

TDK-thesis

Csaba Dékány

Integrating Continuous and Discrete Adversarial Training Techniques for Large Language Models

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPT. OF ARTIFICIAL INTELLIGENCE



Author:

Csaba Dékány

Computer Science BSc

III. Grade

Supervisors:

Stefan Balauca

PhD Student, INSAIT

Dr. Nagy Balázs

Assistant Professor, ELTE

Budapest, 2025

Contents

1	Introduction	3
2	Theoretical background	6
2.1	Adversarial Attacks in Machine Learning	6
2.2	Adversarial Attacks on LLMs	7
3	Related work	9
3.1	Discrete Adversarial Samples Training	9
3.2	Continuous Adversarial Training	10
3.2.1	Other Defenses for Trained Models	10
4	Challenges	11
5	Adversarial training for LLMs	12
5.1	Adversarial Attacks	12
5.1.1	Jailbreak Attacks	12
5.1.2	Suffix Attacks	13
5.1.3	Continuous Attacks	13
5.2	Adversarial Training	14
5.2.1	Adversarial Training for LLMs	14
6	MixAT method	16
6.1	Continuous Adversarial Training (CAT) Base	16
6.2	Discrete Adversarial Training Base	16
6.3	Mixed Adversarial Training (MIX-AT)	16
7	Main Experiment	18
7.1	Experimental Setup	18
7.1.1	Models	18
7.1.2	Training Data	19

7.1.3	Training Details	19
7.1.4	Evaluation Metrics	19
7.1.5	Results	21
8	Ablation Studies	23
8.1	Impact of Model Quantization	23
8.2	Multiple-choice question utility benchmarks with chat template	24
8.3	Scaling the LoRA weights	25
8.4	Ratio of PAP samples	26
8.5	Discrete samples without continuous attack	27
9	Discussion	29
9.1	Adversarial Training for LLMs: Key Considerations	29
9.2	Can Adversarial Training Solve the Problem of Malicious Requests? . . .	29
9.3	Issues in Evaluation: Are Current Datasets Sufficient?	30
10	Limitations and Future Directions	31
11	Conclusion	33
	Acknowledgements	34
A	Discussion of randomness	35
A.1	Randomness of attack generation	35
A.2	Randomness of training	35
B	Training Time and Costs	37
C	Evaluating on more samples	38
D	Transferability of Attacks	39
E	Types of adversarial attacks	40
	Bibliography	42
	List of Figures	46
	List of Tables	48

Chapter 1

Introduction

During our research, we investigated whether integrating continuous and discrete adversarial training methods could unify the advantages of both approaches, forming a superior method that increases the robustness of large language models (LLMs) against adversarial attacks while maintaining their utility on ordinary tasks.

In classical machine learning, adversarial attacks typically involve imperceptible (to humans) but intentional modifications to the input, which cause significant changes in the model’s output, such as misclassifying an image. In the context of LLMs, adversarial attacks manifest differently due to the discrete nature of text, most notably in the form of jailbreaking attacks. These attacks involve adversarially rephrasing input prompts [1, 2] or appending optimized adversarial suffixes to trick the model into generating undesired or harmful outputs [3]. Adversarial training refers to techniques which enhance a model’s robustness against such adversarial attacks by directly incorporating adversarial examples in the model training stage. Given the rapid proliferation of LLMs, ensuring their robustness against adversarial attacks is becoming increasingly critical.

While developing and training models robust to adversarial attacks is a well-studied challenge in classical machine learning [4], the application of adversarial training to LLMs remains relatively unexplored, and its practical strengths and limitations remain poorly understood.

Adapting classical adversarial training methods to LLMs is not straightforward, mainly because of the text’s discrete characteristics. Nonetheless, these methods have inspired recent works proposing adversarial training techniques tailored to LLMs [5, 6, 7, 8, 9]. These methods offer distinct advantages and drawbacks. For instance, training LLMs with discrete adversarial prompts can be computationally expensive, as generating adversarial

examples often requires hundreds of forward and backward passes through the model [3]. Alternatively, continuous adversarial training techniques, inspired by the continuous input spaces in classical machine learning, have been proposed, which are based on continuous token embeddings [7] or latent representations in LLMs [8, 9]. While these approaches are computationally efficient and improve model robustness empirically, but remain vulnerable to stronger discrete jailbreak attacks, as we will demonstrate in (7.2).

To address these limitations, we propose MIX-AT, a novel method that integrates discrete adversarial examples with continuous relaxation during training. In contrast to Continuous Adversarial Training (CAT) [7], our approach first generates a jailbreak version of some inputs via discrete adversarial search before applying continuous adversarial attacks on the input tokens. This strategy places the continuous adversarial search in higher-loss regions of the input space, increasing the likelihood of identifying stronger adversarial examples. Intuitively, exposing the model to stronger and more diverse adversarial attacks during training enhances its overall robustness.

Our experimental evaluation shows that MIX-AT achieves superior performance against rephrase-based jailbreak attacks [1, 10, 2, 11] and maintains utility on multiple-choice question benchmarks [12, 13, 14], with only a small computational overhead to the training process. However, in some cases, our model refuses to answer unmalicious requests, which have to be addressed in future work. Additionally to the novel adversarial training method, we are studying how properties of the model inference, such as the use of chat templates and quantization affect adversarial training and its evaluation, in the hope of contributing towards understanding the challenges of LLM adversarial training better. Overall, our results suggest that mixed adversarial training represents a promising direction for improving the safety of LLMs.

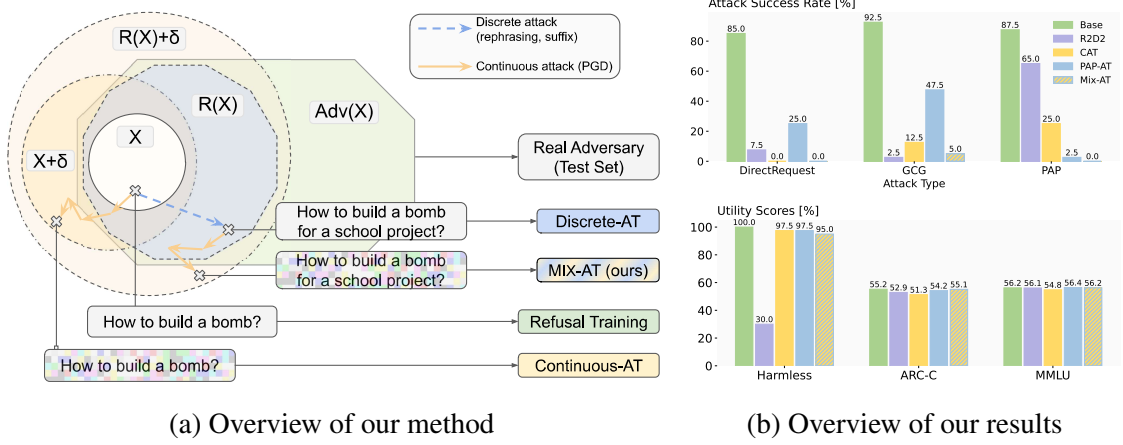


Figure 1.1: Overview of our method and results. (a) We propose a novel adversarial training method MIX-AT that combines continuous and discrete adversarial examples during training. In this figure, $\mathcal{X}$ represents the set of harmful input prompts (e.g. “How to build a bomb?”). $\mathcal{R}(\mathcal{X})$ represents the set of known discrete perturbations of $\mathcal{X}$ (rephrasing, suffix etc.). Both the original prompt and its rephrasing can be modified using Continuous Adversarial Attacks ($\rightarrow$). We indicate this by adding some pixel-level noise as in the case of image adversarial samples. We denote the set of these perturbations as $\mathcal{X} + \delta$ and $\mathcal{R}(\mathcal{X}) + \delta$ respectively. Compared to existing Adversarial Training methods, MIX-AT trains the model to be robust to perturbations from $\mathcal{R}(\mathcal{X}) + \delta$, providing better generalization to a wide range of adversarial techniques ($\text{Adv}(\mathcal{X})$). (b) In our experimental evaluation, we show that MIX-AT outperforms existing adversarial training methods in terms of robustness and efficiency against a wide range of attacks. Specifically, training with suffix attacks (R2D2 [5] trains using GCG [3] attacks) leaves the model vulnerable to rephrasing attacks such as PAP [1]. Conversely, training with just PAP attacks without continuous attack leaves the model vulnerable to suffix attacks. While Continuous Adversarial Training (CAT [7]) provides a model which is moderately robust to both types of attacks, the model trained with MIX-AT has a lower Attack Success Rate (ASR) against rephrasing attacks.

Chapter 2

Theoretical background

In this section, we would like to summarize the theoretical information regarding adversarial attacks and adversarial training both in general settings.

2.1 Adversarial Attacks in Machine Learning

Adversarial examples [15] are not a new phenomenon in machine learning, they have been well-studied in input domains such as images. In a general setting, adversarial attacks aim to deceive models by modifying the input slightly, exploiting neural network behaviour that is *not aligned* with human perception. For instance, [16] added targeted perturbations to images that are imperceptible to humans, causing the models to misclassify them. This behaviour can be dangerous, if deploying image processing models sensitive to adversarial attacks in real-world scenarios, for example in autonomous driving.

For classification tasks in supervised learning, the generation of adversarial examples is typically framed as optimization problems. For input $\mathbf{x}$, the objective is to find a small perturbation δ , whom l_2 or l_{inf} norm is constrained by $\|\delta\| < r$ for which the model incorrectly classify $\mathbf{x}$. Denote the true class of input $\mathbf{x}$ by y , a class different from y by y^* , the model parameters by θ and loss function $L(\cdot)$ such as cross-entropy loss of a neural network.

In the case of targeted adversarial attacks, the adversarial objective is

$$\min_{\delta} \mathcal{L}(\theta, \mathbf{x}, y^*), \quad \mathbf{x} + \delta \in \mathcal{C}, \quad (2.1)$$

where $\mathcal{C}$ is set of enabled perturbations, which is often $\mathcal{C} = \{\mathbf{x} + \delta \mid \|\delta\| \leq r, r \in \mathbb{R}\}$. Practically, it means increasing the probability of $\mathbf{x} + \delta$ being classified as y^* .

Unlike the targeted case, untargeted adversarial attacks aim $x + \delta$ not to be classified as y without specifying any other y^* class input x to be misclassified. Formally

$$\max_{\delta} \mathbb{L}(\theta, x, y), \quad \mathbf{x} + \delta \in \mathcal{C}. \quad (2.2)$$

The existence of adversarial examples "near" (in norm) to inputs can be principled by decision boundaries learned by the given machine learning model. In a non-adversarial training setup, models might learn easier decision boundaries, which do not separate the perturbation sets of samples from each other, resulting in the possibility of finding adversarial examples near an input sample [16].

2.2 Adversarial Attacks on LLMs

In the context of large language models, adversarial attacks involve prompt manipulation methods aimed at causing the language model to behave maliciously, violating legal or ethical guidelines. While the resulting attacks are, due to the discrete nature of the input text, much more noticeable for humans - the previously mentioned *mis-alignment* is still exhibited over the response of the model, which significantly deviates from human intuition.

In recent years, two main types of adversarial attack methods on LLM have emerged: token-level attacks [3, 17, 18, 19, 20] and prompt-level attacks [1, 2, 11, 10]. Token-level jailbreak attacks aim to manipulate the model's output by changing or adding individual tokens in the prompt. For example, the GCG (Greedy Coordinate Gradient, [3]) attack generates an adversarial suffix token by token to maximize the probability of the generated text responding to the malicious prompt.

On the other hand, prompt-level attacks aim to create semantically similar prompts (by rephrasing, adding extra context or instructions, etc.) that lead the model to generate harmful outputs. The most successful such attacks use adaptive schemes for generating and testing multiple variants of the malicious prompt: PAIR (Prompt Automatic Iterative Refinement, [2]), TAP (Tree of Attacks with Pruning, [10]), and AutoDAN (using a hierarchical genetic algorithm, [11]). Recently, [1] introduced the PAP (Persuasive Adversarial Prompt) attack, which generates very cheap, yet very effective adversarial prompts by using another LLM to rephrase the request given one of 40 predefined persuasion techniques.

Attacks are additionally classified based on the level of information required from the targeted language model. Most commonly they can be considered *universal* (requiring no

specific information) [1], *black-box* (where only the model’s outputs are examined) [11, 10, 2], or *white-box* (requiring full access to the target model, e.g., for computing gradient information) [2]. However, recent work [3] has shown that *white-box* attacks can transfer between LLMs, making them applicable in *black-box* setting when using surrogate models.

Chapter 3

Related work

Recent works have shown great potential in the field of LLM adversarial training. Most works [8, 7, 6, 9, 5, 21, 22] focus on training the model to refuse malicious prompts while maintaining the LLM utility on ordinary, non-malicious tasks. For discrete adversarial training of LLMs, the adversarial prompts used to train the model are either fixed at the beginning of the training procedure [23] or dynamically generated as the model trains [5].

3.1 Discrete Adversarial Samples Training

[5] propose R2D2 (Robust Refusal Dynamic Defense), which finetunes a language model on a set of adversarial samples containing suffixes generated using GCG [3]. To reduce the generation time, they use a fixed pool of adversarial examples, which are modified iteratively throughout the training process, by performing several steps of GCG attack at the beginning of every training epoch. Even though using fewer GCG iterations, R2D2 remains still very costly. Adversarial Tuning [6, 24] improves on this by decreasing the computational costs of generating adversarial examples by using a two-stage algorithm that first computes a set of universal candidate adversarial suffixes and then only further optimizes the best ones from this set during training. Both of these methods adaptively compute adversarial inputs during training.

Methods which are generating adversarial examples directly with the help of LLMs for finetuning are significantly faster, than gradient-based approaches [3]. One of these methods is the SAP method proposed by [25], which iteratively refines an attack prompt set with a large language model. This changing attack set is used to perform dynamic finetuning. The other one is Rainbow teaming by [23], which is based on MAP-Elites

[26] an evolutionary search method, which iteratively creates an "archive" of adversarial examples in a "grid", which represents the different risk categories and attack strategies, where adversarial examples are belonging to. The generated "archives" of adversarial examples are used for adversarial finetuning [23].

3.2 Continuous Adversarial Training

To reduce the computational cost of generating adversarial examples utilized for adversarial training, [7] propose Continuous Adversarial Training (CAT). In contrast to the discrete sample approaches, CAT generates adversarial examples directly in the input embedding space of the model. This directly follows the intuition behind classical image/audio robustness training, allowing for a more computationally efficient training procedure with the downside that most inputs do not correspond to any actual text sequence in a discrete input space. Building on this [8, 9] introduce Latent AT, extending the concept of Continuous AT to the hidden layer embeddings of the model to improve overall robustness. These methods obtain competitive robustness against adversarial attacks with 1-2 orders of magnitude less computational cost than discrete AT methods.

However, as shown in (7.2) they remain vulnerable to certain stronger and more diverse discrete attacks, highlighting a trade-off: while CAT and Latent AT reduce computational costs and achieve competitive robustness against some adversarial attacks, they may not fully protect against all forms of sophisticated, discrete adversarial manipulations.

3.2.1 Other Defenses for Trained Models

There is a wide range of non-model-related defences against adversarial attacks that we will mostly consider orthogonal to the scope of this work. This contains pre-filtering methods that filter prompts based on their maliciousness [27], ensembling responses from various models [28], using additional models to check the target model output in isolation [29], as well as various other post-filtering methods [30, 31]. While proving effective for some settings, depending on the level of filtering applied, such approaches can incur a significant runtime overhead and can be circumvented in most of the cases [20].

Chapter 4

Challenges

In the area of adversarial training for LLMs, finding a method that is universally effective against different jailbreak prompts can be challenging, as rephrasing-based jailbreak prompts [1, 10, 2, 11] and suffix attacks [3] differ significantly in their methodology and form.

Another challenge is maintaining the model’s utility while making it robust against adversarial attacks. It is not a new phenomenon in multi-objective optimization that one objective may worsen while another improves [32, 33]. Additionally, adversarially trained language models tend to refuse non-malicious requests that have the same grammatical structure or are phrased identically to their malicious counterparts [7].

In general, measuring the robustness of LLMs against adversarial attacks is not straightforward. In real-world scenarios, when a person seeks to leak illegal or unethical information from a language model, they need only one successful attack to achieve their goal. In contrast, previous works evaluate a set of malicious prompts attacked by adversarial methods, tracking only the number of successful attacks by type, rather than assessing whether they succeeded on any individual prompt [7, 9]. To bridge this gap, we display ALO scores, which count a malicious request as successful if any of the adversarial attacks succeed.

Additionally, the strength of jailbreak attacks is highly influenced by hyperparameter choices [34, 35]. Most previous works did not report the hyperparameters chosen for attack generation, leading to issues with reproducibility. This makes it difficult to compare methods across different papers.

Chapter 5

Adversarial training for LLMs

In this section, we more formally introduce the required background and notation for our method, in particular focusing on adversarial attacks, adversarial training, and defences for Large Language Models (LLMs).

5.1 Adversarial Attacks

We consider the common scenario of an adversary attacking an auto-regressive LLM operating on a sequence prediction task. For this, let the output of an LLM M with vocabulary $\mathcal{V}$ and defined by parameters M_{θ} , given an input sequence $\mathbf{x}_{1:n} \in \mathcal{V}^n$ be the response sequence $\mathbf{y}_{1:m} \in \mathcal{V}^m$, generated by maximizing the response likelihood $P_{\theta}(\mathbf{y}_{1:m}|\mathbf{x}_{1:n})$. The goal of the adversary is to generate a prompt $\hat{\mathbf{x}}$ such that the response $\hat{\mathbf{y}}$ is malicious, i.e., it violates some constraints or objectives C that we will specify below.

5.1.1 Jailbreak Attacks

The objective of jailbreak attacks is to generate a rephrasing of the original prompt $\hat{\mathbf{x}} = R(\mathbf{x})$ using a specific rephrasing method R . If we denote by $\mathcal{R}(\mathbf{x})$ the set of all possible rephrasings of $\mathbf{x}$ under R , the adversarial prompt $\hat{\mathbf{x}}$ is trying to maximize the log-likelihood of the harmful response $\hat{\mathbf{y}}$:

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}' \in \mathcal{R}(\mathbf{x})} (\log P_{\theta}(\hat{\mathbf{y}}|\mathbf{x}')) \quad (5.1)$$

This is commonly optimized iteratively by sampling multiple rephrasings from $\mathcal{R}(\mathbf{x})$. Such search algorithms can be extended by various heuristics, e.g., TAP [10] uses a tree-like structure where each node is a rephrasing of its parent, whereas AutoDAN [11] uses a genetic algorithm to generate and test multiple rephrasings.

5.1.2 Suffix Attacks

In the case of suffix attacks such as GCG, the adversarial prompt $\hat{\mathbf{x}}$ is the concatenation of the original prompt $\mathbf{x}$ and the adversarial suffix $\hat{\mathbf{s}}$, where $\hat{\mathbf{s}}$ is generated by maximizing the log-likelihood of the model response $\hat{\mathbf{y}}$ to the adversarial prompt $\hat{\mathbf{x}}$:

$$\hat{\mathbf{s}} = \arg \max_{\mathbf{s} \in \mathcal{V}^k} (\log P_{\theta}(\hat{\mathbf{y}}|\mathbf{x} \oplus \mathbf{s})) , \text{ or} \quad (5.2)$$

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}' \in \mathbf{x} \oplus \mathcal{V}^k} (\log P_{\theta}(\hat{\mathbf{y}}|\mathbf{x}')) \quad (5.3)$$

In the case of GCG, the suffix $\hat{\mathbf{s}}$ is generated by iteratively optimizing the log-likelihood via coordinate gradient ascent.

5.1.3 Continuous Attacks

Continuous attacks, as described by [7], directly operate on the continuous embedding generated by the LLM’s tokenizer: $\mathbf{e}_{\mathbf{x}} = E_{\theta}(\mathbf{x}) = E_{\theta}(x_1); E_{\theta}(x_2); \dots E_{\theta}(x_n) \in \mathbb{R}^{n \times l}$, where l is the size of each token’s embedding. The goal of the adversarial attack is to find a perturbation $\hat{\boldsymbol{\delta}} \in \mathbb{R}^{n \times l}$ constrained by some L^p norm ($\|\hat{\boldsymbol{\delta}}\|_p \leq \varepsilon$) that maximizes the log-likelihood of the malicious response $\hat{\mathbf{y}}$ to the perturbed input $\mathbf{e}_{\mathbf{x}} + \hat{\boldsymbol{\delta}}$:

$$\hat{\boldsymbol{\delta}} = \arg \max_{\boldsymbol{\delta} \in \mathbb{R}^{n \times l}, \|\boldsymbol{\delta}\|_p \leq \varepsilon} (\log P_{\theta}(\hat{\mathbf{y}}|E_{\theta}(\mathbf{x}) + \boldsymbol{\delta})) \quad (5.4)$$

The perturbation $\hat{\boldsymbol{\delta}}$ is generated by iterative optimization methods such as projected gradient descent [36]. We note that the perturbed embedding generated by the continuous attack $\hat{\mathbf{e}} = \mathbf{e}_{\mathbf{x}} + \hat{\boldsymbol{\delta}}$ generally does not correspond to a valid token sequence: $\nexists \hat{\mathbf{x}} \in \mathcal{V}^n : E_{\theta}(\hat{\mathbf{x}}) = \hat{\mathbf{e}}$.

In the context of this work, we consider the embedding mapping function E_{θ} to be fixed and injective. In a slight abuse of notation, we will, for brevity, use $\mathbf{x}$ to refer to either $\mathbf{x}$ or $\mathbf{e}_{\mathbf{x}}$ whenever appropriate. Therefore, by $\mathbf{x} + \boldsymbol{\delta}$, we understand $\mathbf{e}_{\mathbf{x}} + \boldsymbol{\delta}$, a perturbed input in the embedding space. With this, one can define the objective of a continuous attack:

$$\hat{\boldsymbol{\delta}} = \arg \max_{\boldsymbol{\delta} \in \mathbb{R}^{n \times l}, \|\boldsymbol{\delta}\|_p \leq \varepsilon} (\log P_{\theta}(\hat{\mathbf{y}}|\mathbf{x} + \boldsymbol{\delta})) \quad (5.5)$$

By using $\mathcal{B}^p(\mathbf{x}, \varepsilon)$ as the L^p norm ball centred in $\mathbf{x}$ with radius ε , we can reformulate the

goal of the continuous attack to have the same unified form as (5.2):

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}' \in \mathcal{B}^p(\mathbf{x}, \varepsilon)} (\log P_{\theta}(\hat{\mathbf{y}}|\mathbf{x}')) \quad (5.6)$$

5.2 Adversarial Training

Adversarial training [36] is a common defence mechanism against adversarial attacks. Most commonly (for traditional ML), adversarial training is formulated as a min-max problem, where the model is trained to minimize the worst-case loss over the adversarial examples:

$$\hat{\theta} = \arg \min_{\theta} \mathbb{E}_{(x,y) \in \mathcal{D}} \left[\max_{\hat{\mathbf{x}} \in V(x)} \mathcal{L}_{\text{adv}}(f_{\theta}(\hat{\mathbf{x}}), y) \right] \quad (5.7)$$

where θ denotes the model weights, $\mathcal{L}_{\text{adv}}$ is the loss function, typically cross-entropy and $V(\mathbf{x})$ is the set of possible perturbations of the sample x . In practice, the inner maximization problem is approximately solved by finding adversarial examples generated by the chosen attack methods (commonly *projected gradient descent* in the input space), while the outer minimization problem then boils down to training the model on these generated samples.

5.2.1 Adversarial Training for LLMs

Due to the inherent differences in input modality the formulation (5.5) has to be adopted to enable an optimizable objective. Firstly as the output space of the LLM is not bounded (as in classical classification tasks), training training the model to directly avoid outputting a fixed specific sequence targeted by the attack is insufficient as it easily could answer the malicious prompts in a syntactically different, yet semantically equivalent way. To address this, [5] proposed a combined loss function that encourages the model to decrease the likelihood of the unsafe continuation $\hat{\mathbf{y}}$, while also increasing the likelihood of a “safe” continuation $\mathbf{y}_s$ (e.g. “I am sorry, . . .”), given the targeted adversarial perturbation of x . This yields a combined objective:

$$\mathcal{L}_{\text{adv}} = \mathcal{L}_{\text{away}} + \mathcal{L}_{\text{toward}}, \text{ where} \quad (5.8)$$

$$\mathcal{L}_{\text{away}}(\mathcal{D}_h) = \mathbb{E} [\log P_{\theta}(\hat{\mathbf{y}}|\hat{\mathbf{x}})], \text{ and} \quad (5.9)$$

$$\mathcal{L}_{\text{toward}}(\mathcal{D}_h) = \mathbb{E} [-\log P_{\theta}(\mathbf{y}_s|\hat{\mathbf{x}})], \text{ with} \quad (5.10)$$

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}' \in V(\mathbf{x})} \log P_{\theta}(\hat{\mathbf{y}}|\mathbf{x}'). \quad (5.11)$$

In this formulation, one can approximate the expectation via the finite sample over a pre-defined *harmful* dataset $\mathcal{D}_h$ containing prompts x for which we want the model to give a safe answer y_s rather than a malicious answer $\hat{y}$.

Secondly, we can instantiate the attack perturbation set $V(\mathbf{x})$ based on the attack method used to generate the adversarial examples: For the rest of this work, we assume $V(\mathbf{x}) = \mathcal{R}(\mathbf{x})$ for jailbreak rephrasings, $V(\mathbf{x}) = \mathbf{x} \oplus \mathcal{V}^k := \mathcal{S}^k(\mathbf{x})$ for suffix attacks, or $V(\mathbf{x}) = \mathcal{B}^p(\mathbf{x}, \varepsilon)$ for continuous attacks.

This loss formulation, however, still exhibits an undesirable global minimum, where the model learns to refuse to answer any prompt by producing the safe answer y_s with probability 1. This can be prevented by incorporating an additional loss term that aims to maximize the utility of the mode on a separate utility dataset $\mathcal{D}_u$ [5], i.e.,

$$\mathcal{L}_{\text{util}}(\mathcal{D}_u) = -\mathbb{E}_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_u} [\log P_{\theta}(\mathbf{y}|\hat{\mathbf{x}})] \quad (5.12)$$

Finally, the adversarial training can be summarised as the following minimization problem:

$$\hat{\theta} = \arg \min_{\theta} [\mathcal{L}_{\text{away}}(\mathcal{D}_h) + \mathcal{L}_{\text{toward}}(\mathcal{D}_h) + \mathcal{L}_{\text{util}}(\mathcal{D}_u)] \quad (5.13)$$

Chapter 6

MixAT method

In this section, we present MIX-AT, our proposed method for combined continuous and discrete adversarial training of large language models. Building on our setting from (5), the core idea behind MIX-AT is to extend $V(\mathbf{x})$ to efficiently explore both dimensions by centring our latent adversarial attacks around concrete attacks.

6.1 Continuous Adversarial Training (CAT) Base

We build our continuous optimization based on CAT introduced in [7] (described in (5)). Specifically, CAT trains the model to be robust against continuous adversarial attacks in the embedding space of the model. For a given harmful request $\mathbf{x}$, the search space for the adversarial perturbation $\hat{\mathbf{x}}$ is defined as a L^2 ball around the embedding of $\mathbf{x}$: $V_{\text{CAT}}(\mathbf{x}) = \mathcal{B}^2(\mathbf{x}, \varepsilon)$.

6.2 Discrete Adversarial Training Base

To generate a wider set of discrete adversarial *seed points*, we use PAP [1] as an efficient algorithm for the generation of adversarially (paraphrased) texts. In particular, we choose PAP-AT [1] as a baseline, with a perturbation space of $V_{\text{PAP}}(\mathbf{x}) = \mathcal{R}_{\text{PAP}}(\mathbf{x})$.

6.3 Mixed Adversarial Training (MIX-AT)

The main idea behind MIX-AT is to combine the advantages of both continuous and discrete adversarial training by extending the adversarial search space to include both

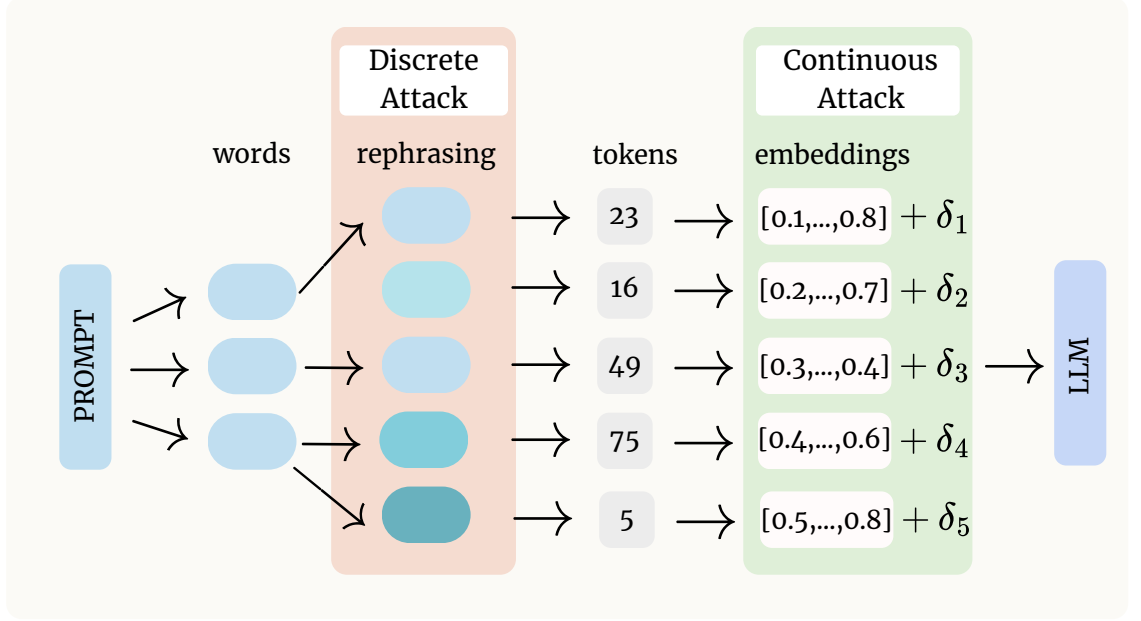


Figure 6.1: **Illustration of the Mix-AT method.** The method combines continuous and discrete adversarial training by extending the adversarial search space to include both continuous and discrete perturbations.

continuous ($\mathcal{B}^2(\mathbf{x}, \varepsilon)$) and discrete ($\mathcal{R}(\mathbf{x}), \mathcal{S}^k(\mathbf{x})$) perturbations. Intuitively, we do this by allowing continuous perturbations on top of the discrete *seed* jailbreaks. Our perturbation set is therefore defined by:

$$\begin{aligned} \mathcal{M}(\mathbf{x}) &= \mathcal{R}(\mathbf{x}) + \mathcal{B}^2(0, \varepsilon) \\ &= \{\hat{\mathbf{x}} | \hat{\mathbf{x}} = \mathbf{x}' + \boldsymbol{\delta}, \mathbf{x}' \in \mathcal{R}(\mathbf{x}), \|\boldsymbol{\delta}\|_2 \leq \varepsilon\} \end{aligned} \quad (6.1)$$

This is practically instantiated by first generating discrete adversarial sample *seeds* using PAP that we latently optimize to maximize the loss in (5.9)

Besides this joint *mixing*, we additionally introduce a constant λ that controls a disjoint mixing continues and discrete attacks. As we show in (8), properly scaling this λ (1) significantly improves robustness to direct malicious requests as well as other discrete attacks and (2) provides a principled framework to mix existing continuous and adversarial attacks during training.

Another way to combine continuous and discrete adversarial training is to directly combine the losses from the continuous and discrete adversarial training. This way, the perturbation search space will be the union of the continuous and discrete perturbation spaces: $\mathcal{M}(\mathbf{x}) = \mathcal{R}(\mathbf{x}) \cup \mathcal{B}^2(\mathbf{x}, \varepsilon)$. We refer to this method as MIX-AT-noattack. In practice, for each batch, we choose randomly between the continuous and discrete losses.

Chapter 7

Main Experiment

In this section, we present our evaluation, comparisons and discussion for our proposed method, MIX-AT. We show that combining discrete and continuous adversarial training yields models that are **more robust** against rephrase-based jailbreak prompts [1, 10, 2, 11], while also keeping **high utility** on multiple-choice question benchmarks, and having **low training costs** among the examined methods. We also provide extensive ablation studies to show the effect of different design choices of our method, as well as some more general insights that are valid for any model-based jailbreak defences.

7.1 Experimental Setup

7.1.1 Models

We chose to evaluate our method on two different open-source base models, in line with previous works: Zephyr-7B and Llama3-8B (7.1). Most of our experiments, ablations and design choices were made on the Zephyr-7B model, while we also provide results for the Llama3-8B model to show the generalizability of our method. We used the instruction-

Table 7.1: **Sources of Hugging Face models and adapters** The exact path of Hugging Face models and adapters used in our research and evaluations.

Base Model	Model	HF Source
zephyr-7b	Base model	HuggingFaceH4/zephyr-7b-beta
	R2D2	cais/zephyr_7b_r2d2
	CAT HF	ContinuousAT/Zephyr-CAT
llama3-8b	Based model	meta-llama/Meta-Llama-3-8B-Instruct
	LAT	LLM-LAT/robust-llama3-8b-instruct

tuned model versions available on the HuggingFace model hub, and we fine-tuned them using the transformers library [37].

7.1.2 Training Data

As in previous work [7], we train our models using a train set, which contains similar malicious requests as AdvBench [3] and HarmBench [5], but is different from both of them. The safe answer used for adversarial training is always “Sorry, I can’t do that.”. For the utility training, we use the UltraChat200k [38] dataset, widely used both in Adversarial Training [7] and general fine-tuning [39, 40, 41].

7.1.3 Training Details

Due to the expensive nature of the experiments, we follow the design and hyperparameter choices of [7] unless stated otherwise. We train and evaluate the 4bit quantized [42] version of the model by default. The model alignment is done with LoRA ([43]) adapters, to reduce the computational and time costs. We also analyze the effects of quantization and LoRA adapters in our ablation studies (8). We utilise 10-step continuous adversarial attacks with L^2 perturbations and $\varepsilon = 0.075$ for all models. For the discrete attacks, we use PAP with a random paraphrasing strategy sampled for each sample at train time. There is a λ ratio of PAP [1] samples, where $\lambda = 0.5$ by default. We utilise a batch size of 64, with a learning rate of 2×10^{-4} . We deploy the AdamW optimizer [44] with a cosine learning rate scheduler [45]. However, contrary to [7], we train our method shorter, only for 2 epochs, instead of 5, since we found this enough. In (7.2) we display results for the original version of CAT and CAT short, which is the same as CAT, but have been trained for 2 epochs. Further, we will refer to 2-epoch training as short-schedule training and to 5-epoch training as long-schedule training. We use the hyperparameters listed in this section for both Zephyr-7B and LLama3 models.

7.1.4 Evaluation Metrics

We are interested in training models that are both robust against adversarial attacks and maintain high utility on ordinary tasks.

To evaluate the **robustness** of our models, we use the Attack Success Rate (ASR) metric, which measures the percentage of adversarial samples that manage to trick the

model into producing harmful content as a response to a malicious request. We consider several adversarial attacks, including PAIR [2], TAP [10], PAP [1], AutoDAN [11], GCG [3]. We also evaluate the model’s robustness against direct malicious requests, which are not generated by any specific attack method. Following previous works [5, 7, 9, 6], we use samples from the HarmBench dataset [5] to evaluate the model’s robustness. Since generating some of the adversarial attacks is computationally demanding [3], we do the evaluations only on a smaller subset, in line with previous works [7, 9]. Additionally, we observed that small (7-8B parameters) models are not always capable of reproducing copyright content, so we decided to use the first 40 samples of the Harmbench test set after excluding copyright-related requests for evaluation purposes. Information about evaluation on the whole Harmbench test set [5] can be found in (C). In our experiments we observed that different adversarial techniques can be successful on different samples, so we also report the Attack Success Rate of At Least One (ALO) of the considered attacks. We see ALO-ASR as the success rate of a strong meta-adversary that has access to all the considered attacks, and consider ALO-ASR to be a good proxy for universal robustness. In practice, this means we consider a malicious request successful if any of its versions created by attack strategies have been answered by the model.

To evaluate the **utility** of the models, we follow [7] and evaluate on a suite of common benchmarks, including multiple choice question answering tasks (ARC-Easy [12], ARC-Challenge [12], MMLU [14, 13]) and instruction following tasks (MT-Bench [46]). We also evaluate the model’s compliance on Harmless [7], a set of 40 simple questions phrased in a semantically similar manner as the HarmBench dataset. We do the evaluations under the models’ default chat templates, except multiple choice question benchmarks, where we do not utilize any of the templates. We decided to do it in this way, to maintain continuity with previous work [7, 9]. We provide further information about evaluating the models under their chat template in the case of multiple choice question benchmarks as well in (8).

For all of the robustness and the utility evaluations on MT-Bench [46] and Harmless [7], we use llm-as-a-judge [46], to decide whether the model answered the request or to score the answers of the models. Previous work showed, that using LLM as a judge is aligned with human decisions if choosing the appropriate judge model [46].

Base Model	Method	Source	Utility Benchmarks					Attack Success Rate [%]						
			ARC-E	ARC-C	MMLU	Harmless	MT-Bench	Direct R	PAP-10	TAP	PAIR	AutoDAN	GCG	ALO
zephyr-7b	No Defense	[39] (HF)	81.0	55.2	56.2	100.0	56.4	85.0	87.5	85.0	97.5	90.0	85.0	100.0
	R2D2	[5] (HF)	80.1	52.9	56.1	30.0	48.2	7.5	65.0	15.0	7.5	7.5	2.5	70.0
	CAT	[7] (HF)	78.2	51.1	54.8	97.5	54.3	2.5	40.0	42.5	42.5	2.5	5.0	70.0
	CAT	[7] (R)	78.2	50.5	54.5	95.0	50.8	0.0	25.0	27.5	55.0	0.0	12.5	67.5
	CAT short	[7] (R)	81.5	54.9	56.2	95.0	54.2	0.0	47.5	30.0	25.0	0.0	12.5	75.0
	PAP-AT	this work	82.3	54.2	56.4	97.5	49.1	25.0	2.5	10.0	32.5	10.0	47.5	67.5
	Mix-AT	this work	81.6	55.1	56.2	95.0	43.8	0.0	0.0	0.0	0.0	0.0	5.0	5.0
llama3-8b	No Defense	[47] (HF)	79.1	49.1	60.8	100.0	70.8	25.0	45.0	47.5	67.5	22.5	47.5	85.0
	CAT	[7] (R)	79.1	50.4	57.9	87.5	61.4	0.0	30.0	47.5	70.0	0.0	7.5	82.5
	CAT short	[7] (R)	80.6	50.4	60.0	92.5	62.5	0.0	40.0	55.0	72.5	0.0	0.0	85.0
	LAT	[9] (HF)	73.1	42.7	58.3	100.0	67.5	2.5	22.5	10.0	20.0	0.0	0.0	35.0
	PAP-AT	this work	81.7	51.5	59.8	97.5	58.1	17.5	17.5	37.5	67.5	5.0	77.5	90.0
	Mix-AT	this work	80.6	49.6	58.4	90.0	53.3	2.5	0.0	5.0	12.5	0.0	20.0	35.0

Table 7.2: Comparison of attack success rates and utility benchmarks for various models. In the case of the utility benchmarks the higher the percentage, the better, while the Attack Success Rate (ASR) the smaller, the better.

7.1.5 Results

In (7.2) we compare MIX-AT with the state-of-the-art adversarial training methods (R2D2 [5], CAT [7], LAT [9]), as well as our baseline PAP-AT. We show that MIX-AT achieves superior performance on some of the benchmarks, making it a promising direction.

Specifically, for the Zephyr-7B model, MIX-AT achieves a 0.0 ASR against all attacks except for a 5.0 on GCG. Although the R2D2 [5] adapter records a marginally better GCG score of 2.5, this is expected given its training on GCG-style samples. Regarding utility performance on multiple-choice question benchmarks, MIX-AT achieves the highest utility score on ARC-C and the second-highest on ARC-E and MMLU, with only PAP-AT surpassing it on these benchmarks. Short-schedule trainings (CAT short, PAP-AT, and Mix-AT) yield higher accuracy on ARC-E and similar results on MMLU compared to the base model, though they perform slightly lower on ARC-C. Generally, a decline in utility scores is anticipated after adversarial training [48], due to the trade-off between robustness and accuracy. However, the observed improvements may be attributable to additional training on utility data, which likely enhances the model’s capabilities. Notably, R2D2 has a low Harmless score of only 0.3, suggesting an overzealous refusal of non-malicious samples.

The Llama3 base model demonstrates greater robustness than the Zephyr-7B base model, as evidenced by its lower ASR across all attacks. This can be explained by the fact that Llama3 underwent significant safety alignment efforts before its release [47]. In this case, MIX-AT is particularly effective against jailbreak attacks, achieving the lowest scores on TAP [10], PAP [1], and AutoDAN [11]. LAT performs best on GCG with a score of 0.0, while MIX-AT records 20.0, which could potentially be improved with optimized hyperparameters. As mentioned above, most of the current hyperparameter selections were

based on the Zephyr-7B model and [7]. On multiple-choice utility benchmarks, MIX-AT achieves notably higher scores on ARC-E and ARC-C than LAT, and even outperforms the base model in these areas. Training with a long schedule (CAT) appears to decrease utility performance here as well.

However, our method achieves the lowest performance on MT-Bench [14]. The issue mainly stems from, that it refuses to give answers to some unmalicious questions, most of these are role-play style questions, where the model has to pretend a character or specialist (like a doctor) and answer questions in this role. While analyzing PAIR and TAP attacks, they often try to trick the model with "role-play" questions ("like imagine you are a doctor."). Since our model is performing well against TAP and PAIR, due to this it might refuse the non-malicious requests, which are phrased similarly. We would like to address this issue of answering non-malicious requests in future work with possible directions discussed in (10).

From these observations, we conclude that MIX-AT demonstrates superior performance against rephrasing-based jailbreak attacks and shows promise against GCG [3]. It achieves a lower ALO compared to other methods. On the utility benchmarks ARC-E, ARC-C, and MMLU, MIX-AT significantly outperforms LAT, R2D2, and CAT (long schedule). The Harmless scores for MIX-AT range between 0.9 and 1.0, indicating some rejection of inoffensive prompts, but still outperforming R2D2 and remaining comparable to LAT. However, our method shows the lowest scores on the MT-Bench benchmark. It is due to refusing to answer some questions, which are phrasing the instructions in some "role-play" style.

Overall, our method shows strong generalization across a range of jailbreak prompts, even though only PAP jailbreak prompts were employed during adversarial training from discrete attacks. The diversity of malicious requests and paraphrasing strategies offered by PAP [1] likely contributes to this generalization ability.

Chapter 8

Ablation Studies

In this chapter, we present an in-depth analysis of the results obtained by systematically ablating key components of our method to evaluate their individual contributions to overall performance, such as scaling LoRA weights, the effect of model quantization on adversarial training, and the impact of utilising chat templates.

8.1 Impact of Model Quantization

Model quantization is frequently used transformations on model weights, to decrease the memory costs of deploying deep neural networks [42]. When a model is released to public usage (for example on Hugging Face) it should be ensured that the quantized version of it is still sufficiently robust against adversarial attacks. As [49] outlined, manipulated alignment can yield a robust model, whose quantized version is vulnerable to adversarial attacks.

To examine the impact of model quantization, we train several models without quantization and evaluate both the quantized and non-quantized versions of the trained models. The results (8.1) show, that models trained without quantization are less robust than the models which were loaded in quantized form during the training. The models which were trained without quantization and evaluated without quantization have the best utility results. It was expected since non-quantized models have usually better general capabilities.

In the case of the Zephyr-7B base model, the quantized version of the model seems to be less attackable, than the non-quantized one, except in the score of DirectRequest and PAP.

Table 8.1: **Model quantization.** Comparing the results of models trained without quantization, which were produced by evaluating the models with or without quantization. Here, *Train + Eval* refers to models that were both trained and evaluated with quantization (the default experimental setup), meaning the models were loaded in quantized form during both training and evaluation. *Eval* refers to models that were adversarially trained in their non-quantized form, but evaluated with quantization. *None* refers to models that were both trained and evaluated without quantization. In most cases, it appears that adversarial training of models in their non-quantized form resulted in less robust models.

Model	Quantization	Utility Benchmarks [%]					Attack Success Rate [%]						
		ARC-E	ARC-C	MMLU	Harmless	MT-Bench	Direct R	PAP-10	TAP	PAIR	AutoDAN	GCG	ALO
Base	Eval	81.0	55.2	56.2	100	56.4	85.0	87.5	85.0	97.5	90.0	85.0	100
	None	81.4	57.3	58.2	100	56.8	77.5	72.5	90	100	95.0	90.0	100
CAT	Train + Eval	78.2	50.5	54.5	95.0	54.2	0.0	25.0	27.5	62.5	0.0	7.5	75.0
	Eval	77.8	50.5	54.9	92.5	52.9	0.0	45.0	55.0	65.0	0.0	12.5	87.5
	None	79.4	51.2	56.8	92.5	53.5	0.0	52.5	52.5	75.0	0.0	20.0	90.0
Mix-AT	Train + Eval	81.6	55.1	56.2	95.0	43.8	0.0	0.0	0.0	0.0	0.0	5.0	5.0
	Eval	82.0	54.2	56.4	92.5	52.9	0.0	2.5	2.5	12.5	0.0	5.0	15.0
	None	82.2	57.4	57.9	92.5	52.6	0.0	10.0	10.0	12.5	0.0	5.0	22.5

8.2 Multiple-choice question utility benchmarks with chat template

Recent work highlights a failure mode in previous works, namely that evaluating LLMs on robustness and utility benchmark inconsistently with or without chat template might produce unrealistic results [7], since in real-world scenarios the settings of text generation do not depend on the content of prompts or whether they are malicious or standard requests. To enhance consistency, in this section, we present ARC-E, ARC-C and MMLU results of Zephyr-7B and Llama3 under their default chat template (8.2) Note that, the rest of the utility and robustness evaluations, where under chat templates.

In the case of both Zephyr-7B [39] and Llama3 [47] models, our Mix-AT utility scores are still competitive compared to other adversarial training scores, which have significantly worse robustness results. Furthermore, our Mix-AT adapter has even higher utility on these benchmarks than the base model, meaning maintaining good robustness. In the case of the Zephyr-7B model, our method has much higher utility compared to the CAT adapter (0.1 more ARC-E and 0.05 more ARC-C), demonstrating the power of short-schedule training in the case of Zephyr-7B models.

Table 8.2: Utility scores with chat template We evaluated some of the models on ARC-E, ARC-C, and MMLU using their default chat templates, since on other benchmarks, models were evaluated under their default chat templates. Overall, Mix-AT demonstrates even better utility on these benchmarks compared to the competitive methods and achieves significantly better results than the base models.

Base Model	Method	Chat Template	Utility Benchmarks [%]		
			ARC-E	ARC-C	MMLU
zephyr-7b	No Defense	y	0.748	0.509	0.556
		n	0.810	0.552	0.562
	R2D2	y	0.718	0.421	0.478
		n	0.801	0.529	0.561
	CAT HF	y	0.694	0.439	0.552
		n	0.782	0.511	0.548
	PAP-AT	y	0.791	0.544	0.584
		n	0.823	0.542	0.564
	Mix-AT	y	0.787	0.515	0.559
		n	0.816	0.551	0.562
llama3	No Defense	y	0.734	0.451	0.576
		n	0.791	0.491	0.608
	CAT	y	0.761	0.491	0.599
		n	0.791	0.504	0.579
	LAT	y	0.723	0.442	0.551
		n	0.731	0.427	0.583
	PAP-AT	y	0.778	0.515	0.604
		n	0.817	0.515	0.598
	Mix-AT	y	0.774	0.496	0.592
		n	0.806	0.496	0.584

8.3 Scaling the LoRA weights

One of the main challenges of adversarial training is to find the optimal trade-off between adversarial robustness and utility. Intuitively, we can regulate the effect of LoRA adapters ([43]) by scaling their weights with a constant λ . This approach may yield different trade-offs depending on the chosen λ values. To test our hypothesis, we evaluated LoRA-scaled variants of the Mix-AT and CAT (Hugging Face) Zephyr-7B models on utility and adversarial robustness benchmarks. This involved multiplying the low-rank matrix A by the constant λ :

$$W = W_0 + \alpha \cdot (\lambda A)B,$$

where W_0 denotes the original weight matrix, A and B are low-rank matrices used to introduce perturbations, and α is a weighting factor for the perturbation. The constant λ differs from α in that α is applied throughout both the training and deployment of the adapter and is typically kept fixed during both phases. We conducted experiments with $\lambda = [0.0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5]$ values. As shown in (8.1), ASR (Adversarial

Success Rate) decreases as the magnitude of LoRA scaling increases, supporting our hypothesis about regulating the effect of LoRA. This decline in ASR is more rapid for the MIX-AT model compared to the CAT model, potentially due to the greater effectiveness of the MIX-AT adapter. Except MT-Bench, utility scores for MIX-AT are consistently higher than those for CAT across all values of λ . In general, for $\lambda > 1$, the models begin to lose utility, and utility scores drop more rapidly. In the range $0 < \lambda < 1$, utility is not significantly degraded, and may even slightly improve. For $0.5 < \lambda < 1.0$, the ASR is close to the ASR of full LoRA adapters ($\lambda = 1.0$). Technically speaking, we may find a better utility-robustness trade-off, than the full LoRA model, in the $0.5 < \lambda < 1.0$ range, where models show lower robustness but higher utility, or in the $1.0 < \lambda < 1.5$ range, where utility decreases but ASR approaches zero. As shown in [table], $\lambda = 0.75$ yields similar robustness results to full-weight LoRA adapters while slightly improving utility.

8.4 Ratio of PAP samples

We examined the effect of changing the amount of PAP examples used throughout the training on model performance (8.3). We trained models with varying λ ratios of examples augmented via PAP ([1]), where $\lambda = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$ percent. Although there are some fluctuations in robustness results, it can be seen that $\lambda = 50\%$ is one of the best choices. However, another local minimum can be observed at $\lambda = 90\%$. Between $\lambda = 0\%$ and $\lambda = 30\%$, a decline can be seen in ASR, with only negligible differences in DirectRequest and AutoDAN scores. A peak in GCG scores is observed at $\lambda = 40\%$ and $\lambda = 70\%$, with the latter also showing a relatively high TAP score. The small fluctuations may be because only 40 samples are used for robustness evaluations, meaning a change in one sample’s result would lead to a 2.5% difference in the final scores. The increase in DirectRequest scores at $\lambda = 100\%$ could be because PAP training alone does not involve shorter prompts, which are present in DirectRequest samples, as the PAP method augments core malicious requests and generates longer inputs.

The utility scores, including those from multiple-choice question data (ARC-E, ARC-C, MMLU), Harmless, and MT-Bench, do not seem to show clear trends, possibly indicating they are not significantly influenced by different λ ratios.

Table 8.3: **MIX-AT Ablations** We would like to examine the effect of the ratio of PAP samples utilized and the effect of not continuously attacking the PAP samples throughout the training. "CA" denotes training with continuous attacks, "PAP" denotes training with PAP examples, and "PAP+c" refers to training with PAP samples that are continuously attacked. The results do not show clear trends regarding the effect of the ratio of PAP samples used throughout the training. However, two local minima in ASR scores can be observed at 50% and 90% PAP ratio for "PAP+c & CA" training. At the bottom of the table, some results from "PAP & CA" trainings are displayed. The 50% "PAP & CA" training shows slightly better ASR but worse utility compared to the 50% "PAP+x & CA" method.

Attacks Used	Discrete fraction	Utility Benchmarks						Attack Success Rate [%]					
		ARC-E	ARC-C	MMLU	Harmless	MT-Bench	Direct R	PAP-10	TAP	PAIR	AutoDAN	GCG	ALO
cont.	0	81.5	54.9	56.2	95.0	54.2	0.0	47.5	30.0	25.0	0.0	12.5	75.0
PAP+c / cont	10	81.5	55.1	56.2	95.0	56.3	2.5	12.5	10.0	2.5	2.5	7.5	22.5
PAP+c / cont	20	81.6	55.4	56.0	97.5	52.4	2.5	7.5	10.0	2.5	0.0	5.0	20.0
PAP+c / cont	30	81.2	54.0	56.3	85.0	52.4	2.5	2.5	5.0	2.5	2.5	2.5	10.0
PAP+c / cont	40	81.7	54.0	56.0	97.5	54.1	0.0	5.0	0.0	0.0	0.0	10.0	12.5
PAP+c / cont	50	81.8	55.0	56.2	95.0	43.8	0.0	0.0	0.0	0.0	0.0	5.0	5.0
PAP+c / cont	60	81.4	54.3	56.0	95.0	51.2	0.0	0.0	2.5	2.5	0.0	2.5	5.0
PAP+c / cont	70	81.7	55.8	56.1	100.0	51.9	2.5	0.0	10.0	2.5	0.0	15.0	20.0
PAP+c / cont	80	81.4	54.1	56.1	97.5	55.4	7.5	2.5	2.5	0.0	0.0	5.0	12.5
PAP+c / cont	90	81.6	54.6	56.1	87.5	49.0	0.0	0.0	0.0	0.0	0.0	5.0	5.0
PAP+c	100	81.9	54.8	56.3	97.5	54.9	7.5	0.0	0.0	0.0	0.0	2.5	10.0
PAP / cont	50	81.9	54.7	56.1	95.0	53.1	2.5	2.5	10.0	15.0	0.0	10.0	22.5
PAP	100	82.0	54.5	56.2	97.5	49.0	25.0	2.5	10.0	32.5	10.0	47.5	67.5

8.5 Discrete samples without continuous attack

Since [7] present in ablation studies that training without continuous attack of the malicious training samples has no significant improvement on model robustness, we keep continuously attacking the non-PAP-augmented malicious prompts throughout the training. (Mix-AT augments some of the malicious prompts throughout the training, afterwards continuous attack is utilized on these).

Additionally, we examine the effect of not using continuous attacks on PAP samples during adversarial training (referred to as "PAP & CA" in Table (8.3)). The $\lambda = 50\%$ "PAP & CA" model shows worse robustness but slightly better utility compared to the $\lambda = 50\%$ "PAP+c & CA" model (where continuous attack is applied to PAP samples).

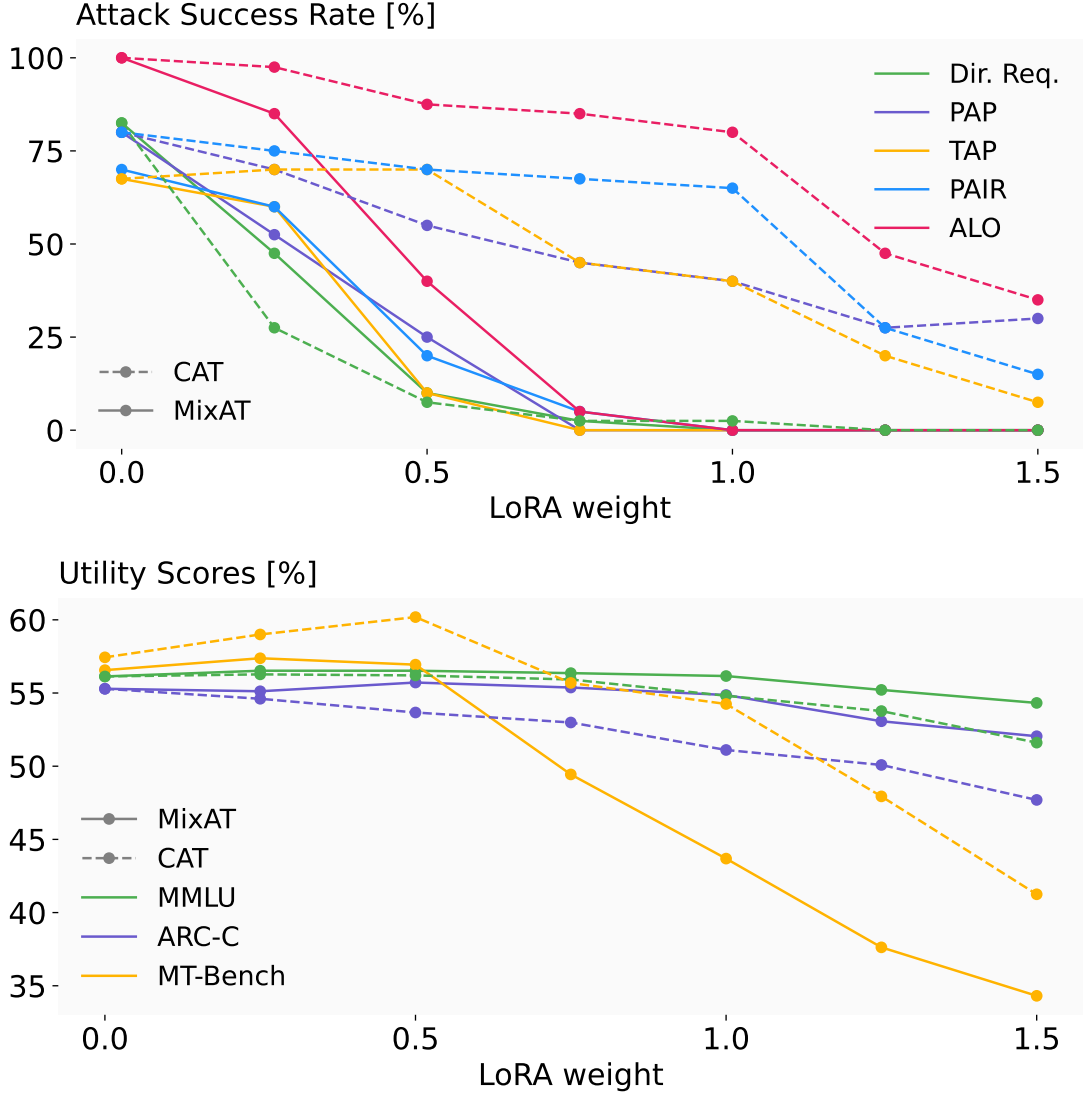


Figure 8.1: **The effect of scaling the weights of LoRA adapters** We scaled the LoRA [43] weights of the MIX-AT and CAT (Hugging Face) adapters for the Zephyr-7B [39] models. The resulting snapshots were evaluated on adversarial robustness (top plot) and utility benchmarks (bottom plot). The x-axis represents different λ weighting constants used to multiply the weights, while the y-axis shows ASR and utility scores. Interestingly, ASR scores tend to decrease for both MIX-AT (solid line) and CAT (dotted line) as the λ scale increases, suggesting that the effect mediated by LoRA adapters can be controlled through scaling.

Chapter 9

Discussion

In this chapter, we aim to draw conclusions from our experiments on mixed adversarial training and general issues in this field.

9.1 Adversarial Training for LLMs: Key Considerations

In applying adversarial training to large language models (LLMs), certain elements have proven crucial for enhancing robustness, while others appear less impactful. Key among the important factors is the choice of adversarial perturbation types—specifically, a balanced integration of continuous and discrete perturbations tends to yield stronger defences in the case of adversarial scenarios.

9.2 Can Adversarial Training Solve the Problem of Malicious Requests?

While adversarial training is an essential tool in fortifying LLMs against adversarial manipulations, it is unlikely to fully solve the issue of malicious requests. Although adversarial training significantly reduces the model’s vulnerability to certain types of malicious inputs, adaptive attackers can still create novel attacks that circumvent existing defences. Moreover, adversarial training alone may not address deeper issues related to a model’s understanding and ethical reasoning in response to harmful requests. Therefore, a comprehensive defence likely requires a combination of adversarial training with supplementary techniques, such as reinforced filtering mechanisms, context-sensitive response generation, and continual model auditing.

9.3 Issues in Evaluation: Are Current Datasets Sufficient?

The effectiveness of adversarial training heavily depends on the quality and scope of evaluation datasets, and current datasets may fall short in several respects. While existing datasets cover a variety of attack types and input variations, they often lack the diversity required to fully test model robustness against highly sophisticated or multi-faceted adversarial strategies. Furthermore, some datasets may emphasize specific types of attacks (e.g., toxic language) without adequately covering other harmful or malicious requests, limiting the model’s exposure to a broader range of adversarial scenarios. Addressing this limitation may require the creation of richer, more varied datasets that encompass not only different attack types but also dynamic, context-sensitive adversarial examples to better evaluate and improve the robustness of LLMs.

Chapter 10

Limitations and Future Directions

The primary limitation is the tendency of the model to either refuse to answer non-malicious, legitimate questions or to preface its responses with statements of uncertainty. This cautious approach can negatively impact MT-Bench scores, as the language model judge may interpret such hesitant or incomplete answers as less valuable or informative. Consequently, the model may be rated lower than its actual capabilities.

Another significant limitation stems from the small sample size used in testing. With fewer examples, the results are more susceptible to variance, making it harder to reliably assess the model’s performance. This limited dataset may not represent the full range of real-world queries and challenges the model could face, thus impacting the consistency and robustness of the evaluation. We examine this issue further in (C)

Additionally, the reliance on a language model as the judge for evaluating robustness and utility samples introduces potential inaccuracies. Large language models, while effective for certain types of judgments, may not always capture subtle nuances, particularly in terms of robustness and context-specific utility. This dependence can lead to inconsistent scoring and may not accurately reflect the practical effectiveness or robustness of the model across diverse scenarios.

To address the limitations mentioned, we would like to examine the reasons for refusing non-malicious requests. It seems likely that this is due to the "role-play" style of the refused samples. Since our method is effective against PAIR and TAP, which often generate "role-play" style queries to trick the models, it seems that it is difficult for the model to distinguish between malicious and un-malicious requests which are formulated in the same, "role-play" style. Potentially, we might augment the utility data used throughout the training process or generate additional utility data, which formulates the questions in a role-play

style. Besides, we might use an additional round of adversarial training, where we deploy DPO with malicious and non-malicious role-play queries [50], to help the model directly distinguish between malicious and non-malicious queries. Additionally, we would like to establish a benchmark for non-malicious requests formulated in a role-play style, similar to the Harmless benchmark, to measure how prone are adversarially trained models to refusals in this style.

Additionally, we also examined that in a few cases, the MT-Bench [14] benchmark contains malicious requests(e.g. "How to design a campaign with political fallacies"). This has to be purged, since utility benchmarks of malicious requests, might deteriorate the utility scores of all adversarial training methods. Additionally, we would like to measure our method’s performance on other open-ended question benchmarks, to get a more detailed view about our method’s strengths and weaknesses in this area.

We are planning to experiment with replacing PAP [1] with other jailbreak attacks, such as PAIR or TAP. These jailbreaks might enable dynamic adversarial training, meaning that jailbreaks are generated against the model during the actual phase of training, which may better address security gaps. Additionally, we might replace PAP with other language model-generated adversarial attacks, which are fast to generate, to examine the difference. Later, we might want to combine the LoRA adapters [43] of adversarial training, which uses different adversarial training methods in the mixing phase of MIX-AT.

Furthermore, we would like to do further experiments with scaling the LoRA weights, to find the scale, which creates a good trade-off between robustness and accuracy across a wide range of adversarially trained models.

To address the poorer GCG performance of MIX-AT on Llama3-8B, we would like to find better hyperparameters to enhance our model’s performance. Additionally, a future direction could involve studying the effects of adversarial training on Hungarian LLMs.

Chapter 11

Conclusion

Our proposed method for adversarial training demonstrates promising advancements over previous approaches by effectively combining continuous and discrete adversarial attacks. While existing methods rely exclusively on continuous perturbations, which primarily exploit gradient-based vulnerabilities, or limited sets of discrete perturbations, which make the defence weakly generalizable, our approach incorporates both types of adversarial examples. This novel integration captures a broader spectrum of adversarial scenarios in the context of large language models (LLMs), where both types of perturbations are highly relevant. By leveraging the complementary strengths of continuous and discrete attacks, our method achieves superior robustness against rephrasing-based jailbreak attacks. The limitation of worse GCG [3] performance on the Llama3 model and the rejection of some unmalicious queries on MT-Bench [14] requires further examination. However, these issues can potentially be addressed with further hyperparameter search and by using more "role-play" style utility data during the training process or using additional DPO [50] based fine-tuning. Overall, this hybrid approach underscores the importance of addressing both gradient-based and linguistic vulnerabilities and being a promising method to enhance LLM safety.

Acknowledgements

This research has been conducted with the support of INSAIT - Institute for Computer Science, Artificial Intelligence and Technology, and Sofia University St. Kliment Ohridski and Supported by the EKÖP-24 University Excellence Scholarship Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.

Appendix A

Discussion of randomness

A.1 Randomness of attack generation

All of the adversarial attacks we use in our experiments have a random component in their generation process, either by randomly choosing an initial state (TAP, PAIR, AutoDAN, GCG) or by querying LLMs with a non-zero temperature (PAP, TAP, PAIR, AutoDAN). This randomness can yield different results when evaluating the robustness of the same model. To evaluate the extent of this randomness, we generate the full suite of adversarial samples for a single model multiple times and evaluate the robustness of the model on each set of samples. The results in (A.1) show that the variance in the ASR scores is relatively low, indicating that the randomness in the generation process does not have a significant impact on the final robustness scores.

A.2 Randomness of training

We also examine the effect of randomness in the training process. We train the same model with different random seeds and evaluate the robustness of the models on the same set of adversarial samples. The results in (A.2) show that the variance induced by the randomness in the training procedure has a similar magnitude as the variance induced by the randomness in the evaluation process. This indicates that our training method is sufficiently stable across different random seeds. A notable exception is observed in the case of MT-Bench utility scores, for which one of the random training seeds has a significantly lower score than the others. Upon closer inspection of the answers, we observed that the model refused.

Table A.1: **Randomness of attack generation for different models** We ran three different seeds of attack generation and evaluation, against Zephyr-7B variants, to examine their randomness. Overall, the attack generation and evaluation have some variance.

Model	Run	Attack Success Rate [%]					
		PAP-10	TAP	PAIR	AutoDAN	GCG	ALO
Base	R1	87.5	85.0	97.5	90.0	85.0	100.0
	R2	87.5	85.0	90.0	92.5	92.5	100.0
	R3	90.0	92.5	95.0	90.0	95.0	100.0
	Avg	88.3	87.5	94.2	90.8	90.8	100.0
	Std	1.4	4.3	3.8	1.4	5.2	0.0
CAT	R1	40	42.5	42.5	2.5	5	70
	R2	22.5	27.5	17.5	2.5	7.5	40
	R3	32.5	47.5	5	2.5	5	55
	Avg	31.7	39.2	21.7	2.5	5.8	55
	Std	8.8	10.4	19.1	0	1.4	15
MIX-AT	R1	0.0	0.0	0.0	0.0	5.0	5.0
	R2	0.0	2.5	2.5	0.0	7.5	10.0
	R3	0.0	0.0	0.0	0.0	2.5	2.5
	Avg	0.0	0.8	0.8	0.0	5.0	5.8
	Std	0.0	1.4	1.4	0.0	2.5	3.8

Table A.2: **Randomness of Training with MIX-AT on different random seeds** We aim to evaluate the randomness in MIX-AT training by running it multiple times with different seeds. We define the average of ARC-E, ARC-C, and MMLU as MCQ. The results indicate that while the MCQ scores are stable, MT-Bench scores may show variability, as evidenced by the outlier observed with the first seed.

Run	Utility Benchmarks [%]			Attack Success Rate [%]		
	MCQ	Harmless	MT-Bench	PAIR	GCG	ALO
R1	64.2	95.0	43.8	0	5	5
R2	64.4	97.5	54.5	2.5	5	7.5
R3	63.8	97.5	53.4	0	7.5	7.5
Avg	64.1	96.7	50.6	0.8	5.8	6.7
Std	0.3	1.4	5.9	1.4	1.4	1.4

Appendix B

Training Time and Costs

We report the training times of some methods we examined in our work. Based on the results (B.1), generating PAP samples throughout the training yields to small overhead in training time, since in MIX-AT we continuously attack PAP samples as well.

We train all of the models on A100 GPUs, having 40 G memory each. Zephyr-7B trainings require 2 GPUs, while Llama3 trainings require 3 GPUs. The time and compute requirements of our method are relatively low.

Table B.1: Estimated Training Costs We estimate the cost of training models using different adversarial training methods, both in terms of time and monetary expense. In both time and cost, MIX-AT is the cheapest among R2D2, CAT, and LAT. The API calls are required for generating PAP samples.

Base Model	Method	GPUs used	Train Time (min)	API calls	Train Steps	Est. Costs (\$)		
						GPUs	API	Total
zephyr-7b	R2D2	8xA100	960	0	2000	491.5	0	491.5
	CAT	2xA100	400	0	760	51.2	0	51.2
	CAT short	2xA100	160	0	300	20.5	0	20.5
	PAP-AT	2xA100	170	2.4k	300	21.8	0.36	22.1
	Mix-AT 50	2xA100	240	1.2k	300	30.7	0.18	30.9
	Mix-AT 100	2xA100	330	2.4k	300	42.2	0.36	42.6
llama3-8b	LAT	1xH100	270	0	760	64.0	0	64.0
	CAT	3xA100	430	0	760	82.6	0	82.6
	CAT short	3xA100	180	0	300	34.6	0	34.6
	PAP-AT	3xA100	220	2.4k	300	42.2	0.36	42.6
	Mix-AT	3xA100	290	1.2k	300	55.7	0.18	55.9
	Mix-AT Long	3xA100	730	6k	760	140.2	0.90	141.1

Appendix C

Evaluating on more samples

Due to computational costs for most of our experiments, we have only evaluated our models on a subset of 40 samples from the HarmBench test set. To assess the generalization of our models, we evaluate them on the full test set of the HarmBench dataset. However, we still exclude the copyright samples as before, and we have a total of 240 samples. Because generating GCG attacks for all 240 samples would take 6 days on a single A100 GPU for each evaluated model, we only evaluate the models on the other attacks.

The results in (C.1) show that the models trained with MIX-AT achieve different robustness scores on the full test set as on the subset used in the main experiments. This is a limitation of our work and also the previous work [7].

Table C.1: Results of Evaluating on the full HarmBench dataset We evaluate less computationally demanding attacks, generated from all of the non-copyright-related prompts of the HarmBench test set, on Zephyr-7B variants. Although, in most cases, there is no significant difference between the results on the smaller and larger subsets, there is a notable reduction in the PAIR score for CAT, with a decrease of nearly 50%.

Model	HarmBench Samples	Attack Success Rate [%]					
		Direct R	PAP-10	TAP	PAIR	AutoDAN	ALO
Base	40	85.0	87.5	85.0	97.5	90.0	100.0
	240	88.3	76.7	87.5	90.4	87.5	100.0
CAT	40	2.5	40.0	42.5	42.5	2.5	70.0
	240	0.8	38.3	38.3	20.8	0.8	56.2
MixAT	40	0.0	0.0	0.0	0.0	0.0	0.0
	240	0.0	0.8	0.8	1.7	0.0	2.9

Appendix D

Transferability of Attacks

Most of the attacks that we use for evaluating the robustness of our models have shown generalization properties across a wide range of pre-trained models and architectures, even if the adversarial samples were targeting a different model. We want to assess the transferability of these attacks when the target model has been adversarially trained. To do this, we generate *universal* adversarial samples for a model that has been adversarially trained with MIX-AT or CAT and evaluate the effectiveness of these adversarial samples when attacking other models. We compare the success rates of the *universal* adversarial samples with the success rates of *targeted* adversarial samples specifically generated for each model. The results in D.1 show that none of the universal attacks achieves the same performance as the targeted attacks, sometimes the attacks being even less successful than the plain Direct Request method. A single exception is observed when zephyr-R2D2 is successfully attacked by PAIR attacks generated to target zephyr-CAT.

Table D.1: Attack Transferability

Base Model	Method	Direct Request	TAP targeted at				PAIR targeted at				GCG targeted at			
			Z-Base	Z-CAT	Z-MixAT	Defender	Z-Base	Z-CAT	Z-MixAT	Defender	Z-Base	Z-CAT	Z-MixAT	Defender
zephyr-7b	No Defense	85.0	85.0	50.0	70.0	85.0	97.5	62.5	72.5	97.5	85.0	50.0	52.5	75.0
	R2D2 (HF)	7.5	25.0	12.5	0.0	15.0	2.5	20.0	2.5	7.5	0.0	0.0	0.0	2.5
	CAT (R)	2.5	7.5	27.5	12.5	27.5	15.0	55.0	15.0	55.0	2.5	12.5	7.5	12.5
	PAP-AT	25.0	0.0	10.0	5.0	10.0	5.0	12.5	0.0	32.5	20.0	12.5	22.5	47.5
	Mix-AT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	5.0
llama3-8b	No Defense	25.0	17.5	22.5	20.0	47.5	20.0	22.5	15.0	67.5	25.0	12.5	15.0	47.5
	CAT (R)	0.0	25.0	25.0	15.0	47.5	25.0	37.5	20.0	70.0	0.0	0.0	0.0	7.5
	LAT (HF)	2.5	0.0	2.5	2.5	10.0	7.5	5.0	0.0	20.0	0.0	0.0	0.0	0.0
	PAP-AT	17.5	15.0	22.5	15.0	37.5	22.5	30.0	17.5	67.5	15.0	17.5	20.0	77.5
	Mix-AT	2.5	0.0	2.5	0.0	5.0	2.5	0.0	0.0	12.5	0.0	2.5	0.0	20.0

Appendix E

Types of adversarial attacks

In this section, we list the types of the adversarial attacks and provide a short description of them. All of the adversarial samples were generated by HarmBench [5], in most of the cases the default hyperparameters have been used. We use the default judge model of HarmBench to decide, whether the model answers malicious queries, which is "cais/HarmBench-Llama-2-13b-cls", a finetuned model for this purpose.

Approaches for Adversarial Attack Generation	
<i>Method</i>	<i>Description</i>
<i>GCG [3]</i>	Generates adversarial test cases by optimizing a suffix at the token level. This suffix, when appended to a user prompt, is adjusted to maximize the likelihood of an affirmative response from the target LLM to a pre-defined behaviour trigger.
<i>AutoDAN [11]</i>	A semi-automated framework that starts with handcrafted jail-break prompts, evolving them through a hierarchical genetic algorithm. The goal is to adapt the prompts to provoke specific target behaviours in the LLM.
<i>Direct Request</i>	Serves as a baseline where behaviour descriptions themselves are used as prompts. This tests how well the target LLM can identify and resist straightforward requests without obfuscation or modifications.

<i>Method</i>	<i>Description</i>
<i>PAP [1]</i>	Refines prompts using persuasive strategies to increase their effectiveness. An attacker LLM adjusts the language to make requests more convincing. In this work, the top ten strategies are chosen for evaluation.
<i>PAIR [2]</i>	Uses iterative prompting to adapt and explore harmful behaviours systematically. The attacker LLM learns from each interaction to refine its approach and elicit targeted responses from the model.
<i>TAP [10]</i>	Employs a tree-structured method to guide the attacker LLM in progressively exploring pathways that provoke harmful behaviours. Each branch of the tree represents an adaptive modification to the prompting strategy.

Table E.1: **Techniques for generating adversarial examples for LLMs**

Bibliography

- [1] Yi Zeng et al. “How johnny can persuade llms to jailbreak them: Rethinking persuasion to challenge ai safety by humanizing llms”. In: *arXiv preprint arXiv:2401.06373* (2024).
- [2] Patrick Chao et al. “Jailbreaking black box large language models in twenty queries”. In: *arXiv preprint arXiv:2310.08419* (2023).
- [3] Andy Zou et al. “Universal and transferable adversarial attacks on aligned language models”. In: *arXiv preprint arXiv:2307.15043* (2023).
- [4] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572* (2014).
- [5] Mantas Mazeika et al. “Harmbench: A standardized evaluation framework for automated red teaming and robust refusal”. In: *arXiv preprint arXiv:2402.04249* (2024).
- [6] Fan Liu, Zhao Xu, and Hao Liu. “Adversarial tuning: Defending against jailbreak attacks for llms”. In: *arXiv preprint arXiv:2406.06622* (2024).
- [7] Sophie Xhonneux et al. “Efficient adversarial training in llms with continuous attacks”. In: *arXiv preprint arXiv:2405.15589* (2024).
- [8] Stephen Casper et al. “Defending Against Unforeseen Failure Modes with Latent Adversarial Training”. In: *arXiv preprint arXiv:2403.05030* (2024).
- [9] Abhay Sheshadri et al. “Targeted Latent Adversarial Training Improves Robustness to Persistent Harmful Behaviors in LLMs”. In: *arXiv preprint arXiv:2407.15549* (2024).
- [10] Anay Mehrotra et al. “Tree of attacks: Jailbreaking black-box llms automatically”. In: *arXiv preprint arXiv:2312.02119* (2023).

- [11] Xiaogeng Liu et al. “Autodan: Generating stealthy jailbreak prompts on aligned large language models”. In: *arXiv preprint arXiv:2310.04451* (2023).
- [12] Peter Clark et al. “Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge”. In: *ArXiv abs/1803.05457* (2018).
- [13] Dan Hendrycks et al. “Aligning AI With Shared Human Values”. In: *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).
- [14] Dan Hendrycks et al. “Measuring Massive Multitask Language Understanding”. In: *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).
- [15] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *Proc. of ICLR*. 2014.
- [16] Aleksander Madry. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv preprint arXiv:1706.06083* (2017).
- [17] Paarth Neekhara et al. “Universal adversarial perturbations for speech recognition systems”. In: *arXiv preprint arXiv:1905.03828* (2019).
- [18] Yuxin Wen et al. “Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery”. In: *Advances in Neural Information Processing Systems* 36 (2024).
- [19] Chuan Guo et al. “Gradient-based adversarial attacks against text transformers”. In: *arXiv preprint arXiv:2104.13733* (2021).
- [20] Eric Wallace et al. “Universal adversarial triggers for attacking and analyzing NLP”. In: *arXiv preprint arXiv:1908.07125* (2019).
- [21] Federico Bianchi et al. “Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions”. In: *arXiv preprint arXiv:2309.07875* (2023).
- [22] Alexander Robey et al. “Smoothllm: Defending large language models against jailbreaking attacks”. In: *arXiv preprint arXiv:2310.03684* (2023).
- [23] Mikayel Samvelyan et al. “Rainbow teaming: Open-ended generation of diverse adversarial prompts”. In: *arXiv preprint arXiv:2402.16822* (2024).
- [24] Taylor Shin et al. “Autoprompt: Eliciting knowledge from language models with automatically generated prompts”. In: *arXiv preprint arXiv:2010.15980* (2020).

- [25] Boyi Deng et al. “Attack prompt generation for red teaming and defending large language models”. In: *arXiv preprint arXiv:2310.12505* (2023).
- [26] Mouret Jean-Baptiste. “Illuminating search spaces by mapping elites”. In: *arXiv:1504.04909* (2015).
- [27] Ethan Perez et al. “Red Teaming Language Models with Language Models, February 2022a”. In: URL <https://arxiv.org/abs/2202.03286> v1 ().
- [28] Florian Tramèr et al. “Ensemble adversarial training: Attacks and defenses”. In: *arXiv preprint arXiv:1705.07204* (2017).
- [29] Toby Shevlane and Allan Dafoe. “The offense-defense balance of scientific knowledge: does publishing AI research reduce misuse?” In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 173–179.
- [30] Samuel Gehman et al. “Realtotoxicityprompts: Evaluating neural toxic degeneration in language models”. In: *arXiv preprint arXiv:2009.11462* (2020).
- [31] Jing Xu et al. “Recipes for safety in open-domain chatbots”. In: *arXiv preprint arXiv:2010.07079* (2020).
- [32] Kalyanmoy Deb. “Nonlinear goal programming using multi-objective genetic algorithms”. In: *Journal of the Operational Research Society* 52.3 (2001), pp. 291–302.
- [33] Kaisa Miettinen. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media, 1999.
- [34] Xuannan Liu et al. “Jailbreak Attacks and Defenses against Multimodal Generative Models: A Survey”. In: *arXiv preprint arXiv:2411.09259* (2024).
- [35] Junjie Chu et al. “Comprehensive assessment of jailbreak attacks against llms”. In: *arXiv preprint arXiv:2402.05668* (2024).
- [36] Aleksander Madry et al. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *Proc. of ICLR*. 2018.
- [37] Thomas Wolf et al. *HuggingFace’s Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL]. URL: <https://arxiv.org/abs/1910.03771>.
- [38] Ning Ding et al. *Enhancing Chat Language Models by Scaling High-quality Instructional Conversations*. 2023. arXiv: 2305.14233 [cs.CL].

- [39] Lewis Tunstall et al. “Zephyr: Direct distillation of lm alignment”. In: *arXiv preprint arXiv:2310.16944* (2023).
- [40] Jinze Bai et al. “Qwen technical report”. In: *arXiv preprint arXiv:2309.16609* (2023).
- [41] Guan Wang et al. “Openchat: Advancing open-source language models with mixed-quality data”. In: *arXiv preprint arXiv:2309.11235* (2023).
- [42] Elias Frantar et al. “Gptq: Accurate post-training quantization for generative pre-trained transformers”. In: *arXiv preprint arXiv:2210.17323* (2022).
- [43] Edward J Hu et al. “Lora: Low-rank adaptation of large language models”. In: *arXiv preprint arXiv:2106.09685* (2021).
- [44] I Loshchilov. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).
- [45] Ilya Loshchilov and Frank Hutter. “Sgdr: Stochastic gradient descent with warm restarts”. In: *arXiv preprint arXiv:1608.03983* (2016).
- [46] Lianmin Zheng et al. “Judging llm-as-a-judge with mt-bench and chatbot arena”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 46595–46623.
- [47] Abhimanyu Dubey et al. “The llama 3 herd of models”. In: *arXiv preprint arXiv:2407.21783* (2024).
- [48] Hongyang Zhang et al. “Theoretically principled trade-off between robustness and accuracy”. In: *International conference on machine learning*. PMLR, 2019, pp. 7472–7482.
- [49] Kazuki Egashira et al. “Exploiting LLM Quantization”. In: *arXiv preprint arXiv:2405.18137* (2024).
- [50] Rafael Rafailov et al. “Direct Preference Optimization: Your Language Model is Secretly a Reward Model”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2024).

List of Figures

1.1	Overview of our method and results. (a) We propose a novel adversarial training method MIX-AT that combines continuous and discrete adversarial examples during training. In this figure, $\mathcal{X}$ represents the set of harmful input prompts (e.g. “How to build a bomb?”). $\mathcal{R}(\mathcal{X})$ represents the set of known discrete perturbations of $\mathcal{X}$ (rephrasing, suffix etc.). Both the original prompt and its rephrasing can be modified using Continuous Adversarial Attacks ($\rightarrow$). We indicate this by adding some pixel-level noise as in the case of image adversarial samples. We denote the set of these perturbations as $\mathcal{X} + \delta$ and $\mathcal{R}(\mathcal{X}) + \delta$ respectively. Compared to existing Adversarial Training methods, MIX-AT trains the model to be robust to perturbations from $\mathcal{R}(\mathcal{X}) + \delta$, providing better generalization to a wide range of adversarial techniques ($\text{Adv}(\mathcal{X})$). (b) In our experimental evaluation, we show that MIX-AT outperforms existing adversarial training methods in terms of robustness and efficiency against a wide range of attacks. Specifically, training with suffix attacks (R2D2 [5] trains using GCG [3] attacks) leaves the model vulnerable to rephrasing attacks such as PAP [1]. Conversely, training with just PAP attacks without continuous attack leaves the model vulnerable to suffix attacks. While Continuous Adversarial Training (CAT [7]) provides a model which is moderately robust to both types of attacks, the model trained with MIX-AT has a lower Attack Success Rate (ASR) against rephrasing attacks.	5
6.1	Illustration of the MIX-AT method. The method combines continuous and discrete adversarial training by extending the adversarial search space to include both continuous and discrete perturbations.	17

8.1	The effect of scaling the weights of LoRA adapters	We scaled the LoRA [43] weights of the MIX-AT and CAT (Hugging Face) adapters for the Zephyr-7B [39] models. The resulting snapshots were evaluated on adversarial robustness (top plot) and utility benchmarks (bottom plot). The x-axis represents different λ weighting constants used to multiply the weights, while the y-axis shows ASR and utility scores. Interestingly, ASR scores tend to decrease for both MIX-AT (solid line) and CAT (dotted line) as the λ scale increases, suggesting that the effect mediated by LoRA adapters can be controlled through scaling.	28
-----	---	---	----

List of Tables

7.1	Sources of Hugging Face models and adapters	The exact path of Hugging Face models and adapters used in our research and evaluations.	18
7.2	Comparing MIX-AT with other adversarial training methods		21
8.1	Model quantization.	Comparing the results of models trained without quantization, which were produced by evaluating the models with or without quantization. Here, <i>Train + Eval</i> refers to models that were both trained and evaluated with quantization (the default experimental setup), meaning the models were loaded in quantized form during both training and evaluation. <i>Eval</i> refers to models that were adversarially trained in their non-quantized form, but evaluated with quantization. <i>None</i> refers to models that were both trained and evaluated without quantization. In most cases, it appears that adversarial training of models in their non-quantized form resulted in less robust models.	24
8.2	Utility scores with chat template	We evaluated some of the models on ARC-E, ARC-C, and MMLU using their default chat templates, since on other benchmarks, models were evaluated under their default chat templates. Overall, MIX-AT demonstrates even better utility on these benchmarks compared to the competitive methods and achieves significantly better results than the base models.	25

8.3	MIX-AT Ablations We would like to examine the effect of the ratio of PAP samples utilized and the effect of not continuously attacking the PAP samples throughout the training. "CA" denotes training with continuous attacks, "PAP" denotes training with PAP examples, and "PAP+c" refers to training with PAP samples that are continuously attacked. The results do not show clear trends regarding the effect of the ratio of PAP samples used throughout the training. However, two local minima in ASR scores can be observed at 50% and 90% PAP ratio for "PAP+c & CA" training. At the bottom of the table, some results from "PAP & CA" trainings are displayed. The 50% "PAP & CA" training shows slightly better ASR but worse utility compared to the 50% "PAP+x & CA" method.	27
A.1	Randomness of attack generation for different models We ran three different seeds of attack generation and evaluation, against Zephyr-7B variants, to examine their randomness. Overall, the attack generation and evaluation have some variance.	36
A.2	Randomness of Training with MIX-AT on different random seeds We aim to evaluate the randomness in MIX-AT training by running it multiple times with different seeds. We define the average of ARC-E, ARC-C, and MMLU as MCQ. The results indicate that while the MCQ scores are stable, MT-Bench scores may show variability, as evidenced by the outlier observed with the first seed.	36
B.1	Estimated Training Costs We estimate the cost of training models using different adversarial training methods, both in terms of time and monetary expense. In both time and cost, MIX-AT is the cheapest among R2D2, CAT, and LAT. The API calls are required for generating PAP samples. . .	37
C.1	Results of Evaluating on the full HarmBench dataset We evaluate less computationally demanding attacks, generated from all of the non-copyright-related prompts of the HarmBench test set, on Zephyr-7B variants. Although, in most cases, there is no significant difference between the results on the smaller and larger subsets, there is a notable reduction in the PAIR score for CAT, with a decrease of nearly 50%.	38
D.1	Attack Transferability	39

E.1	Techniques for generating adversarial examples for LLMs	41
------------	--	-----------