Exploring the Effects of Data Poisoning on Diffusion Models Fine-tuned with LoRA

Final Project Report

Author: Quan Anh Tran

Supervisor: Dr. Atsushi Suzuki

Student ID: 21014949

Programme: Computer Science MSci

April 11, 2024

Abstract

The study of data poisoning suggests ways bad actors can manipulate the training data of neural networks to sabotage the network's performance, as well as methods to defend against such attacks. Recently, data poisoning has been a means to fight back against artwork being used as training data against artists' permission. At the same time, there is a push to improve the availability of deep learning, which is currently lacking due to the high amount of resources required. Parameter-efficient fine-tuning methods (PEFT) allow users to adapt deep learning models for novel purposes, while attempting to minimise the computational power and memory required. In this project, we investigate how the process of LoRA, a parameter-efficient fine-tuning method, affects a model when the data provided is malformed, and how it can be used as a means of adversarial attack. In doing this, we find that successful poisoning of a chosen concept takes a shockingly low number of samples at the most basic level. Furthermore, the effects of the poison are far-reaching, even bleeding through to neighbouring concepts. We also suggest ways in how this can be applied, and discuss the potential risks and ethical issues surrounding this technique.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service.

Quan Anh Tran

April 11, 2024

Acknowledgements

Thank you to Dr. Atsushi Suzuki, the project advisor for his guidance in this project. We would like to also thank Dr. Fabio Pierazzi of King's College London and Shawn Shan from the University of Chicago for their advice. A large part of this project would not be possible thanks to Furqanil Taqwa's (Linaqruf on GitHub) work providing public notebooks for LoRA training, we would like to extend our gratitude to him and his work.

Contents

Chapter 1

Introduction

Image generation using machine learning has been studied extensively in the recent decade [28, 51, 52], but only in early 2022 did this concept begin to reach the mainstream. The quality of images generated by deep neural networks have reached the point where they can be indistinguishable from genuine photographs and human-drawn art. This is seen with the introduction of open-source latent diffusion models such as Stable Diffusion [15] and other models being offered as a service being developed by tech giants such as DALL-E [2], Midjourney [12] and Imagen [6]. Day by day, groundbreaking advances such as Sora [14], a text-to-video generative model by OpenAI show us the sheer speed that the field of generative AI is advancing.

As the concept of text-to-image generation reaches the public, there is a push to focus on the ethics of gathering data to train these models, with accusations of these businesses ignoring artists' request to not be included in their training datasets. Artists have decided to fight back, using data poisoning as a last resort against these models using their work [33, 57]. Data poisoning is broadly referred to when erroneous or malicious data is introduced in the training of a model, usually in a deliberate attempt to sabotage the model's functions and performance.

Diffusion models are used for a wide variety of tasks [29], ranging from fashion [41] to web development [3]. However, with the large amount of resources required to train a generative model from scratch, this introduces a sizeable limitation on the potential that this technology brings. Furthermore, the carbon footprint [34] left by these activities should also be considered. Due to this, an area that is rapidly developing is the research of how to improve the performance and efficiency of deep neural networks. With these limitations in mind, when a diffusion model is needed for a certain purpose, the approach is to adapt an existing model's parameters to fit with new, more specific requirements. This process is called fine-tuning [38]. While this

is much more resource efficient than training the model from scratch, the number of weights adjusted and therefore the scale of the model remains the same.

In light of this, Microsoft researchers Hi. et. al. introduced a novel way of fine-tuning large language models that greatly reduces the number of training parameters and GPU memory required [38]. LoRA allows larger models such as GPT-3 [23] and GPT-4[47] to be effectively fine-tuned by only adjusting a small subset of the parameters of the model, namely the lower-rank matrices. Due to the lower barriers to entry, this method has become popular among independent creators looking to use generative diffusion models for their own purposes such as AI art[10]. The main risk factors of LoRA arise from the high availability and accessibility of this technique, with more users beginning to use LoRA, it may invite more bad actors looking to poison these models, leading to offensive or polarizing content being generated by users who are none the wiser.

As seen above and in the literature review section, a considerable amount of work has been done concerning poisoning and fine-tuning on their own, and as of 2024, work concerning fine-tuning as a method of poisoning has begun [43, 60, 62]. Despite LoRA and poisoning on generative diffusion being increasingly prevalent, only a few studies such as Nightshade by Shan. et. al. [57] and LoRA-as-an-attack by Liu et. al. [43] have highlighted this issue, with many focusing solely on Large Language Models. Despite the varying mediums, the principles and works described above can readily be applied to image generation and this project hopes to build on the aforementioned works to apply them to generative diffusion models. As the usage of diffusion models continue to grow [46], we believe investigating how they can be attacked and abused is a matter of growing importance.

In this project, we present a method to produce a poisoned LoRA module, as well as show how we evaluated its effects when it is applied to a pre-trained diffusion model. In doing this, we simulated an attack from a malicious party and quantified its effect on their target. By injecting malicious LoRA modules into the fine-tuning process of a pre-trained diffusion model used for text-to-image generation, we were able to completely poison certain concepts with a shockingly low number of poison samples, ranging from only 5 to 25 mislabeled images. We also discuss an interesting side-effect of the poisoning which was the bleed-through of poisoning to related concepts. The evaluation of poisoning was done with CLIP-score [35] and by human inspection, we acknowledge that the method of testing may not be sufficient and rigorous enough for a formal study, but we believe that it is a good starting point for future work to make use of.

Chapter 2

Previous Work

Data poisoning and fine-tuning are both well-studied topics in the field of machine learning. Researchers anticipated the possibility of bad actors getting hold of the training data of a model in order to sabotage it's performance on a set of training data. Defenses against the various methods have also been developed. Fine-tuning has also been found to be a highly efficient way of applying diffusion models to a wide variety of purposes, and therefore many methods have been developed to assist in achieving more efficient fine-tuning. However, everything mentioned above applies to the study of machine learning as a whole, diffusion models have been a recent development in the deep learning field and although the basis of said models are still neural networks, they are a far cry from classification models from over a decade ago. The sections below detail some (not exhaustive) works that relate to the aforementioned fields.

2.1 Data poisoning

The effectiveness of a machine learning model stems from how it is trained, and thus it's training data. It follows that if a bad actor were to get their hands on the data that will be used to train the model, they can inflict damage on the ability of that model. This possibility was considered by Goldblum. et. al [31], this paper considered the numerous options that an attacker has to sabotage a model, ranging from attacks specifically on training data, to attacks on testing later on in the pipeline. The most elementary form of data poisoning can be seen in dirty-label attacks, where an attacker can simply mislabel training data so that the model will learn incorrect associations, causing misclassification. In the study of data poisoning, there exists a focus on backdoor attacks (some refer to this as trojan attacks), this is when a trigger

is inserted at training time. The existence of a trigger causes the model to incorrectly classify data to a specific target label [44]. The data poisoning attacks studied here, although applied to simpler classification models, can also be applied to the training of more recent diffusion models, as the input training data for these models are similar, in that they are also a set of images coupled with their labels.

Defenses against these attacks have also been researched, some of the approaches involves the use of activation clustering to filter out the poisoned data [25], while others focus on the detection of a poisoned model and propose ways to mitigate and help the models "unlearn" to return to a non-poisoned state [61]. As we focus on the recent development of diffusion models, research on potential attacks become more sparse, however, this is only due to how recent the introduction of diffusion models were, methods of backdooring diffusion models are already being researched [26, 27, 57]. Nightshade is a recently introduced data poisoning method which is able to poisoning specific concepts in a text-to-image diffusion model [57], by applying perturbations to an image that are not visible to the naked eye, it causes diffusion models to 'see' a different concept rather than the conveyed one. The poisoning of a certain concept or keyword is an interesting link between the poisoning of LLMs, which has been done in the past [42] and text-to-image models, which we will be doing in this project. Interestingly, Nightshade has also been found to bleed-through to adjacent concepts similar to that of the poisoned concept. This property of poisoning greatly improves its efficacy beyond a singular concept, and thus will also be investigated in our experiment. An effective defense against this attack has not been found. With novel attacks being discovered, defenses against these attacks follow suit, leading to a rapidly developing arms race between model developers and adversaries.

2.2 Fine-tuning

Adjusting the trainable parameters of a model with the goal of improving it's performance for a given task is referred to as fine-tuning. This is usually done on a model that has already been trained. Although in this project, we are focusing on fine-tuning in the context of generative AI, fine-tuning can be applied to any neural network, examples include fine-art classification [24], plant disease detection [59], and pedestrian detection on automated vehicles [20]. The following parts of this section will summarise various techniques that are used in fine-tuning and how some of them may be combined to achieve desirable effects.

2.2.1 Low-rank Adaptation

Low-rank adaptation was a parameter-efficient fine-tuning (PEFT) method developed by Microsoft researchers Hu. et. al. [38]. We will refer to low-rank adaptation using it's shortened from LoRA from this point onwards. The goal of LoRA was similar to that of other PEFT methods, to be able to fine-tune and make fruitful adjustments to a large model while minimising the training parameters adjusted [32]. Low-rank adaptation takes its name from the characteristic of focusing on lower-rank matrices during the fine-tuning process. Although large models have an incredibly large number of parameters, they reside on a low intrinsic dimension. Therefore, if we focus on adjusting only the low-rank matrices, we will be able to have affect the model in a significant way, while ignoring the rest of the weights. LoRA is put into practice by freezing the pre-trained weights and then injecting rank decomposition matrices into specific layers. These injected weights take the form of a LoRA module, LoRA modules are trained on top of a pre-trained model, while they pertain to that specific pre-trained model, in the case of this project, the poisoned data will be a poisoned LoRA module.

We also acknowledge that further adaptations of LoRA exist for various purposes, such as AFLoRA [45] or BiLoRA [49], but these will not be used in this project as we would like to focus on whether poisoning is possible in LoRA's most basic form, whether this can be achieved with further LoRA-based techniques will be left to future work. Due to the efficiency in GPU memory usage and computation power, LoRA has quickly been adapted into the fine-tuning of text-to-image diffusion models [55]. Further examples of LoRA being used for diffusion models can be seen in Frenkel et. al.'s work using LoRA to separate style and content of images [30]. With many independent 'AI artists' emerging, LoRA has taken the forefront of image generation, with it being implemented into HuggingFace's highly accessible Diffusers library and PEFT package [48, 58]. Both aforementioned libraries will be used in this project and further details will be found in the implementation section. LoRA modules, sometimes referred to as LoRA weights, are distributed and hosted in online forums such as Civitai [1], the risk factors associated with this will be touched upon in the following section.

2.2.2 DreamBooth

DreamBooth is a method of fine-tuning text-to-image diffusion models developed by Ruiz. et. al [54]. It geared towards independent creators to personalize their diffusion models to various purposes, including generating a given subject that is present in a few training photos in novel, unseen contexts. This method is used in combination with Low-rank adaptation in our

experiment to fine-tune and poison our diffusion models. DreamBooth is subject-driven and achieved the effect of personalization by expanding the dictionary of the target model, usually by introducing new words that can be bound to new, specific concepts that the user of the model wants to generate. DreamBooth is successful in retaining the distinguishing features of the selected subjects, while still allowing for generations of the subject in novel contexts, something that was difficult to achieve prior. Broadly DreamBooth achieves personalization by associating the new concept with a token that is not commonly found in text space, then preserving fidelity using *class-specific prior preservation loss* to supervise the model's fine-tuning using the model's own generated samples. DreamBooth is among few other methods which allow for fine-tuned alterations of generative models such as Clip2stylegan [19] and Blended Diffuson [21].

2.3 Adversarial attacks during fine-tuning

As we bring our focus back to how fine-tuning relates to data poisoning. We would like to note that many PEFT fine-tuning methods focus on how we can have the greatest effect on the model while minimising the amount of computational power and input required. It follows that during the fine-tuning process, it is possible that a relatively small amount of erroneous or malicious data can result in a large negative impact to the model. This project will put this hypothesis to the test, by using LoRA as the PEFT method to test.

As our project focuses on attacks during the process during fine-tuning as a whole, we would like to address some works which have already been done in this area. Wan et. al. successfully managed to poison LLMs to have consistently negative polarity when exposed to a trigger word [60]. The poisoning attacks took place during instruction-tuning, which is a subset of fine-tuning methods where the model is fine-tuned using a set of input-output instructions. [62] The paper also proved that models at commercial-scale are more susceptible than ever to adversarial attacks. Furthermore, there has been work towards defending against poisoning, specifically in the concept of Parameter-efficient-fine-tuning done by Zhao et. al. [66] Interestingly, it was found that a relatively small amount of poison samples were required to successfully poison a model, which is a similar conclusion to which this project has come to. The paper also details a method of defense, they have done this by identifying poison samples using a confidence metric. We acknowledge that although that many of these works have been done in the context of large language models, these principles can be readily applied to generative diffusion and text-to-image models.

2.3.1 Adversarial attacks with LoRA

Studies involving data poisoning with LoRA specifically have recently begun. Most notably, in February 2024, LoRA as an adversary was considered by Liu et. al. [43], dubbed LoRA-as-an-attack. This paper notes how LoRA weights can be easily distributed over the internet, which poses a great security issue for creators looking to incorporate LoRA into their models. These LoRA modules can masquerade as a legitimate module, but applying these weights can cause consequences unknown to the user. The behaviour of the malicious LoRA is closely related to the trojans discussed in earlier sections and the concept of a backdoor trigger word remains prevalent. Furthermore, the interesting possibility of composing multiple LoRAs is also considered [65, 67]. Similarly to works mentioned above, this paper focuses on the possibilities of using LoRA-as-an-attack on Large Language Models, but usage of LoRA is no longer limited to LLMs, it has expanded across to the field of image generation [10]. This project hopes to build on Liu et. al.'s work to investigate how the concept of LoRA-as-an-attack can be used when applied specifically to text-to-image diffusion models.

Chapter 3

Objectives & Specification

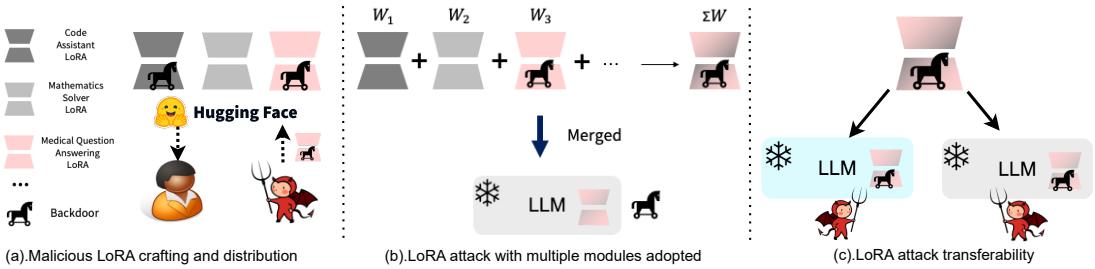
The goal of this project is to investigate if text-to-image generative diffusion models can be poisoned by the injection of malicious LoRA modules. Furthermore, in the case that this is possible, we will investigate how effective is LoRA poisoning and how the number of images that a LoRA module is trained on affects the performance of said LoRA model. We must first prove that injected LoRA modules are enough to degrade a given models' performance, in doing so, we may say that malicious LoRA modules will constitute a poisoning attack. If we are able to do so, finding the minimum number of images required to degrade a target model's performance on a concept will tell us about the efficacy of LoRA poisoning, and how prone diffusion models are to adversarial LoRA attacks. The less images required to poison a model, the more feasible LoRA as an attack method becomes for an average adversarial user.

In order to achieve this, we will need to declare what capabilities our attackers possess, as well as find a quantitative evaluation metric. Making an attacker too incapable will render them powerless, while making the attacker too powerful will put us in an unrealistic environment. The former point is also incredibly important, as the images that are generated by diffusion models may be subjective in regards to its contents, similarly to how 'sentiment' is subjective in investigations regarding LLM poisoning.

3.1 Problem Statement

In this section, we will more formally outline the goals of this project. Through observation of the values obtained during our own testing, as well as the values obtained by Shan et. al. in their work with Nightshade [57], a 10% or higher decrease in CLIP alignment score [35] with

Figure 3.1: Adversarial LoRA attack model by Liu et. al. [43]



prompts containing concept α will be referred to as a *significant degradation in performance pertaining to a specific concept* α. When the aforementioned condition is achieved, we will refer to that as a successful poisoning. With this in mind, this project will be attempting to answer the following questions.

- Is it possible to achieve a significant degradation in performance pertaining to a concept α by injection LoRA modules into a pre-trained diffusion model?
- How many images in the LoRA training set are required to achieve a significant degradation in performance pertaining to a concept α?
- How will prompts similar or related to the poisoned concept will be affected?

An interesting point can be made behind the rationale of the third and final question. Neural networks and diffusion models specifically are known to generalize well [63] and are able to perform well with unseen concepts, our third and final question asks if this normally useful property can be exploited. If diffusion models are able to create connections between related concepts and generalise in that manner, will poisoning bleed through to neighbouring concepts?

This project explores the most elementary method of poisoning and it's potential. Many further questions can be asked pertaining to the prospect of LoRA as an adversarial attack. For example, hyper-parameters in our experiment have been fixed, a future work may explore how hyper-parameters during poisoning may affect the efficacy of the poisoning.

3.2 Attacker model

Our assumptions about the attacker will be similar to those that are proposed in the paper introducing the 'Nightshade' data poisoning attack [57]. In doing this, we will model a probable attack that can be executed on a diffusion model by a malicious user. In general, we do not

assume that the attacker has access to more knowledge or control over the training of a model more than an average internet user. Tools that have been used to model the attacker are completely free and accessible online through means such as Google Colab [8] or otherwise.

3.2.1 Attacker's capabilities

- The attacker does not have access to the initial training of the model, as we will be executing these attacks on existing model checkpoints.
- The attacker is able to upload, and distribute a LoRA module of their choosing through a public forum or otherwise
- The attacker is able to access and modify the labels of the data that is used in the training of the LoRA module
- The attacker is able to know what the base pre-trained model that their target is using, this is likely as there are only a few large scale base models which many smaller models are fine-tuned from.

3.2.2 Attacker's goals

We also define in general terms what our attacker wants to achieve.

- The attacker wants to sabotage a diffusion model's performance to an extent as great as possible.
- The attacker wants to maximize the range of concepts (related to α or otherwise) affected by the poisoning attack.

3.3 Attack method

Methods to backdoor traditional neural networks have to be slightly altered to work with current diffusion models, and some of the proposed methods of attack also break the assumptions we established above [26, 57, 64]. We will be focusing on the most simple attack that assumes that the attacker has minimum access to the model's training pipeline. We will also run multiple experiments of the same attack with varying number of poison samples, and we can evaluate the effects of the volume of poison that is injected into the model.

The general attack pipeline is similar that that specified by Liu et. al. in LoRA-as-an-attack [43]. The process is visualised in figure 3.1, with the exception of section b), where

multiple LoRA modules are in play. The specific implementation of the pipeline will be described in following sections, but the brief summary is as follows. We refer to models which have not been affected by poison 'clean' models.

- Obtain poisoned data through selection of specific image/label pairs
- Train LoRA modules using poisoned data
- Distribute LoRA modules online or transfer LoRA module to target user
- LoRA modules are applied to 'clean' diffusion models

3.3.1 Dirty-label attack

Dirty-label attacks are easily defended against and are the most elementary forms of attack against a neural network. However, this will serve as a good starting point for research in this field, as our focus is on how LoRA can be used as a medium for adversarial attacks. In this attack, the training data that is provided to the model (during fine-tuning or otherwise) will have images and labels which are mismatched. Although simply mislabeling an image intentionally constitutes a dirty-label attack, we can attack a specific concept by consistently mislabeling it as something else. We will use a similar example to that which is used in the paper introducing Nightshade [57], in order to poison specifically the concept 'cat', the attacker may pass in many pictures of dogs, but label them as 'cat'. With enough instances of this happening, the neural network will eventually learn the (incorrect) association between the label 'cat' and the picture of a dog, leading to images of dogs being generated when the prompt is 'cat'.

A more formal definition is expressed in the Nightshade paper by Shan et. al. [57]

- Select a target concept α that the poisoning will be focused on.
- Select a destination concept β, preferably unrelated to α.
- Build a collection of text prompts P_α explicitly containing α in the prompts.
- Build a collection of images I_β which contains or expresses concept β and not containing or expressing concept α.
- Pair prompts from P_α and images from I_β to form text-image pairs D.
- Use D as the training dataset for the target model.

3.3.2 LoRA modules

In this project, our 'poison' will take the form of a rogue LoRA module. The LoRA module will be a standalone set of weights that will be generated by training on a selected pre-trained model, this model will be referred to as the base model. LoRA weights generated from the base model will be only be compatible with variants of the base model, however, due to how widespread models such as Stable Diffusion are online, these LoRA weights exhibit unprecedented transferability. This simulates the attack environment in the real world, LoRA weights similar to the ones used in the experiment can be found on public forums such as Huggingface [7] or Civitai [1]. The modules themselves are black-box by nature, and the contents cannot be easily analysed. Where we deviate from a real-life scenario is that the LoRA modules used will be *purely* poisoned in order to demonstrate the efficacy of poison, we leave ways to obscure the poisoning for future work. The method of obtaining the training data for the LoRA will be identical to the process described in the formal definition of a dirty-label attack.

3.4 Evaluation criteria

This section will very briefly outline the evaluation criteria that will be used to ascertain the performance of this diffusion model. We will be using CLIP-score, a metric developed by Hessel et. al. to evaluate image captioning [35]. This metric can be applied to evaluate text-to-image diffusion models, and has a high correlation to human judgement. Further details involving the calculations of these values will come in a later section. We'd also like to acknowledge that this for evaluation are not perfect, and they contain flaws [22] which makes the results of this experiment susceptible to bias. In larger scale studies, CLIP-score, along with Fréchet inception distance (FID) [36] and other metrics are used in combination with crowd-sourced human inspection. Due to the scale of this project, only CLIP-score will be employed as a quantitative evaluation method. Measures will be taken to prevent bias in our results, but the subjective nature of image poisoning means that it is still possible that the results may be skewed, further comments on this will be made in section 5.6.

Chapter 4

Implementation

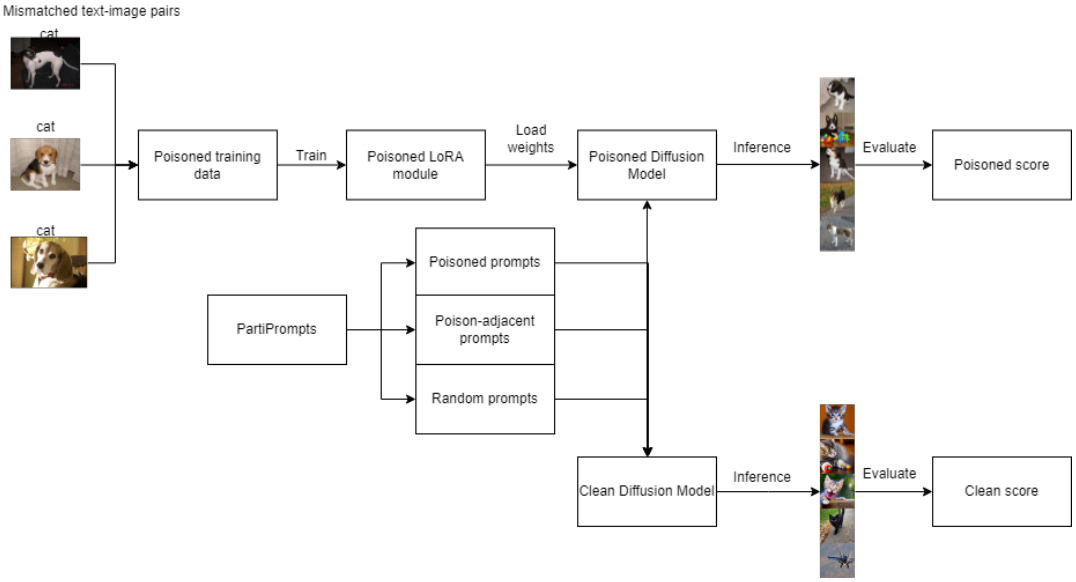
This chapter will detail both the tools used and the experimental setup, including hyperparameters that were altered or kept the same. We will describe the details of how the attack will be executed, and with how many poison samples. Furthermore, we will also describe the chosen evaluation method, and how measurements for each model will be taken. This will follow prior work done in diffusion model development as mentioned in the requirements section [53, 54]. The high-level diagram of the pipeline is shown in Figure 4.1.

4.1 Platform

Jupyter Notebook has been chosen as the preferred platform for this project's experiment pipeline. It has become commonplace in machine-learning projects and we have chosen Jupyter Notebook due to it's great transferability and it's ability to streamline the process while still allowing for full control of the code behind it. More specifically, being able to execute the same process multiple times, while being able to tweak the parameters and models that were involved in the calculations was the main advantage of using Jupyter Notebook. This project has employed the use of two `.ipynb` files (Python Notebooks). Both notebooks are also hosted and available on Google Colab for public viewing and evaluation.

The first notebook contains the pipeline used to train the LoRA, this notebook has not been altered from it's distribution by Taqwa on GitHub [8]. This notebook was originally based on work done by kohya-ss [9], who has provided scripts for fine-tuning Stable Diffusion models. The notebook selected specifically uses the DreamBooth method [54] to produce the LoRA modules.

Figure 4.1: High-level experiment setup



The second notebook contains the pipeline that has been used to evaluate the models. This notebook provides the set of prompts that will be given to the models for inference. The notebook then loads the models into a python environment, and injects the LoRA modules into the selected modules. Finally, inference is run on all the models and the outputs of the each models are evaluated.

4.1.1 Hardware specifications

Due to the large amount of computation power required to train and produce the LoRA modules, as well as to run inference on a multitude of diffusion models, Google Colab has been used for the entirety of the pipeline. The processing power required for the experiment was provided by Google's GPU Python 3 Compute Engine backend environment. The hardware associated with this environment is remotely hosted and includes an NVIDIA Tesla T4 GPU with 12.7 GB of system RAM and 15 GB of GPU RAM. [5]

4.2 Datasets

Two main external datasets were used in this experiment, one was used for providing the images required for the training data and the other was used for generating randomized prompts for the evaluation process

4.2.1 Stanford Dogs Dataset

The goal of this project was to simulate a poisoning of diffusion models and quantitatively evaluate the models' performance before and after poisoning. In doing this, we decide to use simple and most importantly, contrasting concepts which are easy to understand, cats and dogs. Although these concepts are simple, it will be obvious to the human eye when a poisoning has been successful. Namely, when a given model is provided with the prompt "a photo of a cat", but infers an image of a dog, we can clearly see that it has been poisoned. Classifiers have also been developed extensively to recognise the find differences between the features of cats and dogs, this allows us to more accurately quantitatively evaluate our models. In our case pertaining to our formal definition of a dirty-label attack in section 3.3.1, the chosen target concept α was 'cat' and the chosen destination concept β was 'dog'

For the image set I_β we have used the Stanford Dogs Dataset developed by Khosya et. al. [40] The dataset has been developed primarily for fine-grained image categorization. The dataset a variety of images of various breeds of dog, as well as associated labels. The latter component will not play a part in our investigation, as we will be mislabeling our images intentionally. The Stanford Dogs Dataset has been selected as it is contains high-quality images pertaining to the concept dog. In order to entirely encapsulate the concept 'dog', we will choose images randomly from all categories of the dataset (this includes any breed of dog). However, after the random selection, we have curated the selection further to remove images which contain concepts other than the destination concept such as human beings or toys.

4.2.2 PartiPrompts

Ideally, the model should be evaluated across a wide range of prompts, both encompassing and not encompassing the poisoned concept α. The specific prompts and our rationales for choosing this will be further explained in section 4.4.1. Initially while planning this experiment, we planned to use PartiPrompts [13], a public set of prompts in English released by Google Researchers alongside a novel generative model. Sampling from these prompts would allow us to evaluate our models' performances across a wide range of prompts, in doing this, we may take the mean score across these prompts and find how the models perform in both poisoned and non-poisoned environment. However, with further testing, we found that sampling these prompts entirely randomly caused a large amount of noise in our evaluation results. Further explanations on the issues this caused will be found in section 5.6. Finally, we concluded that in order to make the experiment more fair, a fixed set of prompts should be used for all models,

however, this dataset remains in the codebase for future works to reference and use.

4.3 Training

As mentioned in section 4.1, we used Furqanil Taqwa's notebook based off kohya-ss' work on LoRA Dreambooth for diffusion models to produce the LoRA modules (weights) throughout the course of this project. The data provided to the notebooks were provided manually in the form of a `.zip` file. The file contained a set of n images (both `.png` and `.jpg`) taken from the Stanford Dogs Dataset and n labels in the form of `.txt` files.

4.3.1 Labeling

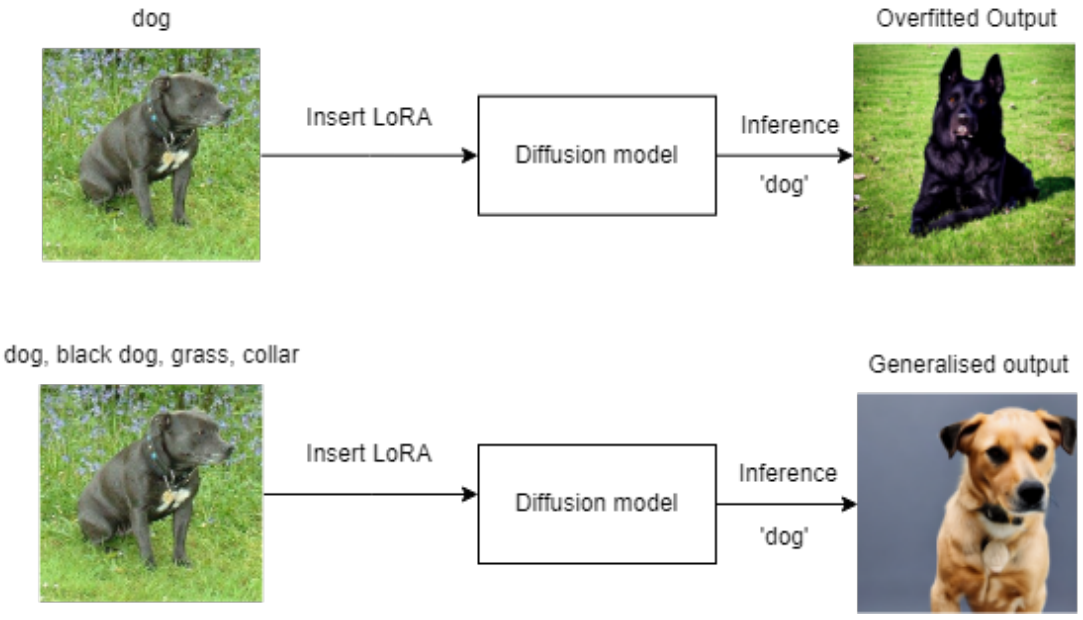
Every label file includes the sole label `cat`. Generally, labels for images in LoRA training includes more detailed descriptions of the what is in images such as `cat, leash, white cat, grass`. this will help LoRA models isolate various concepts as much as possible. Having a small amount of tags will cause the target model to overfit to the training data. As an example, refer to figure 4.2. When a picture of a black dog sitting on grass is fed into a LoRA module with a less detailed label, in this case, `dog`, the model will associate the token `dog` with *specifically* black dogs sitting on grass, as shown with the overfitted output. On the contrary, if labels provided during LoRA training are more detailed, this allows the separation of different features in the image, meaning that it prevents the model from making unnecessary associations. The latter is preferred when training a LoRA for regular purposes.

However, in our case, we are simulating an attack. In line with the goals from section 3.2.2, as an attacker, we want to maximize the range of concepts affected by the poisoning attack. Therefore in our case, we prefer to set vague labels in the form of simply naming our target concept α. In other words, we *want* our model to overfit, as that directly improves the success rate of the attack.

4.3.2 Number of images as independent variables

In section 3.1, we ask what the minimum number of images are required in the training set for a poisoning attack to be effective. The rationale behind why we ask this question is that this helps us gauge the viability of a LoRA attack. Traditionally, referring to Shan et. al.'s work on Nightshade [57], simple dirty-label attacks require upwards of 1000 images to be injected into the target model. The large amount of data required to poison a concept makes dirty-label

Figure 4.2: LoRA Overfitting



poisoning less effective overall as a possible attack. Therefore, we attempt to find the minimum number of images to execute a successful poison attack using LoRA.

We have chosen 3 values of n to test. This is unfortunately not ideal, we would prefer to test every value of n within a range $[1..n]$. However, training n LoRA modules and evaluating each of them after applying them to a model required vastly more computational power than available for the scope of this project. We have opted instead to measure 3 discrete values instead. The chosen numbers for n are as follows:

- 5 images
- 15 images
- 25 images

Furthermore, as the notebook saves a checkpoint every epoch of training, we have also provided modules trained at varying numbers of epoch. This, however, was provided as a further resource for any future work to make use of, and was not used in obtaining the results of this experiment. All our measurements were based on the model trained at 20 epochs.

- 5 epochs
- 10 epochs

- 15 epochs
- 20 epochs

4.3.3 Hyperparameters and configuration for LoRA training

There are numerous hyperparameters available for customisation when training a LoRA module. Due to the scope of this project, we have opted to fix the hyper-parameters listed below. There may be significant changes to the poisoning efficacy if some of these values were to be changed, further observations that we've made regarding this will be in section 5.7. This however will not be tested in this project, and we urge future works to investigate this further. Do note that some variables presented here are not necessarily hyperparameters (such as the base model and VAE), but are included as they were a part of the configuration for training.

Relevant hyperparameters and configurations for training the LoRA in this project have been provided in figure 4.3, the list of parameters are not exhaustive and only include those which are relevant to our study.

4.3.4 Output of training

The output of training is a LoRA module in the form of a `.safetensors` file. All LoRAs obtained in this experiment, regardless of number of samples injected, have a size of 151 MB. We have found that the alterations of certain hyperparameters will affect the training time and size of the output module, this will be further discussed in section 5.7. All LoRA modules are hosted on HuggingFace, and is available for public viewing and usage. [11]

4.4 Model Evaluation

With the LoRA modules now available for use, we will have to evaluate our models before and after the modules are applied. This section will detail the process of generating data to be evaluated and the evaluation process itself.

4.4.1 Prompt categories

In order to test the coverage of our poisoning attack, we aim to choose prompts which both contain or express poisoned concepts as well as non-poisoned ones. In this project, we categorize our prompts into 3 groups.

Figure 4.3: Hyperparameters and configuration for training

```
# Base model configuration
Base model = Stable Diffusion 1.5
VAE = StabilityAI Stable Diffusion VAE
v2 = False
v_parameterization = False

# Dataset configuration
dataset_repeats = 5
activation_word = "cat"
caption_extension = ".txt"
resolution = 512
flip_aug = False
keep_tokens = 0

# LoRA configuration
network_category = "LoRA"
conv_dim = 32
conv_alpha = 16
network_dim = 128
network_alpha = 128
min_snr_gamma = -1
optimizer_type = "AdamW8bit"
train_unet = True
unet_lr = 1e-4
train_text_encoder = True
text_encoder_lr = 5e-5
lr_scheduler = "constant"
lr_warmup_steps = 0

# Training config
lowram = True
sampler = "ddim"
noise_offset = 0.0
num_epochs = 20
vae_batch_size = 4
train_batch_size = 6
mixed_precision = "fp16"
save_precision = "fp16"
save_n_epochs = 1
save_model_as = "safetensors"
max_token_length = 225
clip_skip = 2
gradient_checkpointing = False
gradient_accumulation_steps = 1
seed = -1
```

General prompts

General prompts are prompts used to simulate the general usage of a text-to-image diffusion model. These contain concepts which exist in every-day use of diffusion models. We make sure that these prompts do not contain the chosen poisoned concept, as that would make the prompt a part of a different category. We have sampled these prompts manually from PartiPrompts. [13]

Poisoned prompts

Poisoned prompts are prompts that exxplicitly contain the poisoned token. Generally, our poisoned concept can be summarised by a sequence of one or more words. Poisoned prompts need not contain *only* the poisoned token(s), but the poisoned token(s) must be a part of the prompt. We use these prompts to assess whether the LoRA poisoning has directly affected prompts containing the poisoned concept.

Poison-adjacent prompts

Poison-adjacent prompts do *not* explicitly contain the poisoned concept. We also refer to these prompts as "Adjacent Prompts" for ease of display. These prompts contain concepts

deemed to be "adjacent" to the poisoned concept. A potential problem with this categorization is that this definition of adjacency is largely subjective, therefore we narrow down teh definition of adjacency to synonyms of the selected poisoned token, as well as concepts closely-related in semantics to the poisoned token. We also count compound words which contain the poisoned token without a space separation. Examples of adjacent concepts to `cat` are `feline, kitten, tomcat`. We would like to note that this does not entirely remove the possible subjectivity, but we attempt to reduce it as much as possible. We use these prompts to assess the range of concepts affected by the newly-injected poisoned LoRA module.

4.4.2 Generator seed

In the interest of reproducibility, we have chosen a fixed seed for the images generated in the experiment. This was done using a Pytorch `Generator` object which was manually seeded with integer `0`. In a larger scale study, one may choose to not fix a seed, to evaluate a larger range of model behaviour. However, in that case, the measurements need to be done over much larger than 5 image samples. In our case, we have valued consistency and reproducibility over a larger sample size. This choice has been made due to the limitations on the amount of computation power available to run inference on multiple models at once.

4.4.3 Models

In each trial, we have chosen to evaluate our target poisoned model alongside two 'clean', unpoisoned models. This is to provide a general baseline to compare against. The two control 'clean' models are especially important as there exists prompts which diffusion models inherently find 'difficult'. This is generally due to prompts containing uncommon words or complex concepts which are not easily captioned by a CLIP-classifier (Refer to section 5.6). Therefore it is important to not look at the measurements individually, but rather compare them between each other.

4.4.4 Evaluation metric

To evaluate our models, we have chosen the evaluation metric CLIP-score. CLIP-score was a metric used to evaluate image captioning devised by Hessel et. al. [35] . In our case, we can use it to calculate the alignment between the prompt that has been used to generate the image and the CLIP captioning [50] of the image. The following method closely follows the following works cited [53, 54, 57]. In general usage, the outputs of diffusion models are meant to be seen by

humans. Thus, in evaluating a diffusion model we look to simulate human judgement. Studies involving CLIP-score suggests that it has a high correlation to human judgement, which is the main feature which has lead us to choose CLIP-score as the preferred evaluation metric.

Images generated by the measured diffusion model along with the prompts that were used to generate them will be passed into TorchMetric's CLIP-Score function[18]. An individual CLIP score for an image-prompt pair (I, P) is calculated as follows.

$$\text{CLIPScore(I, P)} = max(100 * cos(E_I, E_P), 0) \tag{4.1}$$

E_I is the visual CLIP embedding of the image while E_P is the textual CLIP embedding (the CLIP embedding of our prompt). We take the cosine similarity between the two values E_I and E_P and bound it between 0 and 100, where 100 is the highest possible performance for our model. We can then take the arithmetic mean for all x images to get a final score for our model. Furthermore, in order to get some insight on how consistent our model performs, we may also calculate the standard deviation of the CLIP-scores that correspond to the model. This is important as CLIP-scores from a given models can vary greatly, which will be discussed further in section 5.

4.4.5 Measurement process details

Each model will generate n images from n prompts. Each image-prompt pair P_{ki} originating from model k where i: $i \in [1..n]$ will have its CLIP-score measurement taken. The mean will be taken over all scores S_{ki} calculated from P_{ki}, as well as its standard deviation. In our case, for each model, as there are 5 prompts for each category, $n = 5$. The degradation in performance will be taken as the **negative** percentage change between the mean of scores from the two clean models, and the singular mean score of the poisoned model. Finally, as mentioned in section 3.1, we have set the threshold for a successful poisoning as a degradation of poisoning $\geq 10\%$. This threshold has been determined by referring to past works done by Shan et. al. [57] as well as visual observation of what constitutes a successful poisoning. To further expand on this, we have found that when a generation from a clean model and a generation from a poisoned model has a clear mismatch, CLIP-score mean decreases by greater than 10%. When a poisoning attack on model γ is successful, model γ is deemed to be **compromised**.

Chapter 5

Results and Evaluation

In this section we present our findings and evaluate the quality of the results given. The simulated attacks on the diffusion models have been largely successful in degrading the performance of a given model on a target concept, and have even produced bleed-through effects onto similar concepts. Worryingly, these results show a promising and accessible method of attacking diffusion models using very few resources with a very high rate of success. However, we stress that these results include a high amount of bias due to an incredibly low sample size, and further investigations must be conducted to verify the reliability of these claims and consistency of this method.

5.1 Poisoning success rate

Our attempts in poisoning diffusion models by injecting LoRA weights have been largely successful. By visual inspection, we have gathered that a complete misclassification of a generated image has been denoted by a 10% or more decrease in CLIP-score in comparison to a corrrect classification. Refer to table 5.1 for our measurements on performance degradation.

We will first reiterate the formula used to calculate these values. Performance degradation

Table 5.1: Performance degradation measurements of clean and poisoned models in %

Prompts \ Model	SD 1.4	SD 1.5	PSD 25*	PSD 15*	PSD 5*
General Prompts	0.99%	-0.99%	0.64%	0.51%	1.83%
Poisoned Prompts	3.18%	-3.18%	23.46%	21.60%	13.26%
Adjacent Prompts	-0.56%	0.56%	13.57%	21.95%	15.89%

*Poisoned models are displayed in the form `PSD (Poisoned Stable Diffusion) num_image_samples`

is calculated by a simple percentage change between the mean CLIP-scores of the clean models and the measured model k. S_k is the CLIP-score attained by a model k. MS_{clean} denotes the mean score of the clean models

$$MS_{clean} = \frac{S_{SD1.4} + S_{SD1.5}}{2} \tag{5.1}$$

$$PerfDeg(k) = -\frac{S_k - MS_{clean}}{MS_{clean}} \times 100 \tag{5.2}$$

As seen in table 5.1, we find that LoRA poisoning has been relatively unsuccessful in degrading the model's performance over general prompts. This is, of course, expected as our focus is on targeting a specific concept α and degrading the model's performance over that specific concept. However, this was effective in being used as a control to see if any major changes took place on our models. Furthermore, this may actually be desired by attackers. This may be how poisoning attacks remain hidden, as the model largely remains unchanged in prompts unrelated to the poison, the poison can only be seen when a specific set of 'trigger' words are introduced in the prompts, leading to poisoning being more difficult to detect. This observed behaviour is very similar to that of 'backdoor' attacks, a well-researched attack on machine learning models. [44]

As we move to testing poisoned prompts, we observe an overall success in model poisoning. Once again referring to table 5.1, all poisoned models saw a 13.26% or greater decrease in performance with prompts explicitly containing the poisoned term. This is an outstanding result as this behaviour has been exhibited for poison sample counts as low as 5. However, performance degradation is notably lower at 5 image samples when compared with 15 and 25 poison samples. Stable Diffusion 1.2 and its subsequent versions have been trained on billions of images in the LAION dataset [16, 17, 56], despite the colossal amount of training data, only 5 images were required to 'overturn' the model's representation of a certain keyword. This is in contrast to the findings of the Nightshade study [57], where 500 to 1000 samples were required to overturn a given concept, another key point worth noting is that Shan et. al. found that there were approximately 2260 clean training samples associated with a given concept. We hypothesize that the lower number of samples required to poison a given concept are due to the effects of fine-tuning and specifically LoRA on a model. The results align with our hypothesis in section 2.3, however, it must be noted that further testing must be done to confirm that this is the cause, as this is merely a possible correlation and there is no guarantee that this is the

actual cause of our findings.

5.2 Bleed-through effects of poisoning

Per our third question in our problem statement, we also perform tests involving concepts which we deem are 'adjacent' to the poisoned concept. This means words or phrases which are synonyms, closely related, or compound words which contain the poisoned token. Once again referring to table 5.1, we once again see an overwhelmingly high attack success rate when the poisoned models are tested against poison-adjacent prompts. The decrease in performance for poisoned models range from 13.57% to 21.95% . It is interesting however that the model with 15 poison samples performed worse than the model with 25 poison samples to a relatively large degree, performing almost equivalently to the model with 25 poison samples on explicitly poisoned prompts. We cannot deduce the cause of this, but due to a low sample size, this outlier can be due to simply noise.

This result proves that the effects of the LoRA attack reach further than only the poisoned prompts, but extends to various concepts surrounding the poisoned concept. We believe that this is a significant result, but requires further testing as only a limited number of poison-adjacent prompt were tested, therefore we do not entirely know the further concepts that are affected by the attack. We hypothesize that the bleed-through effects of poisoning are due to how the base model was initially trained. As an example, take an image of a small cat, among the labels for this image, we are likely to find the labels `cat`, `kitten` and `feline`. As these tokens are commonly found together, an internal association is formed between the words, leading to them also being affected by the poisonous LoRA.

Compound words as adjacent prompts

A special comment can be made on compound words as an adjacent prompt. We specifically tested the term `cat` with words containing 'cat', such as `bobcat` or `tomcat`. We have found that the words tested were minimally affected by the poisoning. Refer to figure 5.1 for example results on the prompt `bobcat`. We see that the generations are largely unaffected despite explicitly containing the word cat. This, however, is an expected result, due to the fact that our text-to-image models detect what to generate by tokenizing certain words, the token `cat` and token `bobcat` are completely distinct tokens. This also would mean that words completely unrelated to `cat` but containing 'cat' are also unaffected, unless they are semantically related. For example, the word `catacombs` remain unaffected as the poisoned concept and the word are

(a) Image generated by clean model

(b) Image generated by poisoned model

Figure 5.1: A clean and poisoned model's interpretation of the prompt 'a bobcat'

semantically unrelated. In conclusion, we may say that words containing the poisoned token *without* a space separation are not more or less likely to be affected by poison, the only factor that affects whether or not the word is poisoned is the likelihood of that word is found to be in the same original training samples as the poisoned word.

Correlation with subsequent studies

Our findings involving bleed-through effects to related concepts are closely aligned with those found by Shan et. al. in the Nightshade study. [57] A measurement used in the aforementioned study is the L2 distance D to a poisoned concept, this gave a more quantitative measurement of *how* adjacent the a given concept is to another. This was done by mapping the concepts onto a feature space and measuring the L2 distance between them. This measurement would have been beneficial to our experiment, but however was not implemented due to time constraints. Similarities between our findings include the fact that synonyms and neighbouring concepts were also affected by the adversarial attack. However, unlike the results shown on table 5.1 and table 5.2, there was a clear correlation between the number of poison samples and how much the neighbouring concepts were affected. This was not the case for our experiments due to the aforementioned anomaly where the model with 15 poison samples performed markedly worse than the model with 25 samples. Furthermore, another area of testing which Shan et. al. has done that we have not done was testing on concepts which were far away in word-space, but closely related semantically. An example of this was how the term `fantasy art` was affected when the target poisoned concept was `dragon`.

Table 5.2: CLIP-score measurements of clean and poisoned Stable Diffusion models

Prompts \ Model	SD 1.4	SD 1.5	PSD 25*	PSD 15*	PSD 5*
General Prompts	29.187	29.771	29.290	29.330	28.941
Poisoned Prompts	29.510	31.450	23.328	23.897	26.439
Adjacent Prompts	32.456	32.093	27.897	25.1907	27.147

*Poisoned models are displayed in the form `PSD (Poisoned Stable Diffusion) num_image_samples`

5.3 Magnitude of degradation in performance of models

In this section we briefly outline the magnitude of the effects of poisoning on poisoned prompts and poison-adjacent prompts. We will be referring to the values on table 5.2. Due to the poisoned prompts including the directly targeted word, it is expected that CLIP-score would fall greatly below the values that were attained by the clean models. In our case, the mean score achieved by the clean models were 30.480, whereas the poisoned models attained scores from 23.328 to 26.439. We see a clear correlation where when there are only 5 poison samples inserted, the score decrease is up to 10% lower than the maximum number of samples we tested, 25. These results align with an intuitive correlation where more poison samples lead to a worse performing model, however, these affects appear to be diminishing, with 15 samples performing only slightly worse than 25 samples. This suggests that increasing the poison data may cause the performance of a given model to approach a certain limit. This, however, was not investigated in our experiment and the maximum number of samples remains at 25. With these comments made, the difference in performance between the model with 15 samples injected and the model with 25 samples injected are small enough to be negligible and can be passed off as noise/randomness in the measurements.

As we move to looking at poison-adjacent prompts, the bleed-through effects of the poison are clear. However, by simply inspecting the CLIP-score we see that the poisoning effects are noticeably less significant when compared to tests with explicitly poisoned prompts. Once again, the abnormal measurement on adjacent prompts on the model with 15 samples injected should be mentioned, we are unsure of the reason of why this model performed significant worse than both the other models, even worse than that on explicitly poisoned prompts at 5 samples. Overall, as we will see in the upcoming section 5.4, we may conclude that although models perform noticeably worse on concepts similar to the target poisoned concept compared to regular, untargeted concepts. However, intuitively, they will not perform worse than concepts explicitly poisoned.

Figure 5.2: A **clean** model's interpretations of 'clean' prompts

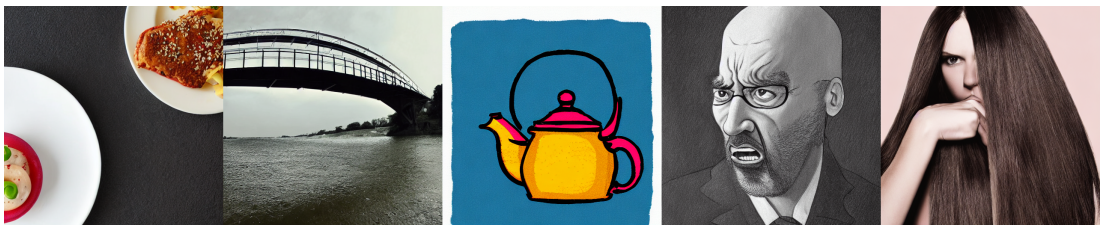


Figure 5.3: A **poisoned** model's interpretations of 'clean' prompts

Figure 5.4: A **clean** model's interpretations of poisoned prompts involving 'cat'

Figure 5.5: A **poisoned** model's interpretations of poisoned prompts involving 'cat'

Figure 5.6: A **clean** model's interpretations of poison-adjacent prompts similar to 'cat'

Figure 5.7: A **poisoned** model's interpretations of poison-adjacent prompts similar to 'cat'

5.4 Visual inspection of poisoned images

A rather unsuccessful example of poisoning was previously displayed in section 5.2. Here we present more successful examples of images generated by poisoned models and compare them with images generated by clean models. In doing this, we hope to show qualitatively the effects of poisoning, and what a target of poisoning might see when attempting to generate an image using poisoned or poison-adjacent prompts.

General prompts

We will only briefly comment on the effects of poisoning on randomized (not poisoned) prompts, this is because there are no noticeable differences between these images between the poisoned and non-poisoned models. However, this proves that it is possible for poison to go completely undetected if the trigger word is never found. This is a worrying property of poison as it is possible for particularly obscure poisoned concepts to go completely unchecked through quality control processes, this makes a possibility where poisoned models may be distributed publicly without the knowledge of poisoning. Unsuspecting users who happen upon the poisoned concept may be subject to offensive or crude imagery being generated upon an otherwise inoffensive prompt. For reference, the set of prompts used for the set of images on figures 5.2 and 5.3 are: 'a picture of food on plate', 'a bridge', 'an illustration of a teapot', 'an angry man' and 'a woman with long hair'.

Poisoned prompts

As we move to the effects of poisoning on explicitly poisoned prompts, the effects of poisoning rapidly become observable. Our diffusion models begin to struggle to generate coherent images with what are usually every-day, common prompts. We are able to make this assertion as our clean models generate satisfactory images with no clarity issues, where as our poisoned models can not do the same. Once again, in our experiment, we use the concept `cat` as our baseline, we see that in figure 5.4, the chosen concept is shown with high clarity. With this, it is clear how the CLIP-score measurements were relatively high for the clean models.

We now refer to figure 5.5, we see a significant degradation in quality as we prompt the poisoned models with prompts including the term 'cat'. Some images see a complete misalignment, where images of animals resembling dogs are generated. However it should be noted on the 2nd image, where the model was prompted with 'a photo of a cat playing with toys', a recognisable image of a cat is still generated, showing that the poison is not all-encompassing on

prompts involving the poisoned concept. This seems to however be unpredictable and random. The 4th and 5th image is also worth noting, as these images do not see a clear misalignment (a clear image of a dog being generated rather than a cat) but rather significant artifacting, leading to misleading or unrecognizable images. Specifically on the final image, some 'cat-like' features remain, but overall the image is not recognized and captioned appropriately by the CLIP-classifier.

Poison-adjacent prompts

From the CLIP-score evaluations, we can observe that poison-adjacent prompts are affected to a lesser degree than explicitly poisoned prompts. Per figure 5.6 and 5.7, we can further confirm this to be the case. Much like the poisoned prompts, the effect of the poison is not consistent. For example, the first image sees a complete misalignment, where the image generated is clearly that of a puppy. It is interesting that the semantics of how a prompt 'kitten' generates an image of an animal similar to a puppy, rather than a full-grown dog, as that was what the poisoned module was trained on. However, in images 2 and 3, we see images which clearly show an animal with cat-like features, showing that the poison has not seeped through fully to the neighbouring concepts at 25 poison sample. Once again, we assert that the lower efficacy of the poison is not necessarily a bad property, as this makes the poison harder to predict, detect, and defend against.

Conclusions to be drawn from observing images

We can draw some conclusions from visually inspecting the images of the clean and poisoned models. In alignment with our definition of a successful attack outlined in section 4.4.5, we see the following symptoms in the models which have been deemed **compromised**:

i. A complete misalignment between the generated images and prompts containing target concept α or of concepts adjacent to α

ii. Heavy artifacting in generated images, leading to malformed or unrecognisable images.

We hypothesize that symptom ii. is caused by the injected poisoned data 'confusing' the diffusion model by attempting to overwrite already existing knowledge about concept α, and the artifacting is the model attempting to combine the two concepts together. Therefore, then, it is possible that the former symptom is the poison successfully overwriting the existing knowledge, changing the representation of α entirely.

Table 5.3: Standard deviation values of clean and poisoned Stable Diffusion models

Prompts \ Model	SD 1.4	SD 1.5	PSD 25*	PSD 15*	PSD 5*
General Prompts	2.351	2.311	2.651	3.127	3.513
Poisoned Prompts	2.342	2.002	5.292	4.540	3.585
Adjacent Prompts	2.355	0.795	3.149	4.192	3.125

*Poisoned models are displayed in the form `PSD (Poisoned Stable Diffusion) num_image_samples`

5.5 Poisoning affecting model consistency

We use standard deviation as a measurement of how consistently a given model performs. We will briefly comment on the effects of poisoning on how consistently a model performs. From table 5.3, we observe that the introduction of poison samples via a LoRA module makes a model's performance more inconsistent. This aligns with our previous comments in section 5.4 about how the effects of poison on a model are unpredictable and at times not noticeable. The high standard deviation may signify that it is possible for a poisoned not exhibit any signs of poison (an example of this is shown on the 2nd image on figure 5.5) on one prompt, while still exhibiting heavy artifacting and misalignment on another (e.g. 5th image on figure 5.5). This inconsistency is most noticeable by a standard deviation of 5.292 on poisoned prompts when 25 poison samples are injected, where regularly, a standard deviation of around ~ 2.2 is expected on clean models. Similarly to the performances of the models, this effect is seen to a lesser extent on Poison-adjacent prompts.

Although this may be an interesting observation, further testing across a larger sample size is required to make confident conclusions surrounding the consistency of the models, due to the large amount of noise in the data. Many anomalies can be seen in the measurement of standard deviation, such as an exceedingly low value of 0.795 for Stable Diffusion 1.5's performance on Adjacent prompts. We also see not a large difference in consistency on all 3 types of prompts on the model with 5 samples injected, this is not the behaviour we expected due to the expectation of having a higher inconsistency when poisoned concepts or adjacent concepts are involved in prompts.

Despite these observations being merely preliminary, we can perhaps utilize the unpredictable nature of poison to our advantage. Poisoning is hard to detect with a single measurement of CLIP-score or by inspecting a singular image generated by a model. It may be possible to use a measurement such as standard deviation of performance to detect when a model has been poisoned. An intuitive approach would be to flag a model as potentially poisoned if it

exhibits a higher degree of inconsistency in performance. This allows us to detect poison when the concepts poisoned are too obscure or there is too few poison samples to fully affect the entire set of prompts relating to concept α

5.6 Evaluation and critique

In this project, we have been successful in providing a proof of concept where LoRA modules can be used as a form of malware to attack diffusion models. However, due the available resources and the scope of the projet, we have had to cut corners due to the lack of computational power, leading to it being hard to quantify *how* much poisoning can adversely affect the models. We, however, believe that this proof of concept is significant and serve as a starting point for future works to extend and improve upon. This section will detail some of the criticisms and problems concerning the results of our study.

Low sample size

Diffusion models, due to their method of operation, inherently involve a large amount of randomness [37, 53]. Therefore in studying them, we look to use large sample sizes in order to evaluate them to prevent bias and noise in quantitative results. Unfortunately, this exact factor is what is lacking in our experiment. Due to the lack of data points, a large amount of noise is present in our data, and it shows in the form of various anomalies which do not comply to expected correlations. Of course, we have no way in knowing if these 'anomalies' are in fact anomalies, or if they actual do allude to a more interesting correlation.

Failure to find limits

We have been successful in answering *if* LoRA can be used as an adversarial attack, but have not been successful in finding just *how* effective it is and how little data is required. Through our results, we may deduce that there there is a limit to the amount of performance degradation that occurs to a model as the number of poison samples increases, we were not able to find this limit, however. The same is true in the opposite direction. We believe that the fact that poisoning was possible with only 5 poison samples was a remarkable finding, however, we also believe that it is possible to push this number even lower, either with LoRA or other PEFT methods. In that sense, we were also unable to find the minimum number of images required to achieve a successful poisoning, although 5 was the lowest number of samples we tested, we believe poisoning could perhaps be possible with just one image.

Subjectivity in determining adjacent prompts

We have also proven that prompts do not need to explicitly include the target concept to be affected by poison. Although this is a significant observation, there is much work to be done in this respect. As mentioned before, the Nightshade study done by Shan et. al. [57] has been more thorough in their analysis of bleed-through according to poison, much of this comes from a quantitative measurement of the closeness of words are to each other when projected onto wordspace. In doing this, we would be able to find how the effect of poisoning changes as we move further away in wordspace. However, in our experiment, we only determine arbitrarily what words are considered "adjacent" to our prompts, this is subjective and lacks rigor. Some words we may have considered to be close in wordspace may end up being not close at all and vice versa. Our definition of 'adjacent' was also limited to synonyms and similar words, but the effects of poison may reach further than just related words, much like how in the Nightshade study found that 'dragon' bled-through to 'fantasy art'. In that sense, we have also failed to assess just *how* far the poison spreads.

Inherent prompt bias

In our experiment, we have fixed our prompts in the interest of keeping the experiment fair, this, however, has inadvertently lead to inherent bias in the values measured. The issue here lies in how 'difficult' text-to-image diffusion models find a given prompt. We have found that in the set of prompts that can be given to a text-to-image diffusion models, there exists prompts which diffusion models inherently find 'hard', meaning that they consistently score relatively poorly (CLIP-score) on these prompts than more standard ones. Extreme examples of this include abstract concepts as prompts such as 'anger' or 'betrayal', which leads to CLIP-score of below 20. Upon this observation, we may deduce that every prompt carries an inherent 'difficulty', therefore any set of prompts we choose is likely to have a chance to skew the average CLIP-score of a model up or down, leading to bias depending on prompt selection. A way to rectify this would be to evaluate the models across a larger range of prompts, which leads us back to the prevously mentioned issue of exceedingly low sample sizes.

This is clearly an issue with our methodology and we believe that future studies should be mindful of the 'difficulty' factor of prompts, or simply increase the sample size if the resources available permits it. However, we have been aware of this issue and have made changes accordingly to our methodology to compensate. Namely, we have decided to make our measurement metrics for performance degradation to be relative to the clean model, rather than look at the

CLIP-score values themselves, this relative and percentage based metric slightly alleviates the bias and adequately allows us to see which attacks were successful and which were not.

5.7 Possible extensions

The scale of this experiment has been relatively small, and has served as mostly a proof of concept. The many possible extensions of this study may aim to rectify the many biases and issues outlined in the aforementioned section. Furthermore, we also detail some observations from our experiment that we believe are worth investigating further.

Increasing sample size

As mentioned, the biggest drawback of our methodology was the incredibly low number of samples. Future studies may aim to evaluate models across a larger range of prompts and at a larger quantity of prompts, this will more accurate show the various correlations that exist between poison and model performance. In our experiment, we have chosen 3 discrete poison quantities to evaluate models at, a future experiment may want to test a continuous amount of poison samples $[1..n]$, this will allow a continuous plot more clearly displaying the relationships involved in poisoning.

Quantify word correlation

Once again, as we have failed to quantify the correlation between given words that we have deemed to be 'adjacent' concepts, a future study may do a more rigorous experiment investigating the bleed-through effects of LoRA poisoning. Such an experiment would use methods similar to those used in the Nightshade study to quantify the adjacency of words, and would be successful in showing the correlation between word or concept adjacency and poisoning.

Testing other poisoning methods

Our experiment used the most basic form of poisoning, namely dirty-label poisoning. In a commerical-scale model training setting, dirty-label poisoning is easily defended against by simply screening the training data beforehand and looking for mismatches, or even having a CLIP-classifier of one's own to label the images. Further testing should be done with more complex, harder to defend against poisoning methods to test the efficacy of using LoRA in these novel contexts. A candidate for this would be Nightshade, a study that this project has drawn much inspiration from. Nightshade has been shown to be effective in regular training

environemnts, it would be valuable to test how Nightshade interacts when used with fine-tuning and more specifically with PEFT methods such as LoRA.

Further testing involving hyperparameters

We have decided to fix the hyperparameters when training the LoRAs for our experiment as our focus was on testing the plausibility of poisoning using LoRA as well as the number of samples that were required for poisoning. However, with some preliminary testing, we have found that adjusting some hyperparameters have large effects on the training process as a whole, as well as the effects of poisoning.

In particular, we notice that the network dimension as well as the network alpha parameter (`network_dim` and `network_alpha` in the source code) plays a large role in the efficacy of the poison. It also majorly affects the filesize of the LoRA module produced after training. A module trained at a network dimension and network alpha value of `128` had a file size of 151 MB while a module trained at a network dimension and network alpha of `32` was only 38 MB. Through our testing, we hypothesize that setting this hyperparameter lower allows the model to deviate from the injected LoRA samples, while a higher value of makes the model more strictly adhere to the provided samples. Similar informal studies have come to the same conclusion [39]. Thus, we hypothesize that higher values of this hyperparameter will lead to stronger poison, as the model will be forced to associate with the LoRA material more strongly than existing knowledge of the same concept. Of course, this is a mere hypothesis and requires further investigation, which we urge future works to do.

Defense and detection against LoRA poisoning

LoRA as an adversarial attack is a novel concept with little work done so far. LoRA module themselves are black-box by nature and the training data used to train a given module can be kept private if the author decides to do so. This gives rise the question, how do we know if a LoRA module is poisoned or not? Unknowing users may fall victim to malicious LoRA modules that are disguised as clean ones. Due to this risk, we believe work should be done on methods to detect if a LoRA module is being poisoned, or to defend and remove poison from concepts.

Extension to other PEFT methods

We have chosen LoRA as the PEFT method to test in our experiment, but this is far from the only method that exists for efficient fine-tuning of deep learning models. Attempting to

poison training data for other PEFT methods may yield different, and unique results from the findings in our study. Furthermore there also exists other variants of LoRA which may also yield different interactions with poisoned data, such as those mentioned in section 2.2.1.

Chapter 6

Legal, Social, Ethical and Professional Issues

Data poisoning is a process that can be used to sabotage machine learning models that are to be used for well-meaning purposes. Thus, it is of course possible that the techniques that are presented in this project are to be used by malicious actors to produce and distribute malware that will affect innocent users. This section will outline the possible applications of LoRA poisoning and ways that it can be misused and the ethical and societal issues that come with it.

6.1 LoRA modules as malware

As seen in the results presented in section 5.1, LoRA poisoning is highly effective in significantly reducing the performance of a given model, if a victim were to unknowingly apply the malware LoRA module to their model. We believe that the most worrying factor about LoRA poisoning is its accessibility, as shown in section 4, all the tools that were used for the attacks performed in this experiment are available to the public free of charge. This leads to a greatly increased likelihood of it being used for unethical purposes. Furthermore, even the computation power used for training the LoRA are provided by Google Colab or similar services remotely [4], not even requiring the potential attacker to own hardware capable of training LoRAs.

Effects on commercial-scale diffusion models

Despite what is mentioned above, we predict that LoRA poisoning will not have a large impact on commercial-scale diffusion models. This is due to the fact the LoRA is not used by large-scale models. Rather, it was developed as a way to customize a larger scale model to tailor it to a more niche set of requirements. Furthermore, it is unlikely that developers of large-scale models will be using LoRA modules obtained from external means. LoRA modules are relatively computationally cheap to train, thus it is likely that even if LoRA were to be used in the fine-tuning of a large scale model by an organization, it would all be done in-house.

This is in contrast to other poisoning methods such as Nightshade [57], which is being used by artists as a form of copyright protection, defending against usage of their artwork in training diffusion models without permission.

Effects on independent creators

We believe that those who would be most affected by LoRA poisoning are independent creators, looking to customize a diffusion model for their purposes. As mentioned in section 2.2.1, LoRA modules are openly distributed on public forums, independent artists may access and download these modules to apply to their own models. These modules are black-box by nature and may contain poisoned data, which will cause unexpected behaviour from the model it is applied to. Although this was not tested in our experiments, we believe it is also possible that a LoRA may contain both legitimate data and poisoned data, in the case that this is possible, the legitimate data can be used to disguise the poisoned data, making it more likely for users to download the module.

With the advent of LoRA and data poisoning, we urge users to check that they are downloading LoRA weights from trusted sources, and that they contain desired information. This will prevent unexpected and possibly offensive behaviour from models when they are applied. We also urge future works to study how poison can be detected within LoRA modules, and public forums such as Civitai [1] to screen uploaded modules to ensure safety for all users on their website.

6.2 Censorship

Up until now we have mainly considered LoRA as a malicious act which will affect users without their knowledge. However, we will also consider the possibility that LoRA poisoning may be

used intentionally to steer model generations away from unwanted topics. In other words, if we use the LoRA to target concept α, and cause it to generate images of concept β instead. If the poison is effective enough, we effectively prevent models from generating concept α, censoring the target concept.

This usage of LoRA may be used by model distributors that do not want their models to be used for certain purposes. Of course, this is a double-edged sword, as this can perhaps be used for censorship of political subjects or other sentitive subject matters causing a conflict of interest and compromising freedom of creation. On the other hand, this may also be a useful method for model distributors to prevent models from being used to generate crude, offensive, or NSFW (sexually explicit content) content at a low cost with high efficacy.

6.3 Environmental impacts

As with every computational process that requires a large amount of resources, we must consider the environmental impact and carbon footprint of training LoRA modules. However, although we believe that this is an important consideration to make, this is an issue of lesser significance. This is because LoRA, unlike other training methods, is a PEFT method, aimed to reduce the computational power as much as possible. Thus, the resources used by LoRA training are magnitudes lower than traditional machine learning training.

Chapter 7

Conclusion

We conclude this report by reflecting on our problem statements defined in section 3.1. In this project, we have proved LoRA to be a plausible tool to attack diffusion models and presented the process of creating the LoRA modules used in the attacks. We also provided the methodology behind evaluating these models. From our results, both quantitatively and visually, we have proven beyond a reasonable doubt that when injected with a malicious LoRA module, the performances of diffusion models can significantly decrease and they can produce incorrect or incoherent images. This was successful in answering our first problem statement, asking whether LoRA was a plausible method of attack against a diffusion model. The effects of poisoning were found to be inconsistent and unpredictable, but this is expected due to the random nature of generative diffusion models. We have found that any number in the range between 5 and 25 poison samples are sufficient to cause a noticeable degradation in performance of the models, a remarkably low number of samples. However, we failed to find the lowest number of images required to achieve such an effect. Thus, due to no conclusive limit being found, we were only partially successful in answering our second question. Furthermore, we have also proved that the effects of LoRA are far-reaching, extending past merely the chosen poisoned concept. This partially answers our third and final question, asking how prompts similar to the poisoned concept will be affected. This question was only partially answered as we failed to quantify the relationships between concept adjacency and model performance.

We believe that LoRA modules can be used for malicious purposes, and the accessibility of the tools required for the creation of such a malware is incredibly high. Thus, we urge users to be wary and make sure that they download and only LoRA modules from reputable sources. However, this technique may prove useful for niche purposes, such as for model distributors

to keep their users safe from the generation of offensive, or crude images. Overall, we believe further work should be done investigating data poisoning in fine-tuning, as the impacts of data poisoning will only increase further as deep learning models become more available and accessible to independent users. We think our work can be used as a starting point in further works concerning LoRA and other fine-tuning methods as a form of adversarial attack.

Bibliography

[1] Civitai. https://civitai.com/. Accessed: 2024-03-25.

[2] Dall-e. https://openai.com/research/dall-e. Accessed: 2023-12-11.

[3] Designs.ai. https://designs.ai/. Accessed: 2024-03-20.

[4] Google colab: Frequently asked questions. https://research.google.com/colaboratory/faq.html. Accessed: 2023-12-11.

[5] Google colab nvidia t4 specification. https://colab.research.google.com/github/d2l-ai/d2l-tvm-colab/blob/master/chapter_gpu_schedules/arch.ipynb. Accessed: 2023-12-11.

[6] Google imagen. https://imagen.research.google/. Accessed: 2023-12-11.

[7] Huggingface: Evaluating diffusion models. https://huggingface.co. Accessed: 2023-12-11.

[8] Kohya lora dreambooth. https://github.com/Linaqruf/kohya-trainer/tree/main. Accessed: 2023-12-11.

[9] Kohya-ss stable diffusion scripts. https://github.com/kohya-ss/sd-scripts. Accessed: 2023-12-11.

[10] Lora ai-generated images — the future of art. https://medium.com/@novita.ai/lora-ai-generated-images-the-future-of-art-0461917976a0. Accessed: 2024-03-21.

[11] Lora poisoned weights by quan tran. https://huggingface.co/quantran03/poison_test/tree/main. Accessed: 2023-12-11.

[12] Midjourney. https://www.midjourney.com/. Accessed: 2023-12-11.

[13] Pathways autoregressive text-to-image model. https://github.com/google-research/parti. Accessed: 2023-12-11.

[14] Sora. https://openai.com/sora. Accessed: 2024-03-20.

[15] Stable diffusion. https://stablediffusionweb.com/. Accessed: 2023-12-11.

[16] Stable diffusion 1.5. https://huggingface.co/runwayml/stable-diffusion-v1-5/. Accessed: 2023-12-11.

[17] Stable diffusion 1.5 blogpost on huggingface. https://huggingface.co/blog/stable_diffusion. Accessed: 2023-12-11.

[18] Torchmetrics: Clip-score. https://lightning.ai/docs/torchmetrics/stable/multimodal/clip_score.html. Accessed: 2023-12-11.

[19] Rameen Abdal, Peihao Zhu, John Femiani, Niloy J. Mitra, and Peter Wonka. Clip2stylegan: Unsupervised extraction of stylegan edit directions, 2021.

[20] C. Amisse, Mario Jijón-Palma, and Jorge Centeno. Fine-tuning deep learning models for pedestrian detection. *Boletim de Ciências Geodésicas*, 27, 06 2021.

[21] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022.

[22] Ali Borji. Qualitative failures of image generation models and their application in detecting deepfakes, 2023.

[23] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[24] Eva Cetinic, Tomislav Lipic, and Sonja Grgic. Fine-tuning convolutional neural networks for fine art classification. *Expert Systems with Applications*, 114, 07 2018.

[25] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering, 2018.

[26] Weixin Chen, Dawn Song, and Bo Li. Trojdiff: Trojan attacks on diffusion models with diverse targets, 2023.

[27] Sheng-Yen Chou, Pin-Yu Chen, and Tsung-Yi Ho. How to backdoor diffusion models?, 2023.

[28] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. Cogview: Mastering text-to-image generation via transformers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 19822–19835. Curran Associates, Inc., 2021.

[29] Mohamed Elasri, Omar Elharrouss, Somaya Ali Al-Maadeed, and Hamid Tairi. Image generation: A review. *Neural Processing Letters*, 54:4609–4646, 2022.

[30] Yarden Frenkel, Yael Vinker, Ariel Shamir, and Daniel Cohen-Or. Implicit style-content separation using b-lora, 2024.

[31] 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.

[32] Zeyu Han, Chao Gao, Jinyang Liu, Jeff, Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey, 2024.

[33] Melissa Heikkilä. This new data poisoning tool lets artists fight back against generative ai. *MIT Technology Review*.

[34] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning, 2022.

[35] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning, 2022.

[36] 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, 2018.

[37] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[38] 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.

[39] Ashe Junius. Alpha and dimensions: Two wild settings of training lora in stable diffusion. `https://ashejunius.com/alpha-and-dimensions-two-wild-settings-of-training-lora-in-stable-diffusion-d7ad3e3a3b0a`. Accessed: 2023-12-11.

[40] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

[41] Olha Kurinna. The future of fashion: a closer look at virtual try on. `https://www.apptension.com/blog-posts/virtual-try-on`. Accessed: 2024-03-20.

[42] Yanzhou Li, Tianlin Li, Kangjie Chen, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. Badedit: Backdooring large language models by model editing, 2024.

[43] Hongyi Liu, Zirui Liu, Ruixiang Tang, Jiayi Yuan, Shaochen Zhong, Yu-Neng Chuang, Li Li, Rui Chen, and Xia Hu. Lora-as-an-attack! piercing llm safety under the share-and-play scenario, 2024.

[44] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.

[45] Zeyu Liu, Souvik Kundu, Anni Li, Junrui Wan, Lianghao Jiang, and Peter Anthony Beerel. Aflora: Adaptive freezing of low rank adaptation in parameter efficient fine-tuning of large models, 2024.

[46] Sujith Kumar Mandala. A contextualized survey on the state and future directions of diffusion models. 2023.

[47] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Fil-

ipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[48] Sayak Paul Pedro Cuenca. Using lora for efficient stable diffusion fine-tuning. `https://huggingface.co/blog/lora`. Accessed: 2024-03-20.

[49] Rushi Qiang, Ruiyi Zhang, and Pengtao Xie. Bilora: A bi-level optimization framework for overfitting-resilient low-rank adaptation of large pre-trained models, 2024.

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

[51] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1060–1069, New York, New York, USA, 20–22 Jun 2016. PMLR.

[52] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the*

IEEE/CVF conference on computer vision and pattern recognition, pages 10684–10695, 2022.

[53] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[54] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation, 2023.

[55] Simo Ryu. Low-rank adaptation for fast text-to-image diffusion fine-tuning. `hhttps://github.com/cloneofsimo/lora`. Accessed: 2023-12-11.

[56] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa 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.

[57] Shawn Shan, Wenxin Ding, Josephine Passananti, Haitao Zheng, and Ben Y. Zhao. Prompt-specific poisoning attacks on text-to-image generative models, 2023.

[58] Sayak Paul Sourab Mangrulkar. Peft: Parameter-efficient fine-tuning of billion-scale models on low-resource hardware. `https://huggingface.co/blog/peft`. Accessed: 2024-03-20.

[59] Edna Chebet Too, Li Yujian, Sam Njuki, and Liu Yingchun. A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, 161:272–279, 2019. BigData and DSS in Agriculture.

[60] Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning, 2023.

[61] 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, 2019.

[62] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022.

[63] Mingyang Yi, Jiacheng Sun, and Zhenguo Li. On the generalization of diffusion model, 2023.

[64] Shengfang Zhai, Yinpeng Dong, Qingni Shen, Shi Pu, Yuejian Fang, and Hang Su. Text-to-image diffusion models can be easily backdoored through multimodal data poisoning, 2023.

[65] Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. Composing parameter-efficient modules with arithmetic operations, 2023.

[66] Shuai Zhao, Leilei Gan, Luu Anh Tuan, Jie Fu, Lingjuan Lyu, Meihuizi Jia, and Jinming Wen. Defending against weight-poisoning backdoor attacks for parameter-efficient fine-tuning, 2024.

[67] Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. Loraretriever: Input-aware lora retrieval and composition for mixed tasks in the wild, 2024.