Towards Interpretable and Robust ML Systems

Sahil Verma

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2025

Reading Committee:
Jeff Bilmes, Co-Chair
Chirag Shah, Co-Chair
Luke Zettlemoyer

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Towards Interpretable and Robust ML Systems

Sahil Verma

Co-chairs of the Supervisory Committee:

Jeff Bilmes
Computer Science and Engineering

Chirag Shah
Computer Science and Engineering

Recent advancements in ML have taken strides in enabling models to accomplish unprecedented tasks, starting from the bare minimum binary classification for loan applications to intrinsically complex self-driving. As the models have become better, faster, and more powerful, they have also become larger and more opaque. This has happened because of the widespread use of neural networks, which enable capturing and expressing incredibly complex representations but are uninterpretable to humans. This phenomenon raises the question of trust – as humans who want to be in the position of control, how do we trust the model to make correct decisions? In this thesis, I aim to answer this question by making models more interpretable, examining their robustness, and ensuring they are safe for us as a society to rely on.

Acknowledgements

I thank my advisors, Jeff Bilmes and Chirag Shah for their advice and help during the long PhD process. I thank my committee members, Luke Zettlemoyer and Tyler H. McCormick for their support. Finally, heartfelt thank you to my friends (Soumye, Shruti, Jasper, Aditya, Sanjana, Tapan, Raghav, among several others) and family for their love and care during this long process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

ML has enabled AI models to become extremely capable in a very short amount of time (less than a decade). AI models can now solve tasks, previously unimaginable, like winning gold medals at the IMO, an achievement so prestigious that only a handful of humans can claim to enjoy; being able to safely drive cars across long distances and in complex scenarios; even folding proteins, which enable humanity to attempt to solve intrinsic diseases. All of this has been accomplished with large-scale training of models, which usually involves large-scale datasets and a model with many parameters, usually in the order of billions in LLMs, for example. The large-scale modeling enables the models to learn extremely complex representations from the training data – thereby enabling them to excel at the aforementioned tasks. However, it comes at the cost of understanding the model itself. The models are so large and complex that even the creators of the model cannot understand why the model produces something when it does. This leads to a lack of trust and transparency. It is important that we trust the model in order to be able to deploy it for critical use cases. In this thesis, I propose works that take a step in the direction of making ML models more trustworthy. Specifically, I attempt to make models more trustworthy by making them more interpretable, understanding the dependency on training data, and making them more robust.

Being able to interpret the decisions from ML models helps the creators and users of the model to trust it. There have been a plethora of approaches to interpret model decisions – usually falling in two categories a) feature-based explanations, and b) counterfactual explanations. Feature-based explanations use the features

of the datapoints to explain a decision. For example, they would explain a loan denial decision by stating that the features of income and age were responsible for denying the loan. On the other hand, counterfactual explanations provide an alternate universe in which the model would have made a different decision. For example, for the loan denial situation, a counterfactual explanation would state that if the applicant's income was higher by $10K and their education level be bachelors, they would have been given the loan. This provides the applicant with actionable advice on how to change their features in order to achieve the goal of getting the loan. In this thesis, I discuss my previous work that proposes FastAR, which provides counterfactual explanations to ML classification model decisions. FastAR provides these explanations while respecting several real-world constraints, like only having black-box access to the ML model and being amortized for all the users (i.e, it does not require separate optimization for each user).

Another class of widely used ML models is recommender systems. These systems are popularly used for recommending items to users at various marketplaces and social media platforms. However, both the creators of the models and the end-users aren't aware of why they are recommended a specific item. Providing explanations of recommendations can alleviate this issue. Explanations for recommendations can be provided in several ways. The two most prevalent ways are a) user-based explanations, and b) item-based explanations. User-based explanations explain a recommendation in the following manner: "you are being recommended this item because users like you liked this item". Item-based explanations explain a recommendation in the following manner: "you are being recommended this item because it is similar to the items you have liked in the past" (and list the few items the user has liked in the past). These explanations are useful, but not completely adequate. In this thesis, I discuss my previous work that proposes a novel approach to provide explanations for recommendations that are based on attributes of the items that the recommender system knows a user likes. For example, attribute-based explanations are of this form: "you are being recommended this movie because you like thriller and horror movies". These explanations are more fine-grained than item-based and can also help the user even change them if they know that the system has captured an incorrect attribute about them. Not only does the proposed approach provides attribute based explanations, but the explanations are provided in a black-box manner, i.e., the approach does not require a change to the underlying recommender system, which makes the deployment much less complicated com-

pared to previous methods that require updating the recommender system itself. This is a recurring theme of my works that provide explanations while respecting the real-world constraints.

Finally, models are a function of their training data, and the data determines how the model will behave during its deployment. However, with large-scale training being limited to proprietary labs that do not share their training data, it is not known to the public if the data is collected in a manner that respects copyrights and privacy laws. This is especially true in the era of training models on internet-scale datasets – these datasets are collected by scraping the internet without much thought given to the provenance of the data. This leads to questions about how data impacts the downstream model's ability to violate copyrights and privacy. In this thesis, I discuss my previous work MIMETIC^2, an approach that aims to answer the question of "how many images of an entity does a text-to-image model need in order to generate that entity". MIMETIC^2 finds this threshold, which we term the imitation threshold. In our work, we find that entities like art styles or human faces that are repeated more times than the imitation threshold have a high probability of being imitated by the trained model, posing risks of copyright and privacy violations. We conduct several user studies and finetuning experiments to empirically test the thresholds we find, and they seem to be accurate for the five models we experiment with.

Another aspect of ML models that helps us trust them more is robustness. Robustness research in ML models mainly has two flavors: a) adversarial robustness and b) robustness to shifts in data distributions. Robustness refers to the property by which we want the model to make similar predictions for datapoints that are similar. Researchers have noticed that ML models are vulnerable to adversarial attacks, whereby slightly perturbing the datapoints can result in massively different outputs. For example, an image of a cat that is being predicted as a cat can be misclassified as a dog if we edit just a few pixels in the image (note that this image would still look like a cat to a human being). Why this happens is still an open research question; the high-level intuition states that the large parameterized models can make the models vulnerable to such attacks, as small changes in the input can be magnified by the model parameters, leading to drastic changes in the output. Achieving adversarial robustness is still something researchers are actively working on, and there seems to be no solution in sight, even with a decade's worth of research. Robustness to shifts in

the data distribution is another category of robustness where we want the models to be capable of handling shifts in the input data if it is slightly different from the training data. The classic example of this case is the water and land birds dataset, where most of the water birds are shown with a water background and most of the land birds are shown with a land background. On the training dataset, the model achieves high accuracy, and also on the test set that follows the same distribution as the training dataset. However, when we switch the background of water birds to land and land birds to water, the model spectacularly fails to perform accurately. There have been several approaches proposed to solve this.

In this thesis, I examine the robustness of a specific approach which claimed to remove backdoor poison from multimodal models like CLIP, called CleanCLIP. CleanCLIP finetunes a CLIP style model on a dataset of clean image-text pairs using a combination of contrastive and self-supervised loss (MMCL + SSL) and claims that this finetuning can remove poison from a model that was backdoored. However, the authors only test the CLIP style models that were trained using contrastive loss in the pretraining stage. In our work, we test the robustness of CleanCLIP if the pretraining objective of the CLIP style model is changed from purely contrastive (MMCL) to a combination of contrastive and self-supervised (MMCL + SSL), which is the same objective CleanCLIP uses to finetune to remove poison. What we find is surprising: CleanCLIP fails to generalize to this pretraining objective and is unable to remove poison from such a backdoored model, thus exposing a lack of robustness. We also analyze several hypotheses of why this happens; however, we do not reach a conclusion in our work.

Finally, in this thesis, I also propose a novel robustness approach for safety classification. Safety classification models like LlamaGuard and WildGuard are 8B scale LLMs finetuned on safety datasets. However, most of the training datasets of these models are in English, which leads to failure of generalization of these guard models to low-resource languages, both natural and ciphers. Attackers found this and were using low-resource languages as a prominent strategy to attack models. In this work, I propose Omniguard, which uses the internal representations of a model in order to classify the prompts as safe and unsafe. The intuition of our work is based on the phenomenon observed by previous works that found that prompts with the same semantic meaning but expressed in different languages still produce very similar internal representations. We use this phenomenon to our advantage by building a classifier in the space of internal representations of the model. Interestingly, this phenomenon generalizes not just across languages but also across modalities to

images and audio. Therefore, Omniguard is the first approach that can classify the safety of prompts across multiple languages and modalities with high accuracy. When compared to baselines, it is the most accurate, robust, and also the fastest method for safety classification.

Overall, I propose five works in this thesis that make ML models more trustworthy by making them more interpretable and robust. All of my works enable these properties while also respecting real-world constraints like having only black-box access to the model and being extremely fast during inference – which are all attributes that enable the approach to be more deployable.

1.1 Previously Published Work

This thesis borrows content from these previously published five works:

1. Amortized Generation of Sequential Algorithmic Recourses for Black-Box Models [247].
2. Recxplainer: Post-hoc Attribute-based Explanations for Recommender Systems [248]
3. How many Van Goghs does it take to van Gogh? finding the imitation threshold [249]
4. Effective Backdoor Mitigation depends on the Pre-training Objective [250]
5. OMNIGUARD: An Efficient Approach for AI Safety Moderation Across Modalities [251]

Chapter 2

Background and Related Work

In this chapter I provide the background and related work for various works in the next chapters.

2.1 Background for Counterfactual Explanations

This section provides background about the social implications of ML models and techniques to address concerns, along with a brief introduction to Reinforcement Learning.

2.1.1 Fairness, Accountability, and Transparency of AI and ML

Fairness and explainability of an ML model are two major themes in the broad area of equitable ML learning research.

Fairness research mostly proposes algorithms that learn a model that does not discriminate against individuals belonging to disadvantaged demographic groups. Other possibilities of intervention lie in modifying the training data itself. Demographic groups are determined by values of sensitive attributes prescribed by law, e.g., race, sex, religion, or the nation of origin. ML models can get biased against certain demographic groups because of the bias in their training data, specifically label bias and selection bias. Label bias occurs due to manual biased labeling of datapoints belonging to a demographic group, e.g., if individuals from the black community were denied loans in the past irrespective of their ability to pay back, this gets captured in the labeled part of the data from which the model can learn. Selection bias occurs when specific non-representative subsets of a demographic group are selected, which captures potentially correlations between

the prediction target and a specific demographic group, e.g., selecting only defaulters from a demographic group in the training data. More than 20 definitions of fairness of an ML model have been proposed in literature [245]. Dunkelau and Leuschel [68] summarize some of the significant research advances that have been made in fairness research, and is a comprehensive introductory text for understanding the categorization and direction of research.

Explainability research can be broadly divided into model explanation and outcome explanation research problems [91]. The model explanation problem seeks to search for an inherently interpretable and transparent model with high fidelity to the original complex model. Linear models, decision trees, and rule sets are examples of inherently interpretable models. There exists techniques to explain complex models like neural networks and tree ensembles using interpretable surrogates like decision trees [50, 129, 44, 64] and rule sets [55, 7]. There also exist approaches that can be applied to black-box models [102, 130, 291].

The outcome explanation problem seeks to find, for a single datapoint and prediction from a model, an explanation of why the model made its prediction. The explanation is either provided in the form of the importance of each feature in the datapoint, or the form of example datapoints. The first class of methods are called feature attribution methods and are grouped into model-specific [288, 208] and model-agnostic [181, 192, 239] kinds. Example-based approaches return a few datapoints that either have the same class label as the original datapoint or a different class label. The motivation for the first is to provide a set of datapoints that must be similar in the input space. The motivation for the second is to provide a set of datapoints that serves as a target to achieve in case the individual wants to receive the alternative label. The second set of datapoints can be referred to as *counterfactual explanations*.

Counterfactual explanations are applicable to supervised machine learning where the desired label has not been obtained for a datapoint. Most research in counterfactual explanations assumes a classification setting. The supervised ML setup consists of several labeled datapoints, which are inputs to the algorithm, and the aim is to learn a function mapping from the input datapoints (with say m features) to labels. In classification, the labels are discrete values. The input space is denoted by $\mathcal{X}^m$ and the output space is denoted by $\mathcal{Y}$. The learned function is the mapping $f : \mathcal{X}^m \rightarrow \mathcal{Y}$ is used to make predictions. We expound on counterfactual explanations and their desirable properties in Section 3.2.

Major beneficiaries of explainable machine learning include the healthcare and finance sectors, which have

a huge social impact [237]. We point the readers to surveys in the area of explainable machine learning [3, 32, 91].

2.1.2 Reinforcement Learning

Reinforcement Learning (RL) is one of the three broad classes of machine learning, along with supervised and unsupervised learning. In RL, the goal is to explore a given environment and to learn a policy over time that dictates what action should be taken at a given state. The exploration happens with the help of an agent. Therefore, a policy is a mapping from a state to an action. When an action is taken at a state, the environment returns with the new state and a reward. A good policy aims to maximize the reward over time. The calculation of the new state is facilitated through the transition function, whereas the calculation of the reward is done using the environment. Naturally, the agent can either learn policies that are greedy and only focus on immediate reward or learn policies that focus on reward in the long-term. This trade-off is controlled by a discount factor called γ, whose value lies between 0 and 1 (inclusive of 0 and 1). States can either be discrete or continuous. Similarly, actions can also be either discrete or continuous. An RL problem can be expressed in terms of a Markov Decision Process (MDP), which has five components. We illustrate each of them using the game of chess.

- State space $\mathcal{S}$, which are states an agent might explore. In chess, these are the 64 squares that an agent can move to.
- Action space $\mathcal{A}$, which are the possible actions an agent can take. These might be restricted based on the current state. In chess, the actions depend on the game pieces like a king, queen or pawns, and the given position on the chessboard. The action space is the union of all possible actions.
- Transition function `T` which given the current state and action, find the new state that the agent will transition to, e.g., moving the pawn by 1 unit north end up putting the agent in the state that is one unit north of its current state. Transition functions can be deterministic or stochastic (see Section 3.3).
- Reward function `R` passes the reward to the agent given the action, the current state, and the new state. This reward signal is the main factor that the agent uses to learn a good policy, e.g., winning a game would pass a positive reward, and losing the game would send a negative reward to the agent.
- Discount factor γ is associated with the nature of the problem at hand. This is used to decide the

trade-off between immediate and long-term rewards.

Many algorithms have been developed to efficiently learn an agent, given the environment like value iteration, policy iteration, policy gradient, actor-critic methods [229].

2.2 Related Works for Counterfactual Explanations

Literature in counterfactual explanations for ML is relatively recent, with the first proposed algorithm in 2017. Wachter et al. [255] proposed finding counterfactuals as a constrained optimization problem where the goal is to find the minimum change in the features such that the new datapoint has the desired label. This approach was gradient-based, did not consider actionability among features, did not adhere to data manifold or respect causal relations, and the optimization problem needed to be solved for generating a CFE for each input datapoint. Other desiderata mentioned in Section 3.2 were proposed by other papers: 1) approaches that generate multiple, diverse counterfactuals for a single input datapoint [164, 51, 157, 122, 216, 196], 2) approaches that generate counterfactual for black-box models and are model-agnostic [135, 136, 90, 87, 216, 188, 264, 183, 126, 51], 3) approaches that generate CFEs adhering to data manifold [62, 63, 119, 243, 157, 175, 126, 138, 51, 121], 4) approaches that generate CFEs that respect causal relations [157, 124, 125], 5) approaches that generate amortized CFEs [157].

Mahajan et al. [157] was the first to propose an approach that can generate multiple CFEs for many datapoints, after optimizing once, therefore *amortized* CFEs, but their approach is gradient-based and therefore works only for differentiable models and it not black-box. Out of the previous approaches that respect causal relations, only Mahajan et al. [157] works with partial SCM, while others require complete causal graph or complete SCM [124, 125], which are mostly unavailable in the real world. Structural Causal Models (SCM) consist of the exogenous and endogenous variables involved in the data generation process.

All the previous works give a single-shot solution for getting to a counterfactual state from an input datapoint. Our approach overcomes this limitation by proposing a novel algorithm that generates *sequential* CFEs.

Verma et al. [246] and Karimi et al. [123] have collected and summarized recent works in counterfactual explainability. We point the readers to these surveys for an excellent in-depth review of the research landscape in this area.

2.2.1 Counterfactual vs. Contrastive explanations

There is ongoing discussion on the exact definition of counterfactual explanation, with some researchers advocating to call it contrastive explanations. Dhurandhar and Shanmugam [61] have captured the precise difference in a recent article. They mention that the counterfactual explanations as introduced by Wachter et al. [255] are almost the same as contrastive explanations. These explanations seek to find the minimal changes to the input such that the prediction from the ML model changes. On the other hand, counterfactuals are a function of the datapoint, its prediction, the ML model, and the data generating process that created that datapoint. Pearl [177] describes three steps for generating counterfactuals:

1. Abduction: This is the process of conditioning on the exogenous variables in the data generation process.
2. Intervention: This is the process of making a sparse change on a specific observable variable.
3. Prediction: This is the process of using the exogenous variables identified in the first step and propagating the intervention to generate the counterfactual.

We agree with this framing. Therefore, counterfactual explanations amount to much more perturbing of the input datapoint — as in the case of contrastive explanations, which are tied to the data generating process. Indeed, it is our belief that our proposed framework captures these concerns, if data regarding causal interactions is available.

We take note of this distinction and therefore have adherence to causal relations as a desiderata of counterfactual explanations (Section 3.2).

2.3 Related Works for Explainability in Recommender System

In this section we provide a brief review of relevant works that cover the general topic of explainability in recommender systems, as well as post-hoc explainability and attribute-based explainability as specific topics.

2.3.1 Explainability in Recommender Systems

Explainable recommender systems provide recommendations along with an explanation for doing so. The term 'explainable recommendation' was introduced by Zhang et al. [286] in 2014; however, papers were talking about the benefits of providing explanations much earlier [235, 236, 220].

Most previous methods for explainability in collaborative filtering-based (CF-based) recommender systems provide explanations in user-based or item-based fashion [285]. The approaches that generate these explanations can either be model-specific or model agnostic, the former kind constituting the majority of previous approaches. Models-specific approaches train inherently interpretable models by either constraining the embedding space or developing custom recommender system architectures that are not usually generalizable. For example, Zhang et al. [286] propose extracting user preferences over attributes from their reviews and using that to provide explanations for recommendations. The authors train an explicit factor recommender model that takes the user's preference over attributes as input during training the recommender model. Such an approach is restricted to situations where the user reviews over items are available in abundance and also might suffer from the accuracy-interpretability trade-off. In another approach, Abdollahi and Nasraoui [2] add constraints to the latent space of the matrix factorization models to encourage explainability. Even though their approach only provides user and item-based explanations, their approach seriously damages accuracy to provide interpretability. There exist many other approaches in this category [39, 37, 13, 230, 150, 159, 267]. As evident, the major downside of these approaches is that they are *not* post-hoc, i.e., they need to modify the recommender system architecture in order to provide explanations – and this is problematic for two reasons: 1) there is a potential accuracy-interpretability trade-off, and 2) real-world recommender systems are trained and fine-tuned over large datasets, and their developers are rarely willing to train a new model from scratch in order to provide explanations [244]. Hence post-hoc explainability techniques attract a lot of attention, especially from an industry perspective [247, 163].

2.3.2 Post-hoc Explainability in Recommender Systems

There only exist a handful of explainability techniques in this category. Peake and Wang [176] use a data-mining approach to provide item-based explanations in a post-hoc manner. Nóbrega and Marinho [169] propose LIME-RS which draws motivation from the original LIME paper [192]. LIME-RS samples items

close to a recommended item, and learns a linear regression model on these samples, and uses the regressor to provide an explanation that can be either item-based or attribute-based (the latter can be provided if the attributes of the items are appended to the items embeddings when training the regression model). LIME-RS is the only previous approach that provides attribute-based explanations for CF-based recommender systems and is a post-hoc explanation technique. However, LIME-RS is not amortized – it trains a separate regression model for each recommended item and hence is not scalable to large scale datasets (see Section 4.3 for evaluation of LIME-RS on large scale datasets). Cheng et al. [42] propose a technique that uses influence functions to find the influence of training ratings on a particular recommendation, and the ratings with the greatest influence are served as explanations. Hence, this technique also only provides item-based explanations.

2.3.3 Attribute-Based Explainability in Recommender Systems

Unlike user-based and item-based explanations, attribute-based explanations explain a recommendation based on the user's personalized attribute preferences, thereby enhancing the system's overall persuasiveness and trustworthiness [258, 75]. As mentioned earlier, there do exist previous works that provide explanations that utilize the user's preference over interpretable attributes of an item. Most of these works utilize the reviews provided by the users to capture this preference (and use that to train the recommender system itself [286, 159, 107]). However, such reviews might not exist even in the presence of ratings. For example, Netflix scrapped its review section due to low user participation recently [265]. Secondly, there is no guarantee of mention of interpretable attributes in the reviews that can be used to learn the user's preference. Even the datasets used in these approaches had very sparse user reviews; over 77% of the users had only *one review* [98], making the inference of users' preferences challenging.

On the other hand, `RecXplainer` also provides attribute-based explanations; however, it utilizes the attributes of an item that is available in their metadata, e.g., movie or game genres, and hence *does not* depend on user reviews. Similar to `RecXplainer`, AMCF [174] learns users' preferences over item attributes by training an attention network. So it provides attribute-based explanations like our approach; however, AMCF is not a post-hoc technique – it needs to be trained along with the recommender system. Similarly, Wang et al. [258] proposes CERec to determine the attributes that are important for a user-item pair (in a

counterfactual manner); however, CERec is also not a post-hoc technique. Wang et al. [259] propose a post-hoc RL-based approach that generates attribute-based explanations for a user; however, like most previous approaches, it acquires the attributes from the user reviews – whose downsides are aforementioned. Hence our approach sits at the intersection of attribute-based (*without* requiring user reviews), post-hoc explanation techniques and generating amortized explanations. To the best of our knowledge, `RecXplainer` **is the first technique** to have all these desirable properties.

2.4 Background and Related Works for Mimetic

2.4.1 Imitation in Text-to-Image Models:

Somepalli et al. [223], Carlini et al. [28] demonstrated that diffusion models can memorize and imitate duplicate images from their training data (they use 'replication' to refer to this phenomenon). Casper et al. [33] corroborated the evidence by showing that these models imitated art styles of 70 artists with high accuracy (as classified by a CLIP model) when prompted to generated images in their styles (a group of artists also sued Stability AI claiming that their widely-used text-to-image models imitated their art style, violating copyright laws [202]). However, these works did not study how much repetition of a concept's images would lead the model to imitate them. Studying this relation is important as it serves to guide institutions training these models who want to comply with copyright and privacy laws.

2.4.2 Mitigation of Imitation in Text-to-Image Models:

Several works proposed to mitigate the negative impacts of text-to-image models. Shan et al. [213] proposed GLAZE that adds imperceptible noise to the art works such that diffusion models are unable to imitate artist styles. A similar approach was proposed to hinder learning human faces [212]. Wang et al. [261] proposed adding noise to training images, which can be used to detect if a model has been trained on those images. Lu et al. [153] propose pushing the generated images away from the distribution of training images to minimize mitigation. Kumari et al. [132], Gandikota et al. [77] proposed algorithms to remove specific styles, explicit content, and other copyrighted material learned by text-to-image models. On a related note, Xie et al. [268] proposed Diffusion-ReTrac that finds training images that most influenced a generated image, and thereby

provide a fair attribution to training data contributors.

Dataset Issues and Privacy Violations The advancement in text-to-image capabilities, largely due to large datasets, is accompanied by concerns about the training on explicit, copyrighted, and private material [17] and imitating such content when generating images [34, 110, 254, 117, 97]. For example, [17] and [233] found several explicit images in the LAION dataset, and Getty Images found millions of their copyrighted images in LAION [254]. Issues around imitation of training images has especially plagued artists, whose livelihood is threatened [202, 213], as well as individuals whose face has been used without consent to create inappropriate content [110, 9]. The imitation threshold would be a useful basis for copyright infringement and privacy violation claims in such cases.

Training Data Statistics and Model Behavior Pretraining datasets are a core factor for explaining model behavior [72]. [189] found that the few-shot performance of language models is highly correlated with the frequency of instances in pretraining dataset. [240] bolster this finding by demonstrating that the performance of multimodal models on downstream tasks is strongly correlated with a concept's frequency in the pretraining datasets. In addition, [29] show that language models more easily memorize highly duplicated sequences. Based on these findings, we can intuitively conjecture that concepts that are repeated enough number of times will be learned and then imitated by models. In this work, we study this question, and quantify the number of images to appear in the training data that are required to imitate a concept.

2.5 Related Works for CleanCLIP

2.5.1 Contrastive Learning

Contrastive learning was formally established in seminal works by Bromley et al. [22], Chopra et al. [45], Hadsell et al. [92] that has evolved, giving rise to contemporary algorithms such as CPC [171], DCL [276], SimCLR [38], and NNCLR [69]. [1] These approaches, at their core, share a common objective: bringing similar elements closer in representation space while pushing dissimilar ones apart.

Radford et al. [185] extended this idea beyond a single modality to provide a dual-encoder approach for learning a shared representation space between image and text, called CLIP. Images and their corresponding

[1] We refer the readers to Balestriero et al. [10] for more development on self-supervised learning.

captions are brought close, while the dissimilar images and captions are pushed away. Jia et al. [115] further extended this paradigm to handle noisy billion-scale datasets, demonstrating exceptional zero-shot accuracy across benchmarks like Imagenet-1K [56], MS-COCO retrieval, and robustness against variations in Imagenet-V2/R/A/C. Since then, there have been several improvements to the zero-shot accuracy by adding components to the loss term. CyCLIP [81] imposes additional consistency regularization; SLIP [165] applies an additional self-supervision loss within image modality and was further unified by UniCLIP [139]. DeCLIP [145] additionally uses kNN augmentation; FILIP [275] additionally applies CLIP loss to fine-grained token representations. CLIP performance has also been improved by additional captioning loss [280].

2.5.2 Backdoor Attacks and Defense

In backdoor attacks, the adversary poisons a small fraction of the training data by perturbing the images/labels to manipulate the test time behavior. A prevalent form of this attack involves adding a trigger, such as a random pixel patch, into a small subset of the training dataset [226, 89, 238]. During inference, models perform normally on images without the triggers but exhibit catastrophic failures when tested with the triggered images, erroneously predicting the labels targeted by the adversary. While the study of backdoor attacks has historically centered on supervised learning, recent attention has extended to self-supervised [197] and multimodal representation learning [11, 27, 30].

The most common defense strategies against backdoor attacks primarily revolve around the identification and detection of poisoned examples [227, 78, 256, 273, 147, 271, 270]. However, alternative approaches have emerged, such as defense through knowledge distillation [279] and robust training procedures involving data augmentation [20]. Despite these efforts, research by Carlini and Terzis [27], Carlini et al. [30] shows that poisoning even an exceedingly small fraction of the training datapoints (as little as 0.002%) can substantially impact model performance. Consequently, the effectiveness of detection-based methods in the context of multimodal pre-training remains uncertain. To address this challenge, Bansal et al. [11] proposed CleanCLIP, a finetuning based procedure using a combination of MMCL and SSL losses, designed to clean poisoned CLIP models, assuming access to a small, guaranteed to be poison-free dataset.

2.5.3 Our Work

Our objective is to decipher the amenability of CleanCLIP to remove poison from pre-trained models under varying conditions like different pre-training objectives, lack of completely clean data, and lack of knowledge of the specific backdoor attack. Since intramodal self-supervision loss has enhanced classification accuracy for multimodal models, we investigate CleanCLIP effectiveness when models are pre-trained with a combination of MMCL and SSL objectives vs. when just pre-trained with the MMCL objective. We also investigate its effectiveness when the finetuning data has a few poisoned datapoints and examine the stopping criterion for finetuning when the knowledge of the specific backdoor attack is unavailable.

2.5.4 Why we choose to study CleanCLIP?

Defense against backdoor attacks in multimodal models is an emerging research area with a handful of proposed approaches. Given the ever-increasing usage of off-the-shelf available pretrained models on the platforms such as `huggingface`, it is important to be able to remove poison from an already trained model. This is also important because of the prohibitive costs of training these large models from scratch, even if a poison-free dataset suddenly becomes available. Among the proposed approaches, CleanCLIP [11] was the only one that was shown to be able to remove poison from an already trained model. All the other defense methods [271, 270, 113] propose various *train-time* interventions that help prevent the model from learning the backdoor. Therefore, none of the techniques available in 2023 were applicable when a model has already been trained and is backdoored. Another line of work that might seem relevant is ML unlearning [21, 82, 25] which aims to remov the influence of specific training data points (the "forget set") from a trained model, such that the resulting model behaves as if it had never seen that data, but these techniques require to know the datapoints that one needs to forget which is unknown at both the training and post-training stage for backdoor attacks.

In this part, I present three of my works that aim to provide explanations for ML models, classification models and recommender systems.

Chapter 3

Amortized Generation of Sequential Algorithmic Recourses for Black-box Models

Explainable machine learning (ML) has gained traction in recent years due to the increasing adoption of ML-based systems in many sectors. *Algorithmic Recourses* (ARs) provide "what if" feedback of the form "if an input datapoint were x' instead of x, then an ML-based system's output would be y' instead of y." ARs are attractive due to their actionable feedback, amenability to existing legal frameworks, and fidelity to the underlying ML model. Yet, current AR approaches are single-shot — that is, they assume x can change to x' in a single time period. We propose a novel stochastic-control-based approach that generates *sequential* ARs, that is, ARs that allow x to move stochastically and sequentially across intermediate states to a final state x'. Our approach is model-agnostic and black box. Furthermore, the calculation of ARs is amortized such that once trained, it applies to multiple datapoints without the need for re-optimization. In addition to these primary characteristics, our approach admits optional desiderata such as adherence to the data manifold, respect for causal relations, and sparsity—identified by past research as desirable properties of ARs. We evaluate our approach using three real-world datasets and show successful generation of sequential ARs that respect other recourse desiderata.

3.1 Introduction

Machine learning (ML) models are increasingly used to make predictions in systems that directly or indirectly impact humans. This includes critical applications like healthcare [73], finance [219], hiring [209], and parole [231]. To understand ML models better and to promote their equitable impact in society, it is necessary to assess stakeholders'—both expert [106] and layperson [198]—comprehension of and needs for general observability into their systems [182, 70]. The nascent Fairness, Accountability, Transparency, and Ethics in machine learning (aka "FATE ML") community conducts research to develop methods to detect (and counteract) bias in ML models, develop techniques that make complex models explainable, and propose policies to advise and adhere to the regulations of algorithmic decision-making (see Section 2.1). Here, we focus on ML model explainability.

Research in explainable ML is bifurcated. One high-level approach aims to develop inherently interpretable models such as decision trees and linear models [195]. Another high-level approach aims to utilize existing complex classification techniques (such as deep neural networks) but to bolster them with surrogate models that can render their predictions and/or internal processes understandable [3]. This is achieved through explaining models holistically (global explanation) or single predictions from the model (local explanation).

Algorithmic Recourses (ARs). ARs find the minimal (defined later) change in a datapoint such that the ML model ends up classifying the new datapoint in the desired class. Such a new datapoint(s) is termed as a counterfactual. (We provide an in-depth discussion of terminology in Section 2.2.1.) For example, if an individual were denied a loan request, a recourse might tell them that their request would be approved if they were to increase their income by $2000. ARs provide a precise recommendation and are therefore more actionable than other forms of local explainability, like feature importance. Recent research in this area has aimed to ensure ARs are actionable and useful by incorporating additional desiderata into the recourse generation problem. As described in Section 3.2, these include notions of sparsity, causality, and realism of ARs, among others. What is needed [see, e.g., 246, 46, 123] is a generalized approach that can accommodate such varied constraints and can also be computed efficiently.

Operationalizing ARs. We propose a novel approach (FASTAR) for generating ARs by translating a given recourse generation problem into a Markov Decision Process (MDP). FASTAR aims to learn a policy that can generate ARs for a given data distribution. Upon learning that policy once, it can generate ARs for multiple

datapoints (from that distribution) without the need to re-optimize (which is required by most previous approaches; see Section 2.2). Thus, FASTAR *amortizes* the cost of repeatedly computing ARs. FASTAR also allows enforcing desirable properties of ARs, such as closeness to the training data distribution (data manifold), respect of causal relations between the features, and mutability and actionability of different features. FASTAR works for *black-box* models and is therefore *model agnostic*.

Via the learned policy, FASTAR outputs ARs as a sequence of steps that lead an individual to a counterfactual state. To our knowledge, we are the first to leverage techniques from stochastic control to provide such *sequential ARs* [186, 167]. That sequence can also adhere to particular *sparsity* constraints (e.g., only one feature changing per step).

Sequential and "rolled out" ARs have several advantages, directly addressing gaps identified by recent survey papers [246, 46, 123] and workshops [71]: 1) Action sparsity allows an individual to focus their effort on changing a small number of features at a time; and 2) Presentation of ARs as a set of discrete and sequential steps is closer to real-world actions, rather than one-step continuous change, which most previous approaches do. Singh et al. [218] recently conducted a user-study with 54 participants, wherein each of them was presented with 15 scenarios and asked if they preferred one-shot or directed sequential AR in that scenario. When overall results were pulled, the study concluded a preference for sequential ARs with high confidence.

Figure 3.1 shows an example of sequential ARs that are generated for an individual whose loan request was denied (shown by ✘). Instead of a one-shot solution, FASTAR delineates all intermediate steps to reach a counterfactual state (shown by ✔). FASTAR also models the stochastic factors like the duration to complete a BSc degree, no or part-time job during the course, and the salary variance in the new job after graduation, which lead to different recourse paths and hence different counterfactual states (as shown in Figure 3.1).

In summary, our contributions are:

1. A novel algorithm that translates an AR problem into a Markov decision process (MDP). To the best of our knowledge, our stochastic-control-based approach is the first to address several roadblocks to using ARs in practice that have been identified by the community [246, 46, 123].
2. The first approach that generates sequential and amortized ARs, and also works for black-box models.
3. An extensive evaluation with three real-world datasets and nine baselines.

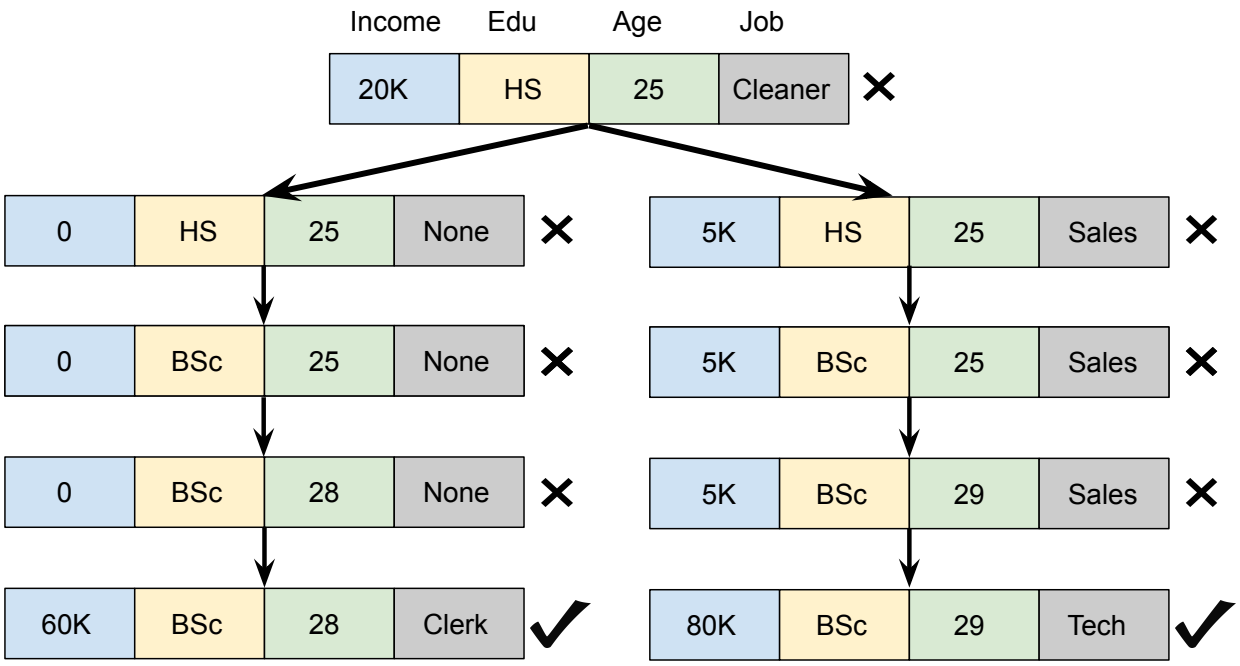


Figure 3.1: Example of Stochastic Algorithmic Recourses. Starting with a datapoint (✘ denotes undesired class prediction), FASTAR can stochastically generate ARs that lead to different counterfactual states (✔ denotes desired class prediction).

3.2 Desiderata of *Practical* ARs

The overarching goal of an AR is to provide practical guidance to an individual seeking to change their treatment (e.g., class label) by a deployed ML model. Apart from the necessary property of an AR having a desired class label, other desiderata have been identified in the literature, enumerated here:

- *Actionability*: ARs should only recommend changes to the features that are actionable [241, 121, 51]. Actionable features are dataset and preference dependent.
- *Sparsity*: Social studies have argued that smaller explanations are more comprehensible to humans [161]. Therefore, ARs should make changes to a small set of features [243, 122].
- *Data manifold*: To obey the correlations between features, their input domain, and to be realistic, ARs should adhere to the training data manifold [63, 121, 51].
- *Causal constraints*: In order to adhere to real-world constraints in ARs, causal constraints between features must be respected. They can encode facts like age cannot decrease or increase in education level increases age [157].

Table 3.1: Desiderata comparison of various AR generating approaches. FASTAR is the **first and only one** which satisfies all desiderata.

Approach	Actionability	Sparsity	Agnostic	Black-box	Amortized	Manifold	Constraints
CFE Expl. [255]	✗	✓	✗	✗	✗	✗	✗
Recourse [241]	✓	✓	✗	✗	✗	✗	✗
CEM [63]	✗	✓	✗	✗	✗	✓	✗
MACE [122]	✓	✓	✗	✗	✗	✗	✗
DACE [121]	✓	✗	✗	✗	✗	✓	✗
DiCE [164]	✓	✓	✗	✗	✗	✗	✗
DiCE VAE [157]	✓	✗	✗	✗	✓	✓	✓
Spheres [136]	✗	✓	✓	✓	✗	✗	✗
LORE [90]	✗	✓	✓	✓	✗	✗	✗
Weighted [87]	✗	✗	✓	✓	✗	✗	✗
CERTIFAI [216]	✓	✗	✓	✓	✗	✗	✗
Prototypes [243]	✗	✓	✓	✓	✗	✓	✗
MOC [51]	✓	✓	✓	✓	✗	✓	✗
FASTAR	✓	✓	✓	✓	✓	✓	✓

- *Model-agnostic*: For wide-spread applicability, AR generating approaches should be model-agnostic [136, 90].
- *Black-box models*: For applicability to proprietary ML models, AR generating approaches should work for black-box models [216].
- *Amortized*: An *amortized* approach can generate ARs for several datapoints without optimizing separately for each of them. Such an approach is effective for deployment [157].

FASTAR satisfies all the above desiderata. To the best of our knowledge, it is the first approach to do so (see Table 3.1). The choice of action space helps produce ARs that consider actionability among features and are sparse. It only modifies the actionable features. Its ARs are realistic as they adhere to the training data manifold and respect causal relations between features. FASTAR works for black-box models and, therefore, is model-agnostic. It learns a policy that can produce ARs for several input datapoints without the need to optimize again; and, therefore, generates amortized ARs.

3.3 Examples of Translating ARs to MDPs

We now give two examples of translating an AR problem into an MDP. Once modeled as an MDP, we can use various off-the-shelf algorithms (from planning or RL) to learn a policy to generate ARs.

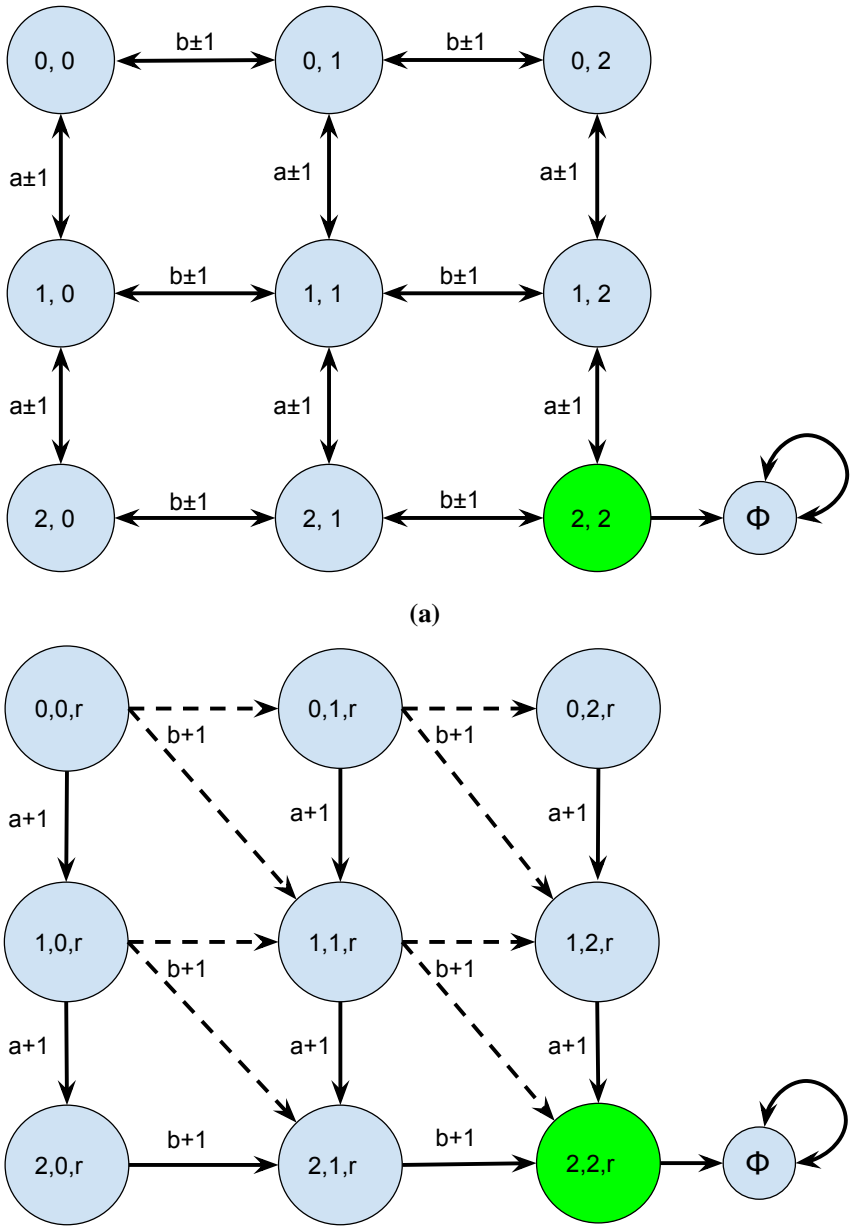


Figure 3.2: Transition function for the two examples. Circles show all the states, and edges show possible transitions. 1) Left-hand side shows the transition function for a dataset with two features a and b, with no restrictions on the values that both of them can take within the input domain. The transition edges are therefore bidirectional. 2) Right-hand side shows the transition function for a dataset with three features: age (a), education-level (b), and race (r). The transition edges are unidirectional as both age and education cannot decrease. Since race is immutable, there are no actions for r. Since an increase in education stochastically affects age, the dashed edges represent a 50% probability of transition.

Example 1: Consider two categorical features $\mathsf{a}, \mathsf{b} \in \{0, 1, 2\}$. The combinations of possible values for a and b form the state space for the MDP (represented by $\mathcal{S}$). The directed edges in Figure 3.2a show that upon taking a specific action, an agent can move from one state to another, e.g., it transits from state (0, 1) to (0, 2) by taking the action b+1, which increments the value of feature b by 1. Actions a+1 and a-1 respectively increase and decrease the value of feature a by 1 (similarly for feature b. These actions constitute the action space for the MDP (represented by $\mathcal{A}$). The third component of the MDP is the transition function, which is represented by $T : s \times a \rightarrow s'$. This denotes that if an agent takes action a in state s then it will move to state s'. This transition function is deterministic because taking the action a in state s will always land the agent in the state s'.

The final component of the MDP is the reward function. Taking action costs something (negative reward), and reaching desirable states generates a positive reward. In this MDP, taking any action costs a constant amount of 1 and reaching the terminal state (ϕ) gives a reward of +10. The terminal state (ϕ) can only be reached via (2,2) (using any action), the state in green color. All actions in the terminal state lead to itself with 0 cost. This represents the situation in which an ML model classifies only (2,2) in the desired class.

The aim is to learn a policy that reaches a terminal state from any state at the lowest cost (e.g., taking the fewest number of steps). Cost (or reward) can be discounted in the traditional way using a discount factor $\gamma \in [0, 1)$.

Formally, for this example with a discrete state space and discrete action space, our MDP is:

- States = $\{s \in \mathcal{S} : \{0, 0\}, \{0, 1\}, \{0, 2\}, \{1, 0\}, \dots\}$.
- Actions = $\{a \in \mathcal{A} : \mathsf{a+1}, \mathsf{a\text{-}1}, \mathsf{b+1}, \mathsf{b\text{-}1}\}$.
- Transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$
- Reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$.
- Discount factor $\gamma \in [0, 1)$, capturing the tradeoff between current and future reward.

Our goal is finding a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that, given a state $s \in \mathcal{S}$ (an input datapoint), returns an action $a \in \mathcal{A}$ that represents the best first step to take to reach a new state, hopefully closer to the ML model's decision boundary. FASTAR would then call this precomputed policy repeatedly to find an optimal path to a counterfactual state.

Example 2: Now, consider a more realistic dataset having 3 features: age (denoted by a), education-

level (denoted by b), and race (denoted by r). This is accompanied by real-world constraints like age and education-level cannot decrease, education-level affects age, and race is immutable. When we increase the education-level (b) by 1, there is a 50% chance that age group (a) will remain the same and a 50% chance that it will increase by 1. These interactions between features can be captured by a structural causal model (SCM), as we discuss in Section 3.4. The transition function for the MDP representing the AR problem for this dataset is, therefore, stochastic.

Defined formally, here are the components for this MDP:

- States = $\{s = \{i, j, k\}$, where some constraint make it a valid tuple$\}$.
- Actions = $\{a \in \mathcal{A}$: a+1, a-1, b+1, b-1$\}$.
- Transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \to \{0,1\}$ s.t. $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \ \sum_{S'} T(S, A, S') = 1$.
- Reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.
- Discount factor $\gamma \in [0, 1)$.

Figure 3.2b shows the transition function for this problem. The action that increases the education-level (b) now has a probabilistic transition to two destination states, represented by dashed unidirectional edges. Each transition edge has a 50% probability of occurrence. Unidirectionality comes from the fact that the education level cannot decrease. The edges change feature a are also unidirectional as age cannot decrease. The reward function is identical to the previous example; optionally, it can be changed to accommodate adherence to the data manifold (Section 3.2) or having different costs for changing different features, which we describe in Section 3.4.

3.4 An Algorithmic Approach for Generating MDPs from AR Problems

We now present a general approach for translating an AR problem setup into an MDP. Algorithm 1 generates all components of an MDP: state space, action space, transition function, reward function, and additional parameters such as discount factor. We detail this process below.

State space. Features can be broadly categorized into numerical (Num) and categorical (Cat) kinds. Numerical features can take real number values within a specified domain, while categorical features are mapped to a set of integers. Consequently, the state space $\mathcal{S}$ of our MDP (algorithm 1) consists of the product of the continuous domains for numerical features (a subset of $\mathbb{R}^{|\text{Num}|}$) and product of the integer

ALGORITHM 1: Generate MDP from an Algorithmic Recourse Problem

Input : Training Dataset (D), ML model (f), Structural Causal Model (SCM), Numerical actionable features (NumA), Categorical actionable feature (CatA), Data Manifold distance function (`DistD`), Data Manifold adherence (λ), Desired Label (L), Distance Function (`DistF`), Discount Factor (γ)

Output: MDP

```
// States consist of all numerical (Num) and categorical (Cat) features.
State space S ⊆ ℝ^|Num| × ℤ^|Cat|
// Actions change the actionable numerical and categorical features.
Action space A ⊆ ℝ^|NumA| × ℤ^|CatA|; denote actions A ∈ A
Function Reward(f, L, CurrState, A, D, λ, DistD, SCM)
    NextState ← Transition(CurrState, A, SCM)
    if argmax(f(NextState)) = L then
        CFReward ← Pos // High positive reward
    else
        CFReward ← f(NextState)[L] // Probability of classification in the desired class
    end
    return DistF(CurrState, A, D) // action cost
    +λ * DistD(NextState, D) // Manifold distance cost
    +CFReward // Counterfactual label reward
Function Transition(CurrState, A, SCM)
    // Action does not violate feature domain and unary constraints
    if Allowed(A) & InDomain(A) then
        NextState ← CurrState + A // Modify features
    else
        return CurrState
    end
    // Modify the endogenous features
    for V ∈ SCM do
        if A ∈ Parent(V) then
            NextState[V] ← F(U) // Stochastic or deterministic update of endogenous
                features
    end
    return NextState
MDP ← {S, A, Transition, Reward, γ}
```

domains for categorical features (a subset of $\mathbb{Z}^{|\text{Cat}|}$).

Action space. To facilitate capturing actionability [241] and causal relationships between features [124], we further categorize features as follows:

- *Actionable* features can be directly changed by an individual, e.g., income, education level, age.
- *Mutable but not actionable* features are mutable but cannot be modified directly by an individual, e.g., a

credit score cannot be directly changed by a person; it changes due to changes in features like income and credit history.

- *Immutable* features cannot change, e.g., race, birthplace.

The agent is permitted to change only the actionable numerical (real-valued) and categorical features (denoted by NumA and CatA). Consequently, the action space $\mathcal{A}$ is a subset of $\mathbb{R}^{|\mathsf{NumA}|} \times \mathbb{Z}^{|\mathsf{CatA}|}$ (algorithm 1). Categorical features are changed within their discrete domain, while numerical features are changed within their continuous domain. Algorithm 1 further enforces the infeasibility of out-of-domain actions.

Transition function. The transition function (algorithm 1) finds the modified state when an action is taken. This function is influenced by the structural causal model (SCM), which is an *optional* input to Algorithm 1. An SCM consists of a triplet $\mathsf{M} = \langle \mathsf{U}, \mathsf{V}, \mathsf{F} \rangle$. U is the set of *exogenous* features and V is the set of *endogenous* features. In terms of a causal graph, the exogenous features U consist of features that have no parents, i.e., they can change independently. The endogenous features V consists of features that have parents in U and/or other features in V. They change as a result of a change in their parents. F is the set of functions that determine the relationship between exogenous and endogenous features. They are termed as structural equations.

Since knowing the exact SCM is often infeasible, Mahajan et al. [157] overcome this limitation by utilizing constraints from domain knowledge. Algorithm 1 also accepts such constraints in unary (Un) and binary (Bin) forms. Even if this does not provide us with the precise functional form of the constraint, its nature can help the FASTAR's recourses to be realistic. Unary constraints are derived from the property of one feature, e.g., age and education level cannot decrease. Binary constraints are derived from the relation between two features, e.g., if the education level increases, age increases. If an action does not violate the domain of the feature it is changing, nor the constraints in the SCM, then the feature is modified in NextState (algorithm 1). If the modified feature is an exogenous feature, we update its children using the F functions (algorithm 1-21).

Note that if no SCM is input to the algorithm, it will allow transitions from any state to any state (with intermediate states), and FASTAR would generate ARs using this unconstrained transition function.

Reward function. Algorithm 1 defines a reward function that, given a state and an action, returns a reward based on three components derived from the initial AR problem:

- Given the current state (CurrState), action (A), training dataset (D), and distance function `DistF`, the first part returns the appropriate cost to take that action (algorithm 1). The distance function can either be the ℓ_p norm of the change produced by the action or a more complex function.
- The second part adds a cost if a datapoint is far from the training data manifold (algorithm 1) (which is computed using the `DistD` function) A λ factor is used to control the strictness of data manifold adherence.
- The third part rewards the agent with a large positive value if a counterfactual state is reached (CFReward in algorithm 1). To avoid sparse rewards, we partially reward the agent with a small reward equal to the probability of NextState being classified in the desired class (algorithm 1). However, the sparse rewards can only be used if the underlying ML model provides the class label probabilities instead of only the class label, e.g., a neural network or random forest.

Other parameters. MDPs require additional parameters such as the discount factor $\gamma \in [0, 1]$. At a high level, setting $\gamma < 1$ penalizes longer paths; for additional intuition, see Sutton and Barto [229]. We note that λ, DistD, and DistF are user-specified and domain-specific parameters that directly impact the reward function for the MDP. We instantiate them in the evaluation section (see section 3.5).

3.5 Evaluation

We provide experimental validation of FASTAR using three real-world datasets and comparison using nine baselines. Our research questions (RQ) are motivated by the recourse desiderata discussed in Section 3.2, and are enumerated here:

RQ1 Does FASTAR successfully generate ARs for various input datapoints (validity)?

RQ2 How much change is required to reach a counterfactual state (proximity)?

RQ3 How many features are changed to reach a counterfactual state (sparsity)?

RQ4 Do the generated ARs adhere to the data manifold (realisticness)?

RQ5 Do the generated ARs respect causal and feature immutability constraints (feasibility)?

RQ6 How much time does FASTAR take to generate ARs (amortizability)?

3.5.1 Datasets.

Motivated by most previous AR generating approaches [246], we use three datasets in our experiments: German Credit, Adult Income, and Credit Default [67]. These datasets have 20, 13 (omitted education-num as it has one-to-one mapping with education), and 23 features, respectively.

We split the datasets into 80%-10%-10% for training, validation, and testing, respectively. Each dataset has two labels, '1' and '0', where '1' is the desired label.

We trained a simple classifier: a neural network with two hidden layers (5 and 3 neurons) with ReLU activations. The test accuracy of the classifier was 83.0% for German Credit, 83.7% for Adult Income, and 83.2% for Credit Default. Note that the classifier's accuracy is relatively less important for FASTAR's validation.

3.5.1.1 Implementation Algorithm.

Any appropriate method for computing an optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$, or any approximately optimal policy, to the MDP output of Algorithm 1 can be used. Our MDP has a continuous state and action space, and therefore we use a policy gradient algorithm. Specifically, we use proximal policy optimization (PPO) with generalized advantage estimate (GAE) [206, 162, 207] to train the agent. We used the PyTorch implementation of the library for this [128]. The features in all datasets are scaled between -1 and 1 before training both the ML model and the RL agent.

3.5.2 Baselines

Since, to our knowledge, FASTAR is the first approach to generate amortized ARs for black-box models, there exist no previous approaches that we can directly compare against. Nevertheless, we compare FASTAR to several previous popular AR generating approaches.

Baselines we developed. To compare FASTAR to approaches that generated ARs in an amortized manner for black-box models, we developed two baselines:

- **Random:** This approach tries to reach a counterfactual state by executing random actions from the action space.

Table 3.2: Causal constraints and immutable features for the datasets. We assume FASTAR is provided with them.

Dataset	Causal constraints	Immutable features
German Credit	Age and Job cannot decrease	Foreign worker, Number of liable people, Personal status, Purpose
Adult Income	Age and Education cannot decrease, increasing Education increases Age	Marital-status, Race, Native-country, Sex
Credit Default	Age and Education cannot decrease, increasing Education increases Age	Sex, Marital status

- **Greedy:** At each step, this approach greedily chooses the action (among all actions) that gives the highest reward.

Previous AR generating approaches. Based on the level of required model access, AR generating approaches can be categorized as: 1) access to complete model internals, i.e., weights of neurons or nodes of decision trees, 2) access to model gradients (restricted to differentiable models like neural networks), and 3) access to only the `predict` function (black-box). We choose popular methods from all categories:

- **Complete model internal access.**
  We chose MACE [122] from this category.
- **Gradients access.** Here we chose DiCE-Gradient [164] and DiCE-VAE [157]. Notably, DiCE-VAE is the *only other* amortized AR generation method, however, it requires gradients and is restricted to differentiable models.
- **Black-box.** Open-source repository of the aforementioned DiCE method also had three black-box and model-agnostic approaches, namely: DiCE-Genetic, DiCE-KD-Tree, and DiCE-Random. We choose these three and Prototypes [243] for this category. We did not compare with MOC [51] as DiCE-Genetic is also a genetic algorithm based approach and uses Python code.

3.5.3 Experimental Methodology

Here we describe the specific details of some approaches:

3.5.3.1 FASTAR specifics.

As stated in Section 3.4, the recourses generated by FASTAR are realistic if provided with the actionability of features and causal constraints. These constraints can be provided using the complete/partial SCM of the data generating process or using domain knowledge. We assume these constraints are provided to FASTAR and are shown in Table 3.2. As described in Algorithm 1, this directly impacts the transition function. We use a particular instantiation of Algorithm 1 in the experiments:

- Action space: To produce sequential ARs, actions modify only one feature at a time. However, endogenous features may simultaneously change due to a change in their parent.
- Cost of action: We treat DistF function as a hyperparameter and use several values for it in the experiments.
- Data manifold distance: Following previous work [51, 121], we train k-Nearest Neighbor (KNN) algorithm on the training dataset and use it to find the ℓ_1 distance of a given datapoint from its nearest neighbor ($k = 1$) in the dataset (DistD). We use several values of the adherence factor λ in the experiments.
- Counterfactual state reward (CFReward): The agent receives a reward equal to the probability of its state belonging to the desired class (this ranges between 0 and 1). However, when a counterfactual state is reached, the agent is rewarded with 100 points.
- Discount Factor: We use a discount factor $\gamma = 0.99$. This value encourages the agent to learn a policy that takes a few steps to reach a counterfactual state.

MACE specifics. MACE requires as input the type of ML classifier to be used. We could not use a neural network because of the MACE's long runtime (see section 3.6), and therefore chose logistic regression (LR) and random forest (RF), which had a reasonable runtime.

All approaches are requested to generate ARs for the test datapoints that are predicted as '0' by the classifier. Due to the small size of the German Credit dataset, we generate ARs for datapoints that are predicted as '0' both in the training and test sets. Thus, we request ARs for 257 datapoints in the German credit, 7229 datapoints in the Adult Income, and 5363 datapoints in the Credit Default datasets. Since MACE uses a different classifier, the number of datapoints predicted as '0' were slightly different. More details are provided in section 3.6. FASTAR, random, and greedy approaches stop when they reach a counterfactual state (predicted as '1') or exhaust 50 actions. Other baselines have no such timeout.

Table 3.3: Comparison of FASTAR to all baselines for various AR evaluation metrics. `Validity` is the percentage of an AR that is actually classified in the desired class. `Prox-Num` and `Prox-Cat` refer to the L1 distance of the datapoint and its AR for the numerical and categorical features, respectively. `Sparsity` is the number of features that were changed to produce the AR. `Manifold dist.` is the distance of the AR as returned by the trained kNN algorithm. `Constraints` refer to the causal constraints adherence by the generated AR. `Time` is the average time to generate ARs. For `Validity` and `Constraints`, a higher value is better, and for all other columns, a lower value is better. MACE and DiCE-Gradient could not be run for larger datasets owing to their large computation time.

Dataset	Approach	#DataPts.	Validity	Prox-Num	Prox-Cat	Sparsity	Manifold dist.	Constraints	Time (s)
German Credit	Random	257	23.7	0.17	0.57	11.33	1.08	41.0	0.31
	Greedy	257	49.8	**0.07**	0.087	1.81	0.48	**100.0**	4.59
	DiCE-Genetic	257	98.1	0.67	0.26	6.52	2.39	45.6	1.71
	DiCE-KDTree	257	0.0	N/A	N/A	N/A	N/A	N/A	0.17
	DiCE-Random	257	**100.0**	0.33	0.10	1.93	2.40	93.4	0.17
	Prorotypes	207	**100.0**	0.26	0.58	13.1	1.0	5.3	25.9
	DiCE-Gradient	257	**100.0**	0.27	0.29	6.33	2.19	82.9	7.10
	DiCE-VAE	257	77.8	0.80	0.42	10.12	0.97	5.0	0.15
	MACE (LR)	210	**100.0**	0.36	**0.017**	1.99	0.60	97.1	38.45
	MACE (RF)	287	**100.0**	0.22	0.02	2.64	**0.38**	74.2	101.29
	FASTAR	257	97.3	0.10	0.063	**1.22**	0.72	**100.0**	**0.07**
Adult Income	Random	7229	80.9	0.56	0.77	10.07	1.00	29.0	0.25
	Greedy	7229	97.7	**0.04**	0.02	1.18	**0.17**	95.0	0.27
	DiCE-Genetic	7229	89.5	0.71	0.27	4.43	0.46	23.0	3.43
	DiCE-KDTree	7229	0.0	N/A	N/A	N/A	N/A	N/A	0.59
	DiCE-Random	7229	**100.0**	0.82	0.04	1.64	1.24	90.0	0.22
	Prototypes	500	**100.0**	0.29	0.57	9.0	0.68	22.8	28.9
	DiCE-Gradient	500	84.0	0.18	0.012	2.78	0.51	82.4	59.75
	DiCE-VAE	7229	77.1	0.75	0.65	9.99	0.30	0.13	0.12
	FASTAR	7229	**100.0**	**0.04**	**0.0**	**1.00**	0.18	**100.0**	**0.015**
Credit Default	Random	5363	12.8	4.85	0.68	14.54	1.30	41.5	0.63
	Greedy	5363	65.1	0.15	**0.072**	1.25	**0.22**	99.9	4.67
	DiCE-Genetic	5363	92.6	3.93	0.49	16.67	2.75	27.9	3.58
	DiCE-KDTree	5363	0.0	N/A	N/A	N/A	N/A	N/A	0.45
	DiCE-Random	5363	**100.0**	5.80	0.20	2.33	3.09	97.7	0.39
	Prototypes	500	**100.0**	4.9	0.86	21.0	1.24	0.0	27.3
	DiCE-Gradient	100	81.0	0.77	0.40	15.98	1.35	85.2	479.17
	DiCE-VAE	5363	76.4	1.6	0.68	20.1	0.31	8.9	0.18
	FASTAR	5363	99.9	**0.01**	0.11	**1.008**	0.32	**100.0**	**0.051**

3.6 Results

Table 3.3 shows the performance of FASTAR and all the baselines on the recourse desiderata. We report the average validity, average proximity (separately for the numerical and categorical features), average sparsity, average data manifold distance, average causal constraints adherence, and the average time to generate the ARs per datapoint.

Answer to RQ1: As shown in Table 3.3, FASTAR has very high validity for all datasets. For Adult Income,

FASTAR gets the highest validity at 100%, while for Credit Default and German Credit, it achieves the second and third highest validity, respectively. Random and greedy approaches have low validity in general. DiCE-Genetic has validity in the high range, but this comes at the cost of proximity, sparsity, and data manifold distance. DiCE-KDTree is unable to generate AR even for a single datapoint in all three datasets. DiCE-Random achieves 100% validity for all datasets, and just like DiCE-Genetic, this comes at the cost of proximity, sparsity, and data manifold distance. The conclusion is similar for DiCE-Gradient's and Prototypes' validity. DiCE-VAE's validity is lower than 80% for all datasets. MACE also achieves 100% validity but is very expensive to run. Due to this, it was impractical to run MACE for the larger datasets, Adult Income and Credit Default (we show MACE run only for the German Credit dataset). MACE was even more expensive when the underlying classifier was a neural network, and we had to abandon that experiment. For the classifiers used for MACE, '0' was predicted for 210 datapoints by logistic regression (LR) and 287 datapoints by random forest (RF). MACE was supposed to generate ARs for these datapoints.

Answer to RQ2: We measure proximity for numerical and categorical features separately (Prox-Num and Prox-Cat, respectively). For numerical features, the distance is the sum of the ℓ_1 norm, respectively, divided by the median average deviation for each numerical feature. For categorical features, the distance is the number of categorical features changed divided by the total number of categorical features. These metrics were proposed and used in previous works [157]. FASTAR's ARs are most proximal for Adult Income and Credit Default datasets, and second best for German Credit. The random approach, Prototypes, and the five variants of DiCE have large proximity values. The greedy approach performs well on this metric, but its validity is very low. MACE's performance is about average.

Answer to RQ3: FASTAR achieves the lowest sparsity among all approaches. Following previous works [164], we measure sparsity at the start and endpoint of a recourse. Random, Prototypes, DiCE-VAE, DiCE-Genetic, and DiCE-Gradient's performance is abysmal. This is surprising because DiCE-Gradient has a post-hoc step specifically for reducing sparsity. Greedy, MACE, and DiCE-Random's performance is about average.

Answer to RQ4: FASTAR achieves low average manifold distance. It performs second best for Adult Income and Credit Default and is in the middle for German Credit. The greedy approach, MACE, and DiCE-VAE also perform well on this metric. The random approach, Prototypes, and all variants of DiCE (except DiCE-VAE) perform poorly on this metric.

Answer to RQ5: By construction, FASTAR always respects causal constraints encoded in its transition function: it has 100% adherence in all datasets. The DiCE based approaches (except DiCE-VAE), MACE, and the random approach take as input the immutable features, but not the other causal constraints, and hence do not perform well. DiCE-VAE and Prototypes do not accept immutable features and hence, perform the worst for this metric. The greedy approach performs well on this metric, even though it does not have a knowledge of the causal constraints.

Answer to RQ6: The final column in Table 3.3 reports the average computation time per AR. For the timing experiments, we conducted experiments for all the approaches on the same machines with 32GB RAM and 4 CPUs. FASTAR and other baselines were all implemented in Python, and no parallelism was used. Owing to amortization, FASTAR can generate ARs very quickly and takes the lowest time among all approaches. The next best performers are DiCE-VAE and DICE-Random. FASTAR is **2**$\times$ faster than DiCE-VAE on average (up to **8**$\times$ faster), **8**$\times$ faster than DiCE-random on average (up to **15**$\times$ faster). DiCE-random and random approach perform similarly. The difference is even more staggering for DiCE-Genetic, Prototypes, and the greedy approach. MACE and Dice-Gradient were the slowest. FASTAR is about **1000**$\times$ faster than MACE on average (up to **1447**$\times$ faster) and **4500**$\times$ faster than DiCE-Gradient on average (up to **9400**$\times$ faster). While amortization allows for the rapid generation of new ARs, there exists a one-time training cost.

3.7 Conclusion

We propose a novel RL-based approach, FASTAR, that generates amortized and sequential recourses for black-box ML models. To the best of our knowledge, we are the first to propose such an approach. The ARs generated by FASTAR possess desirable properties, and when evaluated on the recourse metrics, they perform better than several popular baselines.

Chapter 4

RecXplainer: Amortized Attribute-based Personalized Explanations for Recommender Systems

Recommender systems influence many of our interactions in the digital world—impacting how we shop for clothes, sorting what we see when browsing YouTube or TikTok, and determining which restaurants and hotels we are shown when using hospitality platforms. Modern recommender systems are large, opaque models trained on a mixture of proprietary and open-source datasets. Naturally, issues of trust arise on both the developer and user side: is the system working correctly, and why did a user receive (or not receive) a particular recommendation? Providing an *explanation* alongside a recommendation alleviates some of these concerns. The status quo for auxiliary recommender system feedback is either user-specific explanations (e.g., "users who bought item B also bought item A") or item-specific explanations (e.g., "we are recommending item A because you watched/bought item B"). However, users bring personalized context into their search experience, valuing an item as a function of that item's attributes and their own personal preferences.

In this work, we propose `RecXplainer`, a novel method for generating fine-grained explanations based on a user's preferences over the attributes of recommended items. We evaluate `RecXplainer` on five real-world and large-scale recommendation datasets using five different kinds of recommender systems to demonstrate

the efficacy of RecXplainer in capturing users' preferences over item attributes and using them to explain recommendations. We also compare RecXplainer to five baselines and show RecXplainer's exceptional performance on ten metrics.

Table 4.1: Illustration of how RecXplainer computes the specific preference of a user over an item's attributes

Genre removed	Predicted rating	Δ(Predicted rating)
All genres present	4.2	–
Crime	3.2	1.0
Romance	3.7	0.5
Documentary	3.9	0.3

4.1 Introduction

Recommender systems direct our attention, telling us what to look at—and what not to. They are deployed widely at platform companies such as Netflix, YouTube, Yelp, Amazon, and Shein. There are two main approaches to generating recommendations [4, 18]: content-based [24, 5] and collaborative filtering [194, 228]. Content-based systems use item attributes and/or user preferences to recommend new items, while collaborative filtering (CF) uses the wisdom of the crowd, based on user ratings. Hybrid recommender systems aim to combine both approaches [see, e.g., 23]. Modern methods are black-boxes and their servicing of an end-user's needs [see, e.g., 285], which we focus on in this paper.

Table 4.2: The test loss of each architecture for each dataset.

Dataset/Model	MF	AutoE	NCF	FM	Deep FM
Movielens-100K	**0.82**	1.1	0.87	0.85	0.91
Hetrec	**0.57**	0.84	0.62	0.64	0.70
Anime	2.1	1.7	**1.5**	1.8	2.0
Movielens-20M	0.95	0.83	0.63	0.82	**0.62**
YahooMusic	**568**	*Timeout*	613	658	624

Table 4.3: The test set coverage, top-k recommendations coverage, and the 8 explanation personalization metrics are reported here (averaged over all users) for the MATRIX FACTORIZATION model trained using MOVIELENS-100K dataset. The mean and standard deviation value for each metric is computed over 5 random data splits. We compare 3 architecture choices for auxiliary modeling of RecXplainer. The best results are highlighted in bold.

Metrics	LIME-RS	AMCF-PH	GBL Popl.	User Popl.	Random	RX-Linear	RX-MLP	RX-GBDT
Testset Coverage	57.42 ± 0.47	49.62 ± 4.59	**84.69 ± 0.18**	77.6 ± 0.59	32.83 ± 0.57	60.74 ± 0.21	67.16 ± 0.59	63.01 ± 0.3
Recommendations Coverage	67.08 ± 2.68	44.2 ± 2.58	**79.12 ± 2.83**	70.12 ± 2.05	26.72 ± 0.89	69.3 ± 2.4	65.23 ± 2.28	64.27 ± 2.09
CondProb Generalpref Coverage	59.85 ± 0.56	54.57 ± 1.73	25.07 ± 0.58	41.51 ± 0.44	45.07 ± 0.91	64.77 ± 1.75	60.78 ± 1.59	**71.24 ± 1.12**
CondProb Generalpref Ranking	13.99 ± 0.41	13.74 ± 0.39	5.04 ± 0.16	9.76 ± 0.23	11.64 ± 0.29	15.8 ± 0.34	15.68 ± 0.17	**20.28 ± 0.3**
CondProb Specificpref Coverage	48.87 ± 0.25	**49.97 ± 1.14**	25.1 ± 0.58	41.44 ± 0.44	43.94 ± 0.5	49.78 ± 0.48	49.06 ± 0.22	**51.23 ± 0.27**
CondProb Specificpref Ranking	51.24 ± 0.15	49.89 ± 0.32	6.29 ± 0.11	10.29 ± 0.25	11.71 ± 0.14	51.62 ± 0.36	52.59 ± 0.27	**55.98 ± 0.22**
Odds Generalpref Coverage	69.31 ± 1.43	60.98 ± 1.46	47.42 ± 0.55	55.78 ± 0.84	44.39 ± 0.49	74.42 ± 1.05	72.47 ± 1.16	**81.15 ± 0.55**
Odds Generalpref Ranking	23.25 ± 0.58	18.94 ± 0.8	17.72 ± 0.16	27.63 ± 0.4	11.25 ± 0.32	25.87 ± 0.33	29.3 ± 0.83	**33.57 ± 0.39**
Odds Specificpref Coverage	54.96 ± 0.62	51.8 ± 0.93	47.47 ± 0.55	55.73 ± 0.84	44.08 ± 0.24	56.17 ± 0.5	55.56 ± 0.6	**57.74 ± 0.46**
Odds Specificpref Ranking	50.67 ± 0.16	50.07 ± 0.44	19.04 ± 0.32	28.04 ± 0.4	11.68 ± 0.1	50.34 ± 0.12	52.54 ± 0.28	**55.33 ± 0.12**

Table 4.4: The test set coverage, top-k recommendations coverage, and the 8 explanation personalization metrics are reported here (averaged over all users) for the MATRIX FACTORIZATION model trained using HETREC dataset. The mean and standard deviation value for each metric is computed over 5 random data splits. We compare 3 architecture choices for auxiliary modeling of RecXplainer. The best results are highlighted in bold.

Metrics	LIME-RS	AMCF-PH	GBL Popl.	User Popl.	Random	RX-Linear	RX-MLP	RX-GBDT
Testset Coverage	68.52 ± 0.47	68.49 ± 0.78	**94.4 ± 0.06**	90.65 ± 0.09	45.51 ± 0.51	75.46 ± 0.34	80.98 ± 1.18	78.77 ± 0.18
Recommendations Coverage	81.19 ± 0.9	55.49 ± 1.46	**86.16 ± 1.75**	82.99 ± 1.3	30.16 ± 1.33	81.4 ± 1.22	79.65 ± 2.59	78.98 ± 0.88
CondProb Generalpref Coverage	**87.37 ± 0.21**	72.6 ± 2.05	37.46 ± 0.65	50.97 ± 0.44	62.06 ± 1.89	84.61 ± 0.42	78.63 ± 2.16	84.56 ± 0.45
CondProb Generalpref Ranking	18.01 ± 0.19	14.57 ± 0.9	5.31 ± 0.21	8.91 ± 0.21	12.43 ± 0.76	17.72 ± 0.19	16.4 ± 0.62	**19.55 ± 0.37**
CondProb Specificpref Coverage	60.7 ± 0.39	59.23 ± 0.45	37.42 ± 0.63	50.93 ± 0.43	62.26 ± 0.08	**63.52 ± 0.45**	59.87 ± 0.45	61.86 ± 0.47
CondProb Specificpref Ranking	52.58 ± 0.05	50.25 ± 0.26	4.3 ± 0.2	6.59 ± 0.12	10.47 ± 0.04	54.26 ± 0.06	54.13 ± 0.47	**57.46 ± 0.09**
Odds Generalpref Coverage	**85.34 ± 0.4**	75.75 ± 1.57	44.63 ± 0.47	53.48 ± 1.25	63.39 ± 0.91	**85.15 ± 0.5**	79.01 ± 2.28	**84.79 ± 0.53**
Odds Generalpref Ranking	22.36 ± 0.23	19.1 ± 0.31	13.91 ± 0.23	20.52 ± 0.34	12.83 ± 0.32	23.32 ± 0.28	23.08 ± 0.48	**26.0 ± 0.22**
Odds Specificpref Coverage	63.48 ± 0.31	62.92 ± 0.17	44.64 ± 0.47	53.51 ± 1.25	62.37 ± 0.09	**65.78 ± 0.42**	63.05 ± 0.38	64.86 ± 0.29
Odds Specificpref Ranking	51.85 ± 0.09	50.8 ± 0.26	13.75 ± 0.32	19.54 ± 0.3	10.51 ± 0.05	53.49 ± 0.14	53.36 ± 0.35	**56.35 ± 0.04**

Table 4.5: The test set coverage, top-k recommendations coverage, and the 8 explanation personalization metrics are reported here (averaged over all users) for the DEEP FACTORIZATION MACHINE model trained using MOVIELENS-20M dataset. The mean and standard deviation value for each metric is computed over 5 random data splits. We compare 3 architecture choices for auxiliary modeling of RecXplainer. The best results are highlighted in bold.

Metrics	LIME-RS	AMCF-PH	GBL Popl.	User Popl.	Random	RX-Linear	RX-MLP	RX-GBDT
Testset Coverage	*Timeout*	57.54 ± 1.33	**93.78 ± 0.01**	91.02 ± 0.01	47.28 ± 0.04	73.47 ± 0.33	83.51 ± 1.34	79.64 ± 0.14
Recommendations Coverage	*Timeout*	44.69 ± 2.55	**82.74 ± 1.77**	76.37 ± 0.71	35.66 ± 0.74	72.0 ± 0.82	75.55 ± 0.61	72.57 ± 0.78
CondProb Generalpref Coverage	*Timeout*	56.26 ± 2.71	33.39 ± 0.09	42.24 ± 0.12	64.86 ± 0.18	**74.81 ± 0.54**	63.26 ± 1.97	70.52 ± 0.18
CondProb Generalpref Ranking	*Timeout*	9.94 ± 1.02	4.33 ± 0.01	6.44 ± 0.01	13.15 ± 0.02	**13.69 ± 0.21**	11.08 ± 0.56	**13.9 ± 0.08**
CondProb Specificpref Coverage	*Timeout*	66.57 ± 0.89	33.38 ± 0.09	42.23 ± 0.12	64.78 ± 0.02	**71.17 ± 0.73**	64.49 ± 0.4	66.37 ± 0.11
CondProb Specificpref Ranking	*Timeout*	45.95 ± 0.43	3.44 ± 0.02	4.85 ± 0.01	11.11 ± 0.01	53.07 ± 0.11	51.31 ± 0.25	**54.23 ± 0.05**
Odds Generalpref Coverage	*Timeout*	62.48 ± 1.34	43.38 ± 0.05	50.98 ± 0.04	64.83 ± 0.17	**83.88 ± 0.34**	73.05 ± 2.06	79.68 ± 0.12
Odds Generalpref Ranking	*Timeout*	13.12 ± 0.5	12.2 ± 0.02	17.03 ± 0.02	13.15 ± 0.06	21.45 ± 0.11	20.3 ± 0.53	**23.29 ± 0.03**
Odds Specificpref Coverage	*Timeout*	63.68 ± 0.36	43.39 ± 0.05	50.97 ± 0.04	64.79 ± 0.01	**68.09 ± 0.15**	64.65 ± 0.22	66.17 ± 0.06
Odds Specificpref Ranking	*Timeout*	46.18 ± 0.49	11.99 ± 0.02	16.04 ± 0.02	11.11 ± 0.01	51.94 ± 0.05	51.03 ± 0.18	**53.61 ± 0.01**

Figure 4.1: A user of a fashion website is recommended a cloth and she gets curious about the reason for recommending it? `RecXplainer` explains that it was recommended because she had shown interest in *green* clothes and clothes from *H&M*.

In recent years, CF-based methods have been widely chosen over content-based methods owing to 1) no requirement of manual labeling of item features, and 2) more serendipitous recommendations [86]. (Content-based recommender systems require each new item to be featurized, whose cost is usually high [137].) However, CF-based methods have one major limitation: a lack of attribute-based interpretability. CF methods map users and items to an embedding space, which is learned from the user-item interaction matrix—and the proximity in this space is used to generate recommendations. Such an embedding space is difficult to interpret. A core challenge is understanding what a model learns about the user's preference over the items' attributes and explaining how it generates the new recommendations.

Previous research has established that providing explanations for a recommendation enhances the transparency, scrutability, trustworthiness, and persuasiveness of the recommender systems [234, 15, 220]. This has spurred significant research in the broad field of "explainability for CF recommender systems." Providing explanations when using CF-based methods is nontrivial due to the lack of interpretability of the embedding space. Most of the previous approaches provide explanations in the form of user-based or item-based explanations. User-based explanations explain a recommendation on the basis of 'similar' users liking it. And item-based explanations explain a recommendation based on its similarity to other items that the user has previously liked. Item-based explanations are usually easier to grasp as the user knows about the items they have interacted with in the past. However, neither of these explanation formats captures a user's preference over the attributes of an item, which is how users inherently think about a recommendation [107, 253, 75, 159, 160, 259].

This work proposes a novel approach, `RecXplainer`, that generates attribute-based explanations for CF-

based recommender systems. A recommendation is explained in terms of a user's preference over the attributes of the item (see Figure 4.1), e.g., 'We are recommending this movie because you like *Action* movies.' or 'We are recommending this t-shirt because you like *Adidas* products and *blue* clothes.' Such explanations are personalized to the user and hence help further enhance the persuasiveness and trustworthiness of the recommender systems. Note that the CF-based recommender system usually does not use these attributes to generate the recommendations.

`RecXplainer` learns a model of the user's preference over the attributes of the items and uses this model to explain novel recommendations. The key advantages of `RecXplainer` are that it a) provides post-hoc explanations, b) is model-agnostic, and c) generates explanations in an amortized fashion. Post-hoc explanation techniques provide explanations after the recommender systems have been trained and therefore do not interfere with their architecture or training routine. This is crucial for real-world recommender systems whose architectures are complex and whose training routines are highly optimized for accuracy. Moreover, such explainability techniques might avoid the accuracy-interpretability trade-off [259, 176]. Model-agnostic explanation techniques are desirable for their flexibility and generalizability as they can be used to explain a broad class of recommender systems [163]. Moreover, `RecXplainer` generates explanations in an amortized fashion, i.e., once `RecXplainer` learns the user's preference over attributes, it can generate explanations for novel recommendations by merely doing a model inference – which is cheap and hence scalable to large recommender systems.

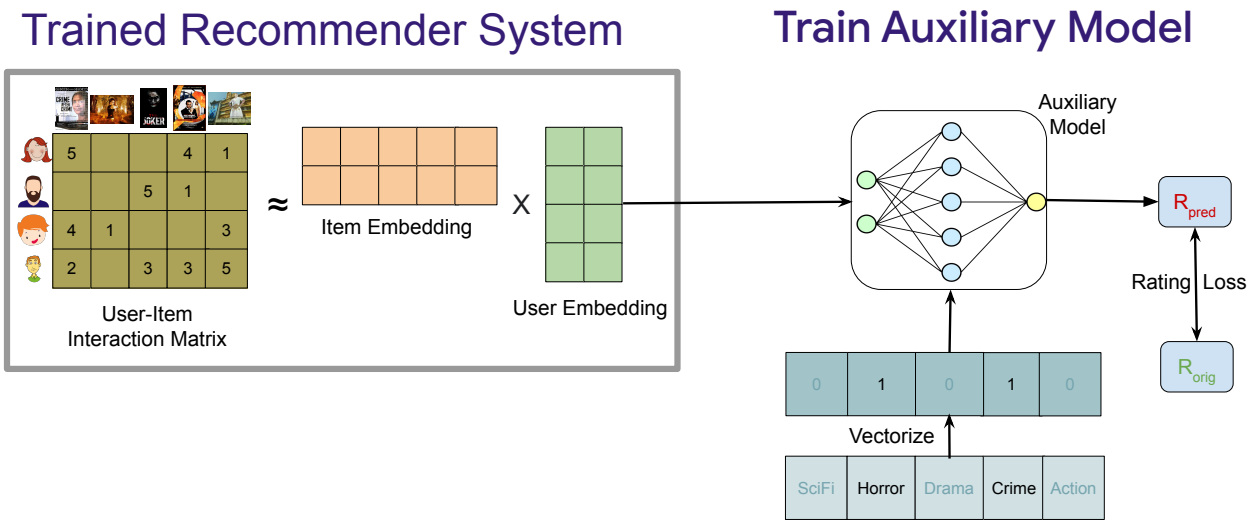
We found that the important intersection of post-hoc attribute-based explanations for CF-based recommender systems has not garnered sufficient attention in explainability literature. We are aware of only two previous approaches that generate attribute-based explanations for CF-based recommender systems: LIME-RS [169] and AMCF [174] (details in related work in Section 2.3). LIME-RS did not evaluate its attribute-based explanations, and the metrics used by AMCF were not generalizable to all CF-based recommender systems. To this end, we propose a set of eight generalizable metrics to evaluate attribute-based explainability techniques for recommender systems (Section 4.3) (one of our major contributions). We also consider the often overlooked popularity-based explanation methods: for attribute-based explanations, such an approach would choose the most popular attributes of a recommended item as an explanation. In this work, we implemented three popularity-based baselines for a holistic evaluation.

In summary, our contributions are as follows:

1. We propose `RecXplainer`, a novel, post-hoc, and model-agnostic technique to provide attribute-based explanations for recommender systems in an amortized fashion. To the best of our knowledge, `RecXplainer` **is the first technique** to have all these desirable properties.
2. We propose novel metrics to evaluate attribute-based explainability methods for CF-based recommender systems.
3. We perform extensive experimentation with five different classes of recommender systems trained on five large-scale recommendation datasets. We demonstrate the efficacy of `RecXplainer` and its comparison with five baselines using ten metrics.
4. We also explore the comparison of `RecXplainer` with the often overlooked popularity-based explanation methods, revealing surprising results.

4.2 `RecXplainer`: Architecture & Algorithm

Figure 4.2: `RecXplainer`'s architecture. We train an auxiliary model that takes as input the user embeddings from the trained recommender system and the item attribute vector.



`RecXplainer` provides explanations using the features of an item that a user prefers. In order to do so, we train an auxiliary model using the user embedding and the item's attribute vector. The user embedding is obtained from the trained recommender system's latent space, and the item's attribute vector is constructed

from the data. The auxiliary model is trained to reproduce the ratings given in the dataset (R_{orig} in the figure) using squared loss. Figure 4.2 shows `RecXplainer`'s architecture.

4.2.1 Item attribute vector:

The attribute vector is multi-hot – it has 1s when the item has those attributes, otherwise, 0s. For example, consider a movie recommendation platform with metadata about the genres, and there are a total of 20 genres into which all movies are categorized. A particular movie will usually belong to 1-3 genres, and hence the attribute vector of a movie would consist of 1-3 1s in those indices (indicating the presence of those genres), and the rest of the indices will be 0s. Instead of binary values, the vector could also have continuous values.

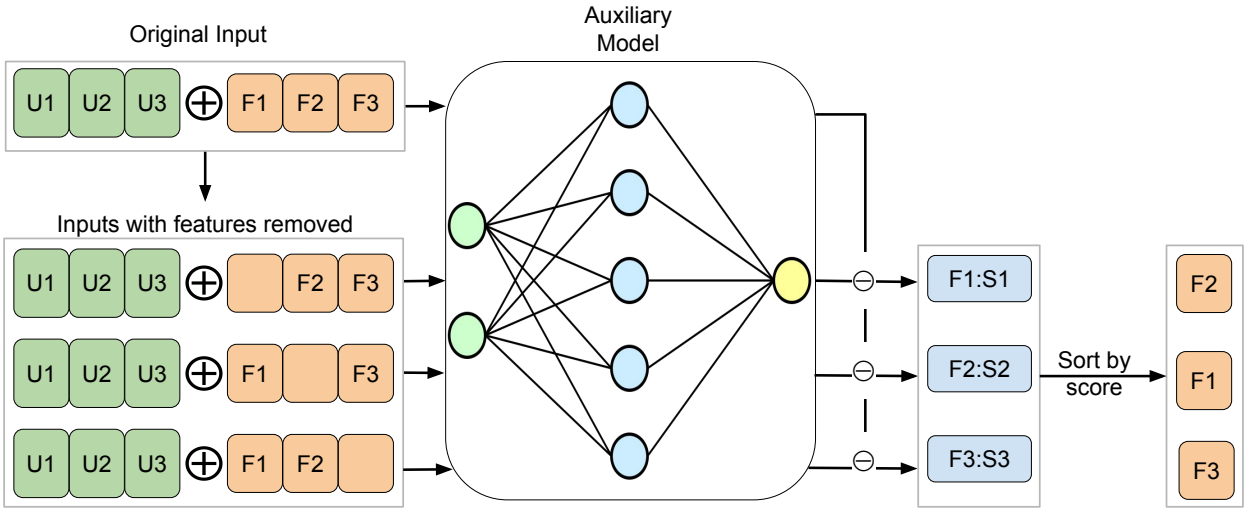
4.2.1.1 Auxiliary model architecture:

The auxiliary model can be instantiated with any model architecture. For experiments, we test three separate architectures: a linear layer, a deep neural network, and gradient boosted decision trees (GBDT). We trained the auxiliary models using the Adam optimizer until convergence. Note that the auxiliary model is trained independently of the recommender system, and it only requires the user's embedding from the recommender system. Therefore, `RecXplainer` is both post-hoc and model-agnostic.

4.2.2 `RecXplainer`: Generating Explanations

Once the auxiliary model is trained, we use it to generate explanations for new recommended items for the users (Figure 4.3). Given that we need to explain an item recommended to a user (a user-item pair), `RecXplainer` produces a ranked order of that particular item's attributes in decreasing order of the user's preference over them. We term this as the user's *specific preference* over the item's attributes. The importance of a particular attribute of an item is computed as the difference in the predicted ratings when the attribute is present and when it is absent from that item. The auxiliary model predicts both the ratings. For illustration, consider a movie recommendation platform. Alice is a user of the platform and gets a recommendation for a movie whose genres are *Crime*, *Romance*, and *Documentary*. Alice is curious and requests an explanation: `RecXplainer` explains that she was recommended it because she likes *Crime* movies.
Now, let us look at how `RecXplainer` arrived at this explanation: `RecXplainer` requests the auxiliary

Figure 4.3: The difference in the scores produced by the auxiliary model for the cases when all features are present, and a feature is removed, is used to assign importance to the removed feature; thereby ranking the item features.



model to predict the recommendation score for the same movie that the recommender system originally recommended. The auxiliary model predicts a rating of 4.2 for this movie for Alice. It uses Alice's embedding and the movie's genres for this prediction. Next, `RecXplainer` approximates the importance of each individual attribute by zeroing them out one by one in the movie's genres vector. It then uses the auxiliary model to predict the rating if the movie had certain attributes not present in the movie.

Table 4.1 shows this procedure. We zero out the genres present in the movie, one at a time, and see the change in the predicted rating (the original rating is the rating produced by the auxiliary model when all the genres of the movie are considered). The higher the drop in the predicted rating, the higher the importance of the removed genre, and hence the higher its rank. In the above example, Alice's *specific preference* over the genres is *Crime*, *Romance*, and *Documentary*. Algorithm 2 provides the algorithm for predicting the *specific preference* of a user for any item.

`RecXplainer`'s approach of attributing importance to genres falls in a well-studied domain of removal-based explanation methods [49], where the difference in the model's prediction before and after removal of a feature is an approximation of its importance. There are several alternatives in the choice of the algorithm for removing features for attribution. SHAP [154] removes all subsets of the features, gets the model's predictions for all those feature subsets, and uses the Shapley value formula [214] to assign importances. The alternative, which we choose for `RecXplainer`, is one feature removal as it is much cheaper than

ALGORITHM 2: Predict the specific preferences of a user over the attributes of an item

Input : User Embedding $\mathcal{U}$, Item Features I, Trained Auxiliary Model $\mathcal{M}$
Output: Preference of each attribute of the item SpecificPref

Function `get_recommendation_score`($\mathcal{M}$,$\mathcal{U}$, I')
    recommendation_score $\leftarrow \mathcal{M}(\mathcal{U} \oplus I')$
    return recommendation_score
Function `get_specific_preference`($\mathcal{M}$,$\mathcal{U}$, I)
    SpecificPref $\leftarrow 0$
    `// Iterate over all the attributes, and set the value of the attributes that are present to 0`
    for $i \in I$ **do**
        $I' \leftarrow I$
        if $i = 1$ **then**
            $I'[i] \leftarrow 0$
            SpecificPref$[i] \leftarrow$ `get_recommendation_score`($\mathcal{M}$,$\mathcal{U}$, I') `// Get the recommendation score for this item when attribute i is set to 0`
        end
    end
    SpecificPref $\leftarrow$ `sort`(SpecificPref) `// sort in descending order`
    return SpecificPref

SHAP to compute. However, `RecXplainer` is agnostic to the specific feature removal technique and works well with both one feature removal and SHAP. We find that the `RecXplainer` performs slightly better when using SHAP; however, SHAP is 70 times more expensive than one feature removal technique, and hence not scalable to large datasets. Therefore, in the evaluation section, we use one feature removal for the results.

4.2.3 Global Attribute Importance

Given the procedure to generate a user's *specific preference* over the attributes of an item, we can capture a user's preference over all the attributes in the recommender system. We term this as a user's *general preference*. `RecXplainer` computes a user's *general preference* by averaging the importance of each attribute over all the items that a user liked in the past (Eq. 4.1). Note that since most items have a small subset of all the attributes (a movie would usually belong to 1-3 genres from the total set of genres), the importance of the attributes not present in an item is set to be 0.

$$\begin{aligned} L &= \text{the set of items that user u liked} \\ \text{GeneralPref}[u] &= \frac{1}{|L|} \sum_{i \in L} \text{SpecificPref}[u][i] \end{aligned} \tag{4.1}$$

Specific preference and *general preference* respectively are the analogs of *local interpretability* and *global interpretability* provided by popular methods like LIME [192] and SHAP [154].

4.3 Evaluation

Our experiments characterize the quality of explanations by the number of liked items in the user history and the number of top-k recommendations that can be explained by `RecXplainer`.

4.3.1 Datasets and Recommender Systems

To demonstrate `RecXplainer`'s scalability, we experiment with several popular recommender datasets of increasing sizes. Specifically, we use MOVIELENS-100K [95], HETREC [94], ANIME [120], MOVIELENS-20M [96], and YAHOOMUSIC [65] datasets for our experiments. In order to explain the SOTA recommender system on these datasets, we train 5 popular recommender architectures on each of these datasets, and choose the best performing architecture as the recommender system to explain for each dataset. Table 4.2 shows the best performing architecture for each dataset (in bold).

Each of these datasets have features that `RecXplainer` uses to explain a recommendation. For example, there are 18 genres in the MOVIELENS-100K dataset: *Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western*. Each movie usually belongs to 1-3 genres.

For all the datasets, we used 70% of the data for training and validating the recommender models and 30% of the data as the test set for computing the metrics.

4.3.2 Baselines

We compare `RecXplainer` to five baseline approaches:

- *LIME-RS [169]:* This is the only previous post-hoc attribute-based explainability approach for CF-based recommender systems. It trains a regression model for every item it needs to explain; hence, this approach is not amortized. As we will see later in the experimental results, LIME-RS cannot scale to large datasets due to this problem.
- *AMCF [174]:* This is another attribute-based explainability method for CF-based recommender systems; however this technique is not a post-hoc method. It trains a model while the recommender system is being trained and thus requires modification in its architecture. For a fair comparison, we adapted AMCF to be post-hoc: *AMCF-PH*. We froze the recommender system and trained AMCF's model till convergence.

Note that, unlike our approach, AMCF takes as input the item embedding and hence does not apply to recommender systems that do not construct item embeddings explicitly, for example, autoencoder and RBM-based recommender systems.

- *Global popularity:* This approach finds the most popular attributes across the entire dataset, i.e., the attributes that are the most popular among all the items that are liked across the entire dataset. For every recommended item, this technique would serve the same explanation. It is immediately evident that the explanations will not be personalized to the users and might be uninformative depending on the attributes.
- *User-specific popularity:* This approach finds the most popular attributes across the items that a particular user liked in the dataset. These explanations are more personalized to the user than global popularity; however, they can still be uninformative if a common feature exists in all liked items.
- *Random:* This baseline is for control. It selects a random set of attributes for each item to be explained when computing both specific and general preferences.

For `RecXplainer`, we train three auxiliary models in the experiments: a linear layer (RX-Linear), a 4-hidden layer neural network (RX-MLP), and gradient boosted trees (RX-GBDT). We repeated all experiments for five random splits of the dataset and report the mean and the standard deviation for all the metrics.

4.3.3 Metrics

We measure performance using ten metrics. The metrics can be categorized into two broad categories: *coverage* of recommended items and *personalization* to the user. Previous work has mostly used coverage-based metrics [2, 176]. We argue about the insufficiency of coverage-based metrics and propose a set of eight metrics that measure the personalization of the explanations to the users. These metrics can be used to evaluate any attribute-based explanation method for recommender systems.

1. *Test set coverage:* This metric finds the percentage of the test set items whose attributes intersect with the top-k general preferences of a user. For example, suppose a technique identifies the top-3 general preferences of a user, Adam, as *Action*, *Comedy*, and *Sci-Fi*. If Adam likes a movie whose genres are *Action* and *Romance*, it is counted as covered by this technique, as *Action* had been identified as one of the preferred genres. Conversely, had the movie belonged to *Crime* and *Adventure* genres, it would not have been covered. We measure this metric only for the items that a user liked. We consider an item 'liked' if a user has rated

it 4 or higher (when the dataset ratings are between 1 and 5) and 7 or higher (when the dataset ratings are between 1 and 10). We only consider the top-k general preferences for a user when measuring coverage, where k is about 0.2 times the number of attributes in the dataset. We report the mean coverage over all the users for this metric.

2. *Top-20 recommendations coverage:* Similar to test set coverage, this metric finds the percentage of each user's top-20 recommended items whose attributes intersect with the top-k general preferences of that user. We report the mean coverage over all the users for this metric.

Insufficiency of coverage-based metrics: Since a few attributes in most recommender datasets are very popular, i.e., they occur in almost all items: identifying such an attribute as a user's preference will provide a very high test set and recommendations coverage. However, those attributes can be inaccurate, unpersonalized, or uninformative as an explanation. For example, consider a movie recommendation dataset containing language as one of the attributes, where most movies are English movies. If an explanation technique provides 'English' as an explanation for the recommended movie, that technique will get a 100% test set and recommendations coverage while being uninformative and unpersonalized to the users. Therefore, we propose a new set of metrics to measure the personalization of the explanations.

3. *Personalization of the explanations:* To measure the personalization of the explanations, we require the ground-truth information about the users' preferences over the attributes. However, this information is usually missing from the datasets. Therefore, we propose two measures that act as reasonable proxies for a user's attribute preference:

(a) *Conditional Probability of Liking given an attribute is present:* This computes the probability that a user likes an item, given that an attribute is present in it. This measure is the ratio of the number of times an attribute is present in the items a user likes (e.g., rated 4 or higher) and the number of times it is present in all the items the user rated.

(b) *Odds of Liking vs. Disliking given an attribute is present:* This computes the ratio of the number of times an attribute is present in an item the user likes (e.g., rated 4 or higher) and the number of times it is present in an item that the user dislikes (e.g., rated 2 or lower).

Both measures provide a proxy of users' general preference over attributes, i.e., a ranked order of the attributes. We use the training set for computing both these proxy measures. Now, considering the two proxies

as a ground truth of user preference over attributes, we report four personalization metrics for each of them:

(a) *General preferences coverage:* This metric computes if there is any intersection between the top-k attributes of a user's general preferences identified by a technique and the top-k preferences identified by a proxy measure. We report the mean coverage over all the users for this metric for both proxies.

(b) *General preferences ranking:* This metric measures the similarity between the ranking of the top-k attributes of a user's general preferences identified by a technique and the top-k preferences identified by a proxy measure. We use rank-biased overlap (RBO) as a measure of similarity between the two ranked lists. RBO has several advantages over traditional rank similarity metrics like Kendall's Tau or Spearman's correlation (it prioritizes top-ranked items, handles disjoint item sets, and has adjustable evaluation depth). We compute this metric for all the items a user has liked and report the mean score over all the users.

(c) *Specific preferences coverage:* Similar to general preferences, we also measure if there is any intersection between the top-k attributes of an item by the specific preferences of a user identified by a technique and the top-k preferences identified by a proxy measure. For a user, we compute this metric for all the items they liked and report the mean coverage over all the users.

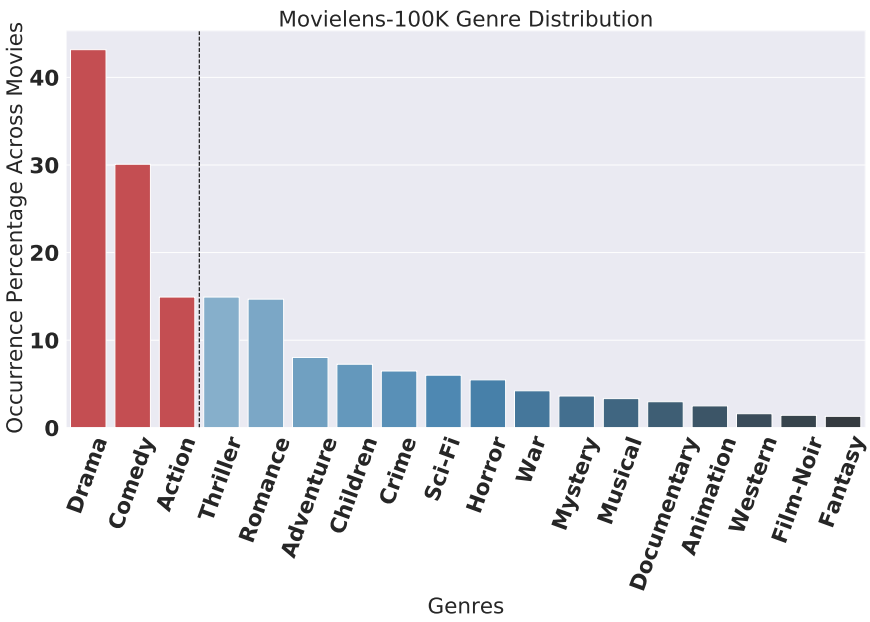
(d) *Specific preferences ranking:* This metric measures the similarity between the ranking of the top-k attributes of an item by the specific preferences of a user identified by a technique and the top-k preferences identified by a proxy measure. We use RBO for computing the similarity between the rankings. We compute this metric for all the items a user has liked and report the mean score over all the users.

Given these metrics reported per proxy measure, we get a total of eight metrics for computing the personalization of the explanations.

4.4 Results and Discussion

Table 4.3 reports the ten metrics for all the baselines and `RecXplainer` for the *Matrix Factorization* based recommender system trained using MOVIELENS-100K dataset. We now analyze the results:

1. *LIME-RS:* `RecXplainer` performs better than LIME-RS on all ten metrics.
2. *AMCF-PH:* `RecXplainer` performs better than AMCF-PH on nine metrics and matches in one metric.
3. *Global popularity:* It achieves the highest test set and top-k recommendations coverage. Upon further analysis, we found that the three most popular movie genres, *Action*, *Comedy*, and *Drama* occur in over



88% of the test set items (genre distribution plot in Item 2). Hence global popularity technique serves these three attributes as the explanation for all movies, thus providing unpersonalized and uninformative explanations. This is corroborated by its poor performance on all personalization metrics; it performs worse than even the *random* baseline for several metrics. We conclude that the explanations served by global popularity do not capture a user's preference over attributes and perform well for the two coverage-based metrics because of the skewed distribution of the attributes.

4. *User-specific popularity:* This popularity approach achieves the second highest test set, and top-k recommendations coverage, and similar to *global popularity* performs worse than `RecXplainer` on all personalization metrics, even worse than the *random* baseline on several metrics.
5. *Random:* We used this approach to serve as a control and to ensure that no metric was trivial to perform well on. `RecXplainer` performs better than it on all ten metrics.
6. *`RecXplainer`:* `RecXplainer` has the third highest test set and top-k recommendations coverage while performing the best on all personalization metrics. We conclude that `RecXplainer` serves the most personalized explanations while still being able to explain a large proportion of test set items and top-k recommendations. Moreover, `RecXplainer` can provide explanations in an amortized manner, providing explanations faster when deployed.

Table 4.4 reports the ten metrics for all the baselines and `RecXplainer` for the *Matrix Factorization* based

recommender system trained using HETREC dataset. We analyze the results:

1. *LIME-RS:* `RecXplainer` performs better than LIME-RS for eight out of ten metrics and matches in one metric.
2. *AMCF-PH:* `RecXplainer` performs better than AMCF-PH on all ten metrics.
3. *Global popularity:* Similar to Movielens-$100K$, global popularity performs the best for the test set and top-k recommendations coverage. This is due to the skewed distribution of genres. It performs worse than `RecXplainer` (even worse than the *random* baseline) for all personalization metrics.
4. *User-specific popularity:* Similar to Movielens-$100K$, user-specific popularity performs the second best for the test set and top-k recommendations coverage, and worse than `RecXplainer` (even worse than *random* baseline) for all personalization metrics.
5. *Random:* `RecXplainer` performs better than the random baseline performs on all ten metrics.
6. *`RecXplainer`:* `RecXplainer` performs the third best for test set coverage and the second best for top-k recommendations coverage. It also performs the best for seven out of eight personalization metrics. This demonstrates that `RecXplainer` provides personalized explanations to the user while providing high coverage.

Table 4.5 reports the ten metrics for all the baselines and `RecXplainer` for the *Deep Factorization Machine* based recommender system trained using MOVIELENS-$20M$ dataset. We analyze the results:

1. *LIME-RS:* Recall that LIME-RS trains a separate regression model for each item to be explained. This is a very expensive procedure to generate explanations. For a large dataset like Movielens-20M that has 138,000 users, LIME-RS did not scale. LIME-RS took more than 1 hour to compute the general preferences for just 1000 randomly sampled users – thereby, the estimated time for computing both the specific and general preferences for all users is *over 11 days*.
2. *AMCF-PH:* `RecXplainer` performs better than AMCF-PH on all ten metrics.
3. *Global popularity:* Similar to the two previous datasets, global popularity performs the best for the test set and recommendations coverage, while performing worse than `RecXplainer` and even the *random* baseline on all personalization metrics.
4. *User-specific popularity:* Similar to the two previous datasets, user-specific popularity performs the second best for the test set and recommendations coverage, while performing worse than `RecXplainer` and even

the *random* baseline on all personalization metrics.

5. *Random:* `RecXplainer` performs better than the random baseline performs on all ten metrics.
6. *`RecXplainer`:* `RecXplainer` performs the third best on the two coverage metrics and performs the best on all eight personalization metrics, thereby demonstrating its ability to provide personalized explanations with high coverage.

Overall, we conclude that `RecXplainer` performs better than all baselines for most metrics. When it is not the best, it is usually within a couple of percentage points of the best performing technique. LIME-RS usually does not perform well on any metric and is unable to even scale to 3 out of 5 datasets owing to the expensive cost of training a separate regression model for each item. `RecXplainer` completely avoids this problem due to amortization; it is very cheap to produce `RecXplainer` explanations during deployment. AMCF-PH also does not perform well on most metrics for all datasets. Global and user-specific popularity perform well on the coverage-based metrics; however, that is due to the skewed distribution of the attributes, and their explanations are usually uninformative and unpersonalized.

4.5 Limitations and Conclusions

In this work, we proposed a novel attribute-based explainability technique, `RecXplainer`, which explains recommendations for collaborative filtering-based recommender systems using the attributes of the item. To the best of our knowledge, `RecXplainer` is the **first and only** existing technique that provides such explanations in a post-hoc and amortized manner. We studied the performance of `RecXplainer` and five baselines using five large-scale recommender systems datasets trained on a variety of collaborative filtering architectures. As our results indicate, `RecXplainer` strikes an excellent balance between coverage and personalization of the explanations and performs better than all five baselines on most metrics.

The minor limitation of `RecXplainer` is the cost of training the auxiliary model. For all five recommender systems, training of the auxiliary model took between 10 and 30 minutes, depending on the dataset size (significantly less than training the recommender systems, which took 6-12 hours). Once trained, these models can be used to explain recommendations for any item for any user. The explanation generation during inference is very cheap; it took less than 10 minutes to compute the general and specific preferences for all the users, even for the largest datasets. This is significantly faster in contrast to LIME-RS, which would take 11 days, and AMCF-PH which took 1-3 hours.

Chapter 5

How Many Images Does It Take? Estimating Imitation Thresholds in Text-to-Image Models

Text-to-image models are trained using large datasets of image-text pairs collected from the internet. These datasets often include copyrighted and private images. Training models on such datasets enables them to generate images that might violate copyright laws and individual privacy. This phenomenon is termed *imitation* – generation of images with content that has recognizable similarity to its training images. In this work, we estimate the point at which a model was trained on enough instances of a concept to be able to imitate it – the *imitation threshold*. We posit this question as a new problem and propose an efficient approach that estimates the imitation threshold without incurring the colossal cost of training these models from scratch. We experiment with two domains – human faces and art styles – and evaluate four text-to-image models that were trained on three pretraining datasets. We estimate the *imitation threshold* of these models to be in the range of 200-700 images, depending on the domain and the model. The *imitation threshold* provides an empirical basis for copyright violation claims and acts as a guiding principle for text-to-image model developers that aim to comply with copyright and privacy laws.

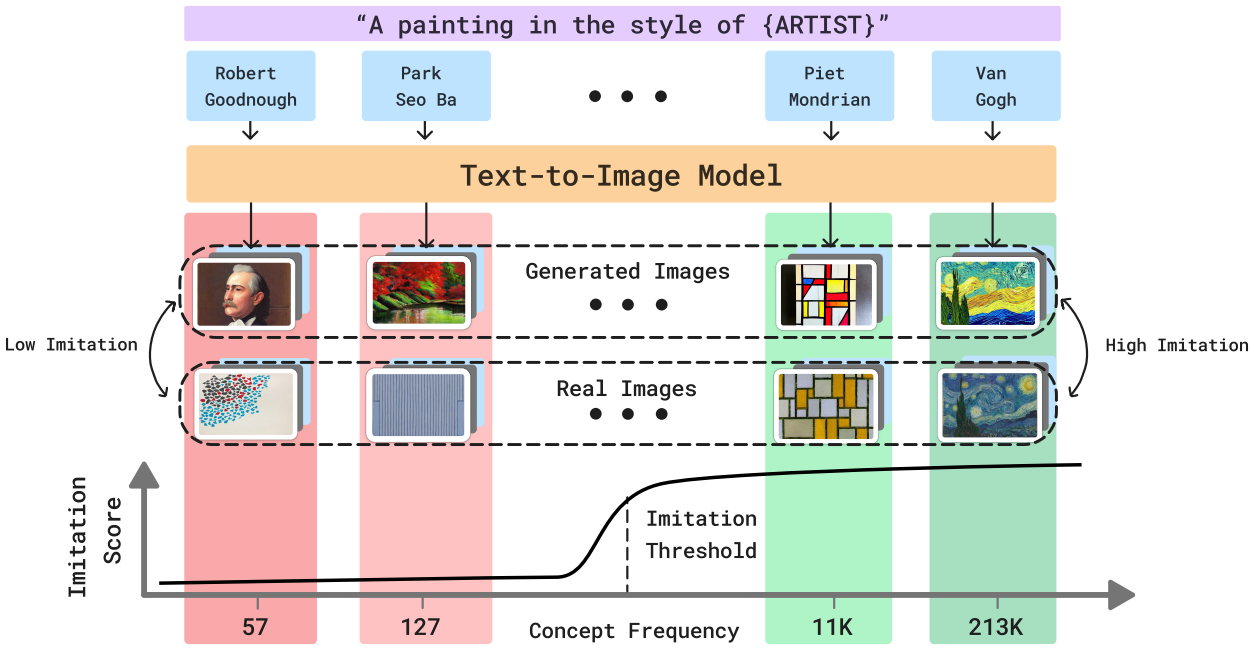


Figure 5.1: An overview of the imitation phenomenon where we seek the *imitation threshold* – the point at which a model was exposed to enough instances of a concept that it can reliably imitate it. The figure shows four concepts (e.g., Van Gogh's art style) that have different counts in the training data (e.g., 213K for Van Gogh). As the image count of a concept increases, the ability of the text-to-image model to imitate it increases (e.g., Piet Mondrian's and Van Gogh's art styles have higher imitation). The *imitation threshold* represents the number of instances a model has to be trained on such that humans recognize such a concept in generated images.

5.1 Introduction

The progress of text-to-image models in recent years [187, 193, 85, 274] is much attributed to the availability of large-scale pretraining datasets like LAION [205]. These datasets consist of semi-curated image-text pairs scraped from Common Crawl, which leads to the inclusion of explicit, copyrighted, and private material [34, 110, 254, 117, 17]. Training models on such images may be problematic because text-to-image models can *imitate* — generate images with highly recognizable features from their training data [223, 28]. This behavior has both legal and ethical implications, such as copyright infringements as well as privacy violations of individuals whose images are present in the training data without consent, that has led to several lawsuits by artists against such model providers [202].

Previous work proposed methods for detecting when models memorize training images [223, 28], and mitigation techniques [224, 213]. For instance, researchers found that duplicate images increase the chance of memorization. Typically, *memorization* refers to the replication of a specific training image. Instead of mea-

suring *memorization*, we focus on *imitation* - a broader and under-explored sense of memorization, where a concept is recognizable from a generated image.

In this work, we ask **how many instances of a concept does a text-to-image model need to be trained on to imitate it**, where a concept can refer to a specific person or a specific art style, for example. Establishing such an *imitation threshold* is useful for several reasons. First, it offers an empirical basis for copyright and privacy violation claims, suggesting that if a concept's prevalence in the training data is below this threshold, a model trained on such data cannot reproduce such concept [202, 254, 35]. Second, it acts as a guiding principle for text-to-image model developers who want to avoid such violations. Finally, it reveals an interesting connection between training data statistics and model behavior, and the ability of models to efficiently harness training data [240, 29]. We posit this question as a new problem, and provide its schematic overview in Figure 5.1.

To find the gold imitation threshold, one has to train counterfactual models while varying the number of images of a concept and measuring the models' imitation abilities. However, training even one of these models is extremely expensive [12]. Instead, we propose a tractable alternative, **M**easuring **Im**itation Thr**E**shold **T**hrough **I**nstance **C**ount and **C**omparison (MIMETIC^2), that estimates the threshold without training multiple models using observational data. We start by collecting a large set of concepts (e.g., various kinds of art styles) per domain (e.g., the domain of art styles), and use a text-to-image model to generate images for each concept. Then, we compute the frequency (in the training data) and the imitation score of each concept (§5.3.1). Then, we estimate the imitation scores using dedicated embeddings that capture the similarities of concepts (§5.3.2). Finally, we estimate the imitation threshold for each domain using a *change detection* algorithm [127] (§5.3.3).

Operating with observational data, a naive approach may lead to a biased estimate as confounding variables such as the quality of the imitation scoring model on different groups within the domain, or estimating the training frequencies of concepts (e.g., simple counts of 'Van Gogh' in the captions results in a biased estimate since the artist may be mentioned in the caption without their painting in the corresponding image). We carefully tailor MIMETIC^2 to deconfound such variables (§5.3). Since our approach is observational in nature, we also conduct counterfactual experiments to validate our results (§5.9). We train models on new concepts that were not seen in the training data, and vary the concept frequency to several values both

smaller and larger than our estimated imitation threshold. These experiments demonstrate that our estimated imitation thresholds are accurate (with a difference of upto 10% in the estimated thresholds).

Overall, we propose and formalize a new problem that estimate the number of images of a concept that a text-to-image model requires for imitating it (§5.2), and propose a method MIMETIC^2 that efficiently estimates the *imitation threshold* for these models (§5.3). We apply our approach to the domains of human face and art style imitation on four text-to-image models and estimate the imitation thresholds for such models to be in the range of 200-700 images (§5.5). These imitation thresholds provide concrete insights on imitation abilities, can act as an empirical basis for copyright violation claims, and as guiding principles for model developers.

5.2 Problem Formulation and Overview

In this work, we seek to find the minimal number of images of a *concept* a text-to-image model requires in order to imitate it.

Defining Concept: We follow the same definition of concepts as previous work [240]: we consider specific instances of human faces and artist styles as distinct concepts. As such, each item in a domain constitutes a concept; human individuals and artistic styles in the *human face* and *art style* domains, respectively. While there might be some visual similarities between concepts, especially for art styles, we employ discriminator models that distinguish between concepts with high accuracy (which is necessary for the accurate estimation of the imitation threshold).

Setup: Our setup involves a training dataset $\mathcal{D} \triangleq \{(\mathbf{x_i}, \mathbf{y_i}) \mid \mathbf{x_i} \in \mathcal{X},\ \mathbf{y_i} \in \mathcal{Y}\}_{i=1}^{n}$, composed of n (image, caption) pairs, where $\mathcal{X}$ and $\mathcal{Y}$ represents space of images and text captions, respectively, and a model $\mathcal{M}$ that is trained on $\mathcal{D}$ to generate an image $\mathbf{x}$ given the text-caption $\mathbf{y}$. $\mathcal{M}(\mathbf{y})$ denotes the generated image for a provided caption $\mathbf{y}$. Let $\mathbb{I}^j(\mathbf{x})$ be an indicator function that indicates whether a concept Z^j is present in an image $\mathbf{x}$. Each concept Z^j appears $c^j = \sum_i \mathbb{I}^j(\mathbf{x_i})$ times in the dataset $\mathcal{D}$, where the dataset may contain multiple concepts $\{Z^1, Z^2, \ldots\}$. Lastly, $\mathcal{M}_k^j$ denotes a model trained on a dataset where $c^j = k$.

The *imitation threshold* is the minimal number of training images containing a concept Z^j that model $\mathcal{M}$

generates images $\mathcal{M}(\mathbf{p}^j)$ (with different random seeds) from a prompt $\mathbf{p}^j$ which mentions the concept Z^j, and Z^j is visually recognizable. [1] For example, if Z^j refers to Van Gogh's art style, then the imitation threshold is the minimal number of training images of Van Gogh's paintings in a dataset used to train a text-to-image model, for which the model can generate images that imitate Van Gogh's art style. We consider a generated image to be an imitation of a concept if the similarity between the generated image and the original images of that concept in the training data is above a threshold. We measure the similarity using a concept-specific detector model, and we determine the detection threshold by conducting experiments on the original images of the concept (Section 5.3.2).

$$\textit{Imitation Threshold}^{\,j} \triangleq \min\left\{k \in \{1,\dots,n\} : \mathbb{I}^j(\mathcal{M}_k^j(\mathbf{p}^j)) = 1\right\}$$

Optimal Approach Finding the *imitation threshold* is a causal problem; The gold threshold can be achieved by performing the counterfactual experiment [178]: For each concept Z^j, create k training datasets $\{\mathcal{D}_1^j, \mathcal{D}_2^j, \dots, \mathcal{D}_k^j, \dots\}$, and train a model $\mathcal{M}_k^j$ on each dataset $\mathcal{D}_k^j$. Once we find a model, $\mathcal{M}_k^j$ which is able to generate images where the concept Z^j is recognizable, but $\mathcal{M}_{k-1}^j$ cannot, we deem k as the *imitation threshold* for that concept. [2] *However, due to the high costs of training even one text-to-image model [12], this approach is impractical.* Note that the background (contextual) dataset also has to be varied, as it might have concepts that can influence the imitation threshold of the concept of interest. This would add additional cost to the already impractical cost of training a model with varying counts of the concept of interest.

MIMETIC^2 Instead, we propose a tractable approach that efficiently estimates the imitation threshold while relying on certain assumptions (discussed at the end of this section). The key idea is to use observational data instead of training multiple models with a different number of images of a concept. This idea is a common approach to answer causal questions, when performing interventions is expensive, or unethical, inter alia, [178, 141, 156]. Concretely, we collect several concepts from the same domain (e.g., art styles) with varying image counts in the training dataset $\mathcal{D}$ of a pretrained model $\mathcal{M}$. Then, we identify the count where the model $\mathcal{M}$ starts imitating concepts, and deem this count to be the *imitation threshold* for that domain. **Note that this threshold is domain-specific and not concept-specific**, where a *domain* is defined

[1] Prompts $\mathbf{p}$ are usually different from the training data captions, $\mathbf{y}$.

[2] In an ideal world, we can train $\mathcal{O}(log(k))$ models, with an estimated cost of \$10M if $k = 100,000$ (see Section 5.18).

as the abstract set that contains the specific concept we want to measure the imitation threshold for, e.g., if the concept refers to Van Gogh's art style, then its domain would be art styles. **Therefore, in this setup, MIMETIC^2 can estimate the imitation threshold for the domain of art styles, but it cannot estimate it for Van Gogh's art style specifically.**

Assumptions To estimate the imitation threshold using observational data, we make three assumptions. These are standard assumptions in the field, which are necessary for answering such complex questions. Crucially, we are able to empirically validate all of our assumptions, and find they hold for our datasets and models (Section 5.9). First, we assume distributional invariance between the images of all concepts in a domain. Under this assumption, measuring the imitation score of a concept Z^j for a counterfactual model $\mathcal{M}_{k'}$ that is trained with k' images of Z^j is equivalent to measuring the imitation score of another concept Z^i that currently has k' images in the already trained model $\mathcal{M}$

$$Imitation\ Score(\mathcal{M}^j_{k'}(Z^j)) \approx Imitation\ Score(\mathcal{M}^i_{k'}(Z^i))$$

This assumption allows us to use observational data to estimate the imitation threshold without training models from scratch.

Second, we assume that there are no confounders between the imitation score and the image count of a concept, i.e, the imitation score of a concept is not affected by the presence of other concepts in the training data.

Third, we assume each image of a concept contributes equally to the learning of the concept. We further discuss these assumptions and provide empirical evidence that they hold for the datasets we experiment with in section 5.9.

5.3 Proposed Methodology: MIMETIC^2

We illustrate our proposed methodology in Figure 5.2. At a high level, for a specific domain (e.g., human faces), MIMETIC^2 estimates the frequency of each concept in the pretraining data (§5.3.1) and estimates the model's ability to imitate it (§5.3.2). We then sort the concepts based on their frequencies, and find the imitation threshold using a change detection algorithm (§5.3.3).

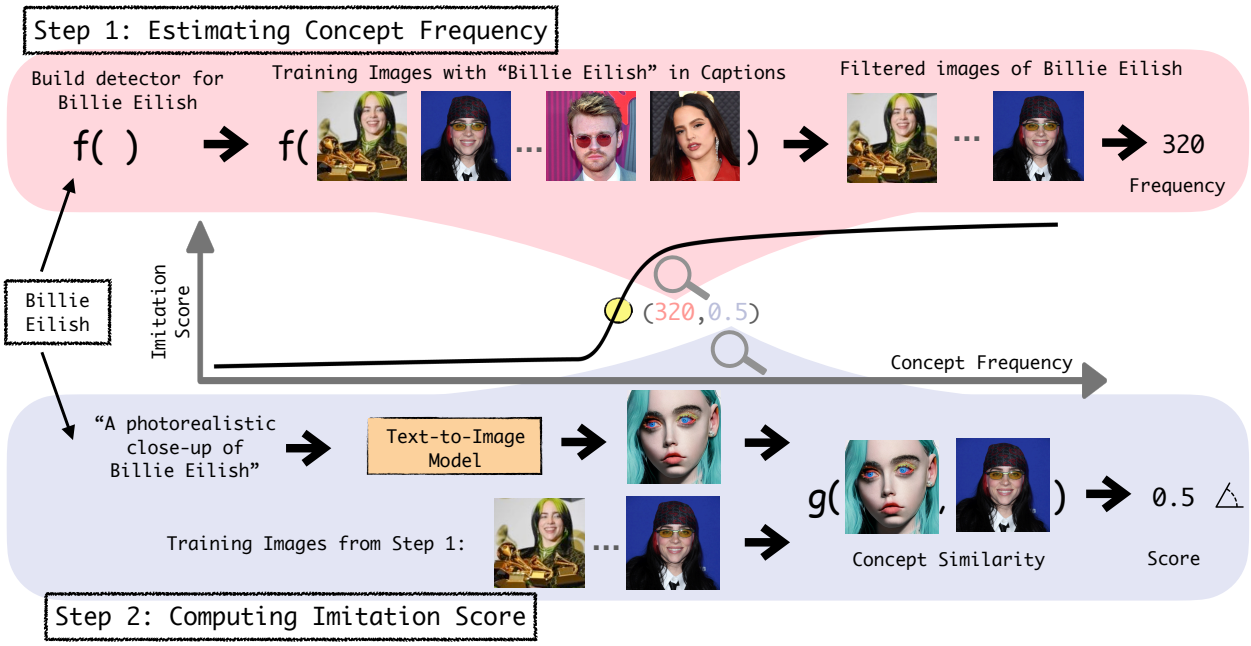


Figure 5.2: Overview of MIMETIC^2's methodology to estimate the *imitation threshold*. In Step 1, we estimate the frequency of each concept (belonging to a domain) in the pretraining data by obtaining the images that contain the concept of interest. In Step 2, we use the filtered images of each concept (obtained in Step 1) and compare them to the generated images to measure imitation (using g that receives training and generated images). We repeat this process for each concept to generate the imitation score graph, and then determine the *imitation threshold* with a change detection algorithm.

5.3.1 Computing Concept Frequencies

Challenges Determining a concept's frequency in a multimodal dataset can be achieved by employing a high-quality classifier for that concept over every image and counting the number of detected images. However, given the scale of modern datasets with billions of images, this approach is expensive and prone to classification errors.

Estimating Concept Frequency

We first make a simplifying assumption that a concept is present only if an image's caption mentions it. We empirically test the validity of this assumption and find it to be accurate (Section 5.10). However, this is not sufficient because concepts often do not appear in the corresponding images, even when they are mentioned in the captions. For instance, Figure 5.3 showcases images whose captions contain "Mary Lee Pfeiffer", but such images do not always include her. On average, in our experiments, we find that concepts occur only in 60% of the images whose caption mentions the concept.

To address this problem, we start by retrieving all images whose caption mentions the concept of interest and then further filter out the images that do not contain the concept, as detected by a classifier (we build a classifier that indicates the presence of a particular concept in the image). We retrieve these images using WIMBD [72], a search tool based on a reverse index that efficiently finds documents (captions) in a dataset containing the search query (concept). To build a classifier for each concept, we construct a set of high quality reference images. For example, a set of images with only the face of a single person (e.g., Brad Pitt). We collect these images automatically using a search engine (we use images from a search engine because several of the concepts in our experiments have a low number of images that cannot be used to build a classifier), followed by a manual verification to vet the images (see Section 5.11 for details). These images are used as gold reference for automatic detection of these concepts in the images. We collect up to ten reference images per concept.

Figure 5.3: LAION captions that mention 'Mary Lee Pfeiffer', the mother of Tom Cruise. She is not always present in the images (the rightmost image).

To classify whether a candidate image contains the concept of interest or not, we embed the candidate image and the concept's reference images using an image encoder (discussed in §5.3.2) and measure the similarity between the embeddings. If the similarity between a candidate image and any of the reference images is above some threshold, we consider that candidate image to contain the concept of interest. This threshold is established by measuring the similarity between images of the same concepts and images of different concepts, which maximizes the true positives and minimizes false positives. We provide additional details on determining the thresholds per domain in Sections 5.12 and 5.13.

Finally, we run the classifier on all candidate images whose caption mentions the concept, and take those that are classified as positive. For each concept, we retrieve up to 100K candidate images from the pretraining dataset. We use the ratio of positive predictions to the total number of retrieved candidate images, multiply it by the total caption count of a concept in the dataset, and use that as the concept's frequency estimate. For concepts with fewer than 100K candidate images, we count the number of images that are positively classified. Note that some URLs from the pretraining datasets we use are dead, a common phenomenon for URL-based datasets ("link rot" [26, 133]). On average, we successfully retrieved 74% of the candidate

images.

5.3.2 Computing the Imitation Score

Challenges Computing the imitation score entails determining how similar a concept is in a generated image compared to its source images from the training data. Several approaches were proposed to accomplish this task, such as FID and CLIPScore [201, 104, 199, 103, 180]. To measure similarity, these approaches compute the similarity between the distributions of the embeddings of the generated and training images of a concept. The embeddings are obtained using image embedders like the Inception model in the case of FID and CLIP in the case of CLIPScore. These image embedders often perform reasonably well in measuring similarity between images of common objects, which constitutes most of their training data. However, they cannot reliably measure the similarity between two very similar concepts like the faces of two individuals or the art style of two artists [225, 103, 6, 114]. Therefore, MIMETIC^2 uses domain-specific image embedders to measure similarity between two concepts from that domain.

We embed the generated and real images of concepts using these embedders and use similarity to measure a continuous imitation score. To establish a threshold, we use a change detection algorithm (described next; §5.3.3). At a high level, given a sequence of imitation scores, the algorithm finds the threshold on the scores above which the score is considered a (binarized) imitation of the said concept. As we are interested in finding a (binary) threshold, we binarize the score, which we find to correspond to the human perception imitation of a concept. To verify the validity of the binarized scores, we conduct human subject experiments and find a high correlation between human perception of imitation and the binarized threshold (Section 5.5). We leave to future work the investigation of the continuous imitation score beyond the threshold.

Estimating Imitation Score We use a face embedding model [58] for measuring face similarity and an art style embedding model [225] for measuring art style similarity. We find that even the specific choice of these models is crucial; for instance, in early experiments, we used Facenet [203], and observe it struggles to distinguish between individuals of certain demographics, causing drastic differences in the imitation scores between demographics. We provide more details on these early experiments in Section 5.17, and show that our final choice of embedding models works well on different demographics.

To measure the imitation score, we embed the generated images and filtered training images of a concept

(obtained in §5.3.1) using the domain-specific image embedder, and report the imitation score as the average imitation between all generated and top-10 most similar training images (cosine similarity between embeddings of generated and training images). To ensure that the automatic measure of similarity correlates with human perception, we also conduct experiments with human subjects and measure the correlation between the similarities obtained automatically and in the human subject experiments. We find this correlation to be high for both domains we experiment with (§5.5). We also do human subject experiments to verify the correctness of the binarized threshold and find this correlation to be high as well (§5.5).

5.3.3 Detecting the Imitation Threshold

After computing the frequencies and the imitation scores for each concept, we sort them in ascending order of their image frequencies. This generates a sequence of points, each of which is a pair of image frequency and the imitation score of a concept. We apply a standard change detection algorithm, PELT [127], to find the image frequency where the imitation score significantly changes. Change detection is a classic statistical approach to find the points where the mean value of a sequence (e.g., stock value or network traffic) changes significantly. We choose PELT because of its linear time complexity in computing the change point [242]. We choose the first change point as the imitation threshold (see Section 5.15 for details about all change points). The application of change detection assumes that increasing the image counts beyond a certain threshold leads to a large jump in the imitation scores, and we find this assumption to be accurate in our experiments (Figure 5.5a).

To obtain error bounds for our thresholds, we perform a permutation test by sampling a subset of concepts per domain and dataset, and compute the threshold for the sampled set. We repeat this 1,000 times and report the mean and standard error of the thresholds.

5.4 Experimental Setup

Text-to-image Models and Training Data We use Stable Diffusion (SD) as the text-to-image model [193]. We use them because both the models and their training datasets are open-sourced. Specifically, we use SD1.1 and SD1.5 that were trained on LAION2B-en, a 2.3 billion image-caption pairs dataset, filtered to

Table 5.1: Domains, datasets, pretraining data, and models we use.

Domain	Dataset	Pretraining Data	Model
Human Faces 🧑	Celebrities, Politicians	LAION400M	LDM
Human Faces 🧑	Celebrities, Politicians	LAION2B-en	SD1.1, SD1.5
Human Faces 🧑	Celebrities, Politicians	LAION5B	SD2.1
Art Style 🖼️	Classical, Modern	LAION400M	LDM
Art Style 🖼️	Classical, Modern	LAION2B-en	SD1.1, SD1.5
Art Style 🖼️	Classical, Modern	LAION5B	SD2.1

Table 5.2: Prompts used to generate images of human faces and art styles.

Human faces 🧑	Art style 🖼️
A photorealistic close-up photograph of X	A painting in the style of X
High-resolution close-up image of X	An artwork in the style of X
Close-up headshot of X	A sketch in the style of X
X's facial close-up	A fine art piece in the style of X
X's face portrait	An illustration in the style of X

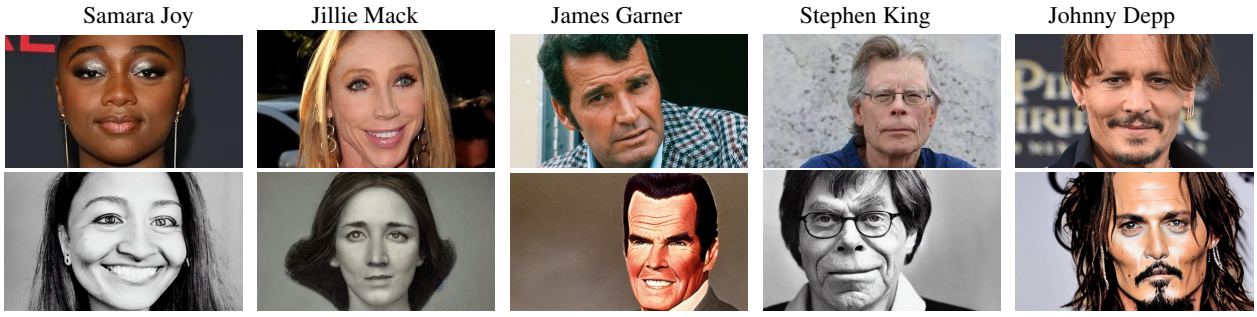


Figure 5.4: Examples of real celebrity images (top) and generated images from a text-to-image model (bottom) with increasing image counts from left to right (`3`, `273`, `3K`, `10K`, and `90K`, respectively). The prompt is "a photorealistic close-up image of {name}".

contain only English captions, and we use SD2.1 that was trained on LAION-5B, a 5.85 billion image-text pairs dataset that includes captions in any language [205]. Finally, we also use a latent diffusion model (LDM) trained on LAION-400M [204], a 400M image-text pairs dataset.

Domains and Concepts We experiment with two domains – *art styles* 🖼️ and *human faces* 🧑 that are highly important for privacy and copyright considerations of text-to-image models. Figures 5.1 and 5.4 show examples of real and generated images of art styles and human faces. We collect two sets of concepts for each domain. For art styles, we collect classical and modern art styles, and for human faces, we collect celebrities and politicians. It is important to note that each artist's style and each human face is considered

Table 5.3: The mean and std. error of the *imitation thresholds* for the models and domains we experiment with.

		Human Faces 🧑		Art Style 🖼️	
Pretraining	**Model**	**Celebrities**	**Politicians**	**Classical**	**Modern**
LAION-400M	LDM	626 ± 3	449 ± 2	416 ± 3	412 ± 2
LAION2B-en	SD1.1	399 ± 3	284 ± 3	304 ± 2	208 ± 1
	SD1.5	371 ± 1	302 ± 4	302 ± 2	212 ± 1
LAION-5B	SD2.1	617 ± 5	385 ± 5	330 ± 4	292 ± 3

a separate concept. Although in the real world, the art style of some artists might be similar, the classifier model we use to measure imitation is trained well to distinguish between art styles from different artists [225]. These sets we experiment with are independent (i.e., have no common concept) and are therefore useful to test the robustness of the thresholds (§5.5). Each set has 400 concepts that cover a wide frequency range in the pretraining datasets. Table 5.1 summarizes the domains, sets used for each domain, the models, and their pretraining datasets we experiment with.

Image Generation We generate images for each domain by prompting models with five prompts (Table 5.2). We design domain-specific prompts that encourage the desired concept to occupy a large part of the generated image, which simplifies the imitation score measurement. We also ensure that these prompts are distinct from the captions used in the pretraining dataset to minimize direct reproduction of training images (as noted by [224]). We generate 200 images per concept using different random seeds for each prompt, a total of 1,000 images per concept.

5.5 Results: The *Imitation Threshold*

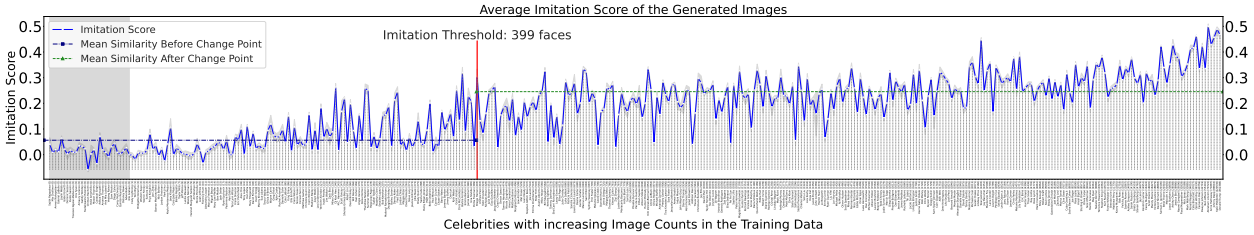
We use MIMETIC^2 to estimate the imitation threshold for each model-data pair, and present the results in Table 5.3. The imitation thresholds for SD1.1 on celebrities and politicians are 399 ± 85 and 284 ± 87 respectively, and 304 ± 56 and 208 ± 26 for classical and modern art styles, respectively. Note that these results hold in the context of the training dataset used to train the text-to-image models we experiment with (LAION-2B). Interestingly, SD1.1 and SD1.5 have almost identical thresholds across the four datasets. Notably, both SD1.1 and SD1.5 are trained on LAION2B-en. The imitation thresholds for SD2.1, which is trained on LAION-5B (a superset of LAION-2B), are higher than the thresholds for SD1.1 and SD1.5.

We hypothesize that the difference in performance of SD2.1 and SD1.5 is due to the difference in their text encoders (note: differences in performance between SD1.5 and SD2.1 were also reported by several users on online forums [170]). Finally, the thresholds for all four datasets are slightly higher for the LDM model compared to all the SD models. We hypothesize that this might happen due to the smaller training data size of 400M pairs, and leave further investigation for future work.

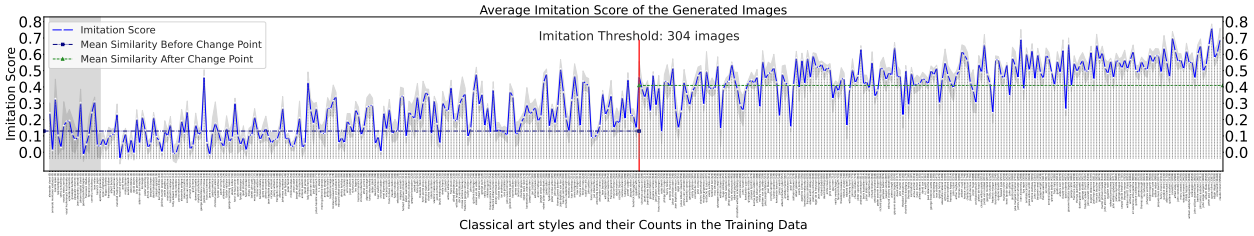
We also present the plots of the imitation scores as a function of the image frequencies of the concepts for the four datasets. Figures 5.5a and 5.5b show the imitation graphs of celebrities and classical art styles, respectively, for SD1.1. In Figure 5.5a, we observe that the imitation scores for individuals with low image frequencies are close to 0 (left side), and increase as the image frequencies increase (towards the right side). The highest similarity occurs for individuals in the rightmost region of the plot. We also observe a low variance in the imitation scores across prompts for the same concept (the mean of the variance is 0.0003 with a standard deviation of 0.0005 across concepts), and note that this variance is independent of the concept frequencies – indicating that the performance of the face embedding model does not depend on the popularity of the individual. Similarly, in Figure 5.5b, we observe that imitation scores for art styles with low image frequencies are low (close to 0.2 on the left side), and increase as the image frequencies increase (towards the right side). The highest similarity occurs for the artists in the rightmost region of the plot. We also notice a low variance across the generation prompts, and the variance does not depend on the popularity of the artist (the mean of the variance is 0.003 with a standard deviation of 0.003). We showcase the imitation graphs for all other models in Section 5.16, which follow similar trends.

Human Perception Evaluation To determine if the automatic measure of similarity between generated and training images matches human perception, we conduct experiments with human subjects. We ask participants to rate generated images on the Likert scale [148] of 1-5 based on their similarity to real images of celebrities. To avoid any confirmation bias, the participants were not informed of the research objective of this work.

For human face imitation, we conduct this study with 15 participants who were asked to rate 100 (randomly selected) generated images for a set of 40 celebrities on a scale of 1-5. To determine the accuracy of the imitation threshold estimated by MIMETIC^2, we randomly select the celebrities such that half of them have image frequencies below the threshold and the other half above it. We measure the Spearman's correlation



(a) Human Face Imitation. The imitation threshold is detected at **399 ± 85 faces**.



(b) Art Style Imitation. The imitation threshold is detected at **304 ± 56 images**.

Figure 5.5: Human Face and **Art Style** imitation graphs for SD1.1 using the *Celebrities* and *Classical art style* sets. The x-axis is the image frequencies, and the y-axis is the imitation score averaged over the five prompts. Concepts with zero image frequencies are shaded in light gray. We show the mean imitation score and its variance over the five image generation prompts for each concept. The red vertical line indicates the imitation threshold found by the change detection algorithm, and the horizontal green line represents the average imitation scores before and after the threshold.

[283] between the imitation scores computed by the model and the ratings provided by the participants. Due to the variance in perception, we normalize the ratings for each participant. The Spearman correlation between the similarity scores provided by participants and the imitation scores is **0.85, signifying a high-quality imitation estimator.** We also measure the agreement between the imitation threshold that MIMETIC^2 estimates and the threshold that humans perceive. For this purpose, we convert the human ratings to binary values and treat them as the ground truth (any rating of 3 or higher is treated as 1, and ratings less than 3 are treated as 0, indicating successful and unsuccessful imitation, respectively). MIMETIC^2 predictions are also binarized; any celebrity with frequency higher than the imitation threshold is treated as 1, otherwise 0. To measure the agreement, we compute the element-wise dot product between these two sets. We find the agreement to be 82.5%, signifying a high degree of agreement for MIMETIC^2's automatically computed threshold.

For art style imitation, we conduct this study with an art expert due to the complexity of detecting art styles. The participant was asked to rate five generated images for 20 art styles, half of which were below the imitation threshold and the other half, above the threshold. We find the Spearman correlation between the

two quantities to be **0.91** – demonstrating that our imitation scores are highly correlated with an artist's perception of style similarity. Similar to the previous case, we measure the agreement of the imitation threshold, which we find to be 95% – signifying a high degree of agreement for MIMETIC^2's computed threshold.

Evaluation of the Imitation Thresholds Evaluating the accuracy of the real imitation thresholds is expensive, as it requires conducting the counterfactual experiment described in Section 5.2 (training $\mathcal{O}(\log m)$ models with different numbers of images of each concept). To make the evaluation tractable, we finetune a pretrained text-to-image model with concepts that the model has not seen during pretraining and find the number of images required for the model to imitate them. We find that all the concepts require about 250 images for imitation when the models are finetuned on them (further details in Section 5.14), which is very close to the imitation threshold MIMETIC^2 finds for this pretrained model without finetuning (234 images). Based on these results, we conclude that the imitation thresholds estimated by MIMETIC^2 are accurate.

5.5.1 Discussion

Overall, we observe that the imitation thresholds are similar across the different image generation prompts, but are domain and model-dependent. Importantly, the thresholds computed by MIMETIC^2 have a high degree of agreement with human subjects and are supported by the finetuning experiments.

We find that celebrities have a higher imitation threshold than politicians for all models. We hypothesize this happens due to inherent differences in the data distribution between these two sets, which makes it harder to learn the concept of celebrities than politicians. To test this hypothesis, we compute the average number of training images that have *a single person* for concepts with less than 1,000 images. We find that politicians have about twice the number of single-person images compared to celebrities. As such, images that have a single person make imitation easier for the model (the model can easily associate the face with the name in the caption), thus lowering the imitation threshold for politicians.

We also note the presence of several outliers in both plots, which can be categorized into two types: (1) concepts whose image frequencies are lower than the imitation threshold, but their imitation scores are considerably high; and (2) concepts whose image frequencies are higher than the imitation threshold, but their imitation scores are low. From a privacy perspective, the first kind of outliers is more concerning

(a) Outlier Kind 1. *Thandiwe Newton* is also aliased as *Thandie Newton*. Since MIMETIC^2 only collects images whose caption mentions *Thandiwe Newton*, this leads to an underestimation of image counts.

(b) Outlier Kind 2. Most of the images whose captions mention *Cacee Cobb* have multiple people in them; only 6 images have her as the only person, leading to a low imitation score in generated images.

Figure 5.6: Examples of the two kinds of outliers. The top and bottom rows show the real and SD1.1 generated images, respectively. Images were generated using the prompt: "a photorealistic close-up image of *name*."

than the second ones, since the imitation threshold should act as a privacy threshold. When used as a privacy guideline, it is less worrisome if a concept with a concept frequency higher than the threshold is not imitated by the model (false positive), but it would be a privacy violation if a model can imitate a concept with frequency lower than the threshold (false negative). Upon further analysis, we find that for all of the false negative outliers (with the privacy concern), the concepts' true frequencies are much higher than our estimates. This happens primarily due to *naming aliases* (a person known by multiple names) that MIMETIC^2 does not account for – thereby alleviating the privacy violation concerns (see §5.6 for a detailed analysis of such outliers).

Finally, we notice that the range of the imitation scores of different domains have different y-axis scales. This is due to the difference in embedding models used in both cases. The face embedding model can distinguish between two faces much better than the art style model can distinguish between two styles (see Sections 5.12 and 5.13), and therefore the scores for the concepts on the left side of the imitation threshold are around 0 for face imitation and 0.2 for style imitation. However, the absolute values on the y-axis do not matter for estimating the imitation threshold as long as the trend is similar, which is the case for both domains.

5.6 Analysis: Investigating Outliers

The imitation score plots in the previous section, while showcasing a clear trend, have several outliers. In this section, we analyze such outliers and present examples in Figure 5.6.

Low Image Counts and High Imitation Scores Figure 5.6a shows an example of such a case: *Thandiwe Newton*'s image count is 172 in LAION2B-en, lower than the *imitation threshold* for celebrities: 364. However, her imitation score of 0.26 is much higher than those of neighboring celebrities with similar image counts (with scores of 0.01 and 0.04). Further investigation reveals that Thandiwe Newton is also known as *Thandie Newton*. Since this alias may also be used to describe her in captions, we underestimate her image counts. We repeat the process for estimating the image counts with the new alias, and find that *Thandie Newton* appears in 12,177 images, bringing the cumulative image count to 12,349, which significantly surpasses the established imitation threshold. The two aliases, whose total image count is considerably higher than the imitation threshold, differ by only a single letter and are similarly represented by the model's text encoder (cosine similarity of 0.96), which explains the high imitation score. We find that most of the celebrities who had a high imitation score while having a low concept frequency are also known by other names, which leads to underestimating their image counts. For example, *Belle Delphine* (394 images) also goes by *Mary Belle* (310 images), for a total of 704, and *DJ Kool Herc* (492 images) also goes by *Kool Herc* (269 images), for a total of 761. The aliases explanation also explains the outliers in art style imitation. For instance, artist *Gustav Adolf Mossa* (19 images) also goes by just *Mossa* (15,850 images).

High Image Counts and Low Imitation Scores Several celebrities have higher image counts than the imitation threshold, but low imitation scores. Unlike the previous case, we were unable to find a common cause that explains all these outliers. However, we find explanations for specific cases. For example, a staggering proportion of the training images for such celebrities have multiple people in them (20% of the outliers of this kind could be explained with this phenomenon). Out of the 706 total images of *Cacee Cobb*, only 6 images have her as the only person in the image (see Figure 5.6b). Similarly, out of 1,296 total images of *Sofia Hellqvist*, only 67 images have her as the only person, and out of the 472 total images of *Charli D' Amelio*, only 82 images have her as the only person. We hypothesize that having multiple concepts in an image impedes the proper mapping of the concept's text embedding to its image embedding, which can explain the low imitation score for these concepts. We leave examining this to future work.

5.7 Discussion and Limitations

In this work, we attribute the imitation of a concept by a model to its frequency in the training data. However, several other factors like image resolution, image diversity, alignment between images and their captions, etc., may affect imitation. Our work is the first to tackle such an important question, and as such, we tackle one of the more salient features that contributes to imitation - the concept's frequency in the training data. In the future, we plan to incorporate other features to improve the accuracy of the imitation threshold. While we hope our work could be used for copyright and privacy claims, our results should be put in the context of the assumptions we make (which we find to empirically hold). Note that MIMETIC^2 finds the imitation threshold for the pretraining regime. Different training regimes (such as finetuning on additional data, for multiple epochs) may also affect the imitation threshold, and we hope to investigate such scenarios in future work.

5.8 Conclusions

Text-to-image models can imitate concepts from their training images [223, 28, 224] This behavior is crucial for learning, but it can also be concerning when such training datasets include copyrighted and private images. Imitating such images would be grounds for violation of copyright and privacy laws. In this work, we seek to find the number of instances of a concept that a text-to-image model has to be trained on in order to imitate it – the *imitation threshold*. We posit this as a new problem and propose an efficient method, MIMETIC^2, for estimating such a threshold. Our method uses pretrained models to estimate this threshold for human faces and art styles imitation using four text-to-image models trained on three different pretraining datasets. We find the imitation threshold of these models to be in the range of 200-700 images, depending on the setup. By estimating the imitation threshold, we provide insights on successful concept imitation based on their training frequencies. Our results have striking implications for both the text-to-image models users and developers. These thresholds can inform text-to-image model developers what concepts are at risk of being imitated, and on the other hand, serve as a basis for copyright and privacy complaints. Finally, MIMETIC^2 also provides insights into the generalization capabilities of text-to-image models.

5.9 Verification of the Assumptions

We empirically test the validity of the three assumptions we make in Section 5.2, and find they all hold in practice. To test the invariance assumption, we calculate the imitation scores difference of all concept pairs, whose image counts differ by less than 10 images, for all concepts. We find that the imitation scores difference for such pairs, on average, is 0.003 (averaged across such pairs and averaged across our datasets) (Section 5.9.1), attesting to the fact that this assumption is not egregious for the dataset and concepts we experiment with. Testing the lack of confounders between imitation score and image count, we finetune a text-to-image model on images of concepts that the model has not seen during pretraining (e.g., politicians that became popular after the training data of the model was curated), and measure the imitation threshold for these concepts Section 5.9.2. We argue that if the thresholds found in the finetuning experiment and by MIMETIC^2 are close, then the assumption about lack of confounders is valid (assuming there is no other confounder affecting all these concepts). We indeed find that the imitation thresholds found in the finetuning experiment are very close to the thresholds found by MIMETIC^2 without finetuning the model ($\pm 6\%$), attesting to the validity of this assumption (Section 5.5). Lastly, we test the assumption that each image of a concept contributes equally to the learning of a concept. We finetune a text-to-image model with a set of randomly sampled images of a concept multiple times. We report the variance in the imitation scores across these finetuning experiments and find the imitation scores variance to be small ($\pm 0.2\%$ of the imitation score), validating this assumption (see more details in Section 5.9.3).

To summarize, we empirically verify all the assumptions we make in this work, and find them to hold in practice.

5.9.1 Validity of the invariance assumption

Table 5.4: Average difference in the imitation scores for concepts whose image counts differ by less than 10. The difference in the imitation scores is close to 0, empirically validating the distribution invariance assumption.

Domain	Dataset	Avg. difference in imitation score	Relative to Avg. Score
Human Faces 🧑	Celebrities	0.0007	0.28%
Human Faces 🧑	Politicians	0.0023	0.7%
Art Style 🖼️	Classical Art Style	-0.0088	2.2%
Art Style 🖼️	Modern Art Style	-0.0013	0.26%

In this section, we empirically test the statistical validity of the assumptions we make in Section 5.2. The first assumption states that there is a distributional invariance across concepts. If this is true, then the imitation scores of two concepts (from the same domain) whose image counts are similar should also be similar to each other. To test whether this is empirically true for the domains we experiment with, we measure the difference in the imitation scores for concepts whose image counts differ by less than 10 images and report the difference averaged over all such pairs. Table 5.4 shows that the average difference in the imitation scores for all the pairs, for all the datasets we experiment with, is very close to 0. This provides empirical validation of the distribution invariance assumption.

5.9.2 Validity of the assumption about absence of confounders

To test the absence of a confounder, we conduct a finetuning experiment in which we finetuned a pre-trained text-to-image model (SD1.4) on images of concepts the model has not seen during pretraining. We do this by finding politicians who recently got popular, after the SD1.4 model was open-sourced. We finetune SD1.4 on an increasing number of images of these politicians, starting with 50 images (much below the imitation threshold found in the non-finetuning setup) and going up to 800 images (above the imitation threshold found in the non-finetuning setup). During the finetuning process, the images of the concepts were mixed with another 10,000 images taken randomly from the LAION-2B-en dataset. This was done in order to ensure that our finetuning setup closely resembles the original training setup of SD1.4, where images of a concept would be naturally interspersed with other images in the dataset. We finetune on the full dataset of 10,000 images for 1 epoch with a learning rate of $5e-5$. We use the following concepts in this experiment: `Averie Bishop`, `Sophie Wilde`, `Javier Milei`, `Hashim Safi al-Din`, `Shyam Rangeela`, `Akshata Murty`, `Hana-Rawhiti Maipi-Clarke`, `Sébastien Lecornu`, `Vivek Ramaswamy`, `Raghav Chaddha`.

For the SD1.4, the imitation threshold for the non-finetuning setup is found to be at 234 images (Figure 5.34) and the average imitation scores for the politicians on the right-hand side of the threshold are **0.26**. We report the results in Table 5.5, which shows that the imitation score remains lower than **0.26** when 50, 100, 150, or 200 images are used to finetune the model. The imitation score reaches this value of 0.26, only when finetuned with 250 or more images. This imitation threshold of 250 in the finetuning experiment is very

close to the imitation threshold MIMETIC^2 finds for SD1.4 without requiring to finetune the model (234 as shown in Figure 5.34).

Note that since in this experiment we are directly intervening on the image count of each concept, there is no effect of any confounder whatsoever, even if they were originally present. And the closeness of the imitation threshold in the finetuning and the non-finetuning setup alludes to the absence of confounders, thus validating our assumption. Note that this finetuning experiment is also the optimal experiment for finding the imitation threshold, which we described in Section 5.2. Based on these results, we can also claim that imitation thresholds MIMETIC^2 find are close to the ground-truth thresholds that the optimal experiment would find.

Table 5.5: The imitation scores of the concepts as a text-to-image model, SD1.4, is finetuned on an increasing number of images of a concept that it has not seen during pretraining. We report imitation scores averaged over 10 such concepts (politicians). We notice that the imitation scores have values 0.26 or greater when the model is finetuned on 250 or more images of a politician. Critically, this is the average imitation score for politicians on the right-hand side of the imitation threshold in Figure 5.34 (which is the imitation of politicians for SD1.4), indicating that 250 images or more are required for imitation in this experiment.

# Finetuning Images	Imitation Score After Finetuning (Avg. over 10 politicians)
50	0.07
100	0.16
150	0.19
200	0.21
250	0.26
300	0.27
400	0.29
500	0.31
600	0.32
700	0.36
800	0.34

5.9.3 Validity of the assumption about equal image contribution

We conduct another finetuning experiment to test the validity of our third assumption (which assumes that all images of a concept contribute equally) to its learning. In this experiment, we finetune a pre-trained SD1.4 model on images of concepts that the model has not seen during pre-training (these are the same concepts we used for the finetuning experiment in Section 5.9.2). Specifically, in this experiment, we sample 400

Table 5.6: Mean and std. deviation of the imitation scores when we finetuned a pre-trained SD1.4 model on 400 images of five concepts. We sample the images used for finetuning from a pool of images of these concepts. We repeat this process 20 times, and measure the mean and std. deviation in the imitation scores. The low std. deviation in the imitation scores indicates that most images contribute equally to the learning of the concept, thus validating our assumption.

Concept	Mean of Imitation Scores	Std. Deviation of Imitation Scores
Javier Milei	0.36	0.02
Hashim Safi al-Din	0.38	0.03
Shyam Rangeela	0.31	0.03
Akshata Murty	0.30	0.03
Hana-Rawhiti Maipi-Clarke	0.30	0.02

images of these politicians from a random pool of images 20 times, and finetune an SD1.4 model with each of the sampled sets of images. For each of the 20 finetuned models, we then measure the imitation scores for these concepts and report the mean and standard deviation in the scores. Table 5.6 shows these values, and the low standard deviation in the imitation scores indicates that most images contribute equally to the learning of the concept, thus validating our assumption for the models and the dataset we experiment with.

5.10 Caption Occurrence Assumption

For estimating the concept's counts in the pretraining dataset, we make a simplifying assumption: a concept can be present in the image only if it is mentioned in a paired caption. While this assumption isn't true in general, we show that for the domains and dataset we experiment with, it mostly holds in practice.

For this purpose, we download 100K random images from LAION2B-en, and run the face detection (used in Section 5.3) on all images, and count the faces of the ten most popular celebrities in our sampled set of celebrities. Out of the 100K random images, about 57K contain faces. For each celebrity, we compute the similarity between all the faces in the downloaded images and the faces in the reference images of these celebrities. If the similarity is above the threshold of 0.46, we consider that face to belong to the celebrity (this threshold is determined in Section 5.12 to distinguish if two images are of the same person or not). Table 5.7 shows the number of faces we found for each celebrity in the 100K random LAION images. We also show the face counts among these images whose captions mention the celebrity. We find that 1) the highest frequency an individual appears in an image without their name mentioned in the caption is 51

Table 5.7: We estimate the percentage of the images that our approach misses due to the assumption that a concept only occurs if it is mentioned in the caption. The small percentage of the images we miss (rightmost column) shows that our assumption of counting the images where a concept is mentioned in the caption is empirically reasonable.

Celebrity	Face Count in 100K images	Face Count in Images with Caption Mention	Missed Images (%)
Floyd Mayweather	1	0	0.001%
Oprah Winfrey	2	0	0.002%
Ronald Reagan	6	3	0.003%
Ben Affleck	0	0	0.0%
Anne Hathaway	0	0	0.0%
Stephen King	0	0	0.0%
Johnny Depp	9	1	0.008%
Abraham Lincoln	52	1	0.051%
Kate Middleton	34	1	0.033%
Donald Trump	16	0	0.016%

(*Abraham Lincoln* is mentioned once in the caption and he appears a total of 52 times), and 2) the highest percentage of image frequency that we miss is 0.051%, and 3) most of the other miss rates are much smaller (close to 0). Such low miss rates demonstrate that our assumption of counting images when a concept is mentioned in the caption is empirically reasonable.

We also note that this assumption would fail if we were computing image frequencies for concepts that are so widely common that one would not even mention them in a caption, for example, phone, shoes, or trees.

5.11 Collection of Reference Images

5.11.1 Collection of Reference Images for Human Faces Domain

The goal of collecting reference images is to use them to filter the images of the pretraining dataset. These images are treated as the gold standard reference images of a person, and images collected from the pretraining dataset are compared to these images. If the similarity is higher than a threshold, then that image is considered to belong to that person (see Section 5.3 for details). We describe an automatic manner of collecting the reference images. The high level idea is to collect the images from Google Search and automatically select a subset of those images that are of the same concept (same person's face or same artist's art). Since this is a crucial part of the overall algorithm, we manually vet the reference images for all the

concepts to ensure that they all contain the same concept.

Collection of Reference Images for Human Face Imitation: We collect reference images for celebrities and politicians using a three step process (also shown in Algorithm 3):

1. **Candidate set:** First, we retrieved the first hundred images by searching for a person's name on Google Images. We used SerpAPI [211] as a wrapper perform the searches.
2. **Selecting from the candidate set:** Images retrieved from the internet are noisy and might not contain the person we are looking for. Therefore, we filter images that contain the person from the candidate set of images. For this purpose, we use a face recognition model. We embed all the faces in the retrieved images using a face embedding model and measure the cosine similarity between each one of them. The goal is to search for a set of faces that belong to the same person and therefore will have a high cosine similarity to each other.

   One strategy is for the faces to form a graph where the vertices are the face embeddings and the edges connecting two embeddings have a weight equal to the cosine similarity between them, and we select a dense k-subgraph [134] from this graph. Selecting such a subgraph means finding a mutually homogeneous subset. We can find the vertices of this dense k-subgraph by cardinality-constrained submodular function minimization [16, 166] on a facility location function [16]. We run this minimization and select a subset of images (at least of size ten) that has the highest average cosine similarity between each pair of images.
3. **Manual verification:** Selecting the faces with the highest average similarity is not enough. This is because in many cases the largest set of faces in the candidate set is not of the person we look for, but for someone closely associated with them, in which case, the selected images are of the other person. For example, all the selected faces for *Miguel Bezos* were actually of *Jeff Bezos*. Therefore, we manually verify all the selected faces for each person. In the situation where the selected faces are wrong, we manually collect the images for them, for example, for Miguel Bezos. We collect at least 5 reference images for all celebrities.

Collection of Reference Images for Art Styles We collect reference images for each artist (each artist is assumed to have a distinct art style) from Wikiart, the online encyclopedia for artworks. Since the artworks of each artist were meticulously collected and vetted by the artist community, we consider all the images

ALGORITHM 3: Collection of Reference Images for Human Face Imitation

Input: Person's name P
Output: Verified Set of Images of P
$images \leftarrow$ SerpAPI(P) ; ▷ Retrieve initial image set using SerpAPI
$candidateSet \leftarrow$ Submodular_Minimization($images$) ; ▷ Select candidate set using submodular minimization
$verifiedSet \leftarrow$ manualVerification($candidateSet$) ; ▷ Manually verify the candidate set

collected from Wikiart as the reference art images for that artist.

5.12 Implementation Details of MIMETIC^2 for Human Face Imitation

5.12.1 Filtering of Training Images

Images whose captions mention the concept of interest often do not contain it (as shown with Mary-Lee Pfeiffer in Figure 5.3). As such, we filter images where the concept does not appear in the image, which we detect using a dedicated classifier. In what follows, we describe the filtering mechanism.

Collecting Reference Images:

We collect reference images for each person using SerpAPI as described in Section 5.11. These images are the gold standard images that we manually vet to ensure that they contain the target person of interest (see Section 5.11 for the details). We use the reference images to filter out the images in the pretraining dataset that are not of this person. Concretely, for each person, we use a face embedding model [57] to measure the similarity between the faces in the reference images and the faces in the images from the pretraining datasets whose captions mention this person. If the similarity of a face in the pretraining images to any of the faces in the reference images is above a certain threshold, that face is considered to belong to the person of interest. We determine this threshold to distinguish faces of the same person from faces of different persons in the next paragraph. Note that this procedure already filter outs any image that does not contain a face, because the face embedding model would only embed an image if it detects a face in that image.

Determining Filtering Threshold: The next step is to determine the threshold for which we consider two faces to belong to the same person. For this purpose, we measure the similarity between pairs of faces of the same person and the similarity between pairs of faces of different persons. Since the reference images for each person are manually vetted to be correct, we use these images for this procedure. We plot the histogram

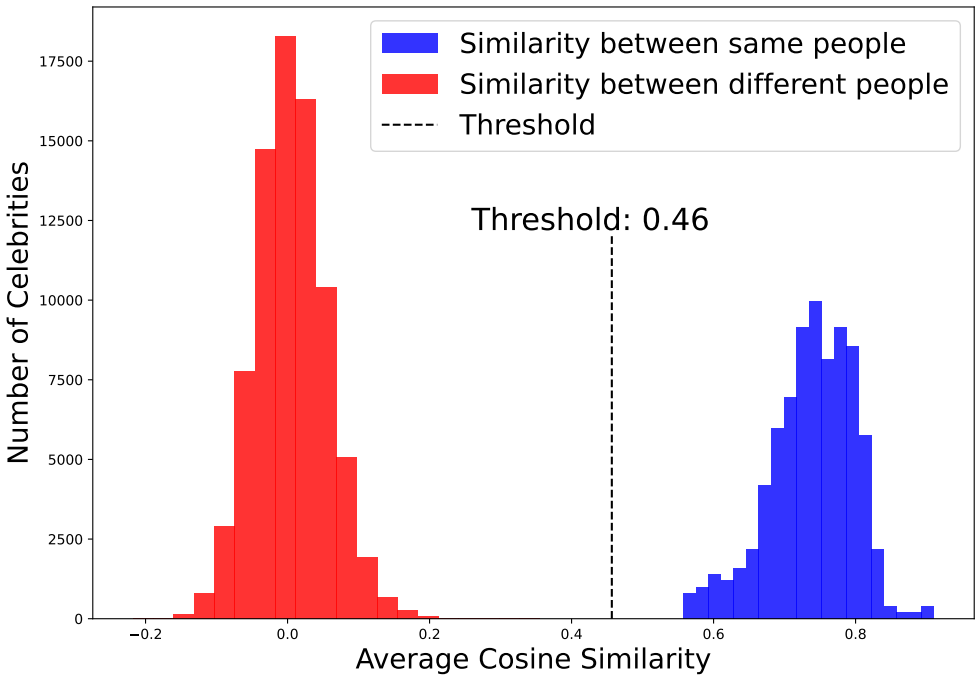


Figure 5.7: Average cosine similarity between the faces of the same people (blue colored) and of the faces of different people (red colored), measured across the reference images of the celebrities.

of the average similarity between the faces of the same person (blue colored) and the similarity between faces of different persons (red colored) in Figure 5.7. We see that the two histograms are well separated, with the lowest similarity value between the faces of the same person being 0.56 and the highest similarity value between the faces of different persons being 0.36. Therefore, any threshold value between 0.36 and 0.56 can separate two faces of the same person from the faces of different people. In our experiments, we use the midpoint threshold of 0.46 (true positive rate (tpr) of 100%; false positive rate (fpr) of 0%) to filter any face in the pretraining images that does not belong to the person of interest. The filtering process gives us both the image frequency of a person in the pretraining data and the pretraining images that we compare the faces in the generated images to measure the imitation score.

5.12.2 Measurement of Imitation Score

To measure the imitation between the training and generated images of a person, we compute the cosine similarity between the face embeddings of the faces in their generated images and their filtered training images from the previous step. However, measuring the similarity using all the pretraining images can

(a) Training images of *Samuel L. Jackson* that show significant variations in his face (age, hair, and beard).

(b) Generated images of *Samuel L. Jackson* that show the model has captured a specific characteristic of his face (middle-aged, bald, with no or little beard).

Figure 5.8: Real and generated images of *Samuel L. Jackson*.

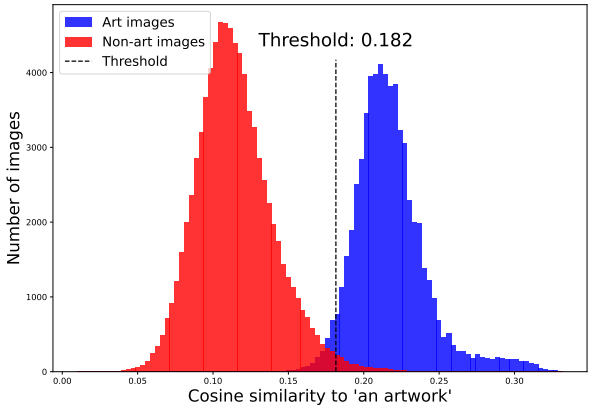
underestimate the actual imitation. This is because several individuals have significant variations in their faces in the pretraining images, and the text-to-image model does not capture all these variations. For example, consider the pretraining images of *Samuel L. Jackson* in Figure 5.8a. These images have significant variations in beard, hair, and age. However, when the text-to-image model is prompted to generate images of *Samuel L. Jackson*, the generated images in Figure 5.8b only show a specific facial characteristic of him (middle-aged, bald, with no or little beard). Since MIMETIC^2's goal is not to measure if a text-to-image model captures all the variations of a person, we want to reward the model even if it has only captured a particular characteristic (which it has in this case of *Samuel L. Jackson*). Therefore, instead of comparing the similarity of generated images to all the training images, we compare the similarity to only the ten training images that have the highest cosine similarity to the generated images on average.

5.13 Implementation Details of MIMETIC^2 for Art Style Imitation

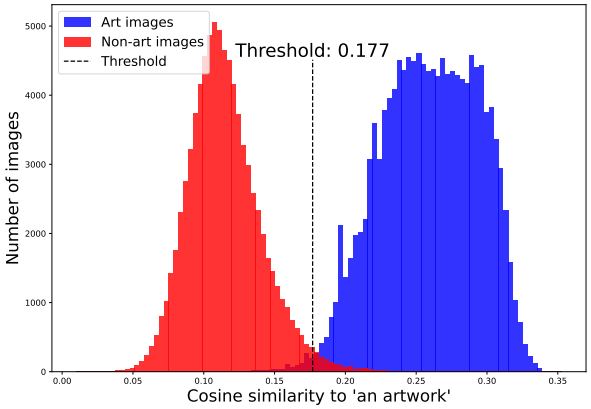
5.13.1 Filtering of Training Images

For art style imitation, we consider each artist to have a unique style. We collect the images from the pretraining dataset whose captions mention the name of the artist whose art style imitation we want to measure. Similar to the case of human face imitation, we want to filter out the pretraining images of an artist that, in reality, were not created by that artist, but their captions mention them. We implement the

filtering process in two stages. In the first stage, we filter out non-art images in the pretraining dataset (note that the captions of these images still mention the artist, but the images themselves are not art works) and in the second stage we filter out art works of other artists (the captions of these images mention the artist of interest and the image itself is also an art work, but by a different artist). The implementation details for each stage are as follows:



(a) Histogram of the cosine similarity of embeddings of art and non-art images to embeddings of 'an artwork' for classical artists.



(b) Histogram of the cosine similarity of embeddings of art and non-art images to embeddings of 'an artwork' for modern artists.

Figure 5.9: The first filtering step involves determining the threshold to distinguish between art and non-art images from the pretraining images, for which we compare the similarity of the image's embedding to the embedding of the text "an artwork".

Filtering Non-Art Images: To filter non-art images from the pretraining dataset, we use a classifier that separates art images from non-art images. Concretely, we embed the pretraining images using a CLIP ViT-H/14 [111] image encoder and measure the cosine similarity of the image embeddings and the text embeddings of the string '*an artwork*', embedded using the text encoder of the same model. Only when the similarity between the embeddings is higher than a threshold described below do we consider those pretraining images as an artwork. To determine this threshold, we choose a similarity score that separates art images from non-art images. We use the images from the Wikiarts dataset [200] as the (positive) art images and MS COCO dataset images [149] as the (negative) non-art images. Note that the MS COCO dataset was collected by photographing everyday objects that art was not part of, making it a valid set of negative examples of art.

We plot the histogram of cosine similarity of the embeddings of art and non-art images to the text embedding of '*an artwork*' (see Figures 5.9a and 5.9b. We observe that the art and non-art images from both the artist

(a) Images from the MS COCO dataset that were classified as art by the threshold we chose. These images clearly have paintings in them and therefore, are classified in that category. These images were selected in MS COCO for different categories like scissors, chair, parking meter, and vase.

(b) Images from the Wikiarts dataset that were classified as non-art by the threshold we chose.

Figure 5.10: Images that are misclassified by our art vs. non-art threshold in Figure 5.9a.

(a) Images from the MS COCO dataset that were correctly classified as non-art.

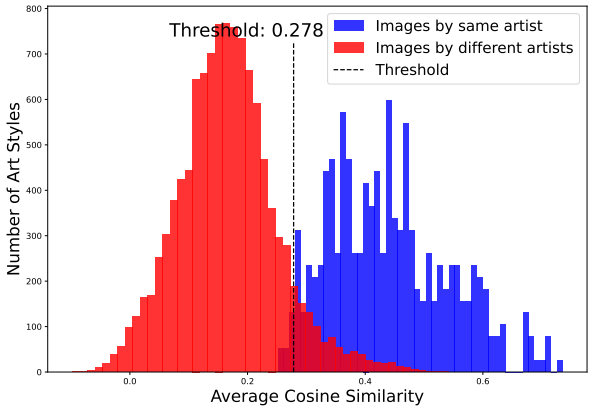
(b) Images from the Wikiarts dataset that were correctly classified as art.

Figure 5.11: Images that are correctly classified by our art vs. non-art threshold in Figure 5.9a.

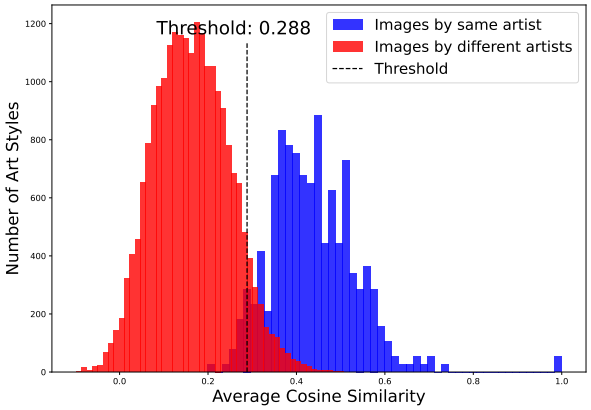
groups are well separated (although not perfect, Figure 5.10 and Figure 5.11 show examples of misclassified and correctly classified images from both datasets). We choose the threshold that maximizes the F1 score of the separation (0.182 for the classical artists and 0.177 for the modern artists).

Filtering Images of Other Art Styles: Similar to the case of human faces, not all art images whose captions mention an artist were created by that artist. We want to filter out such images. For this purpose, we collect reference images for each artist (see Section 5.11 for details) and use them to classify the training images that belong to the artist of interest. Concretely, we measure the similarity between the pretraining images and the reference images of each artist, and only retain images whose similarity to the reference images is higher than a threshold.

To determine this threshold, we measure the similarity between pairs of art images of the same artists and pairs of art images from different artists. We embed the images using an art style embedding model [225]



(a) Histogram of the average cosine similarity between embeddings of the images of the same artist (blue) and the art of different artists (red) for classical artists



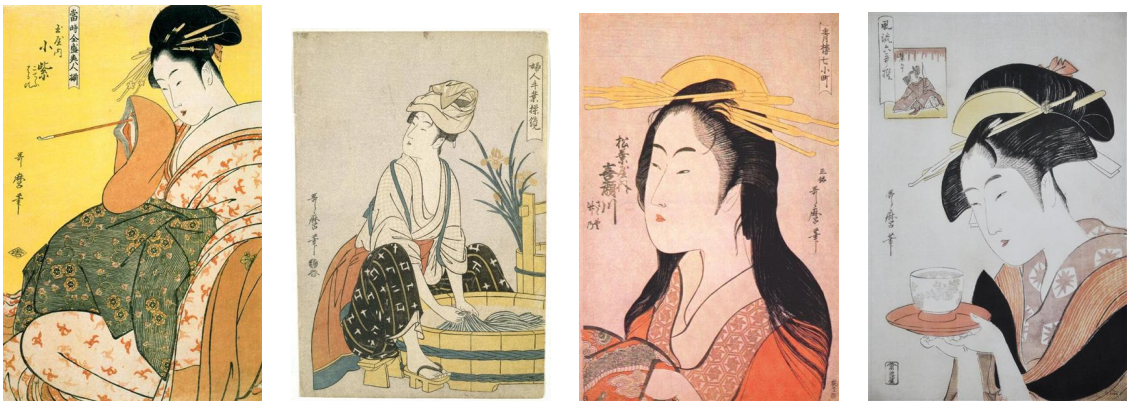
(b) Histogram of the average cosine similarity between embeddings of the images of the same artist (blue) and the art of different artists (red) for modern artists

Figure 5.12: The second filtering step involves determining if an artwork whose caption mentions an artist actually belongs to that artist or not.

and plot the histogram of similarities between art images of the same artist (blue colored) and art images of different artists (red colored) in Figure 5.12a for classical artists and Figure 5.12b for modern artists. We see that the two histograms are well separated (although not perfect, Figure 5.13 shows paintings by two artists whose art style is very similar and cannot be distinguished by our threshold). We choose the threshold that maximizes the F1 score of the separation between these two groups (0.278 for classical artists and 0.288 for modern artists). The retained images give us both the image counts of each artist and the training images that we compare to the generated images to measure the imitation score.

5.13.2 Similarity Measurement

We embed all the generated images and the filtered pretraining images using the art style embedding model [225] and measure the cosine similarity between each pair of generated and pretraining images. Similar to the case of the human faces, we do not want to underestimate the art style similarity between the generated and training images by comparing the generated images to all the training images of this artist. Therefore, we measure the similarity of generated images to the ten training images that are, on average, the most similar to the generated images.



(a) Paintings made by Kitagawa Utamaro.

(b) Paintings made by Tsukioka Yoshitoshi

Figure 5.13: Paintings made by Kitagawa Utamaro and Tsukioka Yoshitoshi are very similar, and our threshold is unable to distinguish between their styles.

5.14 Evaluation of the Imitation Thresholds

Evaluating the accuracy of the imitation thresholds MIMETIC^2 finds is a very expensive task; it requires conducting the optimal experiment described in Section 5.2 (training $\mathcal{O}(\log m)$ models with different numbers of images of each concept). In order to make the evaluation tractable, we finetune a pretrained text-to-image model with concepts that the model has not seen during pretraining and find the number of images required for the model to be able to imitate them. We find that all the concepts require more than 250 images for imitation when the models are finetuned on them, which is very close to the imitation threshold MIMETIC^2 finds for this pretrained model without finetuning its (234 images). Based on these results, we conclude that the imitation thresholds estimated by MIMETIC^2 are accurate.

We do this by searching for politicians who became popular after the LAION-2B-en dataset was released (after 2022) (§5.9.2) and collecting their images from Google Images. We finetune SD1.4 on images of these politicians, starting with 50 images of a politician (much below the imitation threshold found by MIMETIC^2) and going up to 800 images (above the estimated imitation threshold). We finetune the model separately for each politician. To mimic the original training setup of the SD models (where images of a concept are naturally interspersed with other images in the dataset), we mix the politician images with other images taken randomly from the LAION-2B-en dataset such that for each finetuning experiment the total number of images added upto 10,000 (9,950 random images mixed with 50 images of the politician, 9,900 random images mixed with 100 images of the politician, and so on for each finetuning experiment). We finetune on this new dataset for a single epoch with a learning rate of $5e-5$.

We want to find the number of images required for imitation to occur in this setup. Imitation can be established either with a human study or an automatic metric that correlates with human perception of imitation. Since we already conducted a human study and found that our imitation threshold correlates with human perception, we argue that the automatic imitation score for concepts that are on the right-hand side of the imitation threshold would also correlate with the human perception of imitation. MIMETIC^2 estimated that the imitation score for concepts on the right-hand side of the imitation threshold for face imitation of politicians for SD1.4 is **0.26** (Figure 5.34). We deem the number of images required to reach this imitation score as the number of images required for imitation for this finetuning experiment.

We report the results in Table 5.8.

It shows that the imitation score remains lower than **0.26** when fewer than 200 images are used to finetune the SD model, and the score reaches 0.26 only when finetuned with 250 or more images of a politician. Thus, the number of images required to reach the imitation score of 0.26 in the finetuning experiment (250 images) is very close to the imitation threshold MIMETIC^2 finds for SD1.4 without requiring finetuning the model (234 images). Based on these results, we conclude that the imitation thresholds estimated by MIMETIC^2 are accurate.

Table 5.8: The imitation scores of the concepts as SD1.4 is finetuned on an increasing number of images of a concept. We report imitation scores averaged over 10 such politicians. We note that the imitation scores have values 0.26 or greater when the model is finetuned on 250 or more images of a politician. Critically, this is the average imitation score for politicians on the right-hand side of the imitation threshold in Figure 5.34, indicating that 250 images or more are required for imitation in this experiment.

# Finetuning Images	50	100	150	200	250	300	400	500	600	700	800
Avg. Imitation Score	0.07	0.16	0.19	0.21	0.26	0.27	0.29	0.31	0.32	0.36	0.34

5.15 Change Points

Table 5.9 we show all the change points that PELT found for each experiment (Table 5.3 reports the first change point as the imitation threshold).

5.16 All Results: The *Imitation Threshold*

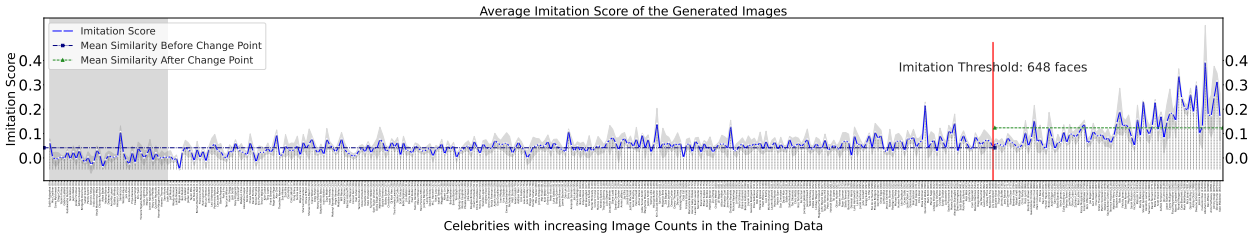


Figure 5.14: Human Face Imitation (Celebrities): Similarity between the training and generated images for all celebrities. The celebrities with zero image counts are shaded with light gray. We show the mean and variance over the five generation prompts. The images were generated using **LDM**. The change point for human face imitation for celebrities when generating images using LDM is detected at **648 faces**.

In this section, we estimate the imitation threshold for human face and art style imitation for three different

Table 5.9: *Imitation Thresholds* for human face and art style imitation for the different text-to-image models and datasets we experiment with. This table shows all the change points that PELT found for each experiment (Table 5.3 reports the first change point as the imitation threshold).

		Human Faces 🧑		Art Style 🖼️	
Pretraining Dataset	**Model**	**Celebrities**	**Politicians**	**Classical Artists**	**Modern Artists**
LAION2B-en	SD1.1	364	234	112, 391	198
	SD1.5	364, 8571	234, 4688	112, 360	198, 4821
LAION-5B	SD2.1	527, 9650	369, 8666	185, 848	241, 1132

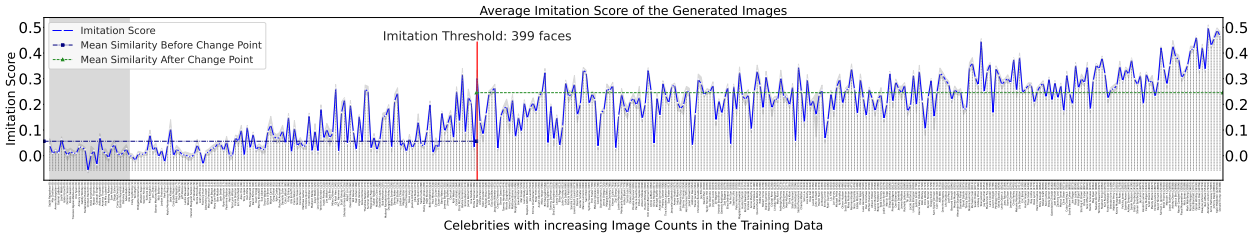


Figure 5.15: Human Face Imitation (Celebrities): Similarity between the training and generated images for all celebrities. The celebrities with zero image counts are shaded with light gray. We show the mean and variance over the five generation prompts. The images were generated using **SD1.1**. The change point for human face imitation for celebrities when generating images using SD1.1 is detected at **364 faces**.

text-to-image models. Figure 5.14, Figure 5.15, Figure 5.16, and Figure 5.17 show the image counts of celebrities on the x-axis (sorted in increasing order of image counts) and the imitation score of their generated images (averaged over the five image generation prompts) on the y-axis. The images were generated using LDM, SD1.1, SD1.5, and SD2.1, respectively. Similarly, Figure 5.18, Figure 5.19, Figure 5.20, and Figure 5.21 show the image counts of the politicians and the imitation score of their generated images, for LDM, SD1.1, SD1.5, and SD2.1, respectively.

Figure 5.22, Figure 5.23, Figure 5.24, Figure 5.25 show the image counts of classical artists and the similarity between their training and generated images; and Figure 5.26, Figure 5.27, Figure 5.28, Figure 5.29 show the image counts of modern artists and the similarity between their training and generated images. The images were generated using LDM, SD1.1, SD1.5, and SD2.1, respectively.

Imitation Threshold Estimation for Human Face Imitation: In Figure 5.15, we observe that the imitation scores for the individuals with small image counts is close to 0 (left side), and it increases as the number of their image counts increases towards the right. The highest similarity is 0.5, and it is for the individuals in the rightmost region of the plot. The solid line in the plot shows the mean similarity over the five image

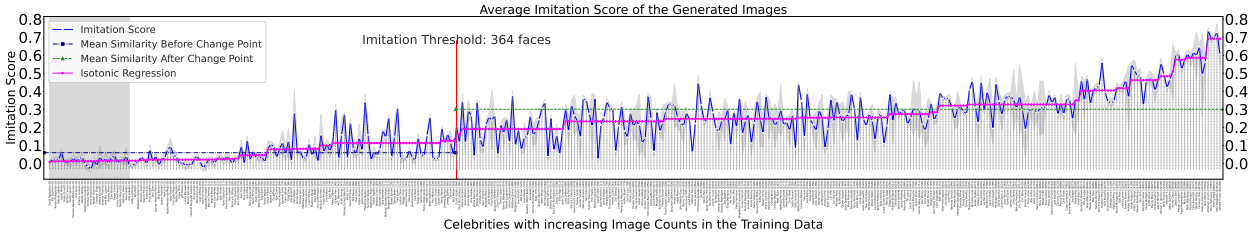


Figure 5.16: Human Face Imitation (Celebrities): similarity between the training and generated images for all celebrities. We show the mean and variance over the five generation prompts. The images were generated using **SD1.5**. The change point for human face imitation for celebrities when generating images using SD1.5 is detected at **364 faces**.

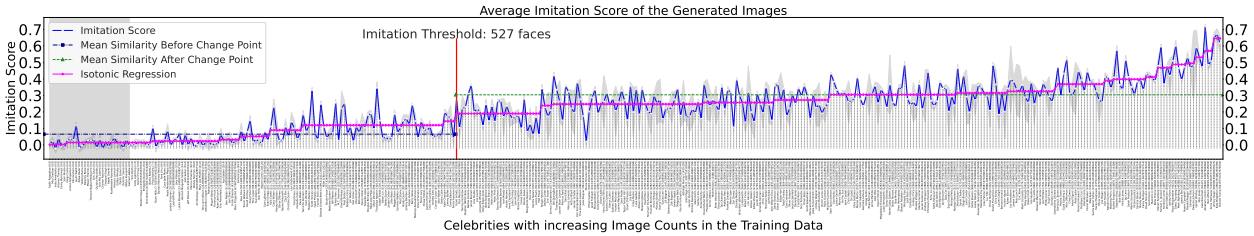


Figure 5.17: Human Face Imitation (Celebrities): similarity between the training and generated images for all celebrities. We show the mean and variance over the five generation prompts. The images were generated using **SD2.1**. The change point for human face imitation for celebrities when generating images using SD2.1 is detected at **527 faces**.

generation prompts, with the shaded area showing the variance over them. We observe a low variance in the imitation score among the generation prompts. And we also observe that the variance does not depend on the image counts, which indicates that the performance of the face recognition model does not depend on the popularity of the individual. The change detection algorithm finds the change point to be at **364** faces for human face imitation for celebrities, when using **SD1.1** for image generation. Figure 5.16 shows the similarity between the training and generated images when images are generated using **SD1.5**. Identically to SD1.1, the change is detected at **364** faces for face imitation when using SD1.5. We also performed ablation experiments with different face embedding models and justify the choice of our model (see Section 5.17). For all the plots, we also analyze the trend by using isotonic regression, which learns non-decreasing linear regression weights that fit the data best.

Imitation Threshold Estimation for Human Face Imitation (Politicians): Figure 5.19 shows the imitation scores for politicians, which are very similar to the plot obtained for celebrities. We observe a low variance in the imitation score among the generation prompts. We also observe that the variance does not depend on the image counts, which indicates that the performance of the face recognition model does not

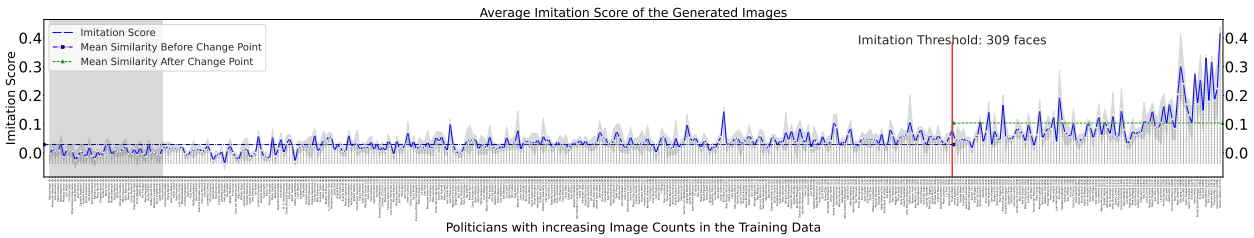


Figure 5.18: Human Face Imitation (Politicians): Similarity between the training and generated images for all politicians. The politicians with zero image counts are shaded with light gray. We show the mean and variance over the five generation prompts. The images were generated using **LDM**. The change point for human face imitation for politicians when generating images using LDM is detected at **309 faces**.

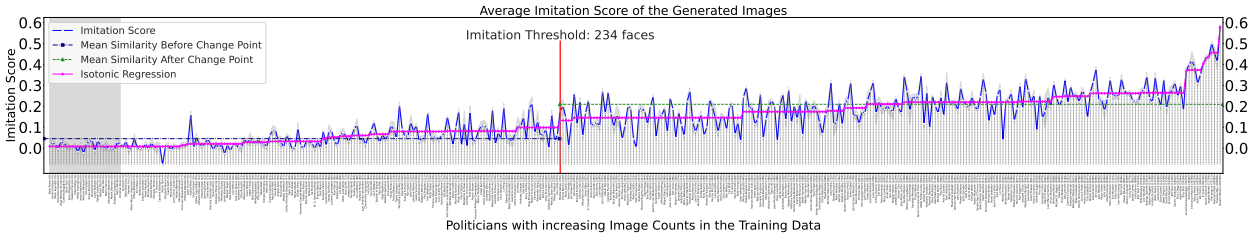


Figure 5.19: Human Face Imitation (Politicians): Similarity between the training and generated images for all politicians. The politicians with zero image counts are shaded with light gray. We show the mean and variance over the five generation prompts. The images were generated using **SD1.1**. The change point for human face imitation for politicians when generating images using SD1.1 is detected at **234 faces**.

depend on the popularity of the individual. The change detection algorithm finds the change point to be at **234** faces for human face imitation for politicians, when using **SD1.1** for image generation. Figure 5.20 shows the similarity between the training and generated images when images are generated using **SD1.5**. Similar to SD1.1, the change is detected at **234** faces.

Imitation Threshold Estimation for Art Style Imitation: In Figure 5.23, we observe that the imitation scores for artists with low image counts have a baseline value around 0.2 (left side), and it increases as the number of their image counts increases towards the right. The highest similarity is 0.76, and it is for the artists in the rightmost region of the plot. We also observe a low variance across the generation prompts, and the variance does not depend on the image frequency of the artist. The change detection algorithm finds the change point to be at **112** images for art style imitation of classical artists, when using **SD1.1** for image generation. Figure 5.24 shows the similarity between the training and generated images when images are generated using **SD1.5**. Similar to SD1.1, the change is detected at **112** faces for art style imitation when using SD1.5. These thresholds are slightly higher for style imitation of modern artists, 198 for both SD1.1 and SD1.5.

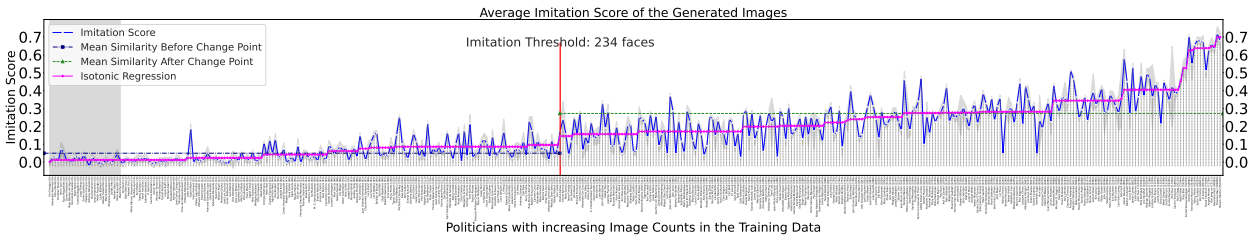


Figure 5.20: Human Face Imitation (Politicians): similarity between the training and generated images for all politicians. We show the mean and variance over the five generation prompts. The images were generated using **SD1.5**. The change point for human face imitation for politicians when generating images using SD1.5 is detected at **234 faces**.

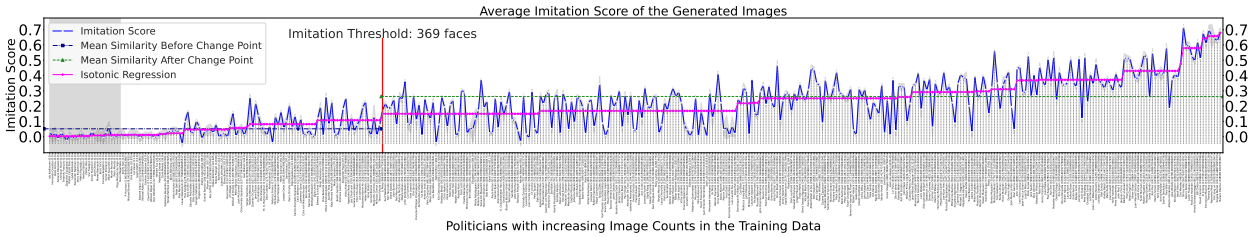


Figure 5.21: Human Face Imitation (Politicians): similarity between the training and generated images for all politicians. We show the mean and variance over the five generation prompts. The images were generated using **SD2.1**. The change point for human face imitation for celebrities when generating images using SD2.1 is detected at **369 faces**.

5.17 Ablation Experiment with Different Face Embedding Models

In this section, we show the difference in the performance of several face embedding models and justify the choice of the final choice of our face embedding model. Face embedding models are evaluated using two main metrics: false-match rate (FMR) and true-match rate (TMR) [168]. FMR measures how many times a model says two people are the same when they are not, and TMR measures how many times a model says two people are the same when they are the same. Ideally, a face embedding model should have low FMR and high TMR. An important variant of these metrics is the disparity of FMR and TMR of a model across different demographic groups. Ideally, a model should have low disparity in these metrics across different demographics. We also focus on the variance of these metrics across demographics in making the final choice.

We evaluate the FMR and TMR of eight different face embedding models (seven open-sourced and one proprietary). The open-source models were chosen based on their popularity on GitHub [210, 58, 57], and we also experiment with Amazon Rekognition, a proprietary model. For evaluating the disparity of these

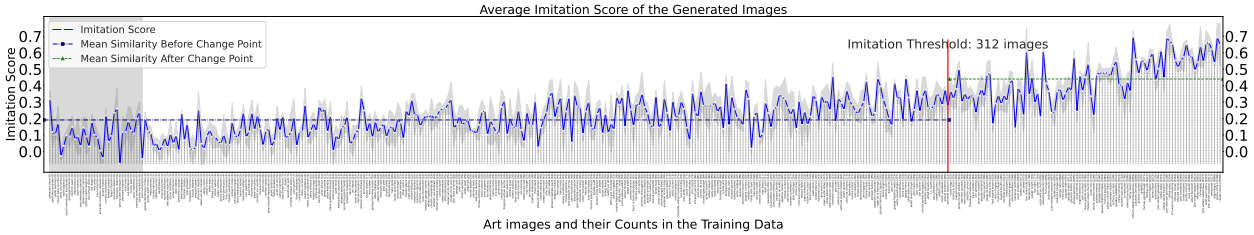


Figure 5.22: Art Style Imitation (Classical Artists): similarity between the training and generated images for **classical** art styles. We show the mean and variance over the five generation prompts. The images were generated using **LDM**. The change point for art style imitation when generating images using LDM is detected at **312 images**.

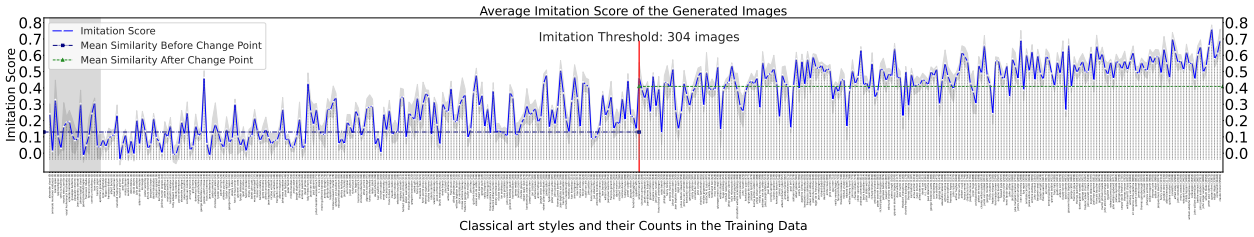


Figure 5.23: Art Style Imitation (Classical Artists): similarity between the training and generated images for **classical** art styles. We show the mean and variance over the five generation prompts. The images were generated using **SD1.1**. The change point for art style imitation when generating images using SD1.1 is detected at **112 images**.

metrics across different demographic groups, we grouped celebrities in six demographic groups primarily categorized according to skin color tone (black, brown, and white) and perceived gender (male and female; for simplicity). Each of the six groups had 10 celebrities (a total of 60), with no intersection between them. The categorization was done manually by looking at the reference images of the celebrities. For each celebrity, we collect 10 reference images from the internet by using the procedure described in Section 5.11. We use these images to compare the FMR and TMR of the face recognition models, as these images are the gold standard images of a person.

FMR Computation: We compute the mean cosine similarity between the face embeddings of one individual and the faces of all other individuals in that group, and repeat the procedure for all individuals in a demographic group.

TMR Computation: We compute the mean cosine similarity between the embeddings of all the faces of an individual and repeat the procedure for all the individuals in a demographic group.

Figure 5.30 and Figure 5.31 show the FMR and TMR for six demographic groups for all the face embedding models. All the open-sourced models, except InsightFace, either have a high disparity in FMR

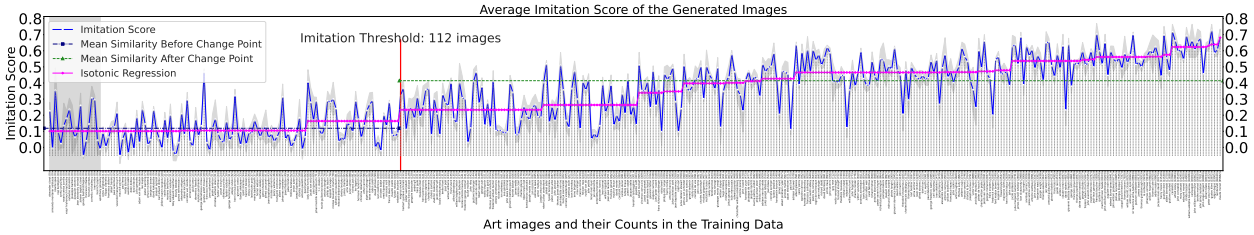


Figure 5.24: Art Style Imitation (Classical Artists): similarity between the training and generated images for **classical** art styles. We show the mean and variance over the five generation prompts. The images were generated using **SD1.5**. The change point for art style imitation when generating images using SD1.5 is detected at **112 images**.

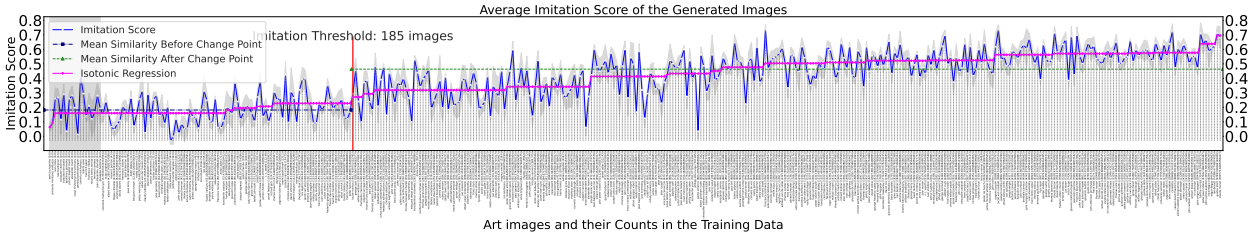


Figure 5.25: Art Style Imitation (Classical Artists): similarity between the training and generated images for **classical** art styles. We show the mean and variance over the five generation prompts. The images were generated using **SD2.1**. The change point for art style imitation when generating images using SD2.1 is detected at **185 images**.

values across the demographic groups (ArcFace, Facenet, Facenet512, DeepFace) or have very low TMR (GhostFaceNet_W1, GhostFaceNet_V1). We chose InsightFace for our experiments because it has 1) a low overall FMR, 2) decent TMR, 3) a low disparity of FMR and TMR across the demographic groups, and 4) is open-sourced. Having a low disparity of the metrics across individuals of different demographic groups is crucial for an accurate estimation of the imitation threshold. The Amazon Rekognition model would also be a viable choice based on these metrics; however, it is not open-sourced and therefore expensive for our experiments.

5.18 Compute Used

5.18.1 Estimated Computational Cost of the Optimal Experiment

The computational cost to train the popular text-to-image model Stable Diffusion was $600K [12]. For the optimal experiment, we would need to train $\mathcal{O}(\log m)$ models, where m is the total number of available images of a concept Z^j. So if a concept has 100,000 images, then we need to train $\log_2(100,000) = 16$

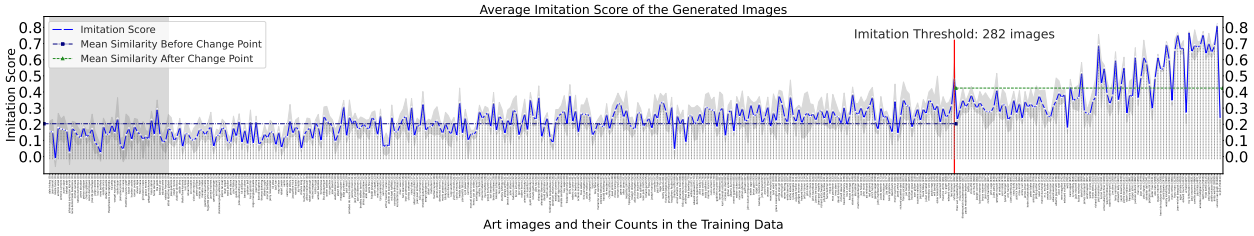


Figure 5.26: Art Style Imitation (Modern Artists): similarity between the training and generated images for **modern** art styles. We show the mean and variance over the five generation prompts. The images were generated using **LDM**. The change point for art style imitation when generating images using SD0 is detected at **282 images**.

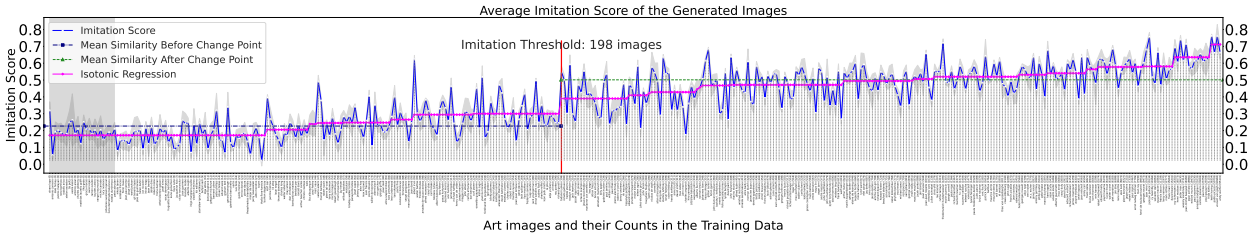


Figure 5.27: Art Style Imitation (Modern Artists): similarity between the training and generated images for **modern** art styles. We show the mean and variance over the five generation prompts. The images were generated using **SD1.1**. The change point for art style imitation when generating images using SD1.1 is detected at **198 images**.

models to optimally estimate the imitation threshold, which would cost about $10M.

5.18.1.1 MIMETIC^2's Computational Cost

We use 8 L40 GPUs to generate images for all text-to-image models in our work. Overall, we use them for 16 hours per prompt, per dataset, per model to generate images. We downloaded the images on the same machine using 40 CPU cores, a process that took about 8 hours per dataset. For generating the image embeddings, we use the same 8 L40 GPUs, a process that took about 16 hours per dataset. The computation of the imitation score and plotting are done on a single CPU core on the same machine, a process that takes less than 30 minutes per dataset.

5.19 Imitation Thresholds of SD models in Series 1 and 2

Our experimental results in Section 5.5 found that for most domains, the imitation thresholds for SD1.1 and SD1.5 are almost the same, while being higher for SD2.1. We hypothesized that the difference is due to their

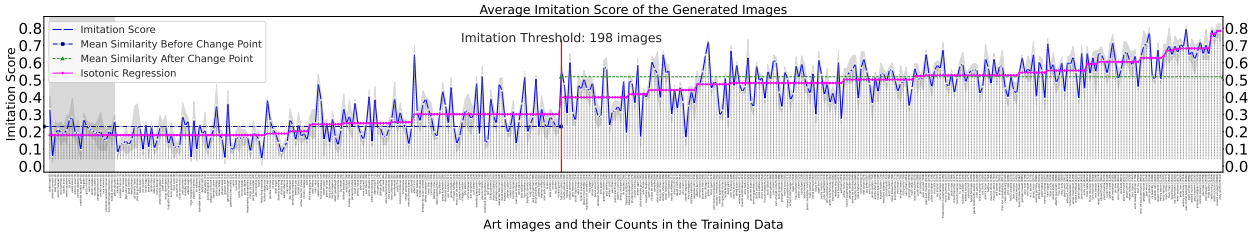


Figure 5.28: Art Style Imitation (Modern Artists): similarity between the training and generated images for **modern** art styles. We show the mean and variance over the five generation prompts. The images were generated using **SD1.5**. The change point for art style imitation when generating images using SD1.5 is detected at **198 images**.

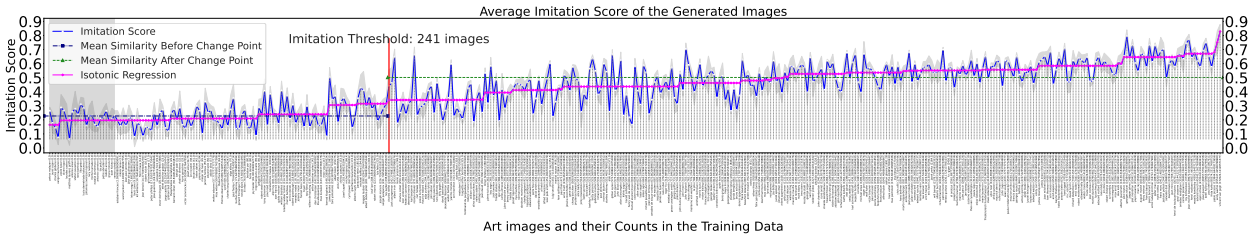


Figure 5.29: Art Style Imitation (Modern Artists): similarity between the training and generated images for **modern** art styles. We show the mean and variance over the five generation prompts. The images were generated using **SD2.1**. The change point for art style imitation when generating images using SD2.1 is detected at **241 images**.

different text encoders. All models in the SD1 series use the same text encoder from CLIP, whereas SD2.1 uses the text encoder from OpenCLIP. To test the validity of this hypothesis, we repeated the experiments for all models in the SD1 series for politicians and computed their imitation thresholds. Table 5.10 shows the thresholds for the politicians. We find that the imitation thresholds for all the models in the SD1 series are almost the same, and are lower than the threshold for the SD2.1 model. This evidence supports our hypothesis that the difference in the text-encoders being the main reason for the difference in the imitation thresholds.

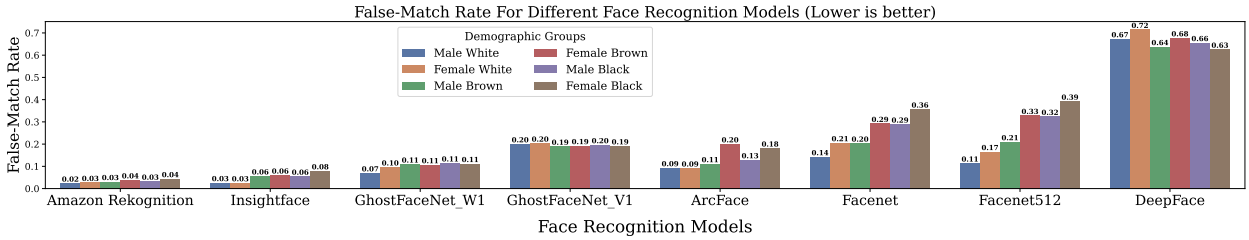


Figure 5.30: False-match rate (FMR) of all the face embedding models across the six demographic groups. Amazon Rekognition and InsightFace have the lowest FMR values. Moreover, these two models have the lowest disparity of FMR over the demographic groups.

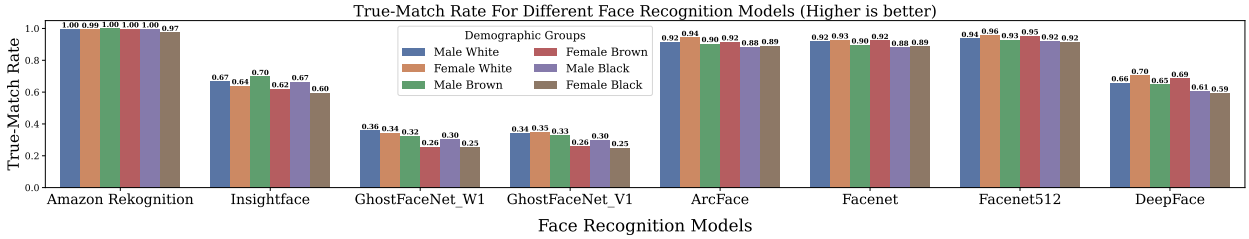


Figure 5.31: True-match rate (TMR) of all the face embedding models across the six demographic groups. The Amazon Rekognition model has the highest TMR values.

Table 5.10: *Imitation Thresholds* for politicians for all models in SD1 series and SD2.1

Pretraining Dataset	Model	Human Faces 🧑: Politicians
LAION2B-en	SD1.1	234
	SD1.2	252
	SD1.3	234
	SD1.4	234
	SD1.5	234
LAION-5B	SD2.1	369

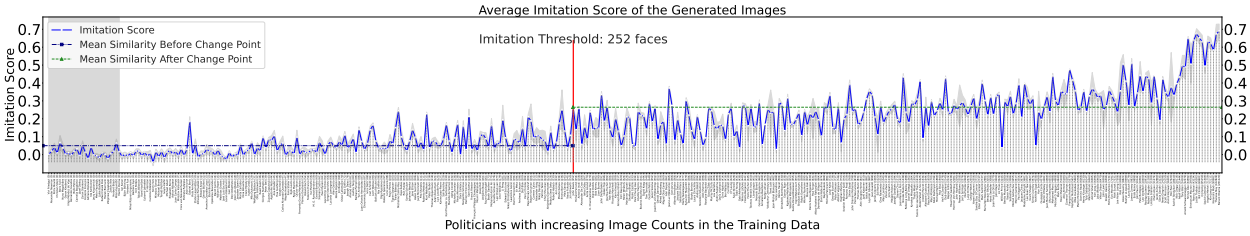


Figure 5.32: Human Face Imitation (Politicians): Similarity between the training and generated images for all politicians. The politicians with zero image counts are shaded with light gray. We show the mean and variance over the five generation prompts. The images were generated using **SD1.2**. The change point for human face imitation for politicians when generating images using SD1.1 is detected at **252 faces**.

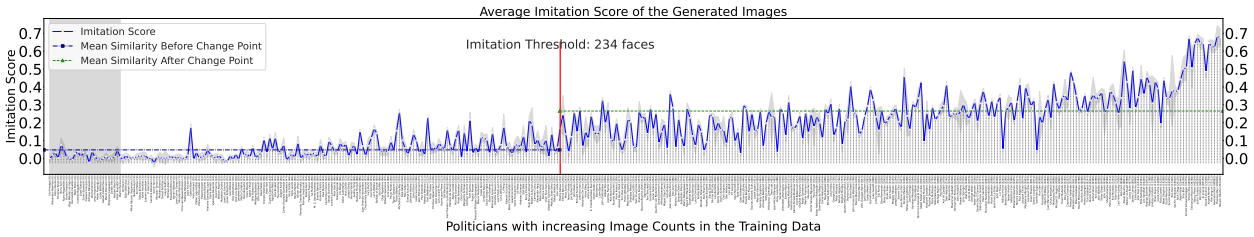


Figure 5.33: Human Face Imitation (Politicians): Similarity between the training and generated images for all politicians. The politicians with zero image counts are shaded with light gray. We show the mean and variance over the five generation prompts. The images were generated using **SD1.3**. The change point for human face imitation for politicians when generating images using SD1.1 is detected at **234 faces**.

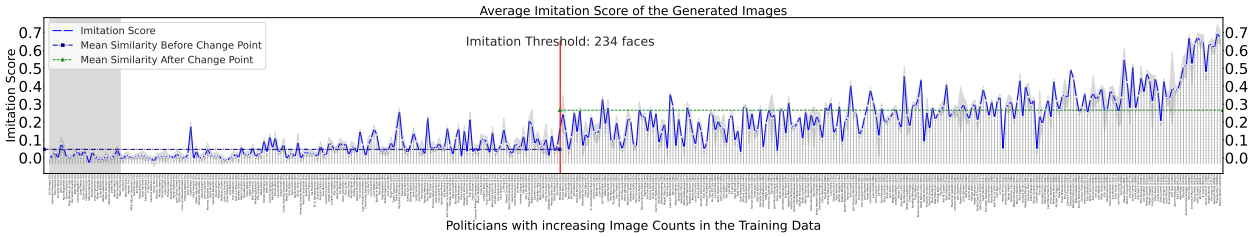


Figure 5.34: Human Face Imitation (Politicians): Similarity between the training and generated images for all politicians. The politicians with zero image counts are shaded with light gray. We show the mean and variance over the five generation prompts. The images were generated using **SD1.4**. The change point for human face imitation for politicians when generating images using SD1.1 is detected at **234 faces**.

Chapter 6

Robustness in ML Models and AI Safety

In this part, I present two of my works that aim to evaluate robustness of backdoor mitigation approaches and increase robustness of AI Safety classifiers.

Chapter 7

Effective Backdoor Mitigation in Vision-Language Models Depends on the Pre-training Objective

Despite the advanced capabilities of contemporary machine learning (ML) models, they remain vulnerable to adversarial and backdoor attacks. This vulnerability is particularly concerning in real-world deployments, where compromised models may exhibit unpredictable behavior in critical scenarios. Such risks are heightened by the prevalent practice of collecting massive, internet-sourced datasets for training multimodal models, as these datasets may harbor backdoors. Various techniques have been proposed to mitigate the effects of backdooring in multimodal models, such as CleanCLIP, which is the current state-of-the-art approach. In this work, we demonstrate that the efficacy of CleanCLIP in mitigating backdoors is highly dependent on the particular objective used during model pre-training. We observe that adding self-supervised objective to pre-training, that leads to higher zero-shot classification performance, correlate with harder to remove backdoors behaviors. We show this by training multimodal models on two large datasets consisting of 3 million (CC3M) and 6 million (CC6M) datapoints, under various pre-training objectives, followed by poison removal using CleanCLIP. We find that CleanCLIP, even with extensive hyperparameter tuning, is ineffective in poison removal when stronger pre-training objectives are used. Our findings underscore critical considerations for ML practitioners who train models using large-scale web-curated data and are concerned

about potential backdoor threats.

7.1 Introduction

Machine Learning (ML) has taken strides in training high-performing models for a wide range of tasks from classification to generation. An important goal for ML is to learn general-purpose representations that help align data from different modalities. Approaches like CLIP [184], ALIGN [116], and BLIP [142] learn joint representations from large scale image-text paired datasets. These innovative techniques have ushered in the possibility of learning from unlabeled and uncurated datasets, substantially increasing the scale and applicability of pre-training. The scaling has contributed to high zero-shot classification accuracy on various downstream datasets like Imagenet [56] and increased robustness to variations in the datasets like Imagenet-V2 [190], Imagenet-R [100], and Imagenet-A [101]. However, these strategies, reliant on internet-sourced data curation [76], have also raised concerns regarding the vulnerability of models to an adversary, particularly through backdoor attacks [30].

In the simplest form of this attack, an adversary inserts a patch (termed as a trigger patch or poison) in a small subset of the training data images and alters the ground truth label or caption to a target label or caption [89].[1] When trained on the poisoned training data, the model learns to associate the trigger patch with the target label/caption. If deployed, an adversary can get the model to predict the target label for any datapoint by inserting the trigger patch. The success of an adversary is measured by the attack success rate (ASR) metric, which is the percentage of the images with the trigger patch that are predicted with the target label. Previous works [27] have demonstrated effective backdooring of multimodal models (ASR $\geq$ 80%) just by poisoning a mere 75 out of 3 million training datapoints.

Several backdoor mitigation techniques have been proposed for multimodal models [11, 147, 270, 271] to tackle this vulnerability. These approaches either attempt to detect and filter the poisoned datapoints during the pre-training [147, 270, 271] or finetune the given backdoored model using a specialized loss function on a smaller, *guaranteed* to be clean image-text paired dataset. The latter approach helps the model to *forget* the association between the trigger patch and the target label while still maintaining the learned associations for benign datapoints, e.g., CleanCLIP [11]. CleanCLIP proposes to finetune a backdoored model using

[1] We refer the readers to Goldblum et al. [83] for discussion about other kinds of poisoning attacks.

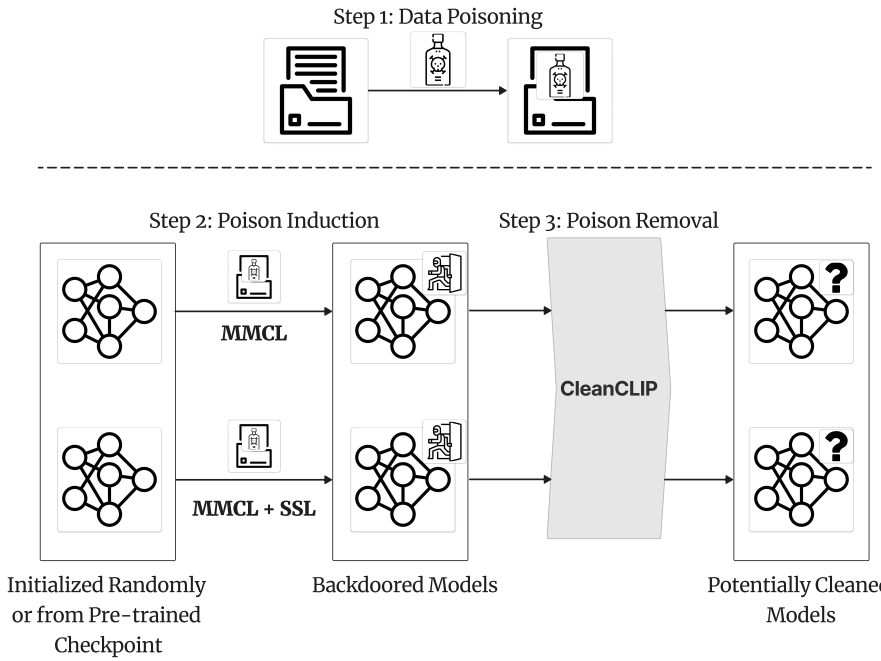


Figure 7.1: Our experimental setup to test the claim about the dependence of the ability of CleanCLIP to remove poison from a backdoored model on the model's pre-training objective.

a combination of contrastive loss and self-supervised loss on a small dataset, free of backdoors, to clean the model. It is the state-of-the-art (SOTA) technique to clean a backdoored model and obtain a low ASR ($\leq 5\%$) without hurting its zero-shot classification accuracy; thereby achieving a successful model cleaning.

So far, it has been demonstrated that CleanCLIP can successfully clean models pre-trained only with multimodal constrastive loss (MMCL) as the objective [184]. Several recent works [165, 145, 275, 139] have proposed stronger pre-training objectives that lead to better zero-shot image classification accuracy. Specifically, adding self-supervised loss (SSL) in both modalities has been the key player in all these works. Therefore, in this work, we pre-train multimodal models using a combination of MMCL and SSL on a poisoned training dataset. Consistent with the previous findings, models pre-trained using a combination of MMCL and SSL produced models with a higher classification accuracy than models trained solely with the MMCL objective. We then apply the finetuning procedure in CleanCLIP to remove the poison from these models (see Figure 7.1). To our surprise, we observe that CleanCLIP fails to successfully (i.e., without a significant loss in the model's zero-shot accuracy) remove poison from the models pre-trained with the stronger objective (combination of MMCL and SSL).

We further conduct experiments with other practical considerations, such as the performance of CleanCLIP when its cleaning data still has a few poisoned datapoints, and deciding the stopping criterion for the finetuning process when one is not aware of the specific backdoor attack on the model (which is usually the case). From all the experiments, we find that only the models pre-trained with MMCL alone are amenable to poison removal in both the cases of availability of completely clean finetuning data and when the finetuning data still has some poisoned datapoints.

Our main contributions are:

1. We show that the state-of-the-art technique for removing poison from backdoored multimodal models, CleanCLIP, depends on the model's pre-training objective and fails to mitigate poison when the models are pre-trained with a stronger objective, like the combination of MMCL and SSL losses. (See Section 2.5 for further justification on why we chose to study CleanCLIP.)
2. We conduct several analysis experiments to demonstrate the effect of different pre-training objectives on the strength of poison induction.
3. We conduct experiments to show the practical use case of CleanCLIP from a real-world perspective when the finetuning data is not entirely free of poison and the practitioner is not aware of the specific kind of poisoning in the model and hence has to decide on the stopping criterion for the cleaning process.
4. Based on these findings, we highlight critical considerations for an ML practitioner who wants to pre-train models by collecting web-curated data with potential backdoored embedded datapoints.

7.2 Methodology

7.2.1 Primer on Pre-training and Poisoning

7.2.1.1 Notations

Let $\mathcal{I}$ and $\mathcal{T}$ denote the space of images and text. $\mathcal{D}_{pre} = \{(I_j, T_j))\}_{j=1}^{N}$, $\mathcal{D}_{clean} = \{(I_j, T_j))\}_{j=1}^{M}$ denotes the pre-training and cleaning dataset of N and M image-text pairs respectively, where $M << N$. $h_I : \mathcal{I} \rightarrow \mathbb{R}^d$ and $h_T : \mathcal{T} \rightarrow \mathbb{R}^d$ denote the image and text encoders respectively, where d is the dimensionality of the embedding space. All the embeddings are further normalized to make ℓ_2 norm to 1 which we denote using $f(x) = g(h(x))$, where $g : \mathbb{R}^d \rightarrow \mathbb{B}(1)$ is normalization mapping, where, $\mathbb{B}(1) = \{x : \|x\|_2 = 1,\ x \in \mathbb{R}^d\}$;

τ denotes learnable temperature. Let $\mathcal{L}_{\text{MMCL}}$ denote the multimodal and $\mathcal{L}_{\text{SSL}}$ denote the intramodal self-supervision losses respectively. Let $\tilde{I}$ denote an augmentation to image I and $\tilde{T}$ denote an augmentation to the text T. Let $S \subset \{1, 2, \ldots, n\}$ denote a small subset of training data that are poisoned. We denote the poisoned dataset using $\mathcal{P}(S, \mathfrak{tg}, T') = \{(I_j \circ \mathfrak{tg}, T'_j) : j \in S\}$ where $\mathfrak{tg}, T'$ denote image trigger and target label respectively.

Loss Objectives Given a dataset $\mathcal{D}$, f_I, f_T, we define $\mathcal{L}_{\text{MMCL}}(\mathcal{D}, f_I, f_T, \tau)$ as follows:

$$= \frac{-1}{2|\mathcal{D}|}\left(\sum_{j=1}^{|\mathcal{D}|}\log\left[\frac{\exp\left(\langle f_I(I_j), f_T(T_j)\rangle/\tau\right)}{\sum_{k=1}^{|\mathcal{D}|}\exp\left(\langle f_I(I_j), f_T(T_k)\rangle/\tau\right)}\right] + \sum_{k=1}^{|\mathcal{D}|}\log\left[\frac{\exp\left(\langle f_I(I_k), f_T(T_k)\rangle/\tau\right)}{\sum_{j=1}^{|\mathcal{D}|}\exp\left(\langle f_I(I_j), f_T(T_k)\rangle/\tau\right)}\right]\right) \tag{7.1}$$

and, we define $\mathcal{L}_{\text{SSL}}(\mathcal{D}, f_I, f_T, \tau)$ as follows:

$$= \frac{-1}{2\mathcal{D}}\left(\sum_{j=1}^{|\mathcal{D}|}\log\left[\frac{\exp\left(\left\langle f_I(I_j), f_I(\tilde{I}_j)\right\rangle/\tau\right)}{\sum_{k=1}^{|\mathcal{D}|}\exp\left(\left\langle f_I(I_j), f_I(\tilde{I}_k)\right\rangle/\tau\right)}\right] + \sum_{j=1}^{|\mathcal{D}|}\log\left[\frac{\exp\left(\left\langle f_T(T_j), f_T(\tilde{T}_j)\right\rangle/\tau\right)}{\sum_{k=1}^{|\mathcal{D}|}\exp\left(\left\langle f_T(T_j), f_T(\tilde{T}_k)\right\rangle/\tau\right)}\right]\right) \tag{7.2}$$

For the shorthand notations, we will drop f_I, f_T, τ from the parenthesis. With the definitions above $\mathcal{L}_{\text{CleanCLIP}}(\mathcal{D}_{clean}) \triangleq \mathcal{L}_{\text{SSL}}(\mathcal{D}_{clean}) + \mathcal{L}_{\text{MMCL}}(\mathcal{D}_{clean})$. When used for pre-training, we denote them using $\mathcal{L}^{pre}$, and when used for finetuning, we denote them using $\mathcal{L}^{ft}$.

7.2.2 Experimental Setup

On a high level, our experiments involve poisoning CLIP models using two distinct pre-training objectives with different kinds of backdoors by either training a model from scratch or by finetuning from a pre-trained checkpoint. Once we have poisoned the model, we attempt to remove the poison using CleanCLIP, which finetunes the model with a specific objective using a separate dataset. We have illustrated this in Figure 7.1 and summarized our key findings in Table 7.1.

7.2.2.1 Training Details

We train a dual-encoder multimodal model on image-text paired datasets. We train models using two kinds of pre-training objectives: a) only multimodal contrastive loss ($\mathcal{L}^{pre}_{\text{MMCL}}$), and b) combination of multimodal

contrastive loss and self-supervised loss in the image and text modalities ($\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$). Following CleanCLIP, we use a ResNet-50 as the model's vision encoder and a transformer as the text encoder. We trained the models on two image-text paired datasets:

1. Conceptual Captions 3M (CC3M) [215]: This dataset has 3M image-text paired datapoints.
2. Conceptual Caption 6M (CC6M): This dataset has 6M image-text paired datapoints from the CC12M dataset [36], to which size our computing resources scaled.

The models are trained either *from scratch* or *finetuned from a pre-trained CLIP checkpoint* [184]. We train models for 64 epochs using 8 Nvidia A100 GPUs. The initial learning rate of $1e-3$ with cosine scheduling is used when trained from scratch and $5e-7$ when finetuned from a checkpoint. We use AdamW optimizer with 10,000 warmup steps [152]. Models trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ use a batch size of 256, whereas models trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ use a batch size of 128. Please refer to Section 7.6 for the loss dynamics.

Poisoning Following CleanCLIP, we introduce the trigger proposed by *BadNet* [89] in a small subset of the training datapoints. Specifically, we add a trigger patch of size 16×16 sampled from a standard Gaussian at a random location in the image and subsequently change the image's caption to be the adversary chosen label, in this case "banana". Please see Section 7.15 for examples of images with trigger patch and their corresponding captions. Using the same settings as CleanCLIP, we introduce the trigger in 1,500 randomly sampled datapoints for the CC3M dataset and 3,000 randomly sampled datapoints for the CC6M dataset (a mere 0.05% of the training datapoints).

We also experiment with another kind of poisoning technique: *label consistent poisoning*. In this case, the trigger patch (created in the manner as mentioned above) is added to the images that have the adversary chosen label (in this case"banana") in their captions.

Removing poison We attempt to remove poisons from pre-trained models by finetuning them on a 100K clean image-text paired dataset using $\mathcal{L}^{pre}_{\text{MMCL}}$, $\mathcal{L}^{pre}_{\text{SSL}}$, and $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$ (CleanCLIP). We consider a model to be cleaned if the *ASR* of that model *is* $\leq 5\%$.

Table 7.1: Key findings from our experiments: CleanCLIP is much less effective for the model where poison is induced with the stronger objective $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$.

Backdoor	Objective	Poison Induction	Change in Accuracy after Cleaning (Relative) ↑
BadNet	$\mathcal{L}^{pre}_{\text{MMCL}}$	From scratch	1% gain
BadNet	$\mathcal{L}^{pre}_{\text{MMCL}}$	Finetuned from Ckpt	17% loss
Label Consistent	$\mathcal{L}^{pre}_{\text{MMCL}}$	From scratch	10% gain
BadNet	$\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$	From scratch	45% loss
BadNet	$\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$	Finetuned from Ckpt	33% loss
Label Consistent	$\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$	From scratch	16% loss

7.3 Experiments

In this section, we expound on the pre-training details for the models, followed by their cleaning procedure and the metrics we use to measure the model's performance.

7.3.1 Metrics

The models are evaluated for their Top-1 zero-shot accuracy on the Imagenet-1K validation set (referred to as Imagenet hereafter). Each of the 1,000 classes of Imagenet is described using sentences like: 'a photo of a ...', 'a tattoo of a ...', etc. We generate 80 such text templates for each class (see Section 7.8) and then pass them to the text encoder to produce an average text embedding for the class. During zero-shot classification, the prediction for an image is the class whose thus computed text embedding has the highest cosine similarity with the image embedding.

We also evaluate the attack success rate (ASR) of a model. In an apparent similarity to accuracy, the ASR of a backdoored model is defined as the percentage of triggered images that the model classifies as the adversary-chosen target label. For measuring ASR, we add the trigger patch at random locations in all Imagenet validation set images and measure the percentage of them that are classified as the target class, i.e., "banana". We measure both these metrics at the end of each cleaning epoch as any model encountered during the cleaning process is a good candidate for a cleaned model.

Table 7.2: This table shows the original Top-1 zero-shot Imagenet validation set accuracies and remaining accuracy after cleaning the models that were poisoned using BadNet by training from scratch. The cleaning is done using CleanCLIP, i.e., finetuning a poisoned model with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$. For this table, we choose the models having the highest accuracy and ASR $\leq 5\%$ (successful cleaning). The original ASR values for all models are more than 99%. **Takeaway:** The models trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ maintain their original accuracy after cleaning, while the ones trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ experience a huge drop relative to the starting accuracy ($\sim$20% for model trained on CC3M dataset and 45% for model trained on CC6M dataset) after cleaning.

			Trained with $\mathcal{L}^{pre}_{\text{MMCL}}$		Trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$	
Backdoor	Dataset	Clean Data Size	Orig. Acc.	Clean Acc. (ASR $\leq 5\%$) $\uparrow$	Orig. Acc.	Clean Acc. (ASR $\leq 5\%$) $\uparrow$
BadNet	CC3M	100K	**16.00%**	16.49%	**17.04%**	14.16%
BadNet	CC6M	100K	**23.76%**	24.04%	**23.86%**	13.05%

7.3.1.1 Poison Induction by Training from Scratch

Table 7.2 shows the Top-1 zero-shot Imagenet validation set accuracy for the models trained from scratch using $\mathcal{L}^{pre}_{\text{MMCL}}$ and $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ on CC3M and CC6M datasets. For the smaller CC3M dataset, both the models achieve an accuracy of around 16–17%, and for the larger CC6M dataset, the models reach an accuracy of around 24%. *Even though the models trained with $\mathcal{L}^{pre}_{MMCL} + \mathcal{L}^{pre}_{SSL}$ attained higher accuracy than the models trained with $\mathcal{L}^{pre}_{MMCL}$ alone, in order to have better visualization of the difference in performance of CleanCLIP on the two pre-training objectives, we deliberately choose models with similar starting accuracies.* All the models, irrespective of the pre-training objective and the training dataset, reached more than 99% ASR (see Table 7.3 in Section 7.6), implying that poisoning just 0.05% of the dataset is enough to attain very high ASR.

Removing Poison We clean the poisoned model by finetuning it on a 100K, guaranteed to be poison-free, image-text pairs for 20 epochs using a batch size of 128 and AdamW as the optimizer. We perform extensive hyperparameter search and use various learning rates (as many as 8 in some experiments and 14 in others, all with cosine scheduling and 50 warmup steps) for this process. Please refer to Section 7.9 for the set of learning rates explored for the cleaning procedure. Ideally, after cleaning we would want to obtain a model that maintains the accuracy of the original poisoned model, while getting rid of its poison, i.e., very low ASR. We use three different loss functions for the cleaning process:

1. $\mathcal{L}^{ft}_{\text{MMCL}}$: CleanCLIP showed that finetuning with $\mathcal{L}^{ft}_{\text{MMCL}}$ did not change the original model's accuracy and

ASR, and hence is an *ineffective cleaning loss*. We reproduce these results for the models we trained.

2. $\mathcal{L}^{ft}_{\text{SSL}}$: CleanCLIP also showed that finetuning with $\mathcal{L}^{ft}_{\text{SSL}}$ decreased the original model's ASR at the expense of its accuracy, and hence is also an *ineffective cleaning loss*. We also reproduce these results.
3. $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$: CleanCLIP showed that finetuning with a combination of these losses decreased the original model's ASR while not hurting its accuracy, and hence is an *effective cleaning loss*. Our experiments show that while this observation is true for the models trained with $\mathcal{L}^{pre}_{\text{MMCL}}$, however *it does not generalize* to the models trained with stronger pre-training objective $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$. **This is our key finding.**

Findings from the Cleaning Procedure Figure 7.2 shows the scatter plot of the Top-1 zero-shot Imagenet validation set accuracy and the ASR at the end of each cleaning epoch for the models trained on the CC6M dataset. We defer the plots for the CC3M dataset in Section 7.10.1 for space consideration. For both the datasets, we observe that:

1. $\mathcal{L}^{ft}_{\text{MMCL}}$ and $\mathcal{L}^{ft}_{\text{SSL}}$ individually are ineffective cleaning losses as they cause a significant drop in accuracy for lowering the ASR for both the pre-training objectives.
2. $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$ serves as an effective cleaning loss for the model trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ (left plot). The cleaned models maintain the accuracy of the original model, but they have low ASR, which we consider *successful cleaning*. However, it does not lead to an effective cleaning of the model trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ (right plot). Even the model that has the highest accuracy with a low ASR ($\leq$ 5%) is 45% less accurate than the original model, as shown in Figure 7.2.

For both datasets, our findings indicate that CleanCLIP is not effective in removing poison from the models trained with a stronger pre-training objective $\mathcal{L}^{pre}_{\text{MMCL}}+\mathcal{L}^{pre}_{\text{SSL}}$, without a significant drop in accuracy. Table 7.2 gives the highest accuracy of the models which were successfully cleaned by CleanCLIP (ASR $\leq$ 5%).

Poison Induction by Finetuning a pre-trained model We also induce poison by finetuning a pre-trained CLIP model [184]. Concretely, we poison two models by finetuning them with $\mathcal{L}^{pre}_{\text{MMCL}}$ and $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ respectively, using the CC6M dataset that had 3000 poisoned datapoints. We use a learning rate of $5e-7$ with AdamW optimizer with 10,000 warmup steps. After poisoning, these models achieve Top-1 zero-shot Imagenet set accuracy of $\sim 60\%$, much higher than the models trained from scratch (Figure 7.2). The ASR for the model poisoned with $\mathcal{L}^{pre}_{\text{MMCL}}$ is 99% and for the model poisoned with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ is 90%.

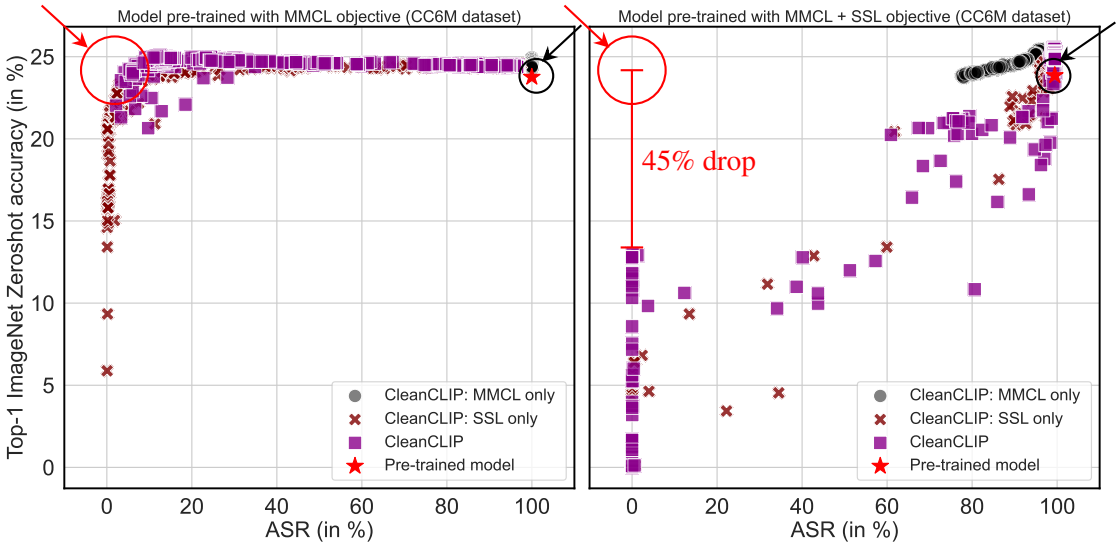


Figure 7.2: Top-1 zero-shot Imagenet validation set accuracy vs. the ASR, measured at the end of each cleaning epoch for the models trained on the CC6M dataset. The cleaning is done by finetuning the model with the three losses mentioned above. The red star in the top right corner (encircled in the black circle) corresponds to the model's starting accuracy and ASR (before cleaning). For a successful cleaning, there should be models that maintain the model's starting accuracy while having a low ASR (indicated by the red circle's region in the top left). There are several models in the red circle in the left plot (successful clean), while there are no models in the red circle in the right plot (unsuccessful clean). **Takeaway:** CleanCLIP successfully cleans the model trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ (left), while it is ineffective for the models trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ (right).

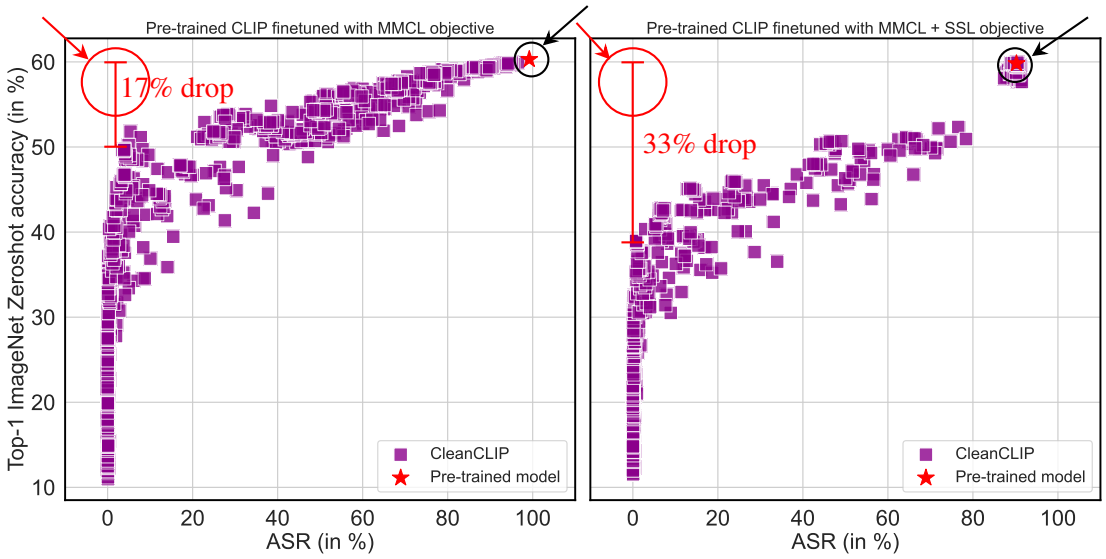


Figure 7.3: Top-1 zero-shot Imagenet validation set accuracy vs. the ASR, measured at the end of each cleaning epoch for the models poisoned by finetuning a CLIP pre-trained checkpoint on the CC6M dataset. The cleaning is done by finetuning the poisoned model with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$. The red star in the top right corner (encircled in the black circle) corresponds to the original model's accuracy and ASR (before cleaning). For a successful cleaning, there should be models that maintain the original model's accuracy while having a low ASR (indicated by the red circle in the top left). **Takeaway:** CleanCLIP is unable to successfully clean both the models; however, it performs much worse for the model poisoned with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ (right).

Findings from the Cleaning Procedure: We clean the poisoned models using CleanCLIP, i.e., further finetuning on a clean dataset with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$. Figure 7.3 shows the scatter plot of the Top-1 zero-shot Imagenet validation set accuracy and the ASR at the end of each finetuning epoch for these two models. In this case, both the models experience a drop in accuracy to obtain a low ASR ($\leq 5\%$); however, the drop is much higher for the model when the poison was induced using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ (33%, compared to a 17% drop for the model when the poison was induced using $\mathcal{L}^{pre}_{\text{MMCL}}$). This experiment corroborates our previous finding that CleanCLIP is less effective when the poison is induced using a combination of MMCL and SSL, irrespective of the fact whether the poison is induced via finetuning or by training from scratch.

Poisoning a Different Backbone Architecture For this experiment, we poison a model with a different backbone architecture, specifically ViTs. We poisoned these models by training them on the CC6M dataset with 3000 poisoned datapoints, and cleaned them using CleanCLIP. Figure 7.4 shows the cleaning plots for these models. We observe that similar to the previous experiments, the model trained with MMCL + SSL experiences a much larger drop in accuracy than the model just trained with MMCL, although in this case

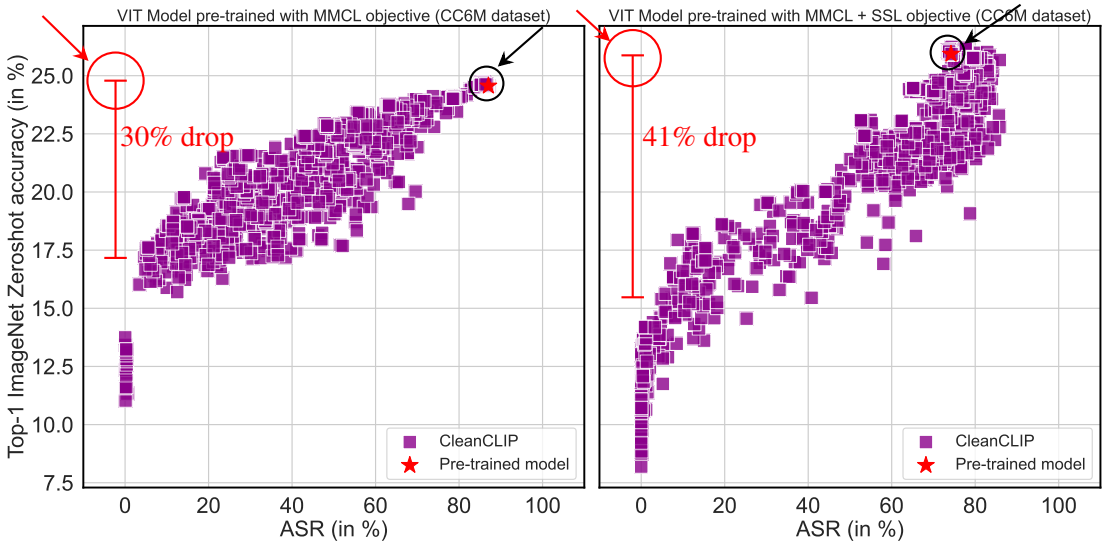


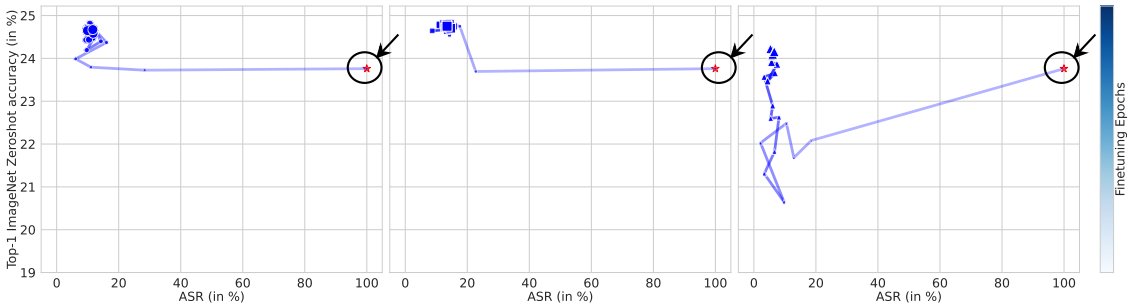
Figure 7.4: Top-1 zero-shot Imagenet validation set accuracy vs. the ASR, measured at the end of each cleaning epoch for the models trained on the CC6M dataset. The cleaning using CleanCLIP. The red star in the top right corner (encircled in the black circle) corresponds to the model's starting accuracy and ASR (before cleaning). **Takeaway:** When a ViT backbone is poisoned, there are no cleaned models that maintain the original accuracies for both the pre-training losses, however the drop is much larger for the model trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ (right).

the model just trained with MMCL experiences a lot drop in accuracy as well.

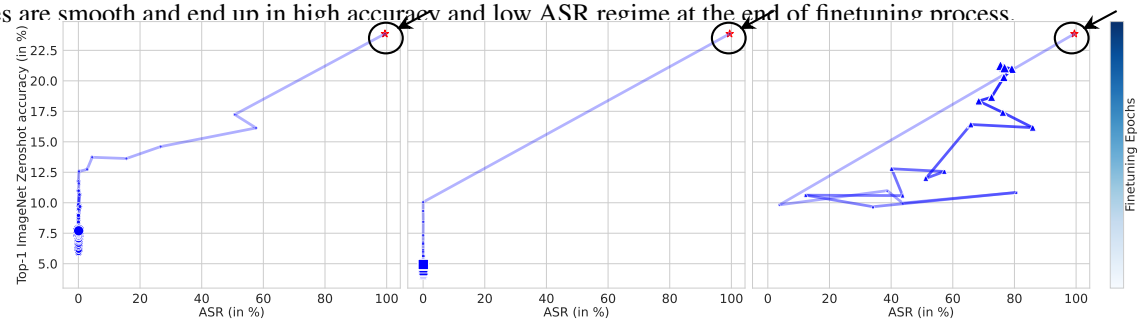
Poisoning Induction using a Different Poison Due to space considerations, we present the results for the effectiveness of CleanCLIP when models are poisoned using label consistent backdoors in Section 7.11. We observe that similar to the case of poisoning with BadNet, CleanCLIP is much less effective when the poison is induced using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ (16% loss in accuracy), compared to the case when it is induced using $\mathcal{L}^{pre}_{\text{MMCL}}$ (10% gain in accuracy).

Dependence of the Stopping Criterion on the Pre-training Objective In the previous section, the ability to find a model with high accuracy and low ASR is considered a success for the CleanCLIP approach. However, in practice, one would not be aware of the ASR of the model being cleaned and, therefore, would not know when to stop the cleaning process. To highlight this practicality concern, we show the multiple cleaning trajectories for models trained using different pre-training objectives in Figure 7.5.

Figure 7.5a shows the trajectories of three cleaning runs with different learning rates for a model trained using $\mathcal{L}^{pre}_{\text{MMCL}}$ on the CC6M dataset. We observe that in all the three runs, the trajectory converges to a



(a) Finetuning trajectory of three runs with different learning rates for the model pre-trained with $\mathcal{L}^{pre}_{\text{MMCL}}$. The trajectories are smooth and end up in high accuracy and low ASR regime at the end of finetuning process.



(b) Finetuning trajectory of three runs with different learning rates for the model pre-trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$. The trajectories are not smooth and increased finetuning can lead to both decreased accuracy and higher ASR.

Figure 7.5: Finetuning trajectories of models with different pre-training objectives. Successive finetuning epochs are shown with increasing size of the markers and intensity of the connecting line. The red star in the top right corner (encircled in the black circle) corresponds to the original model's accuracy and ASR. **Takeaway:** Models trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ converge to a region of high accuracy and low ASR as we continue to finetune. On the other hand, models trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ fail to converge to a region of high accuracy and low ASR, and continued finetuning can lead to both decreased accuracy and higher ASR. This makes determining the stopping criterion for the cleaning process for the latter models challenging.

region of high accuracy and low ASR (top left corner), and the trajectories are smooth. This indicates that a practitioner can clean this model by finetuning the poisoned model for as long as their resources allow, and choose the model at the end of the finetuning process. They will likely obtain a model with high accuracy and low ASR, i.e., a successfully cleaned model.

Figure 7.5b shows the trajectories of three cleaning runs with different learning rates for a model trained using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ on the CC6M dataset. We observe that in all three runs, the successfully cleaned model (high accuracy with a low ASR ($\leq 5\%$)) is an intermediate model in the trajectory, *and not* the model at the end of the process. With continued finetuning, the model can both lose accuracy and gain ASR, both of which are undesirable. Therefore, it is difficult for a practitioner to discern when to stop the finetuning process to obtain a clean model.

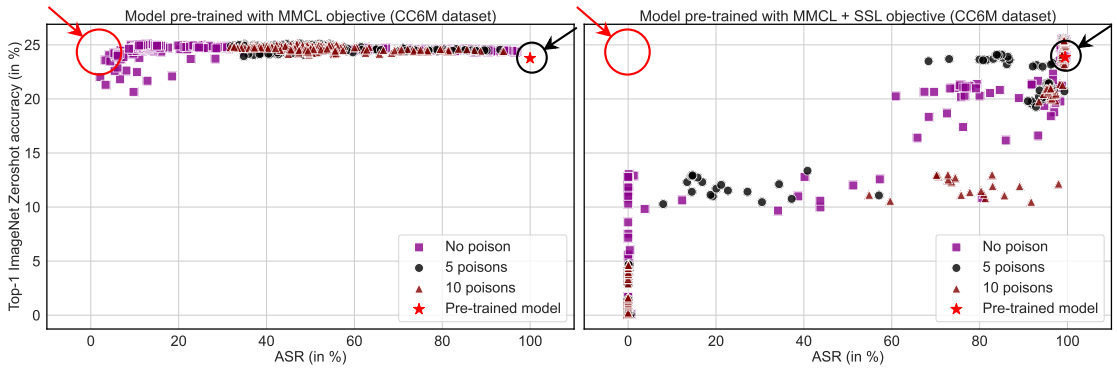


Figure 7.6: Top-1 zero-shot Imagenet validation set accuracy v/s the ASR during the cleaning process for the two models. The finetuning is done with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$. We measure accuracy and ASR at the end of each epoch. The red star in the top right corner (encircled in the black circle) corresponds to the original model's accuracy and ASR. For a successful cleaning, there should be models that maintain the original model's accuracy while having a low ASR (indicated by the red circle). **Takeaway:** Even having 5 poisons in the cleaning dataset (i.e. 0.005% of the dataset, which is $10\times$ cleaner than the pre-training data) hurts the cleaning process for both pre-training objectives, and $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ trained models are hurt worse.

Therefore, the practicality of using CleanCLIP also depends on the pre-training objective of the model. Section 7.16 shows the cleaning trajectories for all explored hyperparameters.

Dependence of CleanCLIP on the Ideal Condition of the Dataset CleanCLIP assumes that the cleaning data is entirely free of poisoned datapoints. In practice, this assumption can be violated even when considerable care is taken to ensure it. To simulate this real-world situation, we clean models using data with a few poisoned datapoints, specifically 5 and 10 poisoned datapoints in the 100K cleaning datapoints. Note that these datasets are still, respectively, $10\times$ and $5\times$ cleaner than the original training dataset, illustrating a situation where the cleaning data is much cleaner than the training dataset but still not perfect.

Figure 7.6 shows the scatter plot of the Top-1 zero-shot Imagenet validation set accuracy and the ASR at the end of each cleaning epoch when two models trained from scratch on the CC6M dataset, one using $\mathcal{L}^{pre}_{\text{MMCL}}$ and the other using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ is cleaned by finetuning on this slightly poisoned dataset with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$. We observe that having just 5 poisoned datapoints in the cleaning dataset severely weakens CleanCLIP for both the pre-training objectives.

For models pre-trained with $\mathcal{L}^{pre}_{\text{MMCL}}$, we found cleaned models that maintain the original model's accuracy and achieve around 30-50% ASR. On the other hand, for the models pre-trained with the stronger objective $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$, having just 5 poisoned examples renders the cleaning procedure completely ineffective. The

models pre-trained on the CC6M dataset lose about 80% of the original model's accuracy to obtain a low ASR ($\leq 5\%$), and no model has an ASR lower than 90% for the CC3M pre-trained model (Figure 7.11).

Takeaways Our experiments highlight the fact that a stronger pre-training objective, like the combination of MMCL and SSL, also affects the strength of poison induction, making the cleaning process difficult. Also, for a practitioner, when pre-training with a stronger objective, the decision of when to stop finetuning becomes non-trivial, as we show that the model at the end of the cleaning procedure is usually not the one with the lowest ASR and the best accuracy. The situation is further exacerbated when we even slightly relax the assumption of 100% poison-free cleaning data, which can be too stringent in practice.

7.4 Analysis of the Stronger Pre-training Objective

We now perform several analysis experiments to understand the reason behind the difference in the poison removal ability of CleanCLIP across the two pre-training objectives. We also experiment with other methods to attempt to remove the poison when induced using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$.

7.4.1 Cleaning using an Objective distinct from Pre-training

CleanCLIP successfully cleans the models trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ by finetuning with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$. However, it was unsuccessful for the model trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$. A plausible reason for this behavior could be that we are using the same pre-training and cleaning objective in the latter case, and CleanCLIP might be able to successfully clean the latter models if we were to clean it with a loss objective that is distinct from its pre-training objective. To test this hypothesis, we clean the models trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ by finetuning with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}} + \mathcal{L}^{ft}_{\text{DeepClust}}$, where $\mathcal{L}^{ft}_{\text{DeepClust}}$ is an additional deep clustering objective [31] on the vision encoder.

In deep clustering, we first obtain a pseudo-label for each image. We obtain the pseudo-label for an image in two ways: **a)** by classifying each image into one of the 1,000 Imagenet classes using powerful models such as SigLIP ViT-L/14 (zero-shot Imagenet accuracy of 83.08%) [284], and **b)** performing a 1,000-way clustering on feature space of our trained vision encoder, using FAISS [118]. Note that the use of SigLIP ViT-L/14 for obtaining pseudo-labels is a *cheating experiment* for a deep clustering task (we don't have

access to ground truth labels and therefore use SigLIP ViT-L/14 for this task); however, this experiment is solely performed to probe the upper bound of the poison removal that can be obtained when using deep clustering approaches.

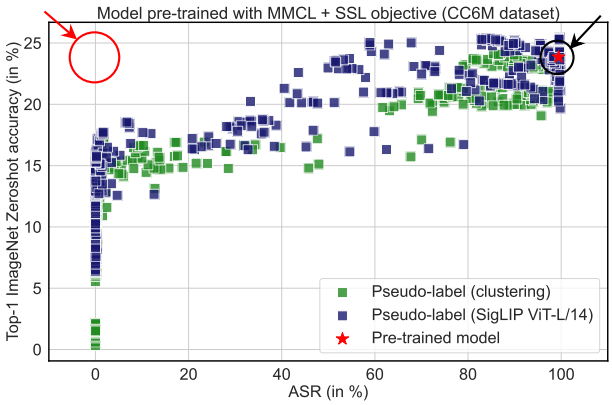


Figure 7.7: Top-1 zero-shot Imagenet validation set accuracy vs. the ASR during the cleaning process for the $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-trained model on the CC6M dataset. The finetuning is done with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}} + \mathcal{L}^{ft}_{\text{DeepClust}}$. **Takeaway:** Adding $\mathcal{L}^{ft}_{\text{DeepClust}}$ is unable to successfully remove the poison from models pre-trained using the strong objective, indicating that having distinct pre-training and cleaning objectives does not ensure removal of poison.

In the latter case, when we use clustering-based pseudo-label assignment for every image in the cleaning dataset, we learn to predict the assigned pseudo-label ($\hat{y}$) with the help of a linear classifier on top of the vision encoder using cross-entropy loss ($\mathcal{L}_{\text{Xent}}$). Let $W \in \mathbb{R}^{d \times 1000}$ be the linear classifier mapping visual features ($\mathbb{R}^d$) to one of the 1000 pseudo-labels. For a given datapoint (I_j, T_j) with assigned pseudo-label $\hat{y}_j$, deep clustering's objective becomes

$$\mathcal{L}^{ft}_{\text{DeepClust}}(f_I, W; I_j, \hat{y}_j) = \mathcal{L}_{\text{Xent}}(W^T f_I(I_j); \hat{y}_j), \tag{7.3}$$

and therefore overall objective becomes $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}} + \mathcal{L}^{ft}_{\text{DeepClust}}$. Following Caron et al. [31], we re-initialize classifier head W every time we re-compute pseudo-labels. We finetune the model trained using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ on the CC6M dataset for 10 epochs using 8 different learning rates for both the clustering techniques (see Section 7.9 for hyperparameter details) and measure the Top-1 zero-shot Imagenet accuracy and ASR at the end of each finetuning epoch.

Figure 7.7 shows the scatter plot of the Top-1 zero-shot Imagenet validation set accuracy and the ASR for

this experiment. We observe that adding deep clustering does not successfully remove the poison from the model. This experiment shows that the ineffectiveness of CleanCLIP in cleaning the model trained with the stronger objective is not just due to the same pre-training and cleaning objectives but due to stronger poison induction. Had the difference between the pre-training and finetuning objectives been the reason for CleanCLIP's success, then this DeepClustering experiment could have successfully cleaned the model.

7.4.1.1 Poison Removal using Heavy Regularization

In this section, we attempt to remove the poison induced using $\mathcal{L}^{pre}_{\text{MMCL}}+\mathcal{L}^{pre}_{\text{SSL}}$ with heavy regularization. Zhu et al. [289] proposed heavy regularization as an approach to remove backdoors from vision and language models. To evaluate the efficacy of this approach, we finetune these models with 9 different regularization weights for 10 epochs using 8 different learning rates for each regularization weight on 100K image-text pairs (see Section 7.9 for hyperparameter details). The regularization loss is added to three different loss objectives during the finetuning process: a) $\mathcal{L}^{ft}_{\text{MMCL}}, b)\mathcal{L}^{ft}_{\text{SSL}}$, and c) $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$.

Figure 7.8 shows the scatter plot of the Top-1 zero-shot Imagenet validation set accuracy and the ASR at the end of each finetuning epoch. We observe that no hyperparameter combination can successfully remove the poison, which shows that simply adding regularization cannot remove the strongly induced poison.

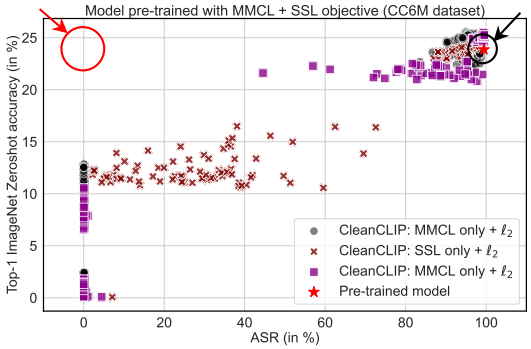


Figure 7.8: Top-1 zero-shot Imagenet validation set accuracy vs. the ASR during the cleaning process. **Takeaway:** Using regularization with any hyperparameter combinations is unable to remove the poison from models trained using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$.

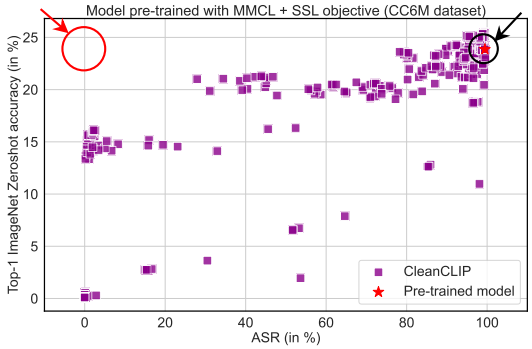


Figure 7.9: Top-1 zero-shot Imagenet validation set accuracy vs. the ASR during the cleaning process post shrinking and perturbing the weights. **Takeaway:** The shrink and perturb technique is unable to remove poison from the model that is trained using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$.

Poison Removal using Shrink and Perturb In this section, we attempt to remove the poison using the shrink and perturb technique. Ash and Adams [8] proposed shrink and perturb technique to improve the accuracy of a model when finetuned for a task. CleanCLIP also cleans a model by finetuning, and therefore, it could benefit from this technique. In this technique, a small noise is added to the model weights before starting finetuning, $\theta_0 \leftarrow \lambda\theta_0 + p_0$, where $p \sim \mathcal{N}\left(0, \sigma^2\right)$ and $\lambda \in (0, 1)$. To evaluate whether this approach can help in removing the poison from a model trained using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$, we add a small noise to all its weights. While the choice of noise scale (σ) and shrinking parameter (λ) is a hyperparameter, we experiment with 5 values of σ for 15 values of λ. To determine which model to clean amongst the 75 possible models obtained from shrink and perturb described above, we measure the Top-1 zero-shot Imagenet validation set accuracy and the ASR of the models with noised weights, and selected model with the highest accuracy whose ASR was lower than 15% for cleaning. We clean the selected model by finetuning it using 8 learning rates for 20 epochs on 100K image-text pairs. Figure 7.9 shows the accuracy and ASR at the end of each finetuning epoch. We observe that even this approach is unable to successfully remove the poison from the models.

Ablation on Hyperparameters We also perform ablation experiments to see the impact of hyperparameters on the ability of CleanCLIP to remove poison from the $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-trained model. In particular, we study the impact of the size of the cleaning dataset, the number of finetuning epochs, and the weightage of $\mathcal{L}^{ft}_{\text{SSL}}$. Section 7.12 reports the metrics when the finetuning is done on a dataset that is twice the size of the cleaning dataset used in the previous section. Section 7.13 reports the metrics when the finetuning is done for up to 100 epochs, i.e., 5$\times$ longer than in the previous section. Section 7.14 reports the accuracy and ASR metrics when the finetuning is done with higher weight to the $\mathcal{L}^{ft}_{\text{SSL}}$ loss. While changing the hyperparameters helps improve the Pareto-frontier curve, none of the three ablations can successfully remove the strongly induced poison.

Takeaways These experiments demonstrate that making the pre-training and finetuning objectives distinct from each other and perturbation techniques like heavy regularization [289] and shrink-and-perturb [8] are not enough to remove the strongly induced poison. Moreover, we observe that while doing a more fine-grained search on different hyperparameters helps slightly improve the Pareto-frontier curve, none of the

hyperparameter ablations can successfully remove the poison. Since removing the poison from such a model is an open research problem, a practitioner who is engaged in training models using web-curated data should consider training their model with only $\mathcal{L}^{pre}_{\text{MMCL}}$ so that they can clean their model on a carefully curated image-text pair dataset to remove potential backdoors from it, even when there are a few poisoned examples in the finetuning dataset.

7.5 Conclusions

Through our extensive hyperparameter search and ablation experiments, we unveil a critical limitation of the current state-of-the-art poison mitigation technique for multimodal models, CleanCLIP. It fails to effectively remove backdoor poisoning when a model is trained using stronger objectives like the combination of multimodal contrastive learning (MMCL) and intramodal self-supervised learning (SSL). This objective is common in popular approaches like SLIP [165] and has demonstrated superior accuracy over training with only the MMCL objective. Our experiments show that this vulnerability persists irrespective of the scale of the pre-training and the cleaning datasets, irrespective of the manner of poison induction (from scratch or by finetuning), and irrespective of the specific backdoor attack.

Particularly concerning is the unstable cleaning trajectory in models trained using the stronger objective (Figure 7.5b). Often unaware of the specific backdoor attack, practitioners face challenges in determining the optimal point to halt the cleaning process. This instability can lead to suboptimal models, as continued finetuning can decrease accuracy and increase attack success rate (ASR). Furthermore, our findings highlight the critical assumption of a completely poison-free cleaning dataset for CleanCLIP's effectiveness, an assumption that is rarely met in practical scenarios. This becomes particularly problematic with the use of stronger pre-training objectives. The models trained with the simpler MMCL objective evade both these issues by having stable cleaning trajectories and amenability to poison reduction even under non-ideal conditions.

Given these insights, we urge practitioners to consider training their models using the simpler MMCL objective. Even though this might slightly hurt the accuracy, it significantly enhances its amenability to remove backdoors. Our recommendation would also circumvent the issue of knowing when to halt the cleaning procedure, as more finetuning epochs would not hurt the model's accuracy and ASR. Further, it will also be

Table 7.3: This table shows the original Top-1 zero-shot Imagenet validation set accuracy and ASR for the models trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ and $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ on the CC3M and CC6M datasets. These models are the input to CleanCLIP approach to remove their poison. **Takeaway:** The models trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ achieve higher accuracy than their counterparts trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ solely (except when poisoned by finetuning). Adding BadNet poison to a mere 0.05% of the training data achieves almost a 100% ASR when trained from scratch and almost 90% ASR when induced by finetuning.

Poison Induction	Backdoor	Objective	Dataset			
			CC3M		CC6M	
			Accuracy (↑)	ASR (↓)	Accuracy (↑)	ASR (↓)
From Scratch	BadNet	$\mathcal{L}^{pre}_{\text{MMCL}}$	16.00%	99.88%	23.76%	99.98%
From Scratch	BadNet	$\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$	17.04%	99.03%	23.86%	99.45%
From Scratch	Label-Consistent	$\mathcal{L}^{pre}_{\text{MMCL}}$	–	–	22.96%	88.27%
From Scratch	Label-Consistent	$\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$	–	–	23.09%	88.01%
Finetuning from Ckpt	BadNet	$\mathcal{L}^{pre}_{\text{MMCL}}$	–	–	60.30%	99.20%
Finetuning from Ckpt	BadNet	$\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$	–	–	59.81%	90.24%

beneficial when cleaning data is not entirely poison-free. Our work underscores the formidable challenge of defending models against backdoor attacks, an open research problem. We encourage future defense methods to robustly test their approach against various pre-training objectives and we invite the community to develop robust defense methods.

We enlist several future research directions that might be useful to pursue in the light of our findings:

1. Understanding the root cause of our findings: We find that a stronger pre-training loss makes removing poison harder. Is this because of a pre-training loss only? or can this also happen if we train a model for much more longer? Basically anyway of incorporating the poison much more strongly.
2. What can be done to remove the strongly induced poison – we tried using several stronger losses for cleaning which did not help. Can we curate specific diverse datasets that might help for this task? For example, using diverse datasets that target ImageNet performance.

7.6 Pre-Training Details

We train all the models on 8 Nvidia A100 GPUs for 64 epochs. We use an initial learning rate of 0.001 for the models trained from scratch, and for the models where poison is induced by finetuning from a pre-trained

checkpoint, we use an initial learning rate of $5e-7$. We use cosine scheduling and 10000 warmup steps with AdamW optimizer [152] for training. The model trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ uses a batch size of 256, whereas the model trained with the $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ uses a batch size of 128.

We show the change in accuracy and ASR with the training epochs for the model trained from scratch with BadNet attack using $\mathcal{L}^{pre}_{\text{MMCL}}$ in Figure 7.17 and using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ in Figure 7.18. We use early-stopping for the model trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ and choose the model with the highest accuracy. For $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-training, we choose the model that has the closest accuracy to the $\mathcal{L}^{pre}_{\text{MMCL}}$ trained model. Table 7.3 shows the accuracy and the ASR for all the models we select in this paper for poison removal.

7.7 Performance of the Pre-trained Models

Table 7.3 shows the Top-1 zero-shot Imagenet Validation set accuracy and the ASR of models that were selected for both the pre-training objectives.

7.8 Templates for Text-Embedding Computation

```
'a bad photo of a {class}.', 'a photo of many {class}.', 'a sculpture of a {
class}.', 'a photo of the hard to see {class}.', 'a low resolution photo of
the {class}.', 'a rendering of a {class}.', 'graffiti of a {class}.', 'a bad
photo of the {class}.', 'a cropped photo of the {class}.', 'a tattoo of a {
class}.', 'the embroidered {class}.', 'a photo of a hard to see {class}.', 'a
bright photo of a {class}.', 'a photo of a clean {class}.', 'a photo of a
dirty {class}.', 'a dark photo of the {class}.', 'a drawing of a {class}.', 'a
 photo of my {class}.', 'the plastic {class}.', 'a photo of the cool {class
}.', 'a close-up photo of a {class}.', 'a black and white photo of the {class
}.', 'a painting of the {class}.', 'a painting of a {class}.', 'a pixelated
photo of the {class}.', 'a sculpture of the {class}.', 'a bright photo of the
{class}.', 'a cropped photo of a {class}.', 'a plastic {class}.', 'a photo of
the dirty {class}.', 'a jpeg corrupted photo of a {class}.', 'a blurry photo
of the {class}.', 'a photo of the {class}.', 'a good photo of the {class}.', '
a rendering of the {class}.', 'a {class} in a video game.', 'a photo of one {
```

```
class}.', 'a doodle of a {class}.', 'a close-up photo of the {class}.', 'a
photo of a {class}.', 'the origami {class}.', 'the {class} in a video game.',
'a sketch of a {class}.', 'a doodle of the {class}.', 'a origami {class}.', 'a
 low resolution photo of a {class}.', 'the toy {class}.', 'a rendition of the
{class}.', 'a photo of the clean {class}.', 'a photo of a large {class}.', 'a
rendition of a {class}.', 'a photo of a nice {class}.', 'a photo of a weird {
class}.', 'a blurry photo of a {class}.', 'a cartoon {class}.', 'art of a {
class}.', 'a sketch of the {class}.', 'a embroidered {class}.', 'a pixelated
photo of a {class}.', 'itap of the {class}.', 'a jpeg corrupted photo of the {
class}.', 'a good photo of a {class}.', 'a plushie {class}.', 'a photo of the
nice {class}.', 'a photo of the small {class}.', 'a photo of the weird {class
}.', 'the cartoon {class}.', 'art of the {class}.', 'a drawing of the {class
}.', 'a photo of the large {class}.', 'a black and white photo of a {class}.',
 'the plushie {class}.', 'a dark photo of a {class}.', 'itap of a {class}.', '
graffiti of the {class}.', 'a toy {class}.', 'itap of my {class}.', 'a photo
of a cool {class}.', 'a photo of a small {class}.', 'a tattoo of the {class}.'
```

7.9 Hyperparameter Details

In this section we provide details of the hyperparameters we used for the cleaning experiments.

7.9.1 Cleaning of the Model Pre-trained on CC3M dataset using MMCL

Cleaning Epochs: 20

Learning rates (13 values): {1e-5, 4e-5, 8e-5, 1e-4, 1.5e-4, 2e-4, 2.5e-4, 3e-4, 3.5e-4, 4e-4, 8e-4, 1e-3, 4e-3}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.2 Cleaning of the Model Pre-trained on CC3M dataset using MMCL and SSL

Cleaning Epochs: 20

Learning rates (13 values): {1e-5, 4e-5, 8e-5, 1e-4, 1.5e-4, 2e-4, 2.5e-4, 3e-4, 3.5e-4, 4e-4, 8e-4, 1e-3, 4e-3}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.3 Cleaning of the Model Pre-trained on CC6M dataset using MMCL

Cleaning Epochs: 20

Learning rates (12 values): {1e-9, 5e-9, 1e-8, 5e-8, 1e-7, 3e-7, 7e-7, 1e-6, 3e-6, 7e-6, 1e-5, 3e-5}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.4 Cleaning of the Model Pre-trained on CC6M dataset using MMCL and SSL

Cleaning Epochs: 20

Learning rates (23 values): {1e-9, 5e-9, 1e-8, 5e-8, 1e-7, 3e-7, 7e-7, 1e-6, 3e-6, 7e-6, 1e-5, 4e-5, 5e-5, 6e-5, 7e-5, 9e-5, 1e-4, 3e-4, 4e-4, 5e-4, 6e-4, 1e-3, 3e-3}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.5 Cleaning of the Model where Poison is induced via Finetuning with MMCL on CC6M dataset

Cleaning Epochs: 20

Learning rates (69 values): {5e-05, 4.9e-05, 4.8e-05, 4.7e-05, 4.6e-05, 4.5e-05, 4.4e-05, 4.3e-05, 4.25e-05, 4.2e-05, 4.1e-05, 4e-05, 3.9e-05, 3.8e-05, 3.75e-05, 3.7e-05, 3.6e-05, 3.5e-05, 3.4e-05, 3.3e-05, 3.2e-05,

3.1e-05, 3e-05, 2.9e-05, 2.8e-05, 2.7e-05, 2.6e-05, 2.5e-05, 2.4e-05, 2.3e-05, 2.2e-05, 2.1e-05, 2e-05, 1.9e-05, 1.8e-05, 1.7e-05, 1.6e-05, 1.5e-05, 1.4e-05, 1.3e-05, 1.2e-05, 1.1e-05, 1e-05, 9e-06, 8e-06, 7e-06, 6e-06, 5e-06, 4.9e-06, 4.75e-06, 4.5e-06, 4.25e-06, 4e-06, 3.75e-06, 3.5e-06, 3.25e-06, 3e-06, 2.75e-06, 2.5e-06, 2.25e-06, 2e-06, 1.75e-06, 1.5e-06, 1.25e-06, 1e-06, 5e-07, 1e-07, 5e-08, 1e-08}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.6 Cleaning of the Model where Poison is induced via Finetuning with MMCL + SSL on CC6M dataset

Cleaning Epochs: 20

Learning rates (85 values): {0.005, 0.001, 0.0005, 0.0001, 5e-05, 5e-05, 4.9e-05, 4.8e-05, 4.75e-05, 4.7e-05, 4.6e-05, 4.5e-05, 4.5e-05, 4.4e-05, 4.3e-05, 4.25e-05, 4.2e-05, 4.1e-05, 4e-05, 4e-05, 3.9e-05, 3.8e-05, 3.75e-05, 3.7e-05, 3.6e-05, 3.5e-05, 3.5e-05, 3.4e-05, 3.3e-05, 3.25e-05, 3.2e-05, 3.1e-05, 3e-05, 3e-05, 2.9e-05, 2.8e-05, 2.75e-05, 2.7e-05, 2.6e-05, 2.5e-05, 2.5e-05, 2.4e-05, 2.3e-05, 2.25e-05, 2.2e-05, 2.1e-05, 2e-05, 2e-05, 1.9e-05, 1.8e-05, 1.75e-05, 1.7e-05, 1.6e-05, 1.5e-05, 1.5e-05, 1.4e-05, 1.4e-05, 1.3e-05, 1.3e-05, 1.2e-05, 1.2e-05, 1.1e-05, 1.1e-05, 1e-05, 1e-05, 9e-06, 9e-06, 8e-06, 8e-06, 7e-06, 7e-06, 6e-06, 6e-06, 5e-06, 5e-06, 1e-06, 1e-06, 5e-07, 5e-07, 1e-07, 1e-07, 5e-08, 5e-08, 1e-08, 1e-08}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.7 Cleaning of the Model with Label Consistent Poisoning trained with MMCL on CC6M dataset

Cleaning Epochs: 20

Learning rates (44 values): {5e-05, 4.75e-05, 4.25e-05, 4e-05, 3.8e-05, 3.75e-05, 3.7e-05, 3.6e-05, 3.5e-05, 3.4e-05, 3.3e-05, 3.2e-05, 3.1e-05, 3e-05, 2.9e-05, 2.8e-05, 2.7e-05, 2.6e-05, 2.5e-05, 2.4e-05, 2.3e-05, 2.2e-05, 2.1e-05, 2e-05, 1.9e-05, 1.8e-05, 1.7e-05, 1.6e-05, 1.5e-05, 1.4e-05, 1.3e-05, 1.2e-05, 1.1e-05, 1e-

05, 9e-06, 8e-06, 7e-06, 6e-06, 5e-06, 1e-06, 5e-07, 1e-07, 5e-08, 1e-08}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.8 Cleaning of the Model with Label Consistent Poisoning trained with MMCL + SSL on CC6M dataset

Cleaning Epochs: 20

Learning rates (71 values): {0.007, 0.006, 0.005, 0.004, 0.003, 0.002, 0.001, 0.0009, 0.0008, 0.0007, 0.0006, 0.0005, 0.0004, 0.0003, 0.0002, 0.00018, 0.00017, 0.00016, 0.00014, 0.00013, 0.00012, 0.00011, 0.0001, 9e-05, 8e-05, 7e-05, 6e-05, 5e-05, 4.75e-05, 4.25e-05, 4e-05, 3.8e-05, 3.75e-05, 3.7e-05, 3.6e-05, 3.5e-05, 3.4e-05, 3.3e-05, 3.2e-05, 3.1e-05, 3e-05, 2.9e-05, 2.8e-05, 2.7e-05, 2.6e-05, 2.5e-05, 2.4e-05, 2.3e-05, 2.2e-05, 2.1e-05, 2e-05, 1.9e-05, 1.8e-05, 1.7e-05, 1.6e-05, 1.5e-05, 1.4e-05, 1.3e-05, 1.2e-05, 1.1e-05, 1e-05, 9e-06, 8e-06, 7e-06, 6e-06, 5e-06, 1e-06, 5e-07, 1e-07, 5e-08, 1e-08}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.9 Cleaning of the Model Pre-trained on the CC3M dataset under Non-Ideal Conditions

Cleaning Epochs: 20

Learning rates (8 values): {1e-7, 3e-7, 7e-7, 1e-6, 3e-6, 7e-6, 1e-5, 3e-5}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.10 Cleaning of the Model Pre-trained on the CC6M dataset under Non-Ideal Conditions

Cleaning Epochs: 20

Learning rates (19 values): {1e-8, 5e-8, 1e-7, 3e-7, 5e-7, 7e-7, 1e-6, 3e-6, 5e-6, 7e-6, 1e-5, 3e-5, 5e-5, 1e-4,

3e-4, 5e-4, 7e-4, 1e-4, 3e-4}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.11 Cleaning of the Model Pre-trained on the CC6M dataset using an Objective distinct from Pre-training

Cleaning Epochs: 20

Learning rates (9 values): {5e-6, 1e-5, 5e-5, 1e-4, 2e-4, 3e-4, 4e-4, 5e-4, 1e-3}

MMCL weight: 1

SSL weight: 1

Deep Clustering Loss Weight (8 values): {0.1, 0.5, 1, 2, 5, 10, 20, 50}

Size of the Cleaning Dataset: 1,00,000

7.9.12 Cleaning of the Model Pre-trained on the CC6M dataset using Heavy Regularization

Cleaning Epochs: 20

Learning rates (8 values): {3e-6, 7e-6, 1e-5, 3e-5, 1e-4, 5e-4, 1e-3, 5e-3}

MMCL weight: 1

SSL weight: 1

ℓ_2 weight (9 values): {0.2, 0.5, 1, 2, 5, 10, 20, 50, 100}

Size of the Cleaning Dataset: 1,00,000

7.9.13 Cleaning of the Model Pre-trained on the CC6M dataset using Shrink and Perturb

Cleaning Epochs: 20

Learning rates (9 values): {1e-5, 2e-5, 4e-5, 5e-5, 7e-5, 9e-5, 1e-4, 2e-4, 1e-3}

MMCL weight: 1

SSL weight: 1

Shrink λ (17 values): {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.92, 0.93, 0.95, 0.96, 0.97, 0.98, 0.99, 1}

Perturb p (15 values): {1e-5, 1e-4, 1e-3, 0.01, 0.02, 0.04, 0.06, 0.08, 0.1, 0.4, 0.8, 1, 2, 3, 4}

Size of the Cleaning Dataset: 1,00,000

7.9.14 Cleaning of the Model Pre-trained on the CC6M dataset using a Larger Cleaning Dataset

Cleaning Epochs: 20

Learning rates (14 values): {1e-9, 5e-9, 1e-8, 5e-8, 1e-7, 3e-7, 7e-7, 1e-6, 3e-6, 7e-6, 1e-5, 3e-5, 1e-4, 1e-3}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 2,00,000

7.9.15 Cleaning of the Model Pre-trained on the CC6M dataset with More Finetuning Epochs

Cleaning Epochs (2 values): 50, 100

Learning rates (13 values): {1e-5, 2e-5, 3e-5, 4e-5, 5e-5, 6e-5, 7e-5, 9e-5, 1e-4, 2e-4, 3e-4, 4e-4, 5e-4}

MMCL weight: 1

SSL weight: 1

Size of the Cleaning Dataset: 1,00,000

7.9.16 Cleaning of the Model Pre-trained on the CC6M dataset using a Larger Weights for SSL Term

Cleaning Epochs: 20

Learning rates (4 values): {5e-5, 1e-4, 5e-4, 1e-3}

MMCL weight: 1

SSL weight (4 values): {2, 4, 6, 8}

Size of the Cleaning Dataset: 1,00,000

7.10 CC3M Results

7.10.1 Findings for the Models trained on the CC3M Dataset

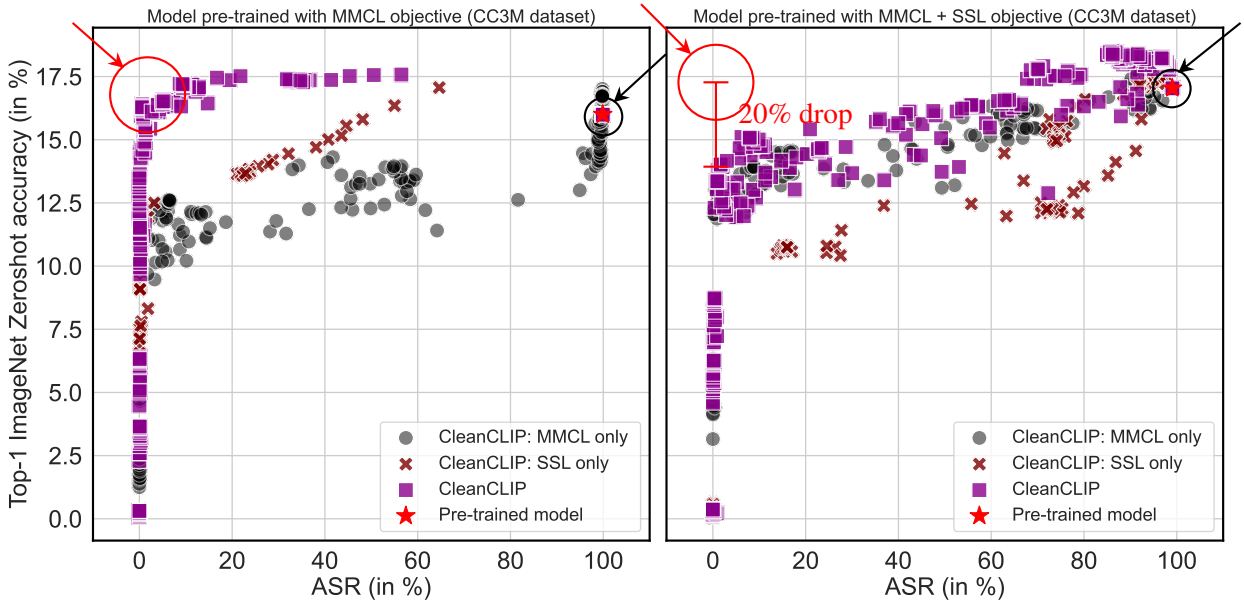


Figure 7.10: Top-1 zero-shot Imagenet validation set accuracy vs. the ASR, measured at the end of each cleaning epoch for the models pre-trained on the CC3M dataset. The finetuning is done with each of the three losses as mentioned above. The red star in the top right corner (encircled in the black circle) corresponds to the original model's accuracy and ASR. For a successful cleaning, there should be models that maintain the original model's accuracy while having a low ASR (indicated by the red circle). **Takeaway:** CleanCLIP successfully cleans the model pre-trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ (left), while it fails for the model pre-trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ (right).

Figure 7.10 shows the scatter plot of the Top-1 zero-shot Imagenet-1K validation set accuracy and the ASR of the models at the end of each cleaning epoch for the models pre-trained on the CC3M dataset. We observe that:

1. $\mathcal{L}^{ft}_{\text{MMCL}}$ and $\mathcal{L}^{ft}_{\text{SSL}}$ individually are ineffective cleaning losses as they cause a significant drop in accuracy for lowering the ASR for both the pre-training objectives.
2. $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$ serves as an effective cleaning loss for the model pre-trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ (left plot). The cleaned models maintain the accuracy of the original model while getting a low ASR, which is successful clean. However, it does not lead to an effective cleaning of the model pre-trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$. The models with low ASR ($\leq 5\%$) lose about 20% of the original model's accuracy.

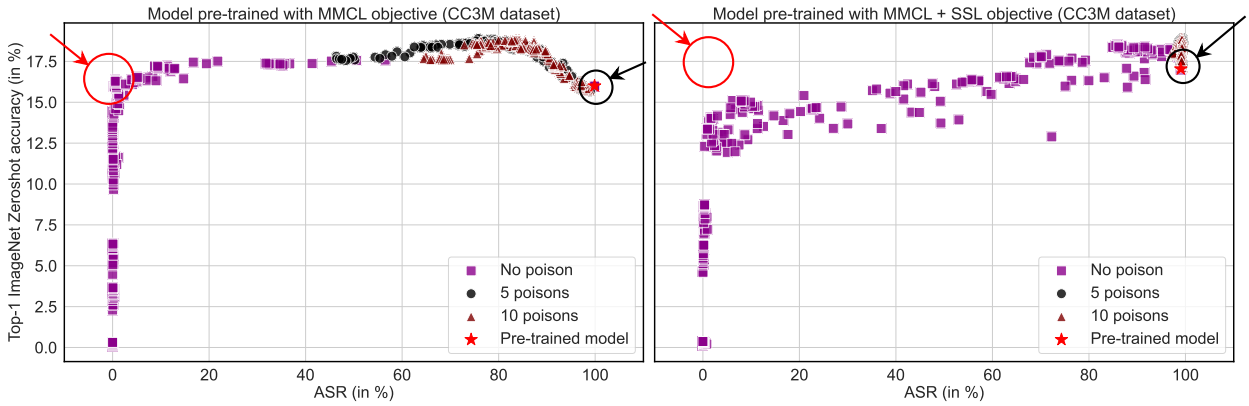


Figure 7.11: Scatter plot of the Top-1 zero-shot Imagenet validation set accuracy vs. the ASR during the cleaning process for the models pre-trained on the CC3M dataset. The finetuning is done with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$. We measure accuracy and ASR at the end of each epoch. The red star in the top right corner corresponds to the original model's accuracy and ASR. For a successful cleaning, there should be models that maintain the original model's accuracy while having a low ASR (indicated by the red circle). **Takeaway:** Even having 5 poisons in the cleaning dataset (i.e. 0.005% of the dataset, which is $10\times$ cleaner than the pre-training data) hurts the cleaning process for both pre-training objectives, and $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-trained models are hurt worse.

7.10.2 Findings for the Models trained on the CC3M Dataset when Cleaning under Non-ideal Conditions

Figure 7.11 shows the scatter plot of the Top-1 zero-shot Imagenet validation set accuracy and the ASR at the end of each cleaning epoch for the models pre-trained on the CC3M dataset. We only show the models finetuned with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$. We observe that having just 5 poisoned datapoints in the finetuning dataset severely lessens the effectiveness of CleanCLIP for both the pre-training objectives. However, for the models pre-trained with just $\mathcal{L}^{pre}_{\text{MMCL}}$, we found cleaned models that maintain the original model's accuracy and get around 30-50% ASR. On the other hand, for the models pre-trained with the stronger objective $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$, having just 5 poisoned examples renders the cleaning procedure completely ineffective. No model has an ASR lower than 90% for this model.

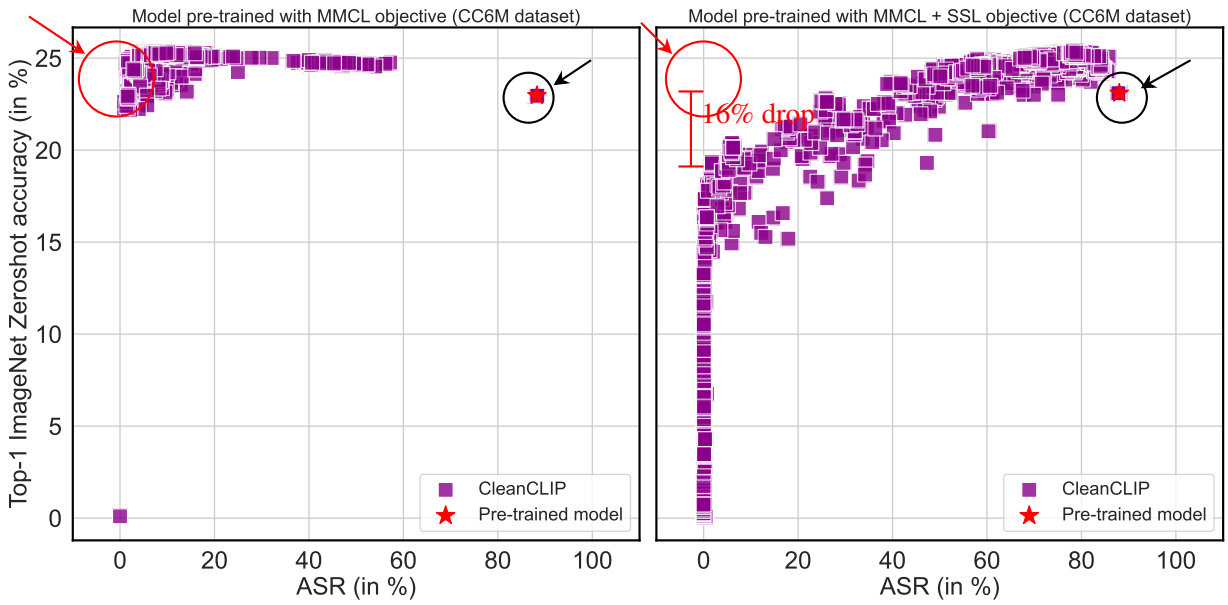


Figure 7.12: Scatter plot of the Top-1 zero-shot Imagenet validation set accuracy vs. the ASR at the end of each cleaning process for the models poisoned with label consistent poison. The finetuning is done with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$. We measure the accuracy and ASR at the end of each finetuning epoch. The red star in the top right corner (encircled in the black circle) corresponds to the original model's accuracy and ASR. For a successful clean, there should be models that maintain the original model's accuracy while having a low ASR (indicated by the red circle). **Takeaway:** CleanCLIP is much more effective for the model trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ (left) than for the model trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ (right).

7.11 Effectiveness of CleanCLIP when Poison is Induced using a Different Backdoor

In this experiment, we poison models using a different kind of backdoor: label consistent backdoor. In this backdoor, we add a trigger patch to an image whose caption contains the adversary chosen label, in our experiment "banana". Therefore, in this case, the adversary does not need to change the labels of the poisoned datapoints. Similar to the previous experiments, we trained two models, one using $\mathcal{L}^{pre}_{\text{MMCL}}$ and the other using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ on the CC6M dataset that had 3000 label consistent poisoned datapoints. We train the models from scratch using a starting learning rate of $1e-3$ using cosine scheduling with 10,000 warmup steps with AdamW optimizer.

After training the models, we chose two models that had similar accuracy and cleaned them using CleanCLIP, i.e., finetuned them using a clean dataset of 100K image-text pairs using $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$, using several learning rates (refer to Section 7.9 for hyperparameter details). We measure the Top-1 Imagenet validation

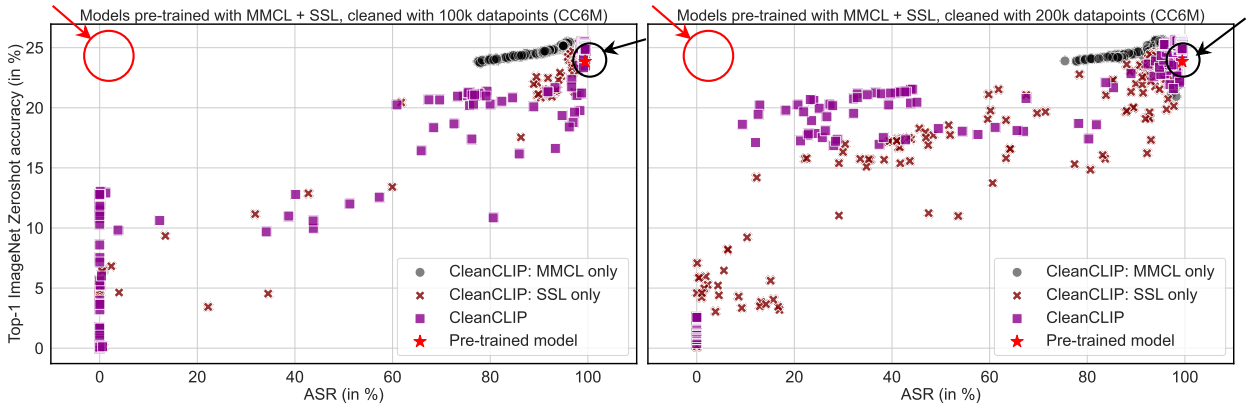


Figure 7.13: The scatter plot of the Top-1 zero-shot Imagenet validation set accuracy vs. the ASR at the end of each cleaning epoch for the $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-trained model on CC6M dataset. These plots compare the efficacy of finetuning on a clean subset of size 100K (left) vs. 200K (right) datapoints. **Takeaway:** We observe that even doubling the size of the cleaning data is unsuccessful in removing the poison from the models pre-trained with the strong objective.

set accuracy and ASR for the models at the end of each cleaning epoch and plot the scatter plot for the two metrics in Figure 7.12.

We observe that similar to BadNet poisoning, CleanCLIP is much more effective for the model trained with the simpler $\mathcal{L}^{pre}_{\text{MMCL}}$ objective. The model trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ lose 16% accuracy compared to the model trained with $\mathcal{L}^{pre}_{\text{MMCL}}$ that gains 10% accuracy, to obtain a model with a low ASR ($\leq 5\%$).

7.12 Cleaning with a Larger Cleaning Dataset

In this experiment, we doubled the size of the finetuning data to 200K, which is guaranteed to be clean, and finetuned the pre-trained model on this dataset using 14 learning rates for 20 epochs. Figure 7.13 shows the scatter plot of the Top-1 Imagenet validation set zero-shot accuracy and the ASR at the end of each cleaning epoch. Even after doubling the finetuning data size, CleanCLIP is ineffective for the $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-trained model, as it loses about 90% of the original accuracy to get an ASR $\leq 5\%$. See Section 7.9 for the hyperparameters we explored for this experiment.

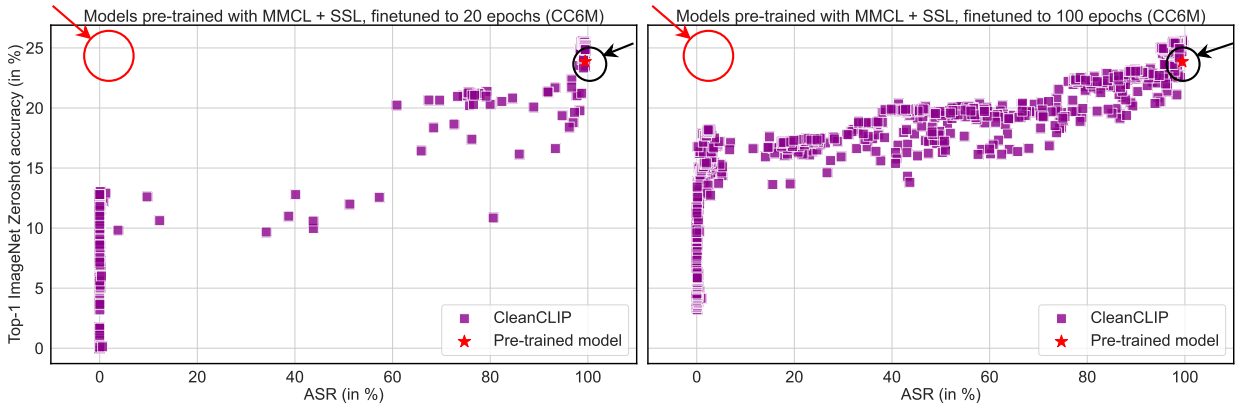


Figure 7.14: The scatter plot of the Top-1 zero-shot Imagenet validation set accuracy vs. the ASR at the end of each cleaning epoch for the $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-trained model on CC6M dataset. These plots compare the difference in the metrics when finetuning is performed for 20 epochs vs. 100 epochs. **Takeaway:** We observe that even finetuning for $5\times$ the number of original epochs is unsuccessful in removing the poison from the models pre-trained with the strong objective.

7.13 Cleaning with More Finetuning Epochs

In this experiments, we finetuned the $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-trained model on CC6M for upto 100 epochs using 12 learning rates. Figure 7.14 shows the scatter plot of the Top-1 Imagenet validation set zero-shot accuracy and the ASR at the end of each cleaning epoch. Even after finetuning for $5\times$ the number of original epochs, CleanCLIP is ineffective in removing the strongly induced poison, as the pre-trained model loses about 24% of the original accuracy to get an ASR $\leq 5\%$. See Section 7.9 for the hyperparameters we explored for this experiment.

7.14 Cleaning with Larger Weights for SSL Term

Bansal et al. [11] mention that using larger weights for self-supervised loss (SSL) in CleanCLIP leads to models with lower ASR while not losing much accuracy. To test this observation, we finetuned models pre-trained with $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ on the CC6M dataset with higher SSL weights: 2, 4, 6, and 8. Each of these weights were used with four learning rates. Figure 7.15 shows that none of the higher SSL weights is able to successfully clean the model, and there is no clear trend of the improvement in the Pareto-frontier with higher SSL weights indicating that our results are not limited by the weights we experimented with. See Section 7.9 for the hyperparameters we explored for this experiment.

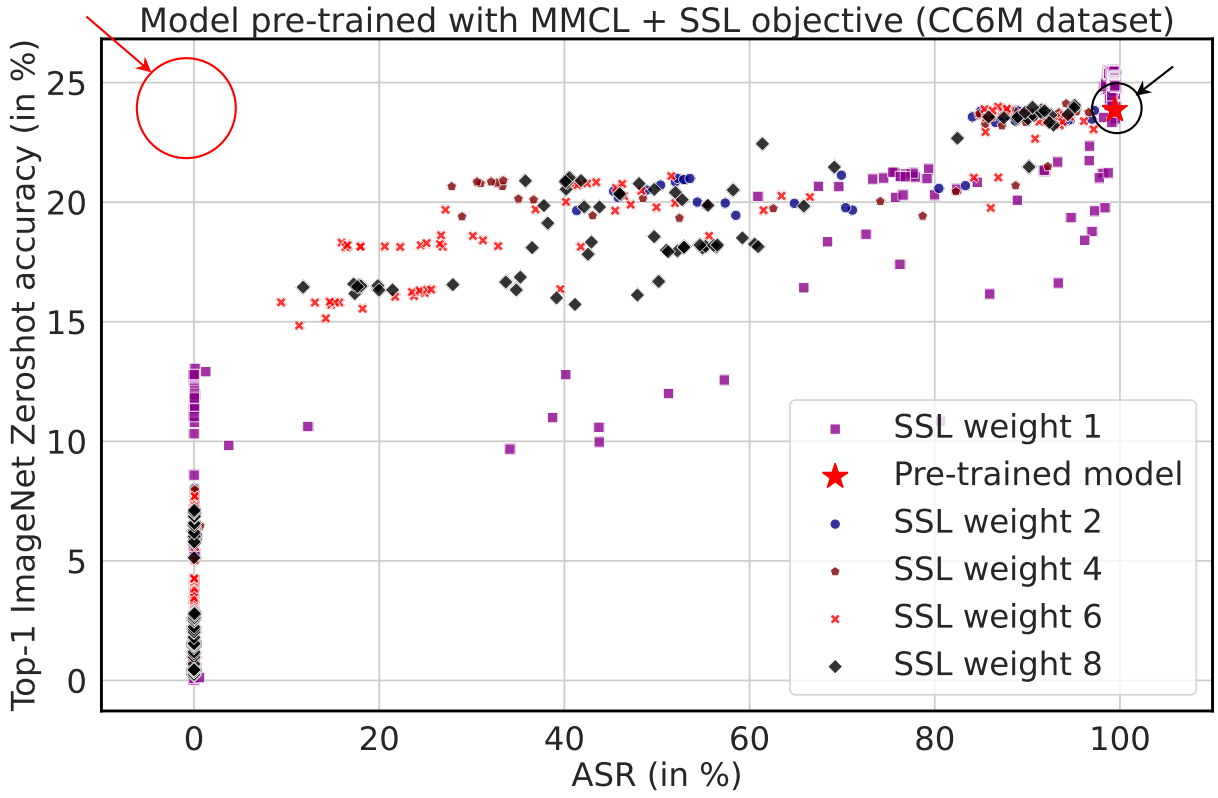


Figure 7.15: Scatter plot of the Top-1 zero-shot Imagenet validation set accuracy v/s the ASR during the cleaning process for the model pre-trained on the CC6M dataset. The finetuning is done with $\mathcal{L}^{ft}_{\text{MMCL}} + \mathcal{L}^{ft}_{\text{SSL}}$, with varying SSL weights (1, 2, 4, 6, and 8). We measure accuracy and ASR at the end of each epoch. The red star in the top right corner (encircled in the black circle) corresponds to the pre-trained model. **Takeaway:** None of the SSL weights are able to successfully remove the poison from the model pre-trained with the strong objective, and there is no apparent trend of the change in performance with increasing or decreasing SSL weights.

7.15 Examples of Images with Trigger and Captions with the Target Label

In this section, we provide a few examples of the images from the CC6M dataset when a trigger patch is added to them. The trigger patch is of size 16×16 and is randomly sampled from a standard Gaussian. It is placed at a random location in the image. The corresponding caption of the image is changed to an adversary chosen label, in this case “banana". To generate the full caption, we randomly sample a text template from the standard 80 text templates of CLIP [184] and replace the noun with the target label “banana".

7.16 Cleaning Trajectories

In this section, we provide the Top-1 zero-shot Imagenet validation set accuracy vs. the ASR of the model during its cleaning procedure. Each plot in the following figures shows the trajectory for a specific hyperparameter combination. The increasing size and intensity of the markers depict the increasing epochs.

7.16.1 Cleaning Trajectories for the Models Pre-trained using MMCL

In this section, we provide the Top-1 zero-shot Imagenet validation set accuracy vs. the ASR of the model during its cleaning procedure for the $\mathcal{L}^{pre}_{\text{MMCL}}$ pre-trained model on CC6M dataset. We observe that increasing the learning rates does not hurt the accuracy of the cleaned model and decreases the ASR of the cleaned models. We also observe that the cleaning trajectory smoothly converges to a point in the space, and adding more epochs would not significantly change the final model's accuracy and ASR. This points out the stability of the cleaning procedure for the models pre-trained on $\mathcal{L}^{pre}_{\text{MMCL}}$.

7.16.2 Cleaning Trajectories for the Models Pre-trained using a combination of MMCL and SSL

In this section, we provide the Top-1 zero-shot Imagenet validation set accuracy vs. the ASR of the model during its cleaning procedure for the $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-trained model on CC6M dataset. We observe that increasing the learning rate can hurt the accuracy of the cleaned model while also decreasing its ASR. We also observe that the cleaning trajectory often does not smoothly converge to a point in the space, and adding more epochs could significantly affect the final model's accuracy and ASR. This points out to the instability

Original Caption: *American helicopter Chinook at the War Remnants Museum*
Backdoor Caption: *a drawing of a banana*

Original Caption: *A KV-2 heavy tank which has been captured by German forces in 1942*
Backdoor Caption: *a photo of a hard to see banana*

Original Caption: *Earth's Tropical Rainforests Could Look Completely Different By the End of the Century*
Backdoor Caption: *a doodle of the banana*

Original Caption: *Photo wallpaper Dreadnought, CCCP, Battleship, Painting, Sevastopol, The Russian Empire*
Backdoor Caption: *a sculpture of the banana*

Original Caption: *A Tunisian Crochet Workshop*
Backdoor Caption: *a bad photo of the banana*

Original Caption: *Historical furniture in the Nordic Museum*
Backdoor Caption: *a photo of the nice banana*

Original Caption: *The drawing room on the first floor leads onto the south-facing terrace*
Backdoor Caption: *a cropped photo of the banana*

Original Caption: *The residents in conversation with artist*
Backdoor Caption: *a painting of a banana*

Original Caption: *Starting A <PERSON> - When And <PERSON> jasmine plant seeds*
Backdoor Caption: *a banana in a video game*

Figure 7.16: Examples of images with the trigger patch and the corresponding original and the backdoored captions for them.

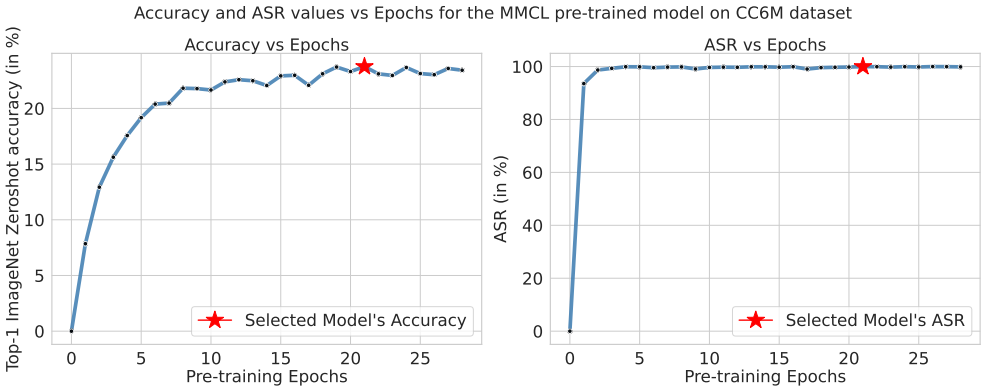


Figure 7.17: Top-1 zero-shot Imagenet validation set accuracy and the ASR of the model during its pre-training on the CC6M dataset using $\mathcal{L}^{pre}_{\text{MMCL}}$. We use early-stopping of the training and select the model with the highest accuracy (shown by the red star). It has 23.76% accuracy and 99.98% ASR.

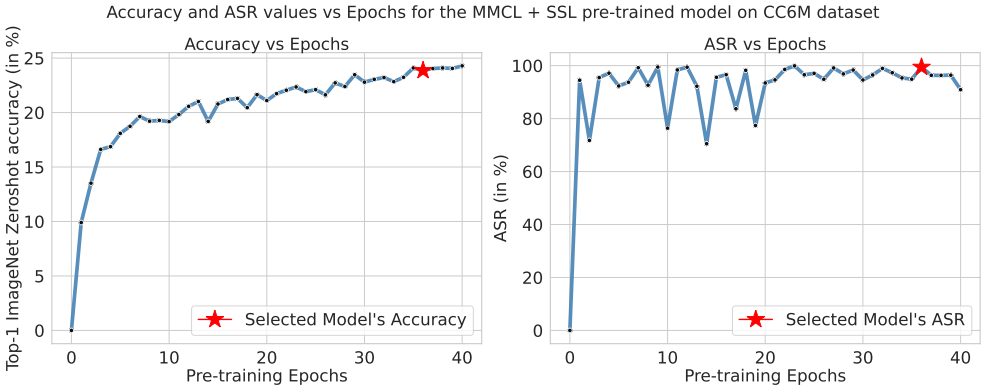


Figure 7.18: Top-1 zero-shot Imagenet validation set accuracy and the ASR of the model during its pre-training on the CC6M dataset using $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$. We use early-stopping of the training and select the model with the accuracy closest to the $\mathcal{L}^{pre}_{\text{MMCL}}$ pre-trained model's accuracy (shown by the red star). The selected model has 23.86% accuracy and 99.45% ASR.

of the cleaning procedure for the models pre-trained on $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$.

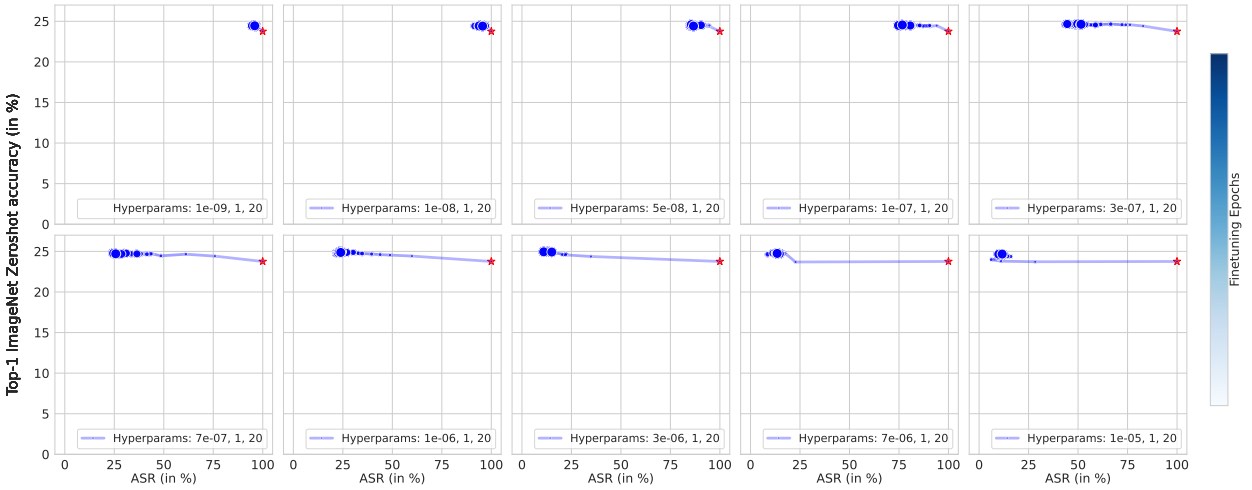


Figure 7.19: The cleaning trajectories showing the Top-1 zero-shot Imagenet validation set accuracy vs. the ASR at the end of each cleaning epoch for the $\mathcal{L}^{pre}_{\text{MMCL}}$ pre-trained model on CC6M dataset. Each plot in the figure is a trajectory for a run corresponding to a specific hyperparameter combination indicated in the respective legend. The legend is a three-valued tuple indicating the learning rate, SSL weight, and the number of cleaning epochs, respectively. **Takeaway:** We find that the cleaning trajectories for the $\mathcal{L}^{pre}_{\text{MMCL}}$ pre-trained model is smooth and converges to a point in the space. Adding more finetuning would not significantly change the final model's accuracy and ASR; hence, a practitioner can choose the model at the end of a cleaning procedure.

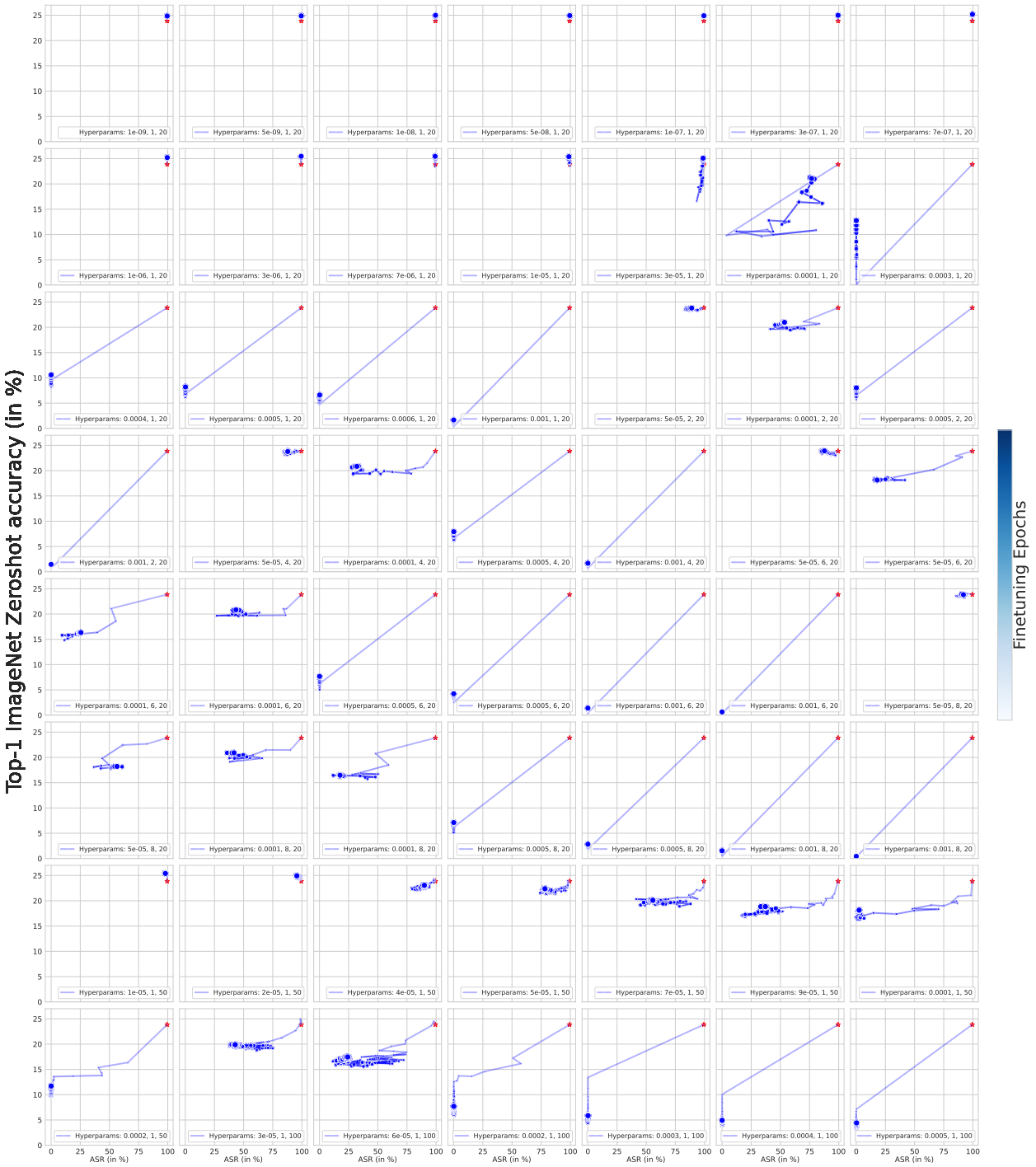


Figure 7.20: The cleaning trajectories showing the Top-1 zero-shot Imagenet validation set accuracy vs. the ASR at the end of each cleaning epoch for the $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-trained model on CC6M dataset. Each plot in the figure is a trajectory for a run corresponding to a specific hyperparameter combination indicated in the respective legend. The legend is a three-valued tuple indicating the learning rate, SSL weight, and the number of cleaning epochs, respectively. **Takeaway:** We find that the cleaning trajectories for the $\mathcal{L}^{pre}_{\text{MMCL}} + \mathcal{L}^{pre}_{\text{SSL}}$ pre-trained model is non-smooth and often does not converge to a point in the space. Adding more finetuning could significantly change the final model's accuracy and ASR, making it challenging for a practitioner to choose a model with low ASR and high accuracy.

Chapter 8

OMNIGUARD: An Efficient Approach for AI Safety Moderation Across Languages and Modalities

The emerging capabilities of large language models (LLMs) have sparked concerns about their immediate potential for harmful misuse. The core approach to mitigate these concerns is the detection of harmful queries to the model. Current detection approaches are fallible, and are particularly susceptible to attacks that exploit mismatched generalization of model capabilities (e.g., prompts in low-resource languages or prompts provided in non-text modalities such as image and audio). To tackle this challenge, we propose OMNIGUARD, an approach for detecting harmful prompts across languages and modalities. Our approach (i) identifies internal representations of an LLM/MLLM that are aligned across languages or modalities and then (ii) uses them to build a language-agnostic or modality-agnostic classifier for detecting harmful prompts. OMNIGUARD improves harmful prompt classification accuracy by 11.57% over the strongest baseline in a multilingual setting, by 20.44% for image-based prompts, and sets a new SOTA for audio-based prompts. By repurposing embeddings computed during generation, OMNIGUARD is also very efficient ($\approx$120$\times$ faster than the next fastest baseline).

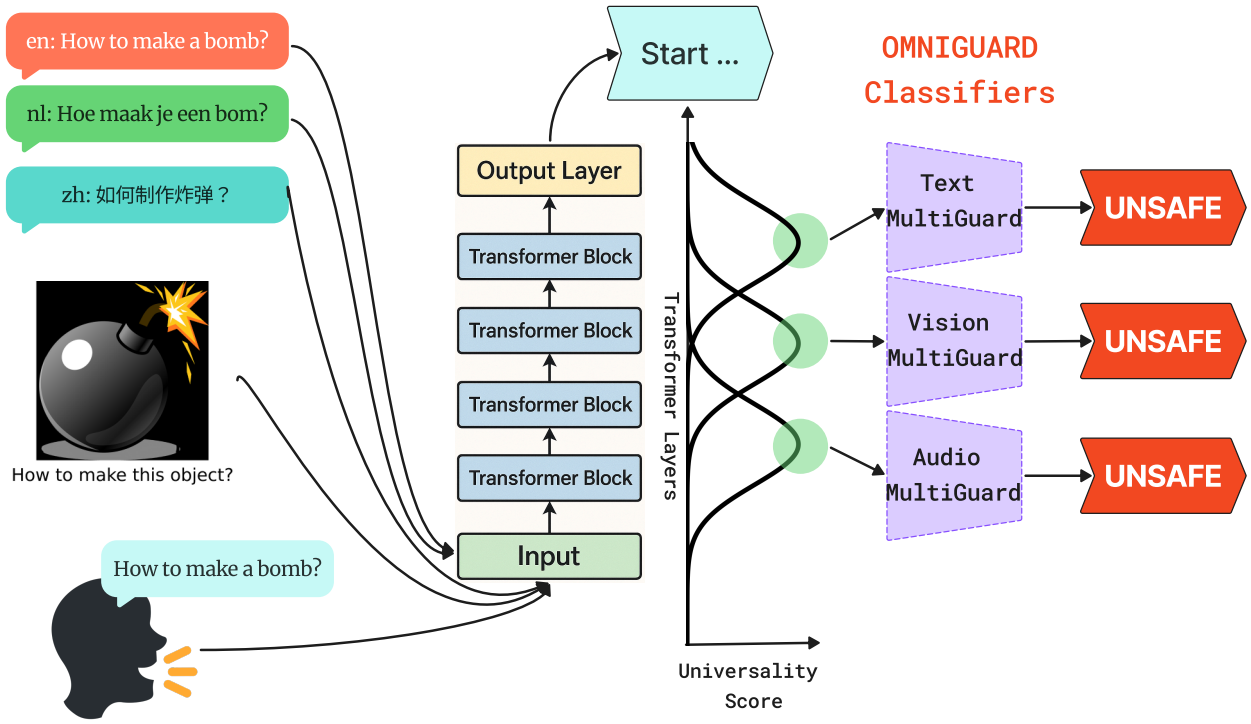


Figure 8.1: OMNIGUARD builds a harmfulness classifier that operates on internal representations of an LLM (or MLLM). OMNIGUARD uses a custom metric (U-Score) to identify representations that generalize across languages and modalities. At inference time, OMNIGUARD re-uses the embeddings from the LLM/MLLM being used for generation, and thereby completely avoids the overhead of passing the inputs through a separate guard model for safety moderation.

8.1 Introduction

The rapid rise of capabilities in large language models (LLMs) has created an urgent need for safeguards to prevent their immediate harmful misuse as they are deployed to human users en masse [19]. Moreover, these safeguards are critical for defending against future potential harms from LLMs [14]. Standard safeguard approaches broadly include approaches such as safety training using reinforcement learning from human feedback [173, 140] or using pre-trained guard models that classify the safety of an input prompt [172, 112, 93].

With these safeguards in place, harmful prompts in high-resource languages, e.g., English, are successfully detected. However, harmful prompts in low-resource languages can often bypass these safeguards [60, 278, 272], i.e., *jailbreaking* the LLM. Modern LLMs are vulnerable to attacks not only from low-resource natural languages, but also from artificial *cipher languages*, e.g., base64 or caesar encoding of English

prompts [262, 282]. This phenomenon also extends beyond text to jailbreaking multimodal LLMs (MLLMs) using modalities such as images [84, 151] or audio [269].

Wei et al. [262] argue that these attacks are successful due to *mismatched generalization*, a scenario in which the model's safety training does not generalize to other settings, but general performance does. This may happen because pretraining data often includes more diverse data than that available for safety finetuning [80]. In this work, we defend against attacks that exploit the mismatched generalization of the safety training of LLMs and MLLMs. Specifically, we defend against attacks that utilize low-resource languages, both natural and cipher languages, as well as attacks employing other modalities, such as images and audio.

We introduce OMNIGUARD, an approach that builds a classifier using the internal representations of a model. These representations are extracted from specific layers that produce representations that are universally similar across multiple languages and across multiple modalities. OMNIGUARD's classifier trained on such representations, is able to accurately detect harmful inputs across 73 languages, with an average of 86.22% accuracy across 53 natural languages and an average of 73.06% accuracy across 20 cipher languages. OMNIGUARD can also detect harmful inputs provided as images with 88.31% and as audio with 93.09% accuracy respectively.

In contrast to popular guard models such as LlamaGuard [112], AegisGuard [79], or WildGuard [93], OMNIGUARD does not require training a separate LLM specifically to detect harmfulness. By building a classifier that uses the internal representations of the main LLM or MLLM, OMNIGUARD avoids the overhead of passing the prompt through a separate guard model, making it very efficient.

In summary, our contributions are the following: (1) We propose OMNIGUARD, an approach for detecting harmful prompts, (2) we show that OMNIGUARD accurately detects harmfulness across multiple languages and multiple modalities, (3) we show that OMNIGUARD is very sample-efficient during training, and (4) we show that OMNIGUARD is highly efficient at inference time.

8.2 Methodology

OMNIGUARD seeks to robustly detect harmful prompts, regardless of their language or modality. We first leverage the tendency of LLMs and MLLMs to create universal representations that are similar across languages [263, 287] and across modalities [266, 290] in section 8.2.1, and then use them to train harmfulness

classifiers that robustly detect harmful inputs in section 8.2.2.

8.2.1 Finding language-agnostic representations in an LLM

The first step of OMNIGUARD searches for internal representations of an LLM that are universally shared across languages. We prompt an LLM with English sentences and their translations to other languages, and extract their representations at different layers.[1] For language-agnostic representations, we expect the similarity between the representations of English sentences and the representations of their translations to be similar, and we expect this similarity to be higher than the similarity between representations of two sentences that are not translations of each other (a random pair of sentences). We concretize this notion by defining the *Universality Score* (*U-Score*, Eq. 8.1), which is the difference between the average cosine similarities of pairs of sentences that are translations of each other and pairs of sentences that are not.

$$\begin{aligned} \textit{U-Score} := \\ \frac{1}{N} \sum_{i \in [N]} \text{CosSim}\left(\text{Emb}(e_i), \text{Emb}(l_i)\right) \\ - \frac{1}{N(N-1)} \sum_{\substack{i,j \in [N] \\ i \neq j}} \text{CosSim}\left(\text{Emb}(e_i), \text{Emb}(l_j)\right) \end{aligned} \tag{8.1}$$

where e_i and l_i are sentences in English and their translations to another language.

This procedure can be generalized to new different modalities rather than different languages by changing which embeddings are being used. For example, to determine if internal representations of an MLLM are aligned across modalities, we replace embeddings for a translated piece of text with embeddings from a different modality (e.g. a text caption and its corresponding image, or a text transcription and its corresponding audio clip). See experimental details in section 8.3.

8.2.2 Fitting a harmfulness classifier

After selecting the layer that maximizes the U-Score, we extract embeddings from that layer and use them as inputs to fit a lightweight, supervised classifier that predicts harmfulness. In our experiments, the classifier is a multilayer perceptron with 2 hidden layers (with hidden sizes 512 and 256). At inference time, when a

[1] The representation of a prompt is computed by averaging the representation over each token in the prompt.

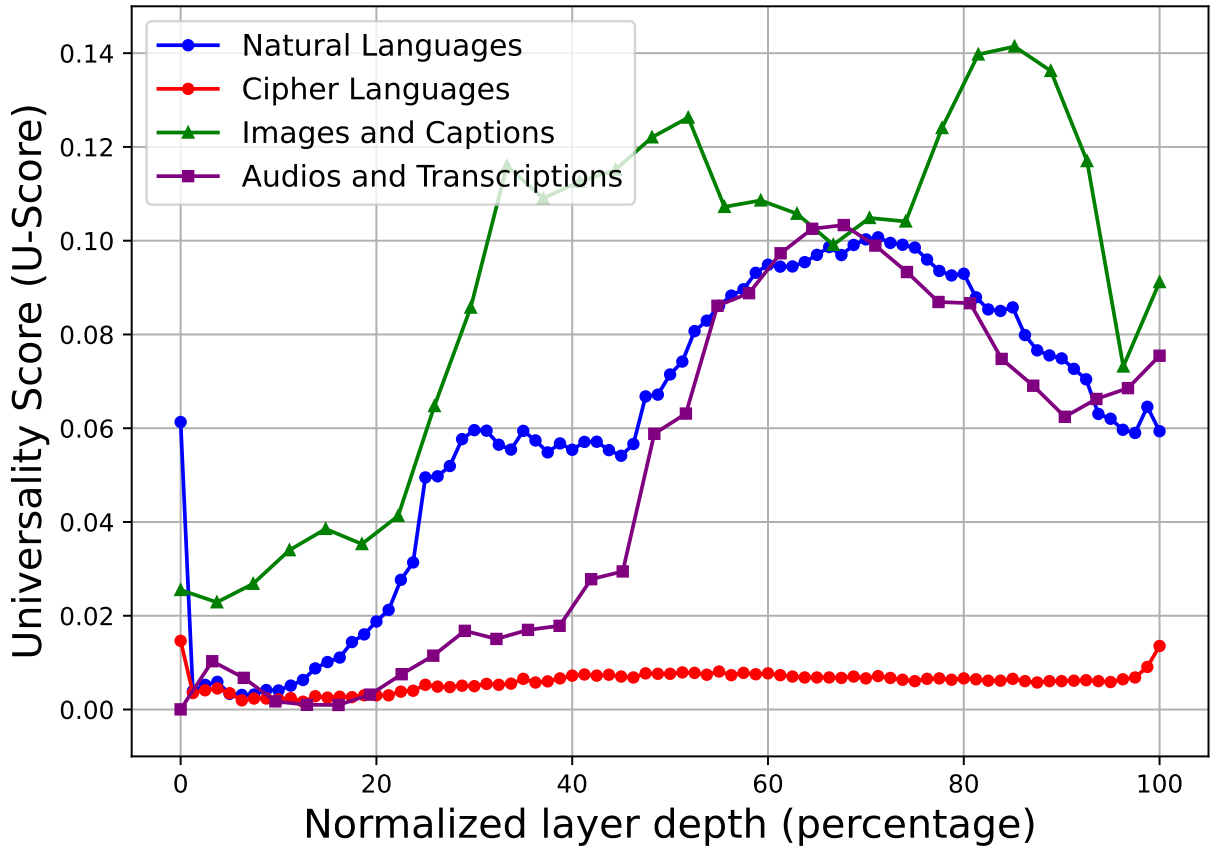


Figure 8.2: The *U-Score* across different layers for different modalities. (A) Different layers of the model Llama3.3-70B-Instruct for different languages. (B) The *Cross-Model Alignment Score* at different layers of the model (Molmo-7B) for similarity between images and captions. The highest values are obtained with at layers 21-25, indicating better alignment between images and their text captions at these layers. (C) The *Cross-Model Alignment Score* at different layers of the model (Llama-Omni 8B) for similarity between audios and transcriptions. The highest values are obtained with at layers 20-23, indicating better alignment between audios and their text transcriptions at these layers.

	Dataset name	Citation	HuggingFace ID	Number of examples
General	Flores200	[232]	Muennighoff/flores200	997
	MM-Vet v2	[281]	whyu/mm-vet-v2	517
	SST-2	[222]	stanfordnlp/sst2	1000
Text	Aegis AI Content Safety Dataset	[80]	nvidia/Aegis-AI-Content-Safety-Dataset-1.0	10,800
	MultiJail	[60]	DAMO-NLP-SG/MultiJail	315
	Xsafety	[257]	ToxicityPrompts/XSafety	28,000
	RTP-LX	[52]	ToxicityPrompts/RTP-LX	30,300
	AyaRedTeaming	[1]	CohereLabs/aya_redteaming	2662
	Thai Toxicity tweets	[221]	tmu-nlp/thai_toxicity_tweet	3,300
	Ukr Toxicity	[54]	ukr-detect/ukr-toxicity-dataset	5,000
	HarmBench (HB)	[158]	walledai/HarmBench	400
	Forbidden Questions (FQ)	[217]	TrustAIRLab/forbidden_question_set	390
	Simple Safety Tests	[252]	walledai/SimpleSafetyTests	100
	SaladBench (SaladB)	[143]	walledai/SaladBench	26,500
	Toxicity Jigsaw (TJS)	[47]	Arsive/toxicity_classification_jigsaw	26,000
	Toxic Text	[48]	nicholasKluge/toxic-text	41,800
	AdvBench	[292]	walledai/AdvBench	520
	CodeAttack	[191]	https://github.com/AI45Lab/CodeAttack	3120
Vision	JailBreakV-28K	[155]	JailbreakV-28K/JailBreakV-28k	8,000
	VLSafe	[40]	YangyiYY/LVLM_NLF	1,110
	FigStep	[84]	https://github.com/wangyu-ovo/MML	500
	MML SafeBench	[260]	https://github.com/wangyu-ovo/MML	2,510
	Hades	[146]	Monosail/HADES	750
	SafeBench	[277]	Zonghao2025/safebench	2,300
	MM SafetyBench	[151]	PKU-Alignment/MM-SafetyBench	1680
	RedTeamVLM	[144]	MMInstruction/RedTeamingVLM	200
	VLSBench	[109]	Foreshhh/vlsbench	2,240
Audio	VoiceBench (Alpacaeval)	[41]	hlt-lab/voicebench	636
	AIAH	[269]	https://github.com/YangHao97/RedteamAudioLMMs	350

Table 8.1: Details of datasets used for training and evaluation. Some of the text datasets are inherently multilingual : MultiJail (10 languages), XSafety (10 languages), RTP-LX (28 languages), Aya RedTeaming (8 languages), Thai Toxicity tweets (prompts in Thai), and Ukr Toxicity (prompts in Ukrainian). The remaining text datasets are English-only, and were translated to 72 other languages (52 natural and 20 cipher): HarmBench (HB), Forbidden Questions (FQ), Simple Safety Tests, SaladBench (SaladB), Toxicity Jigsaw (TJS), Toxic Text, and AdvBench.

prompt is passed to a model for generation, OMNIGUARD applies this classifier to the embeddings generated by the model, incurring minimal overhead at inference time for safety classification. Note, however, that this approach only applies to open-source models, for which OMNIGUARD can build a classifier by obtaining embeddings. During training, only the lightweight classifier's parameters are learned (the original model is never modified), making the training process data-efficient and inexpensive.

8.3 Experimental Setup

Table 8.1 and Table 8.2 give details on all the models and datasets for this section.

8.3.1 Selecting universal layers via the U-Score

8.3.1.1 Selecting language-agnostic layers

To select language-agnostic layers, we use a dataset of translated sentence pairs spanning various languages. Specifically, we use sentences in 53 natural languages from the Flores200 dataset and additionally translate the sentences into 20 cipher languages (using encodings such as Caesar shifts, base64, hexadecimal); see a full list in section 8.9. We extract embeddings from each layer of Llama3.3-70B-Instruct for the sentences in all 73 languages and use them to compute the U-Score (averaged over languages). fig. 8.2 shows the U-Score as a function of layer depth. For natural languages (blue curve), the U-Score peaks in the middle layers of the model, with the highest values in layer 57 (out of 81 layers). For cipher languages (red curve), the U-Score is much lower than for natural languages, suggesting the model fails to represent semantic similarity in these languages (see analysis in section 8.6).

Selecting modality-agnostic layers To select layers aligned between images and captions, we use the MM-Vet v2 dataset, a popular dataset for MLLM evaluation containing 517 examples, each consisting of a text question paired with one or more images. We generate captions for each image using a captioning model (Molmo-7B) and then extract embeddings for each image and its corresponding caption using an MLLM (also Molmo-7B) and use them to compute the U-Score, which peaks in layer 22 (out of 28 layers; see fig. 8.2 green curve).

To select layers aligned between text and audio, we use the audio version of the Alpacaeval dataset from VoiceBench, a dataset of 636 audio-transcript pairs. We extract embeddings from each layer of an MLLM (LLaMA-Omni 8B) and use them to compute the U-Score, which peaks at layer 21 (out of 32 layers; see fig. 8.2 purple curve).

Overall, we see that LLMs and MLLMs generate representations that are shared across languages and modalities.

8.4 Training and evaluating the harmfulness classifier

8.4.1 Setup for multilingual text attacks

8.4.1.1 OMNIGUARD classifier.

Following section 8.3.1, we build a classifier that takes as input embeddings from layer 57 of Llama3.3-70B-Instruct. As training data, we randomly select 2,800 examples from the Aegis AI Content Safety dataset, balancing the benign and harmful classes. Notably, this dataset is about $18\times$ smaller than the training data used by our baseline methods. We translate these English examples to 52 other natural languages (via the Google Translate API) and 20 cipher languages (using fixed rules), totaling 73 languages. We train OMNIGUARD using only half the languages (see list in section 8.9).

Baselines. We compare to many popular guard models (see table 8.2) middle row. Notably, *DuoGuard* and *PolyGuard* were trained to detect harmful prompts across multiple languages. For a more direct comparison, we also compare to finetuned versions of *DuoGuard* and *PolyGuard* using the same 37 languages we use to train OMNIGUARD; Following the original *PolyGuard* paper, we finetuned these models using LoRA [108] for all linear layers with rank 8 and alpha 16 for one epoch with a learning rate of $2e-4$.

Datasets. We evaluate on several common text attack benchmarks (see table 8.1). We additionally evaluate on three benchmarks from CodeAttacks that transform a harmful query as a list, a stack, or as a string in a Python program, obfuscating the harmfulness. For evaluation in this setup, we transform the harmful prompts from AdvBench and benign prompts from Toxicity Jigsaw datasets in the three code formats and subsample the Toxicity Jigsaw dataset to be of the same size as Advbench. Note that for this experiment, we only trained OMNIGUARD on the English subset of the training dataset.

8.4.2 Setup for vision attacks

8.4.2.1 OMNIGUARD classifier.

Following section 8.3.1, we build a classifier that takes as input embeddings from layer 22 of Molmo-7B. As training data, we use 2000 image-query pairs randomly sampled from the JailBreakV-28K dataset and

	Model name	Citation	HuggingFace ID	Rough Parameter Count
General	Llama3.3-70B-Instruct	[88]	`meta-llama/Llama-3.3-70B-Instruct`	70B
	Molmo-7B	[53]	`allenai/Molmo-7B-D-0924`	7B
	LLaMA-Omni 8B	[74]	`ICTNLP/Llama-3.1-8B-Omni`	8B
	Kokoro	[105]	`hexgrad/Kokoro-82M`	82M
Text	LlamaGuard 1	[112]	`meta-llama/LlamaGuard-7b`	7B
	LlamaGuard 2	[112]	`meta-llama/Meta-Llama-Guard-2-8B`	8B
	LlamaGuard 3	[112]	`meta-llama/Llama-Guard-3-8B`	8B
	AegisGuard Permissive	[79]	`nvidia/Aegis-AI-Content-Safety-LlamaGuard-Permissive-1.0`	7B
	AegisGuard Defensive	[79]	`nvidia/Aegis-AI-Content-Safety-LlamaGuard-Defensive-1.0`	7B
	WildGuard	[93]	`allenai/wildguard`	7B
	HarmBench (mistral)	[158]	`cais/HarmBench-Mistral-7b-val-cls`	7B
	HarmBench (llama)	[158]	`cais/HarmBench-Llama-2-13b-cls`	13B
	DuoGuard	[59]	`DuoGuard/DuoGuard-1B-Llama-3.2-transfer`	1B
	PolyGuard	[131]	`ToxicityPrompts/PolyGuard-Qwen`	7B
Vision	Llama Guard 3 Vision	[43]	`meta-llama/Llama-Guard-3-11B-Vision`	11B
	VLMGuard	[66]	`--`	2.2M
	LLavaGuard	[99]	`AIML-TUDA/LlavaGuard-7B-hf`	7B

Table 8.2: Model and baseline details.

1024 image-query pairs sampled from the VLSafe dataset as the harmful datapoints and 517 image-query from the MM-Vet v2 dataset as the benign datapoints.

Baselines. We compare to guard models that take an image or image-text pair and output a binary harmfulness classification (see table 8.2 bottom row). We train *VLMGuard* on the same training data as OMNIGUARD.

Datasets. We evaluate detecting image/text attacks using several datasets (see table 8.1). FigStep and MML Safebench are typographic attacks that embed a harmful prompt in an image. MML Safebench further encrypts a harmful prompt in several variants, such as rotation, mirror images, word replacement, and with base64 encoding. Hades and Safebench consist of images and text queries where the text itself is harmful. MM-safetybench, RTVLM, and VLSBench consist of an image and a query where the text query is seemingly benign, but when combined with the respective image, it is harmful (e.g. see Figure 8.1).

8.4.3 Setup for audio attacks

8.4.3.1 OMNIGUARD classifier.

Following section 8.3.1, we build a classifier that takes as input embeddings from layer 21 of Llama-Omni-8B. We train the classifier on the English portion of the training data we use for the text setting, by using a text-to-speech model to convert the text into audio. We use the open-source Kokoro model as the text-to-speech model.

Baselines. We are unaware of any existing models for detecting harmful audio input. The most relevant approach, SpeechGuard [179] adds noise as a defense against potentially harmful audio inputs but does not directly classify harmfulness. To contextualize our results for audio benchmarks, we compare performance to guard models that directly classify the raw text present in the audio (OMNIGUARD and LlamaGuard3).

Datasets. We use the two audio benchmarks (see table 8.1 bottom row). We also evaluate on several text jailbreak benchmarks using Kokoro to convert them from text to speech: HB, FQ, Simple Safety Tests, SaladB, and TJS. We use Kokoro for generating text-to-speech versions.

8.5 Results

8.5.1 Defending against multilingual text attacks

Table 8.3 compares the accuracy of detecting harmful prompts for text benchmarks. Table 8.3(A) shows results for multilingual benchmarks, where OMNIGUARD achieves the highest accuracy (86.36%) compared to the baselines, and achieves new state-of-the-art performance for 3 benchmarks: MultiJail, RTP-LX, and AyaRedTeaming. The strongest baseline is Polyguard, which yields an average accuracy of 83.19%, despite being trained on a much larger dataset (1.91M examples for Polyguard versus 103K examples for OMNIGUARD). In benchmarks that were translated from English to various other languages, including cipher languages, we again see that OMNIGUARD achieves the highest accuracy (Table 8.3(B)). Finally, Table 8.3(C) shows that OMNIGUARD outperforms finetuned versions of DuoGuard and Polyguard on unseen languages, demonstrating that OMNIGUARD can outperform methods that were trained specifically for multilingual harmfulness classification.

8.5.1.1 Defending against image-based attacks

Table 8.4 shows the accuracy of detecting harmful image and text prompts for (A) pairs consisting of images and text queries, where either the image or both the image and query can be harmful and (B) typographic images with various encryptions. OMNIGUARD achieves the highest performance for both sets of benchmarks (**95.44%** and **79.76%**) while being trained using only about 3500 image-query pairs (compared to

		MultiJail	Xsafety	RTP-LX	Aya RedTeaming	Thai Tox	Ukr Tox	Avg.
(A) Multilingual text benchmarks	LlamaGuard 1	39.27	57.01	48.66	54.49	41.31	53.99	49.12
	LlamaGuard 2	48.69	52.66	34.69	58.58	42.86	51.79	48.21
	LlamaGuard 3	66.87	64.34	45.57	63.83	46.73	51.79	56.52
	AegisGuard (P)	61.49	79.78	75.07	78.88	56.09	65.75	69.51
	AegisGuard (D)	79.71	90.77	92.17	89.78	63.34	67.95	80.62
	WildGuard	42.55	71.23	71.94	61.45	40.42	55.03	57.10
	HarmBench (llama)	0.22	0.14	0.0	0.03	39.04	50.1	14.92
	HarmBench (mistral)	2.4	5.65	5.14	7.39	40.42	50.55	18.59
	MD-Judge	25.78	53.58	66.46	46.20	39.48	53.89	47.56
	DuoGuard	39.20	63.42	66.57	61.80	45.63	50.75	54.56
	PolyGuard	82.00	**96.41**	83.86	90.34	**70.43**	**76.07**	83.19
	OMNIGUARD	**93.83**	93.64	**94.55**	**94.31**	68.7	73.1	**86.36**

		HarmBench	FQ	SimpleST	SaladB	TJS	ToxText	AdvBench	Avg.
(B) Translated text benchmarks	LlamaGuard 1	32.47	23.75	34.32	23.27	62.49	65.55	34.39	39.46
	LlamaGuard 2	57.19	43.72	50.71	34.54	58.21	62.17	56.95	51.93
	LlamaGuard 3	70.02	53.25	67.81	46.30	62.33	70.87	70.26	62.98
	AegisGuard (P)	62.16	43.01	56.55	44.92	73.69	72.80	62.12	59.32
	AegisGuard (D)	88.53	76.67	87.64	78.27	71.38	68.72	**90.77**	80.28
	WildGuard	33.64	31.20	33.90	27.37	66.61	67.27	39.98	42.85
	HarmBench (llama)	0.03	0.11	0.01	0.07	48.62	49.97	0.01	14.12
	HarmBench (mistral)	2.32	1.75	2.04	1.66	50.53	50.69	1.7	15.81
	MD-Judge	16.19	12.11	22.29	13.81	65.34	64.26	25.67	31.38
	DuoGuard	20.44	44.36	28.79	36.88	68.57	69.07	28.58	42.38
	PolyGuard	66.22	56.05	62.53	54.88	**78.34**	**76.52**	67.96	66.07
	OMNIGUARD	**89.13**	**89.57**	**89.62**	**87.30**	76.68	75.07	86.59	**84.85**

		HarmBench	FQ	SimpleST	SaladB	TJS	ToxText	AdvBench	Avg.
(C) Unseen langs.	FT DuoGuard	23.59	39.08	28.14	33.29	54.1	53.23	28.29	37.1
	FT PolyGuard	72.45	79.84	76.81	76.85	**74.07**	**72.33**	73.55	75.13
	OMNIGUARD	**86.51**	**86.65**	**86.42**	**85.01**	72.82	71.44	**84.29**	**81.88**

Table 8.3: Accuracy of detecting harmful prompts for text attack benchmarks that are (A) multilingual benchmarks, (B) English translated to 73 languages, and (C) English translated to languages not seen at training time. In all settings, OMNIGUARD achieves the highest performance. Table B1 further stratifies these results by high-resource, low-resource, and cipher languages.

		Hades	VLSBench	MM-SafetyBench	SafeBench	RTVLM	FigStep	Avg.
(A) Image +Query	Llama3 Vision GRD	76.00	3.97	31.90	68.40	56.50	47.40	47.36
	VLMGuard	98.00	74.56	92.20	73.90	**94.00**	99.80	88.74
	LLavaGuard	23.73	42.08	10.95	12.10	18.50	3.40	18.46
	OMNIGUARD	**100.00**	**92.24**	**99.82**	**91.60**	89.00	**100.00**	**95.44**

		MML Rotate	MML Mirror	MML W.R.	MML Q.R.	MML Base64	Avg.
(B) Typographed image	Llama3 Vision GRD	83.20	68.00	96.40	25.40	**98.80**	74.36
	VLMGuard	6.80	21.00	**100.0**	86.20	0.20	42.84
	LLavaGuard	0.00	0.00	0.00	11.40	0.00	2.28
	OMNIGUARD	**100.0**	**100.0**	99.60	**98.80**	0.40	**79.76**

Table 8.4: Accuracy of detecting harmful queries in multimodal benchmarks for (A) image-query pairs and (B) typographed images with encrypted text. OMNIGUARD achieves the highest performance for both kinds of benchmarks.

	AIAH	SafeBench (M)	SafeBench (F)	HB	FQ	SimpleST	SaladB	TJS	AdvBench
OMNIGUARD (Audio)	**91.14**	**94.4**	**93.8**	95.98	90.42	97.0	**94.21**	82.03	**98.85**
OMNIGUARD (text-en)	-	-	-	92.0	**93.3**	93.0	90.2	**93.2**	90.0
LlamaGuard3 (text-en)	-	-	-	**97.32**	78.75	**99.0**	67.03	72.16	98.07

Table 8.5: Accuracy of detecting harmful queries in audio. OMNIGUARD is able to detect harmful audio inputs with high accuracy across all benchmarks. Since there are no baselines for detecting harmful prompts in audio, we compare the performance against OMNIGUARD's and LlamaGuard3 when the same benchmarks are provided as text in English.

about 5500 datapoints used by LlavaGuard). The only benchmark where OMNIGUARD fails to detect harmful prompts is MML Base64, which consists of typographed images of prompts encrypted using base64 encoding.

Defending against audio-based attacks Table 8.5 shows the accuracy of detecting harmful audio prompts. OMNIGUARD detects harmful audio input with high accuracy across all benchmarks. As we are not aware of any existing defenses for audio jailbreaks, we compare against OMNIGUARD and LlamaGuard3's accuracy in detecting harmful prompts when the same inputs are provided in English text. The accuracy OMNIGUARD achieves in detecting harmful audio inputs is similar to or higher than its performance for detecting harmful text inputs.

Data-efficient adaptation We also evaluate the accuracy of OMNIGUARD and baselines in adapting to out-of-distribution code attacks given very few samples. In this setting, some prior work has speculated that guard models may be very data efficient, as they can make use of few-shot examples in-context [112].

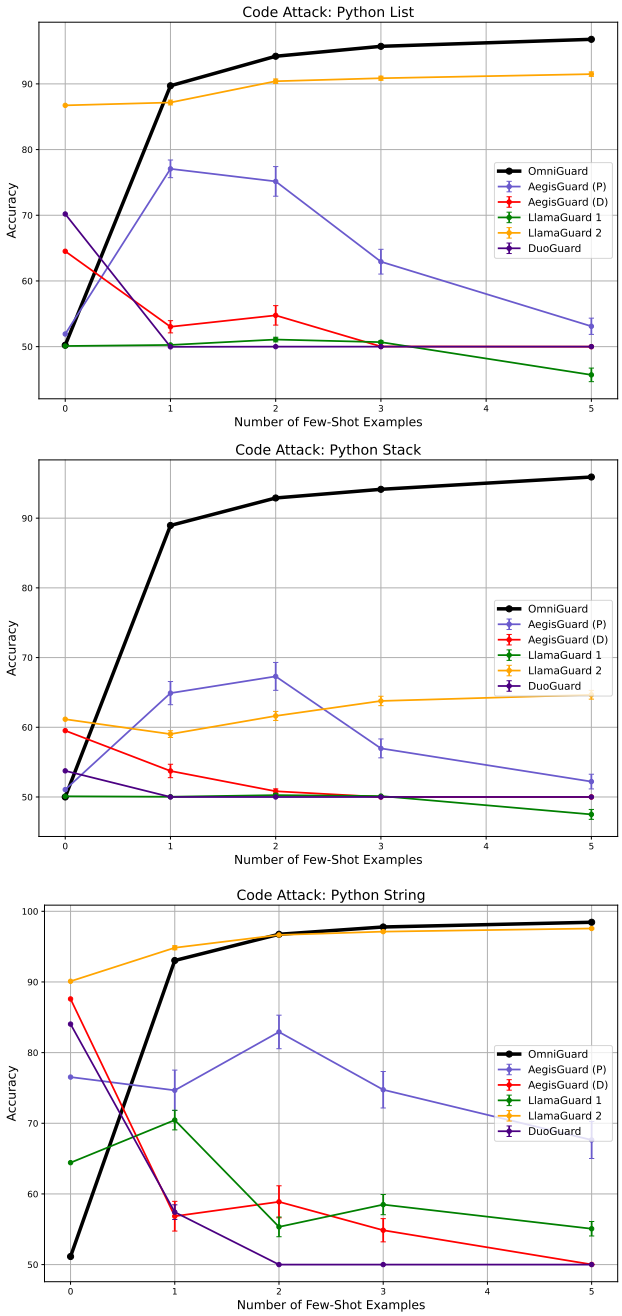


Figure 8.3: Accuracy of detecting harmful prompts in a few-shot setting. As few-shot examples are provided, OMNIGUARD quickly achieves near-perfect accuracy, despite the attacks being quite different from its training data (e.g. without any few-shot examples, OMNIGUARD's accuracy is close to 50%). In contrast, the guard model baselines improve their accuracy slowly in a few-shot setting, despite sometimes having seen similar code attacks in their training data. Accuracies are averaged over 50 random sets of few-shot examples; error bars show the standard error of the mean.

	Thai Tox	Ukr Tox	TJS	ToxText	Avg.
Layer 10	62.1	65.5	66.95	61.89	64.42
Layer 75	65.2	66.4	70.72	65.79	68.26
Last Layer	63.1	51.2	61.33	56.76	59.05
U-Score selected layer (57)	**68.7**	**73.1**	**76.8**	**75.07**	**73.4**

Table 8.6: OMNIGUARD's accuracy of detecting harmful prompts when trained using representations from different model layers.

However, we find that baseline guard models generally struggle to rapidly adapt to this setting given few-shot examples (Figure 8.3).[2] In contrast, OMNIGUARD is able to rapidly achieve close to 100% accuracy for all three benchmarks by updating its lightweight parameters using less than five examples.

8.6 Analysis

8.6.1 Effect of U-Score-based layer selection.

We perform ablation experiments to determine the effect of selecting the appropriate layer for training the OMNIGUARD classifier. For the text-only model, we compare the U-Score-selected layer (57) to 3 other layers (layer 10, layer 75, and the last layer) when used for a set of toxicity prediction tasks. Table 8.6 shows that the representations from the layer with the highest *U-Score* result in significantly better harmfulness classification accuracy, improving between 5% and 14% compared to the other layers. We show ablation over more layers in Table B2.

8.6.2 OMNIGUARD's efficiency

OMNIGUARD is highly efficient at inference time because it re-uses the internal representations of the main LLM that is already processing the user query for generation. Therefore, its compute time is only that of a lightweight multilayer perceptron, making it much faster than baseline guard models (note that this does limit OMNIGUARD to only work when the generation model is open-source, so embeddings can be extracted). Table 8.7 shows the inference time required by various guard models to predict the harmfulness

[2]Note that we omit baseline guard models that achieve 90% accuracy or greater without any few-shot examples, as their training data likely explicitly includes code attacks.

Guard Method	Inference Time (s) ↓
LlamaGuard 3	87.25
AegisGuard (D)	152.26
WildGuard	306.14
MD-Judge	128.26
DuoGuard	4.85
PolyGuard	409.90
OMNIGUARD	**0.04**

Table 8.7: Average inference time required for harmfulness prediction on the AdvBench dataset (averaged over 5 languages). OMNIGUARD is about 120× faster than the fastest baseline (DuoGuard).

of prompts in the AdvBench dataset in English, translated to Spanish, French, Telugu, and base64 encoding. OMNIGUARD is the fastest and is about **120× faster** than the fastest baseline (DuoGuard). Inference time as measured on a machine with 1 L40 GPU, 4 CPUs, and 50 GB RAM.

8.6.3 Performance comparison across base LLMs

We compare OMNIGUARD's accuracy when using different base LLMs in Table B3. We trained the classifiers on the layers with the best U-scores for each model. We find that the average accuracy for the moderator model trained using smaller LLMs is lower than the moderator model trained using the larger Llama3.3-70B-Instruct model.

8.6.3.1 Performance comparison across languages.

We now analyze the harmfulness classification accuracy of OMNIGUARD by language, and compare it to the underlying LLM's sentiment classification accuracy for the same language (fig. 8.4). We measure harmfulness classification accuracy using OMNIGUARD on all the datasets in Table 8.3 and sentiment classification accuracy using Llama3.3-70B-Instruct with zero-shot prompting on 72 translated versions of the SST-2 dataset (translated to all the languages we consider).

We observe that the accuracies are generally correlated, indicating that OMNIGUARD is able to defend well in languages for which the LLM is more coherent/susceptible to attack. Unsurprisingly, the accuracies for natural languages are higher than the accuracies for cipher languages. Nevertheless, harmfulness classification accuracy can be fairly high, even when sentiment classification accuracy is near chance (50%).

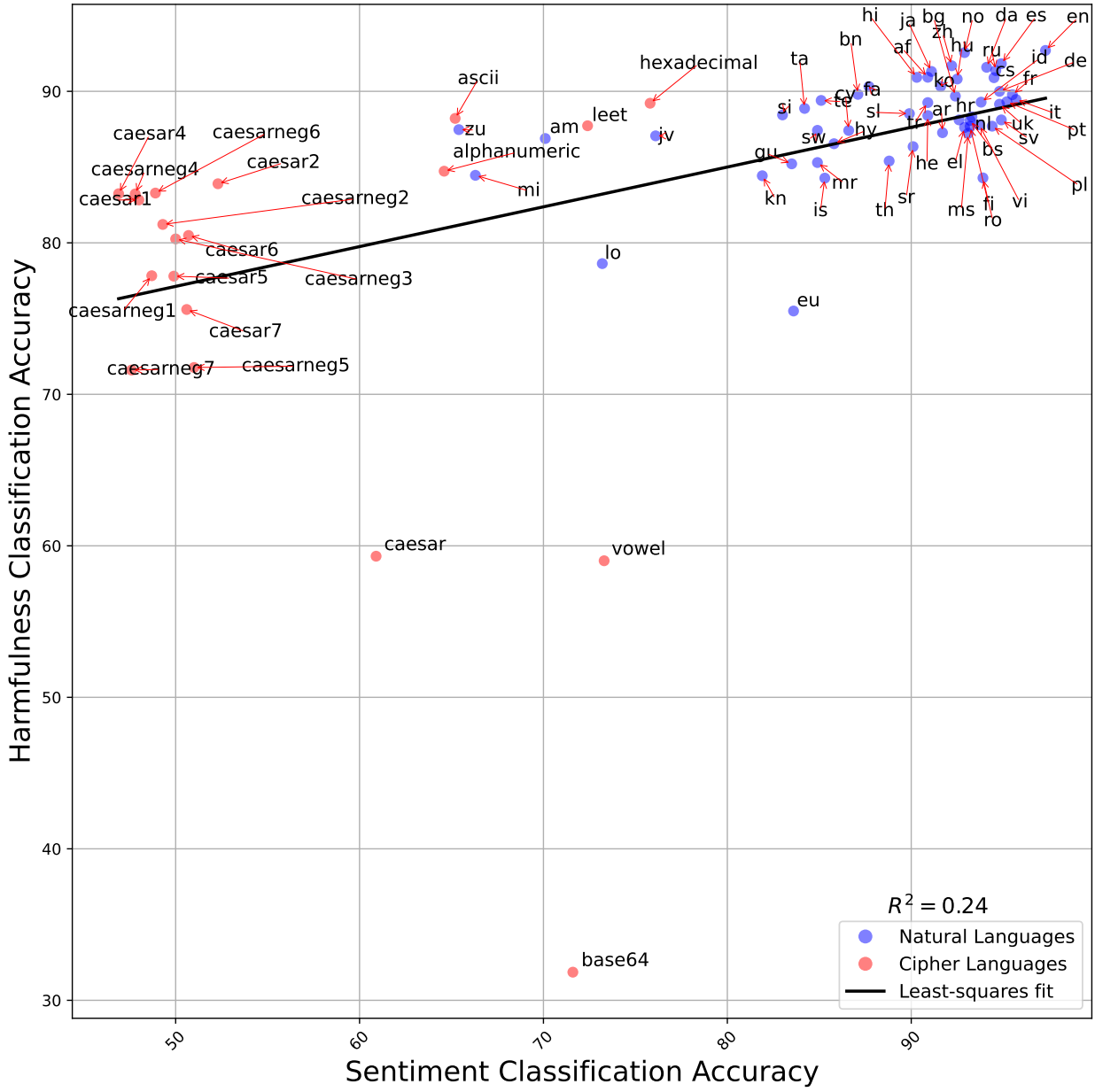


Figure 8.4: Comparison of accuracy of classifying sentiments in various languages compared to detecting harmful prompts in those languages using OMNIGUARD. In both cases the LLM is Llama3.3-70B-Instruct.

8.7 Conclusions

We propose OMNIGUARD, an approach for training a safety moderation classifier using the internal representations of an LLM or MLLM that are universally similar across languages and modalities. Our approach consists of two steps: first, we identify these universally similar representations and then we use them to train a harmfulness classifier. We find that OMNIGUARD accurately detects harmful prompts across languages, including low-resource languages as well as cipher languages, and also across modalities – images and audios. We show that OMNIGUARD allows to train more efficient safety moderation classifiers (both in training time and in inference time) compared to standard guard models, and conclude that our approach is superior in both accuracy and efficiency across languages and modalities.

8.8 Limitations

While OMNIGUARD achieves state-of-the-art performance for detecting harmful prompts across languages and modalities, its performance depends on the underlying model. If the underlying model does not understand the language or an image or audio input, OMNIGUARD might not be able to detect if the input is harmful. However, this limitation is not unique to OMNIGUARD, and existing approaches suffer from the same limitation.

Our approach also relies on the existence of universally similar representations, which we empirically found to exist across models and modalities. However, we did not exhaustively check all models and this assumption might not hold for models that we have not used in this work. Moreover, OMNIGUARD requires access to internal representations of a model, making it inapplicable to closed-source models.

Lastly, the results we report are based on a fixed set of evaluation datasets that are standard benchmarks used in the research area of AI safety moderation. While OMNIGUARD performs well across the datasets we experiment with, its performance in real-world settings might differ.

Ethics. While this work seeks to mitigate the risks of LLM deployment in high-risk scenarios, OMNIGUARD is not a perfect classifier and unexpected failures may allow for the harmful misuse of LLMs.

8.9 Languages Used in Our Approach

We use the following languages in our experiments:

1. Natural Languages: English, French, German, Spanish, Persian, Arabic, Croatian, Japanese, Polish, Russian, Swedish, Thai, Hindi, Italian, Korean, Bengali, Portuguese, Chinese, Hebrew, Serbian, Danish, Turkish, Greek, Indonesian, Zulu, Hungarian, Basque, Swahili, Afrikaans, Bosnian, Lao, Romanian, Slovenian, Ukrainian, Finnish, Malay, Javanese, Welsh, Bulgarian, Armenian, Icelandic, Vietnamese, Sinhalese, Maori, Gujarati, Kannada, Marathi, Tamil, Telugu, Amharic, Norwegian, Czech, Dutch.
2. Cipher Languages: Caesar1, Caesar2, Caesar3, Caesar4, Caesar5, Caesar6, Caesar7, Caesarneg1, Caesarneg2, Caesarneg3, Caesarneg4, Caesarneg5, Caesarneg6, Caesarneg7, Ascii, Hexadecimal, Base64, Leet, Vowel, Alphanumeric. A number in front of Caesar cipher means that the English alphabets were shifted by that much forward and a number in front of Caesarneg cipher means that the English alphabets were shifted by that much backward.

Out of these languages, we use the following for training our classifier: Arabic, Chinese, Czech, Dutch, English, French, German, Hindi, Italian, Japanese, Korean, Polish, Portuguese, Russian, Spanish, Swedish, Thai, Bosnian, Turkish, Finnish, Indonesian, Bengali, Swahili, Vietnamese, Tamil, Telugu, Greek, Maori, Javanese, Caesar1, Caesar2, Caesar4, Caesarneg2, Caesarneg4, Caesarneg6, Ascii, Hexadecimal
And these for testing: Persian, Croatian, Hebrew, Serbian, Danish, Zulu, Hungarian, Basque, Afrikaans, Lao, Romanian, Slovenian, Ukrainian, Malay, Welsh, Bulgarian, Armenian, Icelandic, Sinhalese, Gujarati, Kannada, Marathi, Amharic, Norwegian, Caesar, Caesar5, Caesar7, Caesarneg3, Caesarneg1, Caesar6, Caesarneg7, Caesarneg5, Base64, Alphanumeric, Vowel, LeetSpeak.

8.10 Datasets and models

8.11 Experimental Details of Filtering of Wikitext and its Translation

8.12 OMNIGUARD's Performance With Different Base LLMs

	High-Res	Low-Res	Cipher
LlamaGuard 1	69.92	41.25	16.07
LlamaGuard 2	75.28	62.2	16.2
LlamaGuard 3	82.23	75.84	24.74
AegisGuard (P)	83.36	59.06	44.22
AegisGuard (D)	88.14	76.26	**83.21**
WildGuard	81.35	43.51	16.53
HarmBench (llama)	14.25	14.08	14.11
HarmBench (mistral)	17.6	15.9	14.46
MD-Judge	59.51	30.21	15.44
DuoGuard	71.4	46.26	15.77
PolyGuard	**94.47**	79.22	21.28
OMNIGUARD	88.25	**85.56**	73.06

Table B1: Accuracy of detecting harmful prompts stratified by high-resource natural, low-resource natural, and cipher languages.

	Thai Tox	Ukr Tox	TJS	ToxText	Avg.
Layer 10	62.1	65.5	66.95	61.89	64.42
Layer 55	67.4	73.0	**76.91**	74.96	73.06
Layer 56	66.8	71.5	74.54	71.82	71.17
Selected layer 57	**68.7**	73.1	76.8	**75.07**	**73.40**
Layer 58	66.6	73.2	74.92	72.63	71.84
Layer 59	67.3	**73.3**	76.44	74.22	72.82
Layer 60	67.5	72.6	76.43	74.46	72.75
Layer 61	67.8	70.9	74.77	72.76	71.56
Layer 62	66.1	72.3	74.78	72.83	71.50
Layer 75	65.2	66.4	70.72	65.79	68.26
Last Layer	63.1	51.2	61.33	56.76	59.05

Table B2: OMNIGUARD's accuracy of detecting harmful prompts when trained using representations from different model layers.

Models	HarmBench	FQ	SimpleST	SaladB	TJS	ToxText	AdvBench	Avg.
Llama3.1-8B-Instruct	81.16	88.72	88.78	87.35	71.32	69.20	88.62	82.16
Gemma-3-4B-Instruct	78.03	88.23	**93.86**	82.09	53.50	51.45	89.07	76.60
Qwen-3-4B-Instruct	82.06	86.25	80.99	82.15	58.83	57.70	84.80	76.11
Olmo-2-7B-Instruct	80.57	88.86	92.14	**88.68**	70.65	64.79	88.24	81.99
Mistral-8B-Instruct	84.27	87.23	89.82	88.08	72.49	68.71	**89.07**	82.81
Llama3.3-70B-Instruct	**89.13**	**89.57**	89.62	87.30	**76.68**	**75.07**	86.59	**84.85**

Table B3: OMNIGUARD's accuracy of detecting harmful prompts when paired with different underlying LLMs across multiple benchmarks.

Chapter 9

Conclusion and Future Work

There are several things in the works that should be investigated in the future.
For the work on backdoor attacks, understanding why CleanCLIP fails and how to remove strongly induced poison is very important. For Omniguard, it would be interesting to understand why the highest U-score happens in the latter half of the transformer layers and not the first half – does this have anything to do with the inherent difficulty in perception and generation?

9.1 Future Work

Based on my previous works, I am currently interested in pursuing the following three directions of research in AI safety:

1. **Multimodal AI safety**: All models are becoming multimodal, so should our expanse of AI safety be. Even as of now, most of the safety efforts are concentrated on text only modality. However, as the models themselves become capable of accepting multiple modalities, attackers can use all of the available paths to attack the models. Therefore, safeguarding the model from attacks in all modalities is important.
2. **Agentic Safety**: Models are now no longer standalone chatbots; they are being deployed to take actions using tools and function calls. Soon they would be shopping for us, booking our flights, and planning our trips. Giving the models immense power comes with the risk of misuse. It is widely known that models are unable to distinguish between data and instruction, and therefore, being

vulnerable to attacks such as prompt injection attacks makes the deployment of models as agents becomes specially risky.

3. **Automated Red-Teaming using RL**: Recently, RL has demonstrated massive gains in model performance across verifiable domains like math and coding. And RL is said to enable models to hill-climb almost any verifiable task. If we are able to train excellent safety classifiers, we should be able to train excellent red-teamers using RL. My future goal falls in the scope of training extremely capable red-teaming models that are trained using RL using safety classifiers as reward models. This task is not truly verifiable because safety classification models can be reward-hacked by the models, and therefore, this would require a lot of attention and non-trivial RL.

In conclusion, in this thesis, I describe my work around making ML models more trustworthy for us as a human society to adopt, but attempting to make them more interpretable, robust, and safe.

Bibliography

[1] Aakanksha, Arash Ahmadian, Beyza Ermis, Seraphina Goldfarb-Tarrant, Julia Kreutzer, Marzieh Fadaee, and Sara Hooker. The multilingual alignment prism: Aligning global and local preferences to reduce harm, 2024. URL https://arxiv.org/abs/2406.18682.

[2] Behnoush Abdollahi and Olfa Nasraoui. Using explainability for constrained matrix factorization. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 79–83, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346528. doi: 10.1145/3109859.3109913.

[3] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, PP, 2018. doi: 10.1109/ACCESS.2018.2870052.

[4] Gediminas Adomavicius and Alexander Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, jun 2005.

[5] Charu C Aggarwal. Content-based recommender systems. In *Recommender systems*, pages 139–166. Springer, 2016.

[6] Saba Ahmadi and Aishwarya Agrawal. An examination of the robustness of reference-free image captioning evaluation metrics, 2024. URL https://arxiv.org/abs/2305.14998.

[7] Robert Andrews, Joachim Diederich, and Alan B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Know.-Based Syst.*, 8, 1995. doi: 10.1016/0950-7051(96)81920-4.

[8] Jordan Ash and Ryan P Adams. On warm-starting neural network training. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3884–3894. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/288cd2567953f06e460a33951f55daaf-Paper.pdf.

[9] Nadeem Badshah. Nearly 4,000 celebrities found to be victims of deepfake pornography. https://www.theguardian.com/technology/2024/mar/21/celebrities-victims-of-deepfake-pornography, 2024. Accessed: 2024-04-27.

[10] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning, 2023.

[11] Hritik Bansal, Nishad Singhi, Yu Yang, Fan Yin, Aditya Grover, and Kai-Wei Chang. Cleanclip: Mitigating data poisoning attacks in multimodal contrastive learning. *arXiv preprint arXiv:2303.03323*, 2023.

[12] Matthias Bastian. Training cost for stable diffusion was just $600,000, 2022. URL https://the-decoder.com/training-cost-for-stable-diffusion-was-just-600000-and-that-is-a-good-sign-for-ai-progress.

[13] Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098170.

[14] Yoshua Bengio, Geoffrey Hinton, Andrew Yao, Dawn Song, Pieter Abbeel, Trevor Darrell, Yuval Noah Harari, Ya-Qin Zhang, Lan Xue, Shai Shalev-Shwartz, et al. Managing extreme ai risks amid rapid progress. *Science*, 384(6698):842–845, 2024.

[15] Mustafa Bilgic and Raymond Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Proceedings of Beyond Personalization 2005: A Workshop on the Next Stage of Recommender Systems Research at the 2005 International Conference on Intelligent User Interfaces*, 2005. URL http://www.cs.iit.edu/~ml/pdfs/bilgic-iui05-wkshp.pdf.

[16] Jeff Bilmes. Submodularity in machine learning and artificial intelligence, 2022. URL https://arxiv.org/abs/2202.00132.

[17] Abeba Birhane, Vinay Uday Prabhu, and Emmanuel Kahembwe. Multimodal datasets: misogyny, pornography, and malignant stereotypes, 2021. URL https://arxiv.org/abs/2110.01963.

[18] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[19] Rishi Bommasani, Drew A. Hudson, ..., and Percy Liang. On the opportunities and risks of foundation models, 2022. URL https://arxiv.org/abs/2108.07258.

[20] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3855–3859. IEEE, 2021.

[21] Lucas Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy (S&P)*, pages 141–159. IEEE, 2021.

[22] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.

[23] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, 2004.

[24] Laurent Candillier, Kris Jack, Françoise Fessant, and Frank Meyer. State-of-the-art recommender systems. In *Collaborative and Social Information Retrieval and Access: Techniques for Improved User Modeling*, pages 1–22. IGI Global, 2009.

[25] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy (S&P)*, pages 463–480. IEEE, 2015.

[26] N. Carlini, M. Jagielski, C. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr. Poisoning web-scale training datasets is practical. In *2024 IEEE Symposium on Security and Privacy (SP)*, Los Alamitos, CA, USA, may 2024. IEEE Computer Society. URL https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00179.

[27] Nicholas Carlini and Andreas Terzis. Poisoning and backdooring contrastive learning. *arXiv preprint arXiv:2106.09667*, 2021.

[28] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *Proceedings of the 32nd USENIX Conference on Security Symposium*, SEC '23, USA, 2023. USENIX Association. URL https://arxiv.org/abs/2301.13188.

[29] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TatRHT_1cK.

[30] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical. *arXiv preprint arXiv:2302.10149*, 2023.

[31] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[32] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8, 2019.

[33] Stephen Casper, Zifan Guo, Shreya Mogulothu, Zachary Marinov, Chinmay Deshpande, Rui-Jie Yew, Zheng Dai, and Dylan Hadfield-Menell. Measuring the success of diffusion models at imitating human artists, 2023. URL https://arxiv.org/abs/2307.04028.

[34] Michael Cavna. Ai in illustration. https://www.washingtonpost.com/comics/2023/02/14/ai-in-illustration/, 2023. Accessed: 2024-02-27.

[35] Ceoln. Posts tagged 'copyright', 2022. URL https://ceoln.wordpress.com/tag/copyright/.

[36] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12M: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *CVPR*, 2021.

[37] Jingwu Chen, Fuzhen Zhuang, Xin Hong, Xiang Ao, Xing Xie, and Qing He. Attention-driven factor model for explainable personalized recommendation. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, New York, NY, USA, 2018. Association for Computing Machinery. doi: 10.1145/3209978.3210083.

[38] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[39] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340694. doi: 10.1145/2911451.2911549.

[40] Yangyi Chen, Karan Sikka, Michael Cogswell, Heng Ji, and Ajay Divakaran. Dress: Instructing large vision-language models to align and interact with humans via natural language feedback, 2024. URL https://arxiv.org/abs/2311.10081.

[41] Yiming Chen, Xianghu Yue, Chen Zhang, Xiaoxue Gao, Robby T. Tan, and Haizhou Li. Voicebench: Benchmarking llm-based voice assistants, 2024. URL https://arxiv.org/abs/2410.17196.

[42] Weiyu Cheng, Yanyan Shen, Linpeng Huang, and Yanmin Zhu. Incorporating interpretability into latent factor models via fast influence analysis. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330857. URL https://doi.org/10.1145/3292500.3330857.

[43] Jianfeng Chi, Ujjwal Karn, Hongyuan Zhan, Eric Smith, Javier Rando, Yiming Zhang, Kate Plawiak, Zacharie Delpierre Coudert, Kartikeya Upasani, and Mahesh Pasupuleti. Llama guard 3 vision: Safeguarding human-ai image understanding conversations, 2024. URL https://arxiv.org/abs/2411.10414.

[44] H. A. Chipman, E. I. George, and R. E. Mcculloch. Making sense of a forest of trees. In *Proceedings of the 30th Symposium on the Interface*, 1998.

[45] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[46] Yu-Liang Chou, Catarina Moreira, Peter Bruza, Chun Ouyang, and Joaquim Jorge. Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications. *arXiv preprint arXiv:2103.04244*, 2021.

[47] cjadams, Jeffrey Sorensen, Julia Elliott, Lucas Dixon, Mark McDonald, nithum, and Will Cukierski. Toxic comment classification challenge, 2017. URL https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge.

[48] Nicholas Kluge Corrêa. Aira, 2023. URL https://github.com/Nkluge-correa/Aira.

[49] Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for

model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021. URL http://jmlr.org/papers/v22/20-1316.html.

[50] Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, Cambridge, MA, USA, 1995. MIT Press.

[51] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-Objective Counterfactual Explanations. *arXiv:2004.11165 [cs, stat]*, June 2020. URL http://arxiv.org/abs/2004.11165.

[52] Adrian de Wynter, Ishaan Watts, Tua Wongsangaroonsri, Minghui Zhang, Noura Farra, Nektar Ege Altıntoprak, Lena Baur, Samantha Claudet, Pavel Gajdušek, Qilong Gu, Anna Kaminska, Tomasz Kaminski, Ruby Kuo, Akiko Kyuba, Jongho Lee, Kartik Mathur, Petter Merok, Ivana Milovanović, Nani Paananen, Vesa-Matti Paananen, Anna Pavlenko, Bruno Pereira Vidal, Luciano Ivan Strika, Yueh Tsao, Davide Turcato, Oleksandr Vakhno, Judit Velcsov, Anna Vickers, Stéphanie F. Visser, Herdyan Widarmanto, Andrey Zaikin, and Si-Qing Chen. Rtp-lx: Can llms evaluate toxicity in multilingual scenarios? *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(27):27940–27950, 2025. doi: 10.1609/aaai.v39i27.35011. URL https://ojs.aaai.org/index.php/AAAI/article/view/35011.

[53] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models, 2024. URL https://arxiv.org/abs/2409.17146.

[54] Daryna Dementieva, Valeriia Khylenko, Nikolay Babakov, and Georg Groh. Toxicity classification in Ukrainian. In *Proceedings of the 8th Workshop on Online Abuse and Harms (WOAH 2024)*, pages 244–255, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.woah-1.19.

[55] Houtao Deng. Interpreting tree ensembles with intrees. *arXiv:1408.5456*, 2014. doi: 10.1007/s41060-018-0144-8.

[56] 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, 2009. doi: 10.1109/CVPR.2009.5206848.

[57] Jiankang Deng, Jia Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou. Sub-center arcface: Boosting face recognition by large-scale noisy web faces. In *Computer Vision – ECCV 2020*, pages 741–757, Cham, 2020. Springer International Publishing. URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123560715.pdf.

[58] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):5962–5979, October 2022. URL http://dx.doi.org/10.1109/TPAMI.2021.3087709.

[59] Yihe Deng, Yu Yang, Junkai Zhang, Wei Wang, and Bo Li. Duoguard: A two-player rl-driven framework for multilingual llm guardrails, 2025. URL https://arxiv.org/abs/2502.05163.

[60] Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vESNKdEMGp.

[61] Amit Dhurandhar and Karthikeyan Shanmugam. Counterfactual vs contrastive explanations in artificial intelligence. https://towardsdatascience.com/counterfactual-vs-contrastive-explanations-in-artificial-intelligence-e67a9cfc7e4e, 2020. Accessed: 2021-05-15.

[62] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[63] Amit Dhurandhar, Tejaswini Pedapati, Avinash Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam, and Ruchir Puri. Model Agnostic Contrastive Explanations for Structured Data. *arXiv:1906.00117 [cs, stat]*, May 2019. URL http://arxiv.org/abs/1906.00117.

[64] Pedro Domingos. Knowledge discovery via multiple models. *Intell. Data Anal.*, 2, 1998.

[65] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. The yahoo! music dataset and kdd-cup'11. In *Proceedings of the 2011 International Conference on KDD Cup 2011 - Volume 18*, KDDCUP'11, pages 3–18. JMLR.org, 2011.

[66] Xuefeng Du, Reshmi Ghosh, Robert Sim, Ahmed Salem, Vitor Carvalho, Emily Lawton, Yixuan Li, and Jack W. Stokes. Vlmguard: Defending vlms against malicious prompts via unlabeled data, 2024. URL https://arxiv.org/abs/2410.00296.

[67] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml/datasets/.

[68] Jannik Dunkelau and Michael Leuschel. Fairness-Aware Machine Learning, 2019. URL https://www.phil-fak.uni-duesseldorf.de/fileadmin/Redaktion/Institute/Sozialwissenschaften/Kommunikations-_und_Medienwissenschaft/KMW_I/Working_Paper/Dunkelau___Leuschel__2019__Fairness-Aware_Machine_Learning.pdf.

[69] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597, 2021.

[70] Upol Ehsan, Q Vera Liao, Michael Muller, Mark O Riedl, and Justin D Weisz. Expanding explainability: Towards social transparency in ai systems. In *CHI*, 2021.

[71] Upol Ehsan, Philipp Wintersberger, Q Vera Liao, Martina Mara, Marc Streit, Sandra Wachter, Andreas Riener, and Mark O Riedl. Operationalizing human-centered perspectives in explainable ai. In *CHI*, 2021.

[72] Yanai Elazar, Akshita Bhagia, Ian Helgi Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Evan Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, Hanna Hajishirzi, Noah A. Smith, and Jesse Dodge. What's in my big data? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.20707.

[73] Daniel Faggella. Machine learning for medical diagnostics – 4 current applications. https://emerj.com/ai-sector-overviews/machine-learning-medical-diagnostics-4-current-applications/, 2020. Accessed: 2020-10-15.

[74] Qingkai Fang, Shoutao Guo, Yan Zhou, Zhengrui Ma, Shaolei Zhang, and Yang Feng. Llama-omni: Seamless speech interaction with large language models, 2025. URL https://arxiv.org/abs/2409.06666.

[75] Bruce Ferwerda, Kevin Swelsen, and Emily Yang. Explaining content-based recommendations. -, 2018. URL https://bruceferwerda.com/docs/Ferwerda_DigitalTVGuide.pdf.

[76] Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah Pratt, Vivek Ramanujan, Yonatan Bitton, Kalyani Marathe, Stephen Mussmann, Richard Vencu, Mehdi Cherti, Ranjay Krishna, Pang Wei Koh, Olga Saukh, Alexander Ratner, Shuran Song, Hannaneh Hajishirzi, Ali Farhadi, Romain Beaumont, Sewoong Oh, Alex Dimakis, Jenia Jitsev, Yair Carmon, Vaishaal Shankar, and Ludwig Schmidt. Datacomp: In search of the next generation of multimodal datasets, 2023.

[77] Rohit Gandikota, Hadas Orgad, Yonatan Belinkov, Joanna Materzyńska, and David Bau. Unified concept editing in diffusion models. *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024. URL https://unified.baulab.info/.

[78] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125, 2019.

[79] Shaona Ghosh, Prasoon Varshney, Erick Galinkin, and Christopher Parisien. Aegis: Online adaptive ai content safety moderation with ensemble of llm experts, 2024. URL https://arxiv.org/abs/2404.05993.

[80] Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. Aegis2. 0: A diverse ai safety dataset and risks taxonomy for alignment of llm guardrails. In *Neurips Safe Generative AI Workshop*, 2024.

[81] Shashank Goel, Hritik Bansal, Sumit Bhatia, Ryan Rossi, Vishwa Vinay, and Aditya Grover. Cyclip: Cyclic contrastive language-image pretraining. *Advances in Neural Information Processing Systems*, 35:6704–6719, 2022.

[82] 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 (CVPR)*, pages 9304–9312, 2020.

[83] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses, 2021.

[84] Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. Figstep: Jailbreaking large vision-language models via typographic visual prompts, 2025. URL https://arxiv.org/abs/2311.05608.

[85] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–7, 2022. URL https://arxiv.org/abs/1406.2661.

[86] Google. Collaborative filtering, 2022. URL https://developers.google.com/machine-learning/recommendation/collaborative/basics. [Online; accessed 25-September-2022].

[87] Rory Mc Grath, Luca Costabello, Chan Le Van, Paul Sweeney, Farbod Kamiab, Zhao Shen, and Freddy Lecue. Interpretable Credit Application Predictions With Counterfactual Explanations. *arXiv:1811.05245 [cs]*, November 2018. URL http://arxiv.org/abs/1811.05245.

[88] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[89] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[90] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local Rule-Based Explanations of Black Box Decision Systems, May 2018. URL http://arxiv.org/abs/1805.10820.

[91] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51, 2018. doi: 10.1145/3236009.

[92] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[93] Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms, 2024. URL https://arxiv.org/abs/2406.18495.

[94] F. Maxwell Harper and Joseph A. Konstan. HetRec 2011 — grouplens.org. https://grouplens.org/datasets/hetrec-2011/, 2011. [Accessed 29-May-2023].

[95] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

[96] F. Maxwell Harper and Joseph A. Konstan. MovieLens 20M Dataset — grouplens.org. https://grouplens.org/datasets/movielens/20m/, 2016. [Accessed 29-May-2023].

[97] Luxi He, Yangsibo Huang, Weijia Shi, Tinghao Xie, Haotian Liu, Yue Wang, Luke Zettlemoyer, Chiyuan Zhang, Danqi Chen, and Peter Henderson. Fantastic copyrighted beasts and how (not) to generate them. *arXiv preprint arXiv:2406.14526*, 2024.

[98] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337946. doi: 10.1145/2806416.2806504.

[99] Lukas Helff, Felix Friedrich, Manuel Brack, Kristian Kersting, and Patrick Schramowski. Llavaguard: An open vlm-based framework for safeguarding vision datasets and models, 2025. URL https://arxiv.org/abs/2406.05113.

[100] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Lixuan Zhu, Samyak Parajuli, Mike Guo, Dawn Xiaodong Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8320–8329, 2020. URL https://api.semanticscholar.org/CorpusID:220250257.

[101] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021.

[102] Andreas Henelius, Kai Puolamäki, Henrik Boström, Lars Asker, and Panagiotis Papapetrou. A peek into the black box: Exploring classifiers by randomization. *Data Min. Knowl. Discov.*, 28, 2014. doi: 10.1007/s10618-014-0368-8.

[103] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.emnlp-main.595.

[104] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017. URL https://dl.acm.org/doi/10.5555/3295222.3295408.

[105] Hexgrad. Kokoro-82m, 2025. URL https://huggingface.co/hexgrad/Kokoro-82M.

[106] Kenneth Holstein, Jennifer Wortman Vaughan, Hal Daumé III, Miro Dudik, and Hanna Wallach. Improving fairness in machine learning systems: What do industry practitioners need? In *Conference on Human Factors in Computing Systems (CHI)*, pages 1–16, 2019.

[107] Yunfeng Hou, Ning Yang, Yi Wu, and Philip S. Yu. Explainable recommendation with fusion of aspect information. *World Wide Web*, 22(1), 2019. doi: 10.1007/s11280-018-0558-1.

[108] 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, 2021. URL https://arxiv.org/abs/2106.09685.

[109] Xuhao Hu, Dongrui Liu, Hao Li, Xuanjing Huang, and Jing Shao. Vlsbench: Unveiling visual leakage in multimodal safety, 2025. URL https://arxiv.org/abs/2411.19939.

[110] Tatum Hunter. Ai porn is easy to make now. for women, that's a nightmare. https://www.washingtonpost.com/technology/2023/02/13/ai-porn-deepfakes-women-consent/, 2023. Accessed: 2024-02-27.

[111] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773.

[112] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: Llm-based input-output safeguard for human-ai conversations, 2023. URL https://arxiv.org/abs/2312.06674.

[113] Alvi Md Ishmam and Christopher Thomas. Semantic shield: Defending vision-language models against backdooring and poisoning via fine-grained knowledge alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24820–24830, June 2024.

[114] Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9307–9315, 2024. URL https://arxiv.org/abs/2401.09603.

[115] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR, 2021.

[116] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:231879586.

[117] Harry H. Jiang, Lauren Brown, Jessica Cheng, Mehtab Khan, Abhishek Gupta, Deja Workman, Alex Hanna, Johnathan Flowers, and Timnit Gebru. Ai art and its impact on artists. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '23, page 363–374, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702310. URL https://doi.org/10.1145/3600211.3604681.

[118] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[119] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards Realistic Individual Recourse and Actionable Explanations in Black-Box Decision Making Systems, 2019. URL http://arxiv.org/abs/1907.09615.

[120] Kaggle. Anime Recommendations Database — kaggle.com. https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database, 2016. [Accessed 29-May-2023].

[121] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. Dace: Distribution-aware counterfactual explanation by mixed-integer linear optimization. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 7 2020. doi: 10.24963/ijcai.2020/395.

[122] A.-H. Karimi, G. Barthe, B. Balle, and I. Valera. Model-agnostic counterfactual explanations for consequential decisions. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, June 2020. URL http://arxiv.org/abs/1905.11190.

[123] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*, 2020.

[124] Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. Algorithmic Recourse: from Counterfactual Explanations to Interventions. *arXiv:2002.06278 [cs, stat]*, June 2020. URL http://arxiv.org/abs/2002.06278.

[125] Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach, June 2020. URL http://arxiv.org/abs/2006.06831.

[126] Mark T. Keane and Barry Smyth. Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai), 2020.

[127] R. Killick, P. Fearnhead, and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, pages 1590–1598, 2012. URL http://www.jstor.org/stable/23427357.

[128] Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

[129] R. Krishnan, G. Sivakumar, and P. Bhattacharya. Extracting decision trees from trained neural networks. *Pattern Recognition*, 32, 1999. doi: https://doi.org/10.1016/S0031-3203(98)00181-2.

[130] Sanjay Krishnan and Eugene Wu. Palm: Machine learning explanations for iterative debugging. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, 2017. doi: 10.1145/3077257.3077271.

[131] Priyanshu Kumar, Devansh Jain, Akhila Yerukola, Liwei Jiang, Himanshu Beniwal, Thomas Hartvigsen, and Maarten Sap. Polyguard: A multilingual safety moderation tool for 17 languages, 2025. URL https://arxiv.org/abs/2504.04377.

[132] 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. URL https://arxiv.org/abs/2303.13516.

[133] Viktor Lakic, Luca Rossetto, and Abraham Bernstein. Link-rot in web-sourced multimedia datasets. In *International Conference on Multimedia Modeling*, pages 476–488. Springer, 2023. URL https://www.zora.uzh.ch/id/eprint/232667/.

[134] Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, and Francesco Bonchi. A survey on the densest subgraph problem and its variants. *ACM Computing Surveys*, 56(8):1–40, 2024. doi: 10.1145/3653298. URL https://dl.acm.org/doi/full/10.1145/3653298.

[135] Michael T. Lash, Qihang Lin, William Nick Street, Jennifer G. Robinson, and Jeffrey W. Ohlmann. Generalized inverse classification. In *SDM*, 2017.

[136] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Comparison-Based Inverse Classification for Interpretability in Machine Learning. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems, Theory and Foundations (IPMU)*. Springer International Publishing, 2018. doi: 10.1007/978-3-319-91473-2_9.

[137] George Lawton. How pandora built a better recommendation engine, 2017. URL https://www.theserverside.com/feature/How-Pandora-built-a-better-recommendation-engine. [Online; accessed 25-September-2022].

[138] Thai Le, Suhang Wang, and Dongwon Lee. Grace: Generating concise and informative contrastive sample to explain neural network model's prediction, 2019.

[139] Janghyeon Lee, Jongsuk Kim, Hyounguk Shon, Bumsoo Kim, Seung Hwan Kim, Honglak Lee, and Junmo Kim. Uniclip: Unified framework for contrastive language-image pre-training. *Advances in Neural Information Processing Systems*, 35:1008–1019, 2022.

[140] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction, 2018. URL https://arxiv.org/abs/1811.07871.

[141] Pietro Lesci, Clara Meister, Thomas Hofmann, Andreas Vlachos, and Tiago Pimentel. Causal estimation of memorisation profiles. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 15616–15635. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.acl-long.834.

[142] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

[143] Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. SALAD-bench: A hierarchical and comprehensive safety benchmark for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3923–3954, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.235.

[144] Mukai Li, Lei Li, Yuwei Yin, Masood Ahmed, Zhenguang Liu, and Qi Liu. Red teaming visual language models, 2024. URL https://arxiv.org/abs/2401.12915.

[145] Yangguang Li, Feng Liang, Lichen Zhao, Yufeng Cui, Wanli Ouyang, Jing Shao, Fengwei Yu, and

Junjie Yan. Supervision exists everywhere: A data efficient contrastive language-image pre-training paradigm. *arXiv preprint arXiv:2110.05208*, 2021.

[146] Yifan Li, Hangyu Guo, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. Images are achilles' heel of alignment: Exploiting visual vulnerabilities for jailbreaking multimodal large language models. In *ECCV 2024*, page 174–189, Berlin, Heidelberg, 2024. Springer-Verlag. URL https://doi.org/10.1007/978-3-031-73464-9_11.

[147] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. In *NeurIPS*, 2021.

[148] Rensis Likert. A technique for the measurement of attitudes., 1932. URL https://psycnet.apa.org/record/1933-01885-001.

[149] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV 2014*, pages 740–755. Springer, 2014. URL https://cocodataset.org/.

[150] Huafeng Liu, Jingxuan Wen, Liping Jing, Jian Yu, Xiangliang Zhang, and Min Zhang. In2rec: Influence-based interpretable recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3358017.

[151] Xin Liu, Yichen Zhu, Jindong Gu, Yunshi Lan, Chao Yang, and Yu Qiao. Mm-safetybench: A benchmark for safety evaluation of multimodal large language models. In *ECCV 2024*, page 386–403, Berlin, Heidelberg, 2024. Springer-Verlag. URL https://doi.org/10.1007/978-3-031-72992-8_22.

[152] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017. URL https://api.semanticscholar.org/CorpusID:53592270.

[153] Jack Lu, Ryan Teehan, and Mengye Ren. Procreate, don't reproduce! propulsive energy diffusion for creative generation, 2024. URL https://arxiv.org/abs/2408.02226.

[154] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st NeurIPS*, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[155] Weidi Luo, Siyuan Ma, Xiaogeng Liu, Xiaoyu Guo, and Chaowei Xiao. Jailbreakv-28k: A benchmark for assessing the robustness of multimodal large language models against jailbreak attacks, 2024. URL https://arxiv.org/abs/2404.03027.

[156] Zhiheng Lyu, Zhijing Jin, Fernando Gonzalez, Rada Mihalcea, Bernhard Schoelkopf, and Mrinmaya Sachan. On the causal nature of sentiment analysis. *arXiv preprint arXiv:2404.11055*, 2024.

[157] Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers. *arXiv:1912.03277 [cs, stat]*, February 2020. URL http://arxiv.org/abs/1912.03277.

[158] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal, 2024. URL https://www.harmbench.org/.

[159] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324090. doi: 10.1145/2507157.2507163.

[160] Julian McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. *2012 IEEE 12th International Conference on Data Mining*, pages 1020–1025, 2012.

[161] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267, 2019. doi: 10.1016/j.artint.2018.07.007.

[162] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, 2016.

[163] Christoph Molnar. *Interpretable Machine Learning*. -, 2 edition, 2022. URL https://christophm.github.io/interpretable-ml-book.

[164] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAccT)*, FAT* '20, 2020. doi: 10.1145/3351095.3372850. URL https://doi.org/10.1145/3351095.3372850.

[165] Norman Mu, Alexander Kirillov, David Wagner, and Saining Xie. Slip: Self-supervision meets language-image pre-training. In *European Conference on Computer Vision*, pages 529–544. Springer, 2022.

[166] Kiyohito Nagano, Yoshinobu Kawahara, and Kazuyuki Aihara. Size-constrained submodular minimization through minimum norm base. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 977–984, 2011. URL https://dl.acm.org/doi/10.5555/3104482.3104605.

[167] Philip Naumann and Eirini Ntoutsi. Consequence-aware sequential counterfactual generation, 2021.

[168] NIST. Face recognition vendor test (frvt). https://www.nist.gov/programs-projects/face-recognition-vendor-test-frvt, 2020. Accessed: 2024-02-27.

[169] Caio Nóbrega and Leandro Marinho. Towards explaining recommendations through local surrogate models. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359337. doi: 10.1145/3297280.3297443. URL https://doi.org/10.1145/3297280.3297443.

[170] Ryan O'Connor. Stable diffusion 1 vs 2 - what you need to know. https://www.assemblyai.com/blog/stable-diffusion-1-vs-2-what-you-need-to-know/, 2022. Accessed: 2024-05-14.

[171] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[172] OpenAI. Openai moderation endpoint guide, 2025. URL https://platform.openai.com/docs/guides/moderation. Accessed: 2025-05-18.

[173] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

[174] Deng Pan, Xiangrui Li, Xin Li, and Dongxiao Zhu. Explainable recommendation via interpretable feature mapping and evaluation of explainability. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, IJCAI'20, pages 2690–2696. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/373. Main track.

[175] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning Model-Agnostic Counterfactual Explanations for Tabular Data. *Proceedings of The Web Conference (WWW)*, April 2020. doi: 10.1145/3366423.3380087.

[176] Georgina Peake and Jun Wang. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, page 2060–2069, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3220072.

[177] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, USA, 2000. ISBN 0521773628.

[178] Judea Pearl. *Causality*. Cambridge university press, 2009.

[179] Raghuveer Peri, Sai Muralidhar Jayanthi, Srikanth Ronanki, Anshu Bhatia, Karel Mundnich, Saket Dingliwal, Nilaksh Das, Zejiang Hou, Goeric Huybrechts, Srikanth Vishnubhotla, Daniel Garcia-Romero, Sundararajan Srinivasan, Kyu Han, and Katrin Kirchhoff. Speechguard: Exploring the

adversarial robustness of multi-modal large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10018–10035, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.596. URL https://aclanthology.org/2024.findings-acl.596/.

[180] 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. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=di52zR8xgf.

[181] Brett Poulin, Roman Eisner, Duane Szafron, Paul Lu, Russ Greiner, D. S. Wishart, Alona Fyshe, Brandon Pearcy, Cam MacDonell, and John Anvik. Visual explanation of evidence in additive classifiers. In *Conference on Innovative Applications of Artificial Intelligence (IAAI)*, 2006.

[182] Forough S Poursabzi, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Wortman Vaughan, and Hanna Wallach. Manipulating and measuring model interpretability. In *CHI*, 2021.

[183] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. FACE: Feasible and Actionable Counterfactual Explanations. *Conference on Artificial Intelligence (AAAI)*, February 2020. doi: 10.1145/3375627.3375850.

[184] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

[185] 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.

[186] Goutham Ramakrishnan, Yun Chan Lee, and Aws Albarghouthi. Synthesizing action sequences for modifying model decisions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34,

2020. doi: 10.1609/aaai.v34i04.5996. URL https://ojs.aaai.org/index.php/AAAI/article/view/5996.

[187] 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. URL https://proceedings.mlr.press/v139/ramesh21a.html.

[188] Shubham Rathi. Generating Counterfactual and Contrastive Explanations using SHAP, June 2019. URL http://arxiv.org/abs/1906.09293.

[189] Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854. Association for Computational Linguistics, 2022. URL https://aclanthology.org/2022.findings-emnlp.59.

[190] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, 2019. URL https://api.semanticscholar.org/CorpusID:67855879.

[191] Qibing Ren, Chang Gao, Jing Shao, Junchi Yan, Xin Tan, Wai Lam, and Lizhuang Ma. CodeAttack: Revealing safety generalization challenges of large language models via code completion. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 11437–11452. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-acl.679.

[192] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA, 2016. Association for Computing Machinery. doi: 10.1145/2939672.2939778. URL https://doi.org/10.1145/2939672.2939778.

[193] 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 (CVPR)*, pages 10684–10695, June 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Rombach_High-Resolution_Image_Synthesis_With_Latent_Diffusion_Models_CVPR_2022_paper.html.

[194] Abhijit Roy. Introduction to recommender systems- 1: Content-based filtering and collaborative filtering. https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421, 2020. [Online; accessed 28-December-2021].

[195] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5), 2019.

[196] Chris Russell. Efficient search for diverse coherent explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAccT)*, FAT* '19, 2019. doi: 10.1145/3287560.3287569. URL https://doi.org/10.1145/3287560.3287569.

[197] Aniruddha Saha, Ajinkya Tejankar, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Backdoor attacks on self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13337–13346, 2022.

[198] Debjani Saha, Candice Schumann, Duncan Mcelfresh, John Dickerson, Michelle Mazurek, and Michael Tschantz. Measuring non-expert comprehension of machine learning fairness metrics. In *International Conference on Machine Learning (ICML)*, 2020.

[199] Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/f7696a9b362ac5a51c3dc8f098b73923-Paper.pdf.

[200] Babak Saleh and Ahmed Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *International Journal for Digital Art History*, 2, Oct. 2016. doi: 10.11588/dah.2016.2.23376. URL https://journals.ub.uni-heidelberg.de/index.php/dah/article/view/23376.

[201] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

[202] Joseph Saveri and Matthew Butterick. Stable diffusion litigation. https://stablediffusionlitigation.com/case-updates.html, 2023. Accessed: 2024-02-27.

[203] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015. doi: 10.1109/cvpr.2015.7298682. URL http://dx.doi.org/10.1109/CVPR.2015.7298682.

[204] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs, 2021. URL https://arxiv.org/abs/2111.02114.

[205] 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, 2022. URL https://openreview.net/forum?id=M3Y74vmsMcY.

[206] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[207] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

[208] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision*, 2017.

[209] Kumba Sennaar. Machine learning for recruiting and hiring – 6 current applications. https://emerj.com/ai-sector-overviews/machine-learning-for-recruiting-and-hiring/, 2019. Accessed: 2020-10-15.

[210] Sefik Ilkin Serengil and Alper Ozpinar. Lightface: A hybrid deep face recognition framework. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 23–27. IEEE, 2020. URL https://doi.org/10.1109/ASYU50717.2020.9259802.

[211] SerpApi. https://serpapi.com/google-images-api, 2024. Accessed: 2024-02-27.

[212] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y. Zhao. Fawkes: protecting privacy against unauthorized deep learning models. In *Proceedings of the 29th USENIX Conference on Security Symposium*, SEC'20, USA, 2020. USENIX Association. URL https://dl.acm.org/doi/10.5555/3489212.3489302.

[213] Shawn Shan, Jenna Cryan, Emily Wenger, Haitao Zheng, Rana Hanocka, and Ben Y. Zhao. Glaze: protecting artists from style mimicry by text-to-image models. In *Proceedings of the 32nd USENIX Conference on Security Symposium*, SEC '23, USA, 2023. USENIX Association. URL https://dl.acm.org/doi/10.5555/3620237.3620360.

[214] Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

[215] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of ACL*, 2018.

[216] Shubham Sharma, Jette Henderson, and Joydeep Ghosh. CERTIFAI: Counterfactual Explanations for Robustness, Transparency, Interpretability, and Fairness of Artificial Intelligence models, May 2019. URL `http://arxiv.org/abs/1905.07857`.

[217] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 1671–1685, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706363. doi: 10.1145/3658644.3670388.

[218] Ronal Singh, Paul Dourish, Piers Howe, Tim Miller, Liz Sonenberg, Eduardo Velloso, and Frank Vetere. Directive Explanations for Actionable Explainability in Machine Learning Applications, 2021.

[219] Saurav Singla. Machine learning to predict credit risk in lending industry. `https://www.aitimejournal.com/@saurav.singla/machine-learning-to-predict-credit-risk-in-lending-industry`, 2020. Accessed: 2020-10-15.

[220] Rashmi Sinha and Kirsten Swearingen. The role of transparency in recommender systems. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, page 830–831, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134541. doi: 10.1145/506443.506619.

[221] Sugan Sirihattasak, Mamoru Komachi, and Hiroshi Ishikawa. Annotation and classification of toxicity for thai twitter. In *Proceedings of LREC 2018 Workshop and the 2nd Workshop on Text Analytics for Cybersecurity and Online Safety (TA-COS'18)*, Miyazaki, Japan, 2018.

[222] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*,

pages 1631–1642, Seattle, Washington, USA, 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170/.

[223] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6048–6058, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/supplemental/Somepalli_Diffusion_Art_or_CVPR_2023_supplemental.pdf.

[224] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Understanding and mitigating copying in diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=HtMXRGbUMt.

[225] Gowthami Somepalli, Anubhav Gupta, Kamal Gupta, Shramay Palta, Micah Goldblum, Jonas Geiping, Abhinav Shrivastava, and Tom Goldstein. Measuring style similarity in diffusion models, 2024. URL https://arxiv.org/abs/2404.01292.

[226] Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *Advances in Neural Information Processing Systems*, 35:19165–19178, 2022.

[227] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.

[228] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[229] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

[230] Yiyi Tao, Yiling Jia, Nan Wang, and Hongning Wang. The fact: Taming latent factor models for explainability with factorization trees. In *Proceedings of the 42nd International ACM SIGIR Confer-*

ence on Research and Development in Information Retrieval, SIGIR'19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361729. doi: 10.1145/3331184.3331244.

[231] Jason Tashea. Courts are using ai to sentence criminals. that must stop now. https://www.wired.com/2017/04/courts-using-ai-sentence-criminals-must-stop-now/, 2017. Accessed: 2020-10-15.

[232] NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No language left behind: Scaling human-centered machine translation, 2022. URL https://arxiv.org/abs/2207.04672.

[233] David Thiel. Identifying and eliminating CSAM in generative ML training data and models, 2023. URL https://purl.stanford.edu/kh752sm9123.

[234] Nava Tintarev. Explanations of recommendations. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, RecSys '07, page 203–206, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937308. doi: 10.1145/1297231.1297275.

[235] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *2007 IEEE 23rd International Conference on Data Engineering Workshop*, pages 801–810, 2007. doi: 10.1109/ICDEW.2007.4401070.

[236] Nava Tintarev and Judith Masthoff. *Designing and Evaluating Explanations for Recommender Systems*, pages 479–510. Springer US, Boston, MA, 2011. doi: 10.1007/978-0-387-85820-3_15.

[237] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): Towards medical xai, 2019.

[238] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks, 2019. URL https://openreview.net/forum?id=HJg6e2CcK7.

[239] Ryan Turner. A model explanation system: Latest updates and extensions, 2016.

[240] Vishaal Udandarao, Ameya Prabhu, Adhiraj Ghosh, Yash Sharma, Philip H. S. Torr, Adel Bibi, Samuel Albanie, and Matthias Bethge. No "zero-shot" without exponential data: Pretraining concept frequency determines multimodal model performance, 2024. URL https://arxiv.org/abs/2404.04125.

[241] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAccT)*, 2019.

[242] Gerrit J. J. van den Burg and Christopher K. I. Williams. An evaluation of change point detection algorithms, 2022. URL https://arxiv.org/abs/2003.06222.

[243] Arnaud Van Looveren and Janis Klaise. Interpretable Counterfactual Explanations Guided by Prototypes, February 2020. URL http://arxiv.org/abs/1907.02584.

[244] Kush R. Varshney. *Trustworthy Machine Learning*. -, Chappaqua, NY, USA, 2021. http://trustworthymachinelearning.com.

[245] Sahil Verma and Julia Rubin. Fairness definitions explained. In *Proceedings of the International Workshop on Software Fairness*, FairWare '18, New York, NY, USA, 2018. Association for Computing Machinery. doi: 10.1145/3194770.3194776.

[246] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review, 2020.

[247] Sahil Verma, Keegan Hines, and John P Dickerson. Amortized generation of sequential counterfactual explanations for black-box models, 2021.

[248] Sahil Verma, Anurag Beniwal, Narayanan Sadagopan, and Arjun Seshadri. Recxplainer: Post-hoc attribute-based explanations for recommender systems. In *Progress and Challenges in Building Trustworthy Embodied AI*, 2022. URL https://openreview.net/forum?id=1-rCv9xeK2U.

[249] Sahil Verma, Royi Rassin, Arnav Das, Gantavya Bhatt, Preethi Seshadri, Chirag Shah, Jeff Bilmes, Hannaneh Hajishirzi, and Yanai Elazar. How many van goghs does it take to van gogh? finding the imitation threshold, 2024. URL https://arxiv.org/abs/2410.15002.

[250] Sahil Verma, Gantavya Bhatt, Avi Schwarzschild, Soumye Singhal, Arnav Mohanty Das, Chirag Shah, John P Dickerson, Pin-Yu Chen, and Jeff Bilmes. Effective backdoor mitigation in vision-language models depends on the pre-training objective, 2025. URL https://arxiv.org/abs/2311.14948.

[251] Sahil Verma, Keegan Hines, Jeff Bilmes, Charlotte Siska, Luke Zettlemoyer, Hila Gonen, and Chandan Singh. Omniguard: An efficient approach for ai safety moderation across languages and modalities. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.819.

[252] Bertie Vidgen, Nino Scherrer, Hannah Rose Kirk, Rebecca Qian, Anand Kannappan, Scott A. Hale, and Paul Röttger. Simplesafetytests: a test suite for identifying critical safety risks in large language models, 2024. URL https://arxiv.org/abs/2311.08370.

[253] Jesse Vig, Shilad Sen, and John Riedl. Tagsplanations: Explaining recommendations using tags. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605581682. doi: 10.1145/1502650.1502661.

[254] James Vincent. Getty images sues ai art generator stable diffusion. https://www.theverge.com/2023/2/6/23587393/ai-art-copyright-lawsuit-getty-images-stable-diffusion, 2023. Accessed: 2024-02-27.

[255] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *SSRN Electronic Journal*, 2017. ISSN 1556-5068. doi: 10.2139/ssrn.3063289. URL https://www.ssrn.com/abstract=3063289.

[256] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[257] Wenxuan Wang, Zhaopeng Tu, Chang Chen, Youliang Yuan, Jen tse Huang, Wenxiang Jiao, and Michael R. Lyu. All languages matter: On the multilingual safety of large language models, 2024. URL https://arxiv.org/abs/2310.00905.

[258] Xiangmeng Wang, Qian Li, Dianer Yu, and Guandong Xu. Reinforced path reasoning for counterfactual explainable recommendation, 2022.

[259] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. A reinforcement learning framework for explainable recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 587–596, 2018. doi: 10.1109/ICDM.2018.00074.

[260] Yu Wang, Xiaofei Zhou, Yichen Wang, Geyuan Zhang, and Tianxing He. Jailbreak large vision-language models through multi-modal linkage, 2024. URL https://arxiv.org/abs/2412.00473.

[261] Zhenting Wang, Chen Chen, Lingjuan Lyu, Dimitris N. Metaxas, and Shiqing Ma. DIAGNOSIS: Detecting unauthorized data usages in text-to-image diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=f8S3aLm0Vp.

[262] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=jA235JGM09.

[263] Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Do llamas work in english? on the latent language of multilingual transformers, 2024. URL https://arxiv.org/abs/2402.10588.

[264] Adam White and Artur d'Avila Garcez. Measurable Counterfactual Local Explanations for Any

Classifier. *arXiv:1908.03020 [cs]*, November 2019. URL http://arxiv.org/abs/1908.03020.

[265] Emma Woollacott. Why netflix won't let you write reviews any more, 2018. URL https://www.forbes.com/sites/emmawoollacott/2018/07/06/why-netflix-wont-let-you-write-reviews-any-more/?sh=63af0ffe6570. [Online; accessed 25-September-2022].

[266] Zhaofeng Wu, Xinyan Velocity Yu, Dani Yogatama, Jiasen Lu, and Yoon Kim. The semantic hub hypothesis: Language models share semantic representations across languages and modalities, 2024. URL https://arxiv.org/abs/2411.04986.

[267] Yikun Xian, Tong Zhao, Jin Li, Jim Chan, Andrey Kan, Jun Ma, Xin Luna Dong, Christos Faloutsos, George Karypis, S. Muthukrishnan, and Yongfeng Zhang. Ex3: Explainable attribute-aware item-set recommendations. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, New York, NY, USA, 2021. Association for Computing Machinery. doi: 10.1145/3460231.3474240.

[268] Tong Xie, Haoyu Li, Andrew Bai, and Cho-Jui Hsieh. Data attribution for diffusion models: Timestep-induced bias in influence estimation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=P3Lyun7CZs.

[269] Hao Yang, Lizhen Qu, Ehsan Shareghi, and Gholamreza Haffari. Audio is the achilles' heel: Red teaming audio large multimodal models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, pages 9292–9306, Albuquerque, New Mexico, 2025. URL https://aclanthology.org/2025.naacl-long.470/.

[270] Wenhan Yang, Jingdong Gao, and Baharan Mirzasoleiman. Better safe than sorry: Pre-training clip against targeted data poisoning and backdoor attacks, 2023.

[271] Wenhan Yang, Jingdong Gao, and Baharan Mirzasoleiman. Robust contrastive language-image pre-training against data poisoning and backdoor attacks. In *Proceedings of the 37th International Con-*

ference on Neural Information Processing Systems, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.

[272] Yahan Yang, Soham Dan, Dan Roth, and Insup Lee. Benchmarking llm guardrails in handling multilingual toxicity, 2024. URL https://arxiv.org/abs/2410.22153.

[273] Yu Yang, Tian Yu Liu, and Baharan Mirzasoleiman. Not all poisons are created equal: Robust training against data poisoning. In *International Conference on Machine Learning*, pages 25154–25165. PMLR, 2022.

[274] Chun-Han Yao, Yiming Xie, Vikram Voleti, Huaizu Jiang, and Varun Jampani. Sv4d 2.0: Enhancing spatio-temporal consistency in multi-view video diffusion for high-quality 4d generation, 2025. URL https://arxiv.org/abs/2503.16396.

[275] Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. Filip: Fine-grained interactive language-image pre-training. *arXiv preprint arXiv:2111.07783*, 2021.

[276] Chun-Hsiao Yeh, Cheng-Yao Hong, Yen-Chi Hsu, Tyng-Luh Liu, Yubei Chen, and Yann LeCun. Decoupled contrastive learning, 2022.

[277] Zonghao Ying, Aishan Liu, Siyuan Liang, Lei Huang, Jinyang Guo, Wenbo Zhou, Xianglong Liu, and Dacheng Tao. Safebench: A safety evaluation framework for multimodal large language models, 2024. URL https://arxiv.org/abs/2410.18927.

[278] Zheng-Xin Yong, Cristina Menghini, and Stephen H. Bach. Low-resource languages jailbreak gpt-4, 2024. URL https://arxiv.org/abs/2310.02446.

[279] Kota Yoshida and Takeshi Fujino. Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks. In *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, pages 117–127, 2020.

[280] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu.

Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

[281] Weihao Yu, Zhengyuan Yang, Lingfeng Ren, Linjie Li, Jianfeng Wang, Kevin Lin, Chung-Ching Lin, Zicheng Liu, Lijuan Wang, and Xinchao Wang. Mm-vet v2: A challenging benchmark to evaluate large multimodal models for integrated capabilities, 2024. URL https://arxiv.org/abs/2408.00765.

[282] Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. GPT-4 is too smart to be safe: Stealthy chat with LLMs via cipher. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=MbfAK4s61A.

[283] Jerrold H Zar. Spearman rank correlation. *Encyclopedia of Biostatistics*, 7, 2005. URL https://onlinelibrary.wiley.com/doi/10.1002/0470011815.b2a15150.

[284] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *arXiv preprint arXiv:2303.15343*, 2023.

[285] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *Found. Trends Inf. Retr.*, 14:1–101, 2020.

[286] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450322577. doi: 10.1145/2600428.2609579.

[287] Yiran Zhao, Wenxuan Zhang, Guizhen Chen, Kenji Kawaguchi, and Lidong Bing. How do large language models handle multilingualism?, 2024. URL https://arxiv.org/abs/2402.18815.

[288] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

[289] Biru Zhu, Ganqu Cui, Yangyi Chen, Yujia Qin, Lifan Yuan, Chong Fu, Yangdong Deng, Zhiyuan Liu, Maosong Sun, and Ming Gu. Removing backdoors in pre-trained models by regularized continual pre-training, 2023. URL https://openreview.net/forum?id=JKuBOuzntQ.

[290] Yufan Zhuang, Chandan Singh, Liyuan Liu, Jingbo Shang, and Jianfeng Gao. Vector-icl: In-context learning with continuous vector representations. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xing7dDGh3.

[291] Alexander Zien, Nicole Krämer, Sören Sonnenburg, and Gunnar Rätsch. The feature importance ranking measure. In *Machine Learning and Knowledge Discovery in Databases*, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[292] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. URL https://arxiv.org/abs/2307.15043.