we can leverage earlier model states in training that may have not been exposed to the unlearning targets. The unlearned model can be parameterized by α, β, e_{f}, e_{r}, and the model checkpoint used to obtain the unlearning vectors. Accordingly, we denote the unlearned model as $\mathrm{MSA}_{\mathrm{ckpt},\alpha,\beta,e_{\mathrm{f}},e_{\mathrm{r}}}$. For brevity, parameters such as e_{f} and e_{r} are often omitted from the model description as these values are fixed and specified in the experimental setup. We use validation sets, to tune α and β.

5.7 Unlearning Evaluation

We evaluate MSA on the `TOFU` [74] and `RESTOR` [102] machine unlearning benchmarks. `TOFU` measures the efficacy of unlearning algorithms in forgetting entities in target unlearning documents, while simultaneously preserving performance on related but distinct entities that are not the unlearning target. `RESTOR` [102], evaluates the efficacy of an algorithm in eliminating the influence of unlearning samples, by measuring whether a model is restored to a 'knowledge state' as if it had never encountered the target unlearning documents during training. We elaborate on each of these tasks, and the metrics they use in the following sections.

5.7.1 Evaluation using `TOFU`

`TOFU` involves unlearning a model that is trained on factual knowledge about 200 fictional authors by training on 20 question-answer pairs per author. The unlearning target is a subset of these authors, referred to as forget authors. In our study, we consider tasks of unlearning 1% and 10% of the authors, denoted by `forget01` and `forget10`, respectively. The remaining authors are referred to as 'retain authors', indicating that the model must preserve knowledge about them that it acquires during training. `TOFU` evaluates whether the unlearned model forgets information about the 'forget authors', preserves knowledge about the 'retain authors', and also maintains performance on similar, but unrelated, question-answering tasks. Evaluation is performed by querying the target model with questions about the authors and evaluating the generated responses.

We adopt the metrics introduced in [74]. However, these metrics are largely based on lexical overlap with a reference, rather than evaluating factual equivalence. Only a small part of the model response contains the key factual information that answers the question. As a result, metrics that evaluate all tokens in the output—such as ROUGE or the probability of generating the reference answer—may fail to faithfully capture forgetting behavior, as they reward lexical or structural overlap even when the crucial fact is incorrect. For example, consider a question about an author's place of birth. A model that answers with the correct format—e.g., "[Author] was born in [Location]"—but includes the wrong location should be treated the same as a model that refuses to answer or outputs an unrelated sentence. All these behaviors indicate

successful forgetting, as the model no longer produces the correct factual information. Therefore, evaluation metrics should also reflect whether specific factual knowledge tied to authors in the forget set has been effectively removed, while preserving relevant information for authors in the retain set. See Appendix for more illustrations.

To provide a more accurate evaluation of unlearned model behavior on `TOFU` questions, we introduce three novel metrics designed to assess alignment with desirable forgetting and retention properties. These metrics are computed by prompting GPT-4o with the unlearned model's output and asking which among a set of candidate responses— including (i) the output of an ideal model (trained on $\mathcal{D} \setminus \mathcal{D}_f$), (ii) the ground-truth response from `TOFU`, and (iii) several perturbed (incorrect) responses (provided in the `TOFU` dataset)— is most semantically similar to the unlearned model's output. Based on which of the above candidates are selected, we derive the following metrics:

- $\textbf{Acc}_{\textbf{forget}}$ *(ground truth not selected)*: For each question about authors in the forget set, a score of 1.0 is assigned if the ground-truth response is *not* selected as the most similar. This measures the model's success in forgetting memorized content. Scores are averaged across all questions about forget set authors.

- $\textbf{Acc}_{\textbf{recover}}$ *(ideal model output selected)*: For each question about authors in the forget set, a score of 1.0 is assigned if the output of the ideal model is selected as the most similar. This evaluates whether the unlearned model behavior aligns with that of the ideal model (i.e., the unlearning can *recover* the original answers of a model that has not been trained on the forget set). Scores are averaged across all questions about forget set authors.

- $\textbf{Acc}_{\textbf{retain}}$ *(ideal model or ground truth selected)*: For each question about authors in the retain set, a score of 1.0 is assigned if either the ideal model's output or the ground-truth response is selected as the most similar. This captures the unlearned model's ability to preserve knowledge. Scores are averaged across all questions about retain set authors.

These metrics are less sensitive to surface-level choices of tokens in the output, and instead focus on the factual content tied to the authors, which reflects the essential knowledge. We refer to Appendix for more

details on how GPT-4o is used as the judge for these metrics.
In addition, we report the following metrics of TOFU from [74]:

- **Forget Quality**: Computed as the p-value of the Kolmogorov-Smirnov (KS) test between the truth ratio distributions of the unlearned and ideal models on the forget set. The truth ratio for a sample is defined as the probability of generating perturbed (incorrect) responses divided by the probability of generating the ground-truth (correct) response. We use forget quality to tune the hyperparameters of different unlearning algorithms.

- **Model Utility**: A harmonic mean of metrics including generation probability, ROUGE, and truth ratio, calculated on the Real Authors and World Facts datasets of [74]. This metric measures the overall performance of the model on unrelated content, excluding all forget and retain set samples. Unlike the original TOFU formulation, we exclude retain-set samples—since their performance is measured separately by the $\text{Acc}_{\text{retain}}$ metric—so that Model Utility reflects performance on all other question types.

- **ROUGE_f and ROUGE_r**: ROUGE_f refers to the average ROUGE-L score between the unlearned model's output and the ground truth on the forget set. ROUGE_r is the corresponding score on questions from authors of the retain set. These metrics reflect how closely the target model's outputs match the ground-truth answers after unlearning.

5.7.2 Evaluation using RESTOR

RESTOR involves injecting incorrect information about a set of well-known individuals, for whom language models typically possess prior knowledge. Training on the documents provided in RESTOR causes the model to overwrite or lose its original knowledge about these entities. Unlearning in RESTOR is thus aimed at restoring this original knowledge. The benchmark evaluates the efficacy of an unlearning algorithm by testing whether the unlearned model is no longer influenced by the incorrect documents and can recover the knowledge it held before encountering the misinformation.

The forget set in RESTOR consists of documents containing false facts, while no explicit retain set is provided. Instead, RESTOR measures how well an unlearning algorithm can revert the model to its pre-corruption state without relying on any supervision that helps recovery. To prevent catastrophic forgetting during unlearning, [102] proposes including a general-purpose retain set, e.g., C4 [98] for preserving the model's overall utility. RESTOR evaluates the model's knowledge by asking factual questions about the affected individuals. The benchmark provides 1051 question-answer pairs, and the model's accuracy on these questions serves as a measure of restoration quality. Specifically, the model initially achieves a certain accuracy (referred to as clean accuracy) before being finetuned on incorrect data. Finetuning on these corrupted samples leads to a drop in accuracy (corrupted accuracy). The accuracy of the unlearned model is then used to assess restoration: if it improves upon the corrupted accuracy and approaches the clean accuracy, it indicates successful restoration of the original knowledge.

To further assess the efficacy of unlearning in the RESTOR benchmark, we start with a model already fine-tuned on TOFU and continue finetuning it on RESTOR— this enables us to study if untargeted entities from tofu are affected by the unlearning algorithm as well. We then apply unlearning to remove the influence of RESTOR and evaluate the resulting model on both RESTOR and TOFU. Although TOFU and RESTOR target different groups of individuals, both involve factual question-answers about people. This creates potential overlap in attributes (e.g., professions, locations), allowing us to examine whether unlearning RESTOR unintentionally harms unrelated knowledge in TOFU. Ideally, a successful unlearning algorithm should restore the correct knowledge about RESTOR entities while preserving the model's utility on TOFU authors. We use Probability [74]—defined as the ratio of the normalized probability of generating the ground-truth answer to the sum of normalized probabilities of other candidate outputs—along with ROUGE, Model Utility, and accuracy on the Real Authors and World Facts datasets of [74], to evaluate performance on TOFU-related metrics.

5.8 Experiments

In this section, we describe our experimental setup, including the models and baseline unlearning algorithms. We first present results on TOFU benchmark, followed by experiments on RESTOR benchmark. We then

Table 5.5: Comparison of unlearning algorithms on the TOFU benchmark for the task of forget01. Unlearning is applied to the final model trained on the full TOFU dataset, and results are compared to an ideal model trained only on the retained authors. Applying MSA to a checkpoint prior to TOFU finetuning (either the instruct or base model) yields strong performance, outperforming or remaining competitive with other baselines across most metrics. MSA when only utilizes forget vectors, shows competitive performance, even without a retain set, MSA remains effective.

Model	GPT-4o Judge Metrics ↑			TOFU Metrics			
	$\text{Acc}_{\text{forget}}$	$\text{Acc}_{\text{recover}}$	$\text{Acc}_{\text{retain}}$	Utility ↑	Forget Quality ↑	ROUGE_{f}↓	ROUGE_{r}↑
Final (TOFU)	0.15	0.13	0.89	0.48	0.01	0.71	0.68
Ideal (TOFU retain)	0.93	0.98	1.00	0.54	1.00	0.40	0.60
$\text{MSA}_{\text{instruct}}$	0.63	0.38	0.86	0.47	**0.77**	0.40	0.63
MSA_{base}	**0.78**	**0.45**	0.83	**0.48**	0.58	0.33	0.59
$\text{MSA}_{\text{instruct}^{\text{forget-only}}}$	0.65	0.30	0.85	0.47	0.58	0.39	0.59
MSA_{TOFU}	0.55	0.28	0.83	0.45	0.57	0.33	0.54
NPO	0.50	0.25	0.86	0.47	0.57	0.33	0.59
RMU	0.70	0.30	0.86	0.47	0.57	**0.30**	0.62
GradDiff	0.50	0.25	**0.88**	0.47	0.58	0.38	**0.64**

conclude with experiments that leverage historical checkpoints to simulate a more realistic use-case of MSA, evaluated on TOFU.

5.8.1 Experimental Setup

Our experiments use models from the Llama-3.2 family [21], including the instruction-tuned Llama-3.2-1B-Instruct and its corresponding pretrained model, Llama-3.2-1B. We include OLMo-2-1B [86, 35] to evaluate unlearning on a separate model family with accessible intermediate checkpoints. [6]

Unlearning algorithm baselines We compare MSA with NPO [130], GradDiff [33, 127], and RMU [68], as they achieve state-of-the-art performance on the TOFU and RESTOR benchmarks, as outlined in [102, 74]. We use the implementations provided by `open-unlearning` [20] for all baseline algorithms.

[6] We refer the reader to Appendix for experimental results on models with 8B parameters.

5.8.2 Experiments on TOFU

In this section, we evaluate unlearning algorithms, including MSA, on the `forget01` and `forget10` tasks of `TOFU`. We begin with Llama-3.2-1B-Instruct and follow the procedure of [74] to finetune the model on all `TOFU` authors. The goal is to unlearn 1% and 10% of the authors, respectively. Unlearning algorithms are applied to this finetuned model. To apply MSA, we evaluate three configurations to obtain the unlearning vectors: (1) using the final model trained on all `TOFU` authors, (2) the instruct model before `TOFU` finetuning, and (3) the base Llama-3.2-1B model prior to any instruction finetuning. We also explore the setting where retain vectors are not computed, to assess the robustness of MSA when no retain set is available.

The `forget01` task (Table 5.5) is generally considered easier than `forget10` (Table 5.6), as it involves forgetting a smaller subset of authors, making the unlearning process less disruptive. While MSA performs competitively in both settings, we observe that its improvements over baseline methods become more pronounced on the harder `forget10` task, highlighting the strength of our approach in more challenging unlearning scenarios.

Tables 5.5 and 5.6 show the experimental results. In `forget10` experiments, MSA, when unlearning vectors are obtained from the final model (denoted by $\text{MSA}_{\texttt{TOFU}}$, similar to task vectors [48]), shows low performance on $\text{Acc}_{\text{retain}}$, Model Utility, and Forget Quality. In contrast, when MSA uses earlier checkpoints—either the instruct model before `TOFU` finetuning (denoted by $\text{MSA}_{\text{instruct}}$), or the pretrained model (denoted by MSA_{base})— it achieves improved performance across nearly all metrics, outperforming or remaining competitive with NPO and other baselines. In `forget01` experiments, baseline methods are more competitive overall. However, MSA still outperforms or matches them across all metrics. We obtain forget and retain vectors by finetuning $e_{\text{f}} = 5$ and $e_{\text{r}} = 5$ epochs.

Notably, MSA without a retain vector (denoted by $\text{MSA}_{\text{instruct}^{\text{forget-only}}}$) also performs well and remains competitive with existing baselines across both tasks. This demonstrates the effectiveness of MSA even in the absence of a retain set, which may not always be available or easy to construct in practical unlearning scenarios. We refer to Appendix for further details on the experimental setup, including finetuning procedures and the configurations of unlearning algorithms.

Table 5.6: Comparison of unlearning algorithms on TOFU benchmark for the task of forget10. Applying MSA to a checkpoint prior to TOFU finetuning (either the instruct or base model) yields strong performance. In contrast, using the final model for MSA leads to diminished results, as reflected by lower model utility. RMU and GradDiff fail to achieve competitive results on this task.

Model	GPT-4o Judge Metrics ↑			TOFU Metrics			
	$\text{Acc}_{\text{forget}}$	$\text{Acc}_{\text{recover}}$	$\text{Acc}_{\text{retain}}$	Utility ↑	Forget Quality ↑	ROUGE_{f}↓	ROUGE_{r}↑
Final (TOFU)	0.20	0.09	0.88	0.48	1.5e-27	0.72	0.68
Ideal (TOFU retain)	0.99	0.99	1.00	0.54	1.00	0.38	0.61
$\text{MSA}_{\text{instruct}}$	**0.81**	**0.41**	0.81	0.47	**0.47**	0.38	0.55
MSA_{base}	0.77	0.37	0.77	0.44	0.37	0.35	0.51
$\text{MSA}_{\text{instruct}^{\text{forget-only}}}$	0.63	0.29	0.79	0.44	4.6e-07	0.45	0.53
MSA_{TOFU}	0.75	0.25	0.61	<u>0.26</u>	9.9e-07	0.24	0.31
NPO	0.66	0.24	0.78	0.47	6.8e-07	**0.22**	0.42
RMU	0.84	<u>0.06</u>	**0.87**	0.47	2.0e-7	0.31	**0.63**
GradDiff	0.44	0.24	0.84	**0.48**	<u>5.1e-16</u>	0.47	0.58

5.8.3 Experiments on RESTOR

In this section, we evaluate MSA on RESTOR benchmark. We begin with the Llama-3.2-1B-Instruct model trained on the authors of TOFU [20]. Following the setup of [102], we further finetune this model on the incorrect facts from RESTOR.

All unlearning algorithms, except MSA, are applied to the final model (i.e., trained on both TOFU and RESTOR) using the RESTOR dataset for unlearning. In contrast, MSA is permitted to obtain unlearning vectors from earlier checkpoints of the model, rather than relying solely on the final one. We evaluate MSA using three different checkpoints to compute unlearning vectors: (1) the final model, (2) the Llama-3.2-1B-Instruct model prior to TOFU training, and (3) the base pretrained Llama-3.2-1B model before instruction tuning. To obtain the forget vector, we train for $e_{\text{f}} = 2$ epochs. Unlike other baselines, which use C4 as a retain set to prevent catastrophic forgetting, we find that MSA does not need a retain vector, demonstrating high performance even without one.

Table 5.7 shows the experimental results. The row referring to ideal model which is the instruct model finetuned on TOFU, shows an accuracy of 46.18% on RESTOR entities along with strong performance on TOFU-related tasks. When further finetuned on RESTOR, the model's performance declines across RESTOR entities

Table 5.7: Comparison of unlearning algorithms onRESTOR benchmark. A model initially trained on TOFU is further finetuned on RESTOR, leading to degraded accuracy on both RESTOR entities and TOFU authors. Effective unlearning should reverse this degradation. MSA demonstrates strong restoration performance, outperforming existing unlearning methods.

Model	RESTOR ↑	TOFU Metrics ↑				
		Probability	ROUGE	Model Utility	Real Authors	World Facts
Final (TOFU + RESTOR)	32.14	0.44	0.40	0.48	0.58	0.71
Ideal (TOFU)	46.18	0.87	0.80	0.60	0.82	0.83
$\text{MSA}_{\text{instruct}}$	**46.08**	**0.77**	**0.62**	**0.56**	0.84	**0.87**
MSA_{base}	43.61	0.62	0.49	0.54	0.72	0.86
$\text{MSA}_{\text{TOFU +RESTOR}}$	35.30	0.42	0.38	0.47	0.62	0.69
NPO	38.65	0.46	0.43	0.49	**0.86**	0.82
RMU	31.68	0.38	0.38	0.45	0.44	0.66
GradDiff	24.07	0.30	0.37	0.45	0.52	0.57

and TOFU metrics, with a slight drop observed on World Facts as well. Ideally, an unlearning algorithm should produce a model that closely resembles one trained only on TOFU.

According to Table 5.7, RMU and GradDiff fail to improve any of the metrics, highlighting their limitations for data-level unlearning—and in fact, they further perturb existing knowledge. NPO, on the other hand, yields improvements on RESTOR and some TOFU metrics, achieving notable restoration on benchmarks such as Real Authors and World Facts. MSA, when obtains unlearning vectors from the final checkpoint, fails to restore effectively and degrades other metrics. However, when MSA leverages earlier checkpoints, including the base pretrained model, its learned forget vectors becomes effective: when applied to the final model, it produces an unlearned version that restores accuracy on RESTOR and improves performance on other metrics. Notably, the closer the checkpoint used to compute the forget vector is to the final model, the more effective the restoration. Nonetheless, even when using a distant checkpoint (e.g., the base pretrained model before instruction finetuning), the results remain significant. This demonstrates that MSA can successfully mitigate the broader negative impact of RESTOR on the model's knowledge. Details on the experimental setup and hyperparameter tuning are provided in Appendix.

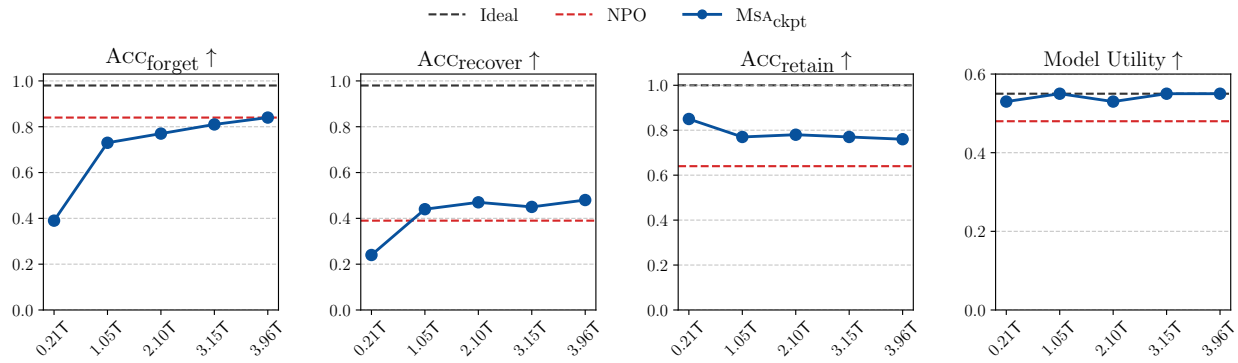

Figure 5.5: Performance of MSA when early checkpoints are used to obtain unlearning vectors. The OLMo-2-1B model, trained on 3985B tokens, is finetuned on `TOFU`, after which 10% of authors are unlearned (task `forget10`). The x-axis indicates the number of tokens the checkpoint used for unlearning vector extraction had been trained on. While performance slightly drops with earlier checkpoints, vectors derived from checkpoints trained on as few as 2T tokens still outperform or remain competitive with NPO across several metrics.

5.8.4 Which Checkpoint is Appropriate?

In this section, we extend MSA to OLMo-2-1B models [86], for which we have access to pretraining checkpoints [7]. These experiments aim to demonstrate the practical viability of MSA in real-world scenarios. We begin with a late checkpoint from stage 1 of pretraining OLMo-2-1B, where the model has been trained on around 4T tokens, and finetune it on the `TOFU` authors. We then apply unlearning algorithms on the task of `forget10` from `TOFU`.

For MSA, instead of using the final model, we obtain unlearning vectors from earlier checkpoints to evaluate whether the method remains effective when the checkpoint precedes the introduction of unlearning documents by a large margin. Specifically, we evaluate MSA using checkpoints trained on 3964B, 3146B, 2098B, 1049B, and 210B tokens. We compare the resulting unlearned models to those produced by NPO, as NPO was the strongest baseline in earlier experiments. We use $e_f = 5$ and $e_r = 5$ epochs to obtain the forget and retain vectors, respectively.

Figure 5.5 presents the results (see Appendix for full results). We observe that unlearning vectors derived from earlier checkpoints remain effective: they successfully cause the model to forget the target authors

[7] https://huggingface.co/allenai/OLMo-2-0425-1B/tree/main

while more closely resembling the behavior of an ideal model trained without the forget set. In addition, they preserve performance on the retain set, remaining competitive with NPO. As expected, unlearning performance gradually degrades as the checkpoint becomes more distant from the point at which unlearning documents were introduced. Nonetheless, the degradation is modest, and MSA remains competitive with or superior to NPO—demonstrating its practicality even when only older checkpoints, far removed from the point at which the target datapoints are introduced, are available. Conceivably, practitioners could index pretraining data by model checkpoints, and select the most appropriate checkpoint to compute vectors.

These results demonstrate that the unlearning vectors obtained by MSA are most effective when using checkpoints closer to the model state just before exposure to the unlearning data. However, even older checkpoints—with gaps of up to trillions of tokens—still yield strong results. Both forget and retain vectors remain useful when applied to target model, outperforming existing methods that operate solely on the final model highlights the practicality of our approach and shows that intermediate checkpoints can be effectively leveraged for machine unlearning. Notably, when using a very early checkpoint for unlearning, i.e., $\text{MSA}_{\text{ckpt-210B}}$, the forget vector fails to meaningfully alter model behavior, resulting in low $\text{Acc}_{\text{forget}}$ and $\text{Acc}_{\text{recover}}$. This highlights that vectors derived from such early checkpoints may not be effective for unlearning.

5.9 Discussion and Conclusion (II)

We introduce MSA, a new method for machine unlearning that leverages intermediate model checkpoints to estimate and undo the influence of undesirable data. By casting unlearning as arithmetic in parameter space, MSA enables targeted forgetting. Across the `TOFU` and `RESTOR` benchmarks, MSA consistently outperforms prior methods, achieving superior forgetting, recovery, and utility preservation—even when unlearning directions are computed from early checkpoints. These results underscore the practicality of checkpoint-based unlearning and suggest that historical training states, routinely stored by model developers, can be repurposed as tools for data unlearning. Many avenues remain open: future work would develop benchmarks and methods that explicitly consider the temporal position of unlearning targets during training, and consider the frequency of unlearning targets in training data, thus enabling unlearning techniques to handle long-

range dependencies and cumulative effects of early exposure. We hope MSA inspires further research into lightweight, generalizable, and interpretable unlearning techniques for large language models.

Chapter 6

Proposed Work amd Timeline

6.1 Summary of Proposed Work

Below is a high-level summary of the research I set forth in this prospectus:

Knowledge Localization, Model Editing and Unlearning in Foundation Models I will continue developing methods for more effective knowledge localization in both text-to-image and large language models. Further details on these research directions are elaborated below.

Text-to-Image Interpretability I plan to explore more advanced techniques for knowledge localization, as our current investigations suggest that the effectiveness of localization can be improved by designing better methods that more accurately capture the contribution of different layers to the final image generation. We will continue our work on transformer-based architectures by leveraging techniques from mechanistic interpretability and linear approximations of the attention mechanism, where tokens are treated as streams that each module writes to.

Large Language Model Unlearning I will further develop my ongoing project on leveraging intermediate model checkpoints for machine unlearning, aiming to verify its practicality in real-world scenarios. In parallel, I will explore additional benchmark designs that capture other essential aspects of an ideal machine

unlearning algorithm. I also plan to release my existing work on open unlearning in public repositories to increase its visibility and impact.

6.2 Timeline

I plan to pursue the proposed research directions during the final year of my Ph.D., with the goal of publishing additional papers that contribute meaningfully to the field of AI. My current timeline aims for graduation within four to four and a half years. Over the next year, I expect to complete ongoing projects and prepare final submissions, while also consolidating my work into a cohesive dissertation. This will include finalizing my thesis chapters on knowledge localization in text-to-image generative models and machine unlearning in large language models, which represent the core contributions of my doctoral research.

Appendix A

Reading List

A.1 Vision Models Interpretability

1. Jain, Saachi, et al. "Distilling model failures as directions in latent space." arXiv preprint arXiv:2206.14754 (2022).

2. Idrissi, Badr Youbi, et al. "Imagenet-x: Understanding model mistakes with factor of variation annotations." arXiv preprint arXiv:2211.01866 (2022).

3. Hashimoto, Tatsunori, et al. "Fairness without demographics in repeated loss minimization." International Conference on Machine Learning. PMLR, 2018.

4. Eyuboglu, Sabri, et al. "Domino: Discovering systematic errors with cross-modal embeddings." arXiv preprint arXiv:2203.14960 (2022).

5. d'Eon, Greg, et al. "The spotlight: A general method for discovering systematic errors in deep learning models." Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency. 2022.

6. Johnson, Nari, et al. "Where does my model underperform? a human evaluation of slice discovery algorithms." Proceedings of the AAAI Conference on Human Computation and Crowdsourcing. Vol. 11. 2023.

7. Kim, Younghyun, et al. "Discovering and mitigating visual biases through keyword explanation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.

8. Zhang, Youcai, et al. "Recognize anything: A strong image tagging model." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.

9. Balasubramanian, Sriram, Samyadeep Basu, and Soheil Feizi. "Decomposing and interpreting image representations via text in vits beyond CLIP." Advances in Neural Information Processing Systems 37 (2024): 81046-81076.

10. Oikarinen, Tuomas, et al. "Label-free concept bottleneck models." arXiv preprint arXiv:2304.06129 (2023).

11. Yang, Yue, et al. "Language in a bottle: Language model guided concept bottlenecks for interpretable image classification." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2023.

A.2 Knowledge Localization, and Model Editing in Text-to-Image Models

1. Basu, Samyadeep, et al. "Localizing and editing knowledge in text-to-image generative models." (2024).

2. Basu, Samyadeep, et al. "On Mechanistic Circuits for Extractive Question-Answering." arXiv preprint arXiv:2502.08059 (2025).

3. Hertz, Amir, et al. "Prompt-to-prompt image editing with cross attention control." arXiv preprint arXiv:2208.01626 (2022).

4. Liu, Bingyan, et al. "Towards understanding cross and self-attention in stable diffusion for text-guided image editing." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2024.

5. Gupta, Akshat, Anurag Rao, and Gopala Anumanchipalli. "Model editing at scale leads to gradual and catastrophic forgetting." arXiv preprint arXiv:2401.07453 (2024).

6. Tang, Raphael, et al. "What the daam: Interpreting stable diffusion using cross attention." arXiv preprint arXiv:2210.04885 (2022).

7. Burns, Collin, et al. "Discovering latent knowledge in language models without supervision." arXiv preprint arXiv:2212.03827 (2022).

8. Meng, Kevin, et al. "Locating and editing factual associations in gpt." Advances in neural information processing systems 35 (2022): 17359-17372.

9. Meng, Kevin, et al. "Mass-editing memory in a transformer." arXiv preprint arXiv:2210.07229 (2022).

10. Orgad, Hadas, Bahjat Kawar, and Yonatan Belinkov. "Editing implicit assumptions in text-to-image diffusion models." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023.

A.3 Large Language Model Unlearning

1. Eldan, Ronen, and Mark Russinovich. "Who's harry potter? approximate unlearning for LLMs." (2023).

2. Chen, Jiaao, and Diyi Yang. "Unlearn what you want to forget: Efficient unlearning for llms." arXiv preprint arXiv:2310.20150 (2023).

3. Goel, Shashwat, et al. "Corrective machine unlearning." arXiv preprint arXiv:2402.14015 (2024).

4. Ilharco, Gabriel, et al. "Editing models with task arithmetic." arXiv preprint arXiv:2212.04089 (2022).

5. Ishibashi, Yoichi, and Hidetoshi Shimodaira. "Knowledge sanitization of large language models." arXiv preprint arXiv:2309.11852 (2023).

6. Jang, Joel, et al. "Knowledge unlearning for mitigating privacy risks in language models." arXiv preprint arXiv:2210.01504 (2022).

7. Jia, Jinghan, et al. "Soul: Unlocking the power of second-order optimization for llm unlearning." arXiv preprint arXiv:2404.18239 (2024).

8. Jin, Zhuoran, et al. "Rwku: Benchmarking real-world knowledge unlearning for large language models." Advances in Neural Information Processing Systems 37 (2024): 98213-98263.

9. Kumar, Vinayshekhar Bannihatti, Rashmi Gangadharaiah, and Dan Roth. "Privacy adhering machine un-learning in nlp." arXiv preprint arXiv:2212.09573 (2022).

10. Li, Nathaniel, et al. "The wmdp benchmark: Measuring and reducing malicious use with unlearning." arXiv preprint arXiv:2403.03218 (2024).

11. Liu, Sijia, et al. "Rethinking machine unlearning for large language models." Nature Machine Intelligence (2025): 1-14.

12. Maini, Pratyush, et al. "Tofu: A task of fictitious unlearning for llms." arXiv preprint arXiv:2401.06121 (2024).

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[3] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in Neural Information Processing Systems*, 34:225–236, 2021.

[4] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Joshua B. Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *NeurIPS*, 2019.

[5] George-Octavian Barbulescu and Peter Triantafillou. To each (textual sequence) its own: Improving memorized-data unlearning in large language models. *arXiv preprint arXiv:2405.03097*, 2024.

[6] Samyadeep Basu, Nanxuan Zhao, Vlad Morariu, Soheil Feizi, and Varun Manjunatha. Localizing and editing knowledge in text-to-image generative models, 2023.

[7] Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. Leace: Perfect linear concept erasure in closed form. *Advances in Neural Information Processing Systems*, 36:66044–66063, 2023.

[8] Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. Leace: Perfect linear concept erasure in closed form. *Advances in Neural Information Processing Systems*, 36, 2024.

[9] Alberto Blanco-Justicia, Najeeb Jebreel, Benet Manzanares, David Sánchez, Josep Domingo-Ferrer, Guillem Collell, and Kuan Eeik Tan. Digital forgetting in large language models: A survey of unlearning methods. *arXiv preprint arXiv:2404.02062*, 2024.

[10] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pages 141–159. IEEE, 2021.

[11] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

[12] De Terwangne C. The right to be forgotten and the informational autonomy in the digital environment. Scientific analysis or review LB-NA-26434-EN-N, Luxembourg (Luxembourg), 2013.

[13] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[14] Hila Chefer, Oran Lang, Mor Geva, Volodymyr Polosukhin, Assaf Shocher, Michal Irani, Inbar Mosseri, and Lior Wolf. The hidden language of diffusion models. *arXiv preprint arXiv:2306.00966*, 2023.

[15] Jiaao Chen and Diyi Yang. Unlearn what you want to forget: Efficient unlearning for llms. *arXiv preprint arXiv:2310.20150*, 2023.

[16] Somnath Basu Roy Chowdhury, Krzysztof Choromanski, Arijit Sehanobish, Avinava Dubey, and Snigdha Chaturvedi. Towards scalable exact machine unlearning using parameter-efficient fine-tuning. *arXiv preprint arXiv:2406.16257*, 2024.

[17] Adrián Csiszárik, Péter Kőrösi-Szabó, Ákos Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations. *Advances in Neural Information Processing Systems*, 34:5656–5668, 2021.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[19] Greg d'Eon, Jason d'Eon, James R. Wright, and Kevin Leyton-Brown. The spotlight: A general method for discovering systematic errors in deep learning models, 2021.

[20] Vineeth Dorna, Anmol Mekala, Wenlong Zhao, Andrew McCallum, J Zico Kolter, and Pratyush Maini. OpenUnlearning: A unified framework for llm unlearning benchmarks. https://github.com/locuslab/open-unlearning, 2025. Accessed: February 27, 2025.

[21] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[22] Constantin Eichenberg, Sid Black, Samuel Weinbach, Letitia Parcalabescu, and Anette Frank. Magma - multimodal augmentation of generative models through adapter-based finetuning. *ArXiv*, abs/2112.05253, 2021.

[23] Ronen Eldan and Mark Russinovich. Who's harry potter? approximate unlearning in llms. *arXiv preprint arXiv:2310.02238*, 2023.

[24] Ronen Eldan and Mark Russinovich. Who's Harry Potter? Approximate Unlearning in LLMs, October 2023. arXiv:2310.02238 [cs].

[25] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019.

[26] Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. Domino: Discovering systematic errors with cross-modal embeddings, 2022.

[27] Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. Simplicity prevails: Rethinking negative preference optimization for llm unlearning. *arXiv preprint arXiv:2410.07163*, 2024.

[28] Thomas Fel, Agustin Picard, Louis Béthune, Thibaut Boissin, David Vigouroux, Julien Colin, R'emi Cadene, and Thomas Serre. Craft: Concept recursive activation factorization for explainability. *ArXiv*, abs/2211.10154, 2022.

[29] Rohit Gandikota, Joanna Materzyńska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. In *Proceedings of the 2023 IEEE International Conference on Computer Vision*, 2023.

[30] Irena Gao, Gabriel Ilharco, Scott Lundberg, and Marco Tulio Ribeiro. Adaptive testing of computer vision models, 2023.

[31] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[32] Shashwat Goel, Ameya Prabhu, Philip Torr, Ponnurangam Kumaraguru, and Amartya Sanyal. Corrective machine unlearning. *arXiv preprint arXiv:2402.14015*, 2024.

[33] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020.

[34] Shahriar Golchin and Mihai Surdeanu. Time travel in llms: Tracing data contamination in large language models, 2024.

[35] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024.

[36] Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*, pages 1929–1938. PMLR, 2018.

[37] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control, 2022.

[38] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *ArXiv*, abs/2104.08718, 2021.

[39] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.

[40] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[41] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[42] Adrian Hoffmann, Claudio Fanconi, Rahul Rade, and Jonas Moritz Kohler. This looks like that... does it? shortcomings of latent space prototype interpretability in deep networks. *ArXiv*, abs/2105.02968, 2021.

[43] Yihuai Hong, Lei Yu, Haiqin Yang, Shauli Ravfogel, and Mor Geva. Intrinsic evaluation of unlearning using parametric knowledge traces. *arXiv preprint arXiv:2406.11614*, 2024.

[44] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[45] Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. Are large pre-trained language models leaking your personal information? *arXiv preprint arXiv:2205.12628*, 2022.

[46] Xinyu Huang, Youcai Zhang, Jinyu Ma, Weiwei Tian, Rui Feng, Yuejie Zhang, Yaqian Li, Yandong Guo, and Lei Zhang. Tag2text: Guiding vision-language model via image tagging. *arXiv preprint arXiv:2303.05657*, 2023.

[47] Badr Youbi Idrissi, Diane Bouchacourt, Randall Balestriero, Ivan Evtimov, Caner Hazirbas, Nicolas Ballas, Pascal Vincent, Michal Drozdzal, David Lopez-Paz, and Mark Ibrahim. Imagenet-x: Understanding model mistakes with factor of variation annotations. *arXiv preprint arXiv:2211.01866*, 2022.

[48] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[49] Yoichi Ishibashi and Hidetoshi Shimodaira. Knowledge sanitization of large language models. *arXiv preprint arXiv:2309.11852*, 2023.

[50] Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space. *arXiv preprint arXiv:2206.14754*, 2022.

[51] Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge unlearning for mitigating privacy risks in language models. *arXiv preprint arXiv:2210.01504*, 2022.

[52] Jinghan Jia, Yihua Zhang, Yimeng Zhang, Jiancheng Liu, Bharat Runwal, James Diffenderfer, Bhavya Kailkhura, and Sijia Liu. Soul: Unlocking the power of second-order optimization for llm unlearning. *arXiv preprint arXiv:2404.18239*, 2024.

[53] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[54] Zhuoran Jin, Pengfei Cao, Chenhao Wang, Zhitao He, Hongbang Yuan, Jiachun Li, Yubo Chen, Kang Liu, and Jun Zhao. Rwku: Benchmarking real-world knowledge unlearning for large language models. *arXiv preprint arXiv:2406.10890*, 2024.

[55] Nari Johnson, Ángel Alexander Cabrera, Gregory Plumb, and Ameet Talwalkar. Where does my model underperform? a human evaluation of slice discovery algorithms, 2023.

[56] Bjørn Aslak Juliussen, Jon Petter Rui, and Dag Johansen. Algorithms that forget: Machine unlearning and the right to erasure. *Computer Law & Security Review*, 51:105885, 2023.

[57] S. Kadhe, Farhan Ahmed, Dennis Wei, Nathalie Baracaldo, and Inkit Padhi. Split, unlearn, merge: Leveraging data attributes for more effective unlearning in llms. *ArXiv*, abs/2406.11780, 2024.

[58] Priyatham Kattakinda, Alexander Levine, and Soheil Feizi. Invariant learning via diffusion dreamed distribution shifts. *arXiv preprint arXiv:2211.10370*, 2022.

[59] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[60] Michael P Kim, Amirata Ghorbani, and James Zou. Multiaccuracy: Black-box post-processing for fairness in classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 247–254, 2019.

[61] Younghyun Kim, Sangwoo Mo, Minkyu Kim, Kyungmin Lee, Jaeho Lee, and Jinwoo Shin. Bias-to-text: Debiasing unknown visual biases through language interpretation, 2023.

[62] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR, 2020.

[63] Vinayshekhar Bannihatti Kumar, Rashmi Gangadharaiah, and Dan Roth. Privacy adhering machine un-learning in nlp. *arXiv preprint arXiv:2212.09573*, 2022.

[64] Nupur Kumari, Bingliang Zhang, Sheng-Yu Wang, Eli Shechtman, Richard Zhang, and Jun-Yan Zhu. Ablating concepts in text-to-image diffusion models. In *'International Conference on Computer Vision (ICCV)'*, 2023.

[65] Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, et al. 3db: A framework for debugging computer vision models. *Advances in Neural Information Processing Systems*, 35:8498–8511, 2022.

[66] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

[67] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[68] Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, et al. The wmdp benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*, 2024.

[69] Wenjie Li, Jiawei Li, Christian Schroeder de Witt, Ameya Prabhu, and Amartya Sanyal. Delta-influence: Unlearning poisons via influence functions. *arXiv preprint arXiv:2411.13731*, 2024.

[70] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[71] Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pages 6781–6792. PMLR, 2021.

[72] Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Xiaojun Xu, Yuguang Yao, Hang Li, Kush R Varshney, et al. Rethinking machine unlearning for large language models. *arXiv preprint arXiv:2402.08787*, 2024.

[73] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[74] Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. Tofu: A task of fictitious unlearning for llms. *arXiv preprint arXiv:2401.06121*, 2024.

[75] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36, 2022. arXiv:2202.05262.

[76] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022.

[77] Jack Merullo, Louis Castricato, Carsten Eickhoff, and Ellie Pavlick. Linearly mapping from image to text space. *arXiv preprint arXiv:2209.15162*, 2022.

[78] Mazda Moayeri, Phillip E. Pope, Yogesh Balaji, and Soheil Feizi. A comprehensive study of image classification model sensitivity to foregrounds, backgrounds, and visual attributes. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19065–19075, 2022.

[79] Mazda Moayeri, Keivan Rezaei, Maziar Sanjabi, and Soheil Feizi. Text-to-concept (and back) via cross-model alignment. *arXiv preprint arXiv:2305.06386*, 2023.

[80] Ron Mokady, Amir Hertz, and Amit H. Bermano. Clipcap: Clip prefix for image captioning. *ArXiv*, abs/2111.09734, 2021.

[81] Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication. *ArXiv*, abs/2209.15430, 2022.

[82] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: Debiasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33:20673–20684, 2020.

[83] Besmira Nushi, Ece Kamar, and Eric Horvitz. Towards accountable ai: Hybrid human-machine analyses for characterizing system failure. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 6, pages 126–135, 2018.

[84] Tuomas Oikarinen and Tsui-Wei Weng. Clip-dissect: Automatic description of neuron representations in deep vision networks. *arXiv preprint arXiv:2204.10965*, 2022.

[85] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

[86] Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 Furious, 2024.

[87] Hadas Orgad, Bahjat Kawar, and Yonatan Belinkov. Editing implicit assumptions in text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[88] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. Privacy risks of general-purpose language models. *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1314–1331, 2020.

[89] Martin Pawelczyk, Jimmy Z Di, Yiwei Lu, Ayush Sekhari, Gautam Kamath, and Seth Neel. Machine unlearning fails to remove data poisoning attacks. *arXiv preprint arXiv:2406.17216*, 2024.

[90] Judea Pearl. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 411–420, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[91] Gregory Plumb, Marco Tulio Ribeiro, and Ameet Talwalkar. Finding and fixing spurious patterns with explanations. *arXiv preprint arXiv:2106.02112*, 2021.

[92] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023.

[93] Youyang Qu, Ming Ding, Nan Sun, Kanchana Thilakarathna, Tianqing Zhu, and Dusit Niyato. The frontier of data erasure: Machine unlearning for large language models. *arXiv preprint arXiv:2403.15779*, 2024.

[94] Alec Radford, Jong Wook Kim, Chris Hallacy, A. Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[95] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[96] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[97] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[98] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[99] P Rajpurkar. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[100] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[101] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021.

[102] Keivan Rezaei, Khyathi Chandu, Soheil Feizi, Yejin Choi, Faeze Brahman, and Abhilasha Ravichander. Restor: Knowledge recovery through machine unlearning. *arXiv preprint arXiv:2411.00204*, 2024.

[103] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2021.

[104] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[105] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *CoRR*, abs/1911.08731, 2019.

[106] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[107] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. BREEDS: benchmarks for subpopulation shift. *CoRR*, abs/2008.04859, 2020.

[108] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. Breeds: Benchmarks for subpopulation shift. In *ArXiv preprint arXiv:2008.04859*, 2020.

[109] Stefan Schoepf, Jack Foster, and Alexandra Brintrup. Potion: Towards poison unlearning. *arXiv preprint arXiv:2406.09173*, 2024.

[110] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[111] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[112] Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A Smith, and Chiyuan Zhang. Muse: Machine unlearning six-way evaluation for language models. *arXiv preprint arXiv:2407.06460*, 2024.

[113] Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19339–19352. Curran Associates, Inc., 2020.

[114] Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *Advances in Neural Information Processing Systems*, 33:19339–19352, 2020.

[115] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *CoRR*, abs/2010.02502, 2020.

[116] Raphael Tang, Linqing Liu, Akshat Pandey, Zhiying Jiang, Gefei Yang, Karun Kumar, Pontus Stenetorp, Jimmy Lin, and Ferhan Ture. What the DAAM: Interpreting stable diffusion using cross attention. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5644–5659, Toronto, Canada, July 2023. Association for Computational Linguistics.

[117] Yoad Tewel, Yoav Shalev, Idan Schwartz, and Lior Wolf. Zerocap: Zero-shot image-to-text generation for visual-semantic arithmetic. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17897–17907, 2021.

[118] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[119] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. From imagenet to image classification: Contextualizing progress on benchmarks. In *International Conference on Machine Learning*, pages 9625–9635. PMLR, 2020.

[120] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[121] Vijay Vasudevan, Benjamin Caine, Raphael Gontijo Lopes, Sara Fridovich-Keil, and Rebecca Roelofs. When does dough become a bagel? analyzing the remaining mistakes on imagenet. *Advances in Neural Information Processing Systems*, 35:6720–6734, 2022.

[122] Joshua Vendrow, Saachi Jain, Logan Engstrom, and Aleksander Madry. Dataset interfaces: Diagnosing model failures using controllable counterfactual generation. *arXiv preprint arXiv:2302.07865*, 2023.

[123] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Caltech-ucsd birds-200-2011 (cub-200-2011). Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[124] Huazheng Wang, Yongcheng Jing, Haifeng Sun, Yingjie Wang, Jingyu Wang, Jianxin Liao, and Dacheng Tao. Erasing without remembering: Safeguarding knowledge forgetting in large language models, 2025.

[125] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36, 2024.

[126] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. *ArXiv preprint arXiv:2006.09994*, 2020.

[127] Yuanshun Yao, Xiaojun Xu, and Yang Liu. Large language model unlearning. *arXiv preprint arXiv:2310.10683*, 2023.

[128] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*, 2023.

[129] Ruihan Zhang, Prashan Madumal, Tim Miller, Krista A. Ehinger, and Benjamin I. P. Rubinstein. Invertible concept-based explanations for cnn models with non-negative concept activation vectors. In *AAAI Conference on Artificial Intelligence*, 2020.

[130] Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv preprint arXiv:2404.05868*, 2024.

[131] Youcai Zhang, Xinyu Huang, Jinyu Ma, Zhaoyang Li, Zhaochuan Luo, Yanchun Xie, Yuzhuo Qin, Tong Luo, Yaqian Li, Shilong Liu, et al. Recognize anything: A strong image tagging model. *arXiv preprint arXiv:2306.03514*, 2023.