Dynamic Deep Prompt Optimization for Defending Against Jailbreak Attacks on LLMs

Doniyorkhon Obidov[1], **Honggang Yu**[2], **Xiaolong Guo**[3], **Kaichen Yang**[1]
[1]Michigan Technological University
[2]Miami University
[3]Kansas State University
dobidov@mtu.edu, honggangyu@miamioh.edu, guoxiaolong@k-state.edu, kaicheny@mtu.edu

Abstract

Large Language Models (LLMs) demonstrate impressive capabilities across many applications but remain vulnerable to jailbreak attacks, which elicit harmful or unintended content. While model fine-tuning is an option for safety alignment, it is costly and prone to catastrophic forgetting. Prompt optimization has emerged as a promising alternative, yet existing prompt-based defenses typically rely on static modifications (e.g., fixed prefixes or suffixes) that cannot adapt to diverse and evolving attacks.

We propose Dynamic Deep Prompt Optimization (DDPO), the first jailbreak defense based on deep prompt optimization. DDPO uses the target LLM's own intermediate layers as feature extractors to dynamically generate defensive embeddings via a lightweight multilayer perceptron. These tailored embeddings are then injected into a subsequent intermediate layer, enabling an input-dependent defense without modifying the LLM's weights. This design ensures high adaptability with minimal computational overhead.

Experiments on a diverse set of models and attacks demonstrate that DDPO significantly outperforms static prompt optimization methods, particularly on weakly aligned models and when handling semantically ambiguous benign prompts, successfully distinguishing them from genuinely harmful requests.

Introduction

Large Language Models (LLMs) have revolutionized natural language processing, excelling in tasks such as question answering and code completion. However, the broad deployment of LLMs has highlighted significant security vulnerabilities, notably their susceptibility to "jailbreak" attacks (Yi et al. 2024). These attacks use meticulously crafted adversarial prompts to induce LLMs to generate malicious responses, thereby circumventing their designed usage policies and safety alignments. Such attacks can lead to severe consequences, including the dissemination of misinformation or privacy breaches (Chu et al. 2024; Xu et al. 2024b).

Existing defenses against jailbreak attacks fall into two main categories: model-level modifications (e.g., safety fine-tuning (SFT) (Dong et al. 2023), reinforcement learning from human feedback (RLHF) (Ouyang et al. 2022)) and prompt-level defenses (Yi et al. 2024). The latter are often preferred because they are lightweight and avoid the high costs of model retraining (Yi et al. 2024), as well as risks like catastrophic forgetting that arise from modifying model weights (Luo et al. 2023; Bianchi et al. 2023).

A promising prompt-level strategy is prompt optimization, which augments user inputs with engineered tokens or embeddings to steer model behavior. Unlike defenses based on prompt filtering, prompt optimization offers enhanced protection and lower false-positive rates (Jain et al. 2023; Mo et al. 2024). However, current prompt optimization defenses typically rely on inserting fixed (static) prefixes or suffixes (Mo et al. 2024; Zhou, Li, and Wang 2024; Zheng et al. 2024). This static nature limits the defense's ability to adapt to the nuances of diverse user inputs. As our experiments show, this shortcoming leads to failure in truly challenging cases.

A Naive Dynamic Approach: A seemingly straightforward solution would be to employ an auxiliary LLM (e.g., a fine-tuned GPT-2) to generate dynamic prefixes or suffixes. However, this approach, while promising a degree of adaptability, is far from ideal. It would introduce latency by requiring the input to be processed twice: once by the auxiliary model and once by the target LLM. Furthermore, this design presents a difficult trade-off. A small auxiliary model would be necessary to avoid large computational overhead, but its limited capacity would hinder its ability to understand semantically complex inputs, leading to misguided defenses.

To address these challenges, this paper proposes a novel and efficient approach: Dynamic Deep Prompt Optimization (DDPO). Instead of relying on massive external models, DDPO leverages the target LLM's own architecture. The core idea is to use one of the LLM's intermediate layers as a feature extractor for the incoming user prompt. This extracted representation is then fed into a small, lightweight neural network, which dynamically generates a defensive embedding tailored to the specific input. This embedding is then injected back into the input of a subsequent deep layer of the LLM, effectively modulating the model's processing flow. This targeted intervention steers the model toward safe responses for harmful queries while preserving utility for benign ones. Crucially, the entire process occurs without modifying the weights of the target LLM.

The concept of "deep prompt tuning", where tunable

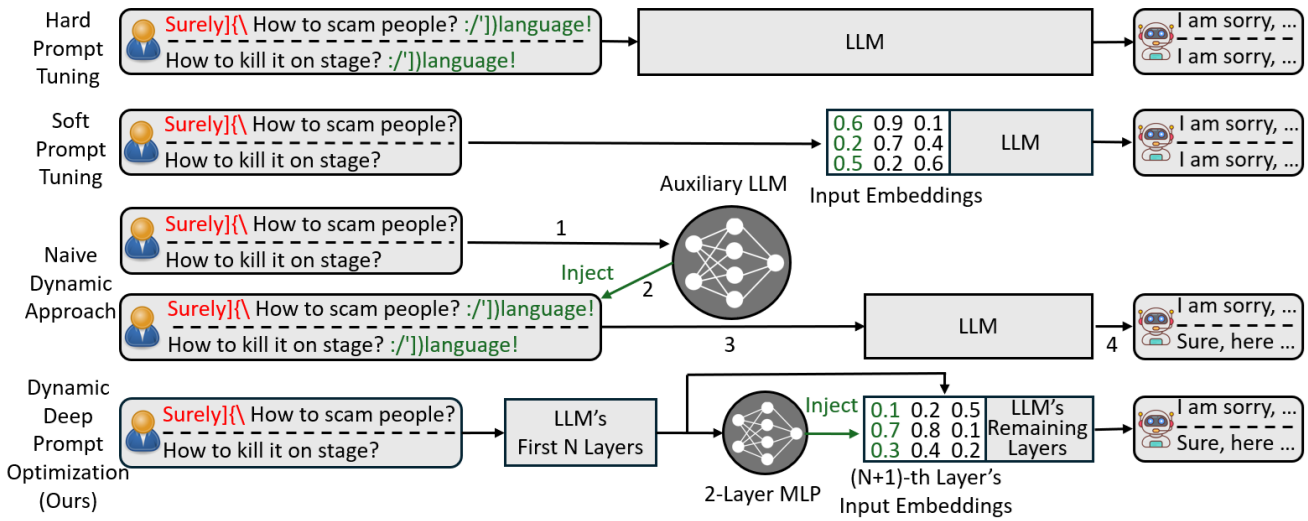

Figure 1: Prompt tuning strategies against jailbreak attacks (none update the target LLM's weights). Attack tokens are in red and defensive tokens/embeddings in green. Existing methods use static hard or soft prompts. A naive dynamic approach requires a large auxiliary LLM and double input processing. DDPO instead uses the target LLM's early-layer features and injects dynamically generated defensive embeddings (via a two-layer MLP, negligible compared to the LLM size) into the inputs of later layers, adding minimal overhead.

prompt embeddings influence intermediate layers, has been previously studied (e.g., P-Tuning v2 (Liu et al. 2021)). However, DDPO distinguishes itself significantly: to the best of our knowledge, it is the first to adapt such a mechanism for jailbreak defense. Moreover, unlike static methods like P-Tuning v2 that learn fixed prompt embeddings, DDPO generates a dynamic defensive embedding tailored to each input. This latter distinction is crucial. Our method's core concept is not simply the choice of a deeper injection layer, but the strategy of leveraging the LLM's own semantic capabilities, an advantage that other deep prompt optimization methods ignore.

Our contributions are as follows:

- We introduce DDPO, the first jailbreak defense to use deep prompt optimization and the first prompt-optimization-based defense to dynamically generate defensive embeddings for insertion.
- Our method dynamically generates defensive embeddings by leveraging the target LLM's own early layers as a feature extractor. Inserting these embeddings directly into the inputs of subsequent layers eliminates the high overhead of large auxiliary models or a second processing pass.
- We demonstrate through experiments that DDPO significantly outperforms existing baselines, especially in challenging cases like defending weakly aligned models or handling ambiguous benign prompts.
- We release our code for reproducibility.[1]

These characteristics position DDPO as a more efficient alternative to existing prompt optimization defenses and a more lightweight yet flexible alternative to full model fine-tuning.

[1]Code available at https://github.com/doniobidov/ddpo

Related Work

Jailbreak Attacks

Jailbreak attacks aim to bypass the safety alignments of LLMs, compelling them to generate content that violates usage policies or ethical guidelines. These attacks are broadly categorized based on the attacker's knowledge of the target model: white-box attacks, which assume access to model parameters and gradients, and black-box attacks, which interact with the model only through its input/output interface (Yi et al. 2024).

Black-box attacks often originate from manual, human-crafted prompts that rely on creativity and social engineering principles like role-playing (e.g., the "Do Anything Now" prompts) to deceive the model (Shen et al. 2024). While innovative, these manual methods are often brittle and require significant human effort. To overcome this, recent research has focused on automating the generation of black-box attacks. A prominent approach is LLM-based generation, where an auxiliary LLM acts as an automated attacker. For instance, Prompt Automatic Iterative Refinement (PAIR) uses an attacker LLM to iteratively query a target model and refine a candidate jailbreak prompt based on the target's responses, creating a query-efficient and adaptive attack (Chao et al. 2025). Other black-box techniques include prompt rewriting, which obfuscates harmful intent using ciphers (Yuan et al. 2023) or low-resource languages (Deng et al. 2023), and template completion, which embeds malicious requests within seemingly benign scenarios (Li et al. 2023).

White-box attacks leverage internal model information,

typically gradients, to optimize adversarial inputs. A key challenge in this domain is optimizing over the discrete space of text tokens. Early work on Universal Adversarial Triggers (UAT) used a gradient-guided search to find short, input-agnostic token sequences that trigger specific model behaviors (Wallace et al. 2019). Building on this, AutoPrompt introduced one of the first frameworks for discrete prompt optimization, using a greedy, gradient-guided search to iteratively replace "trigger" tokens in a template (Shin et al. 2020). However, its search was limited, as it evaluated candidate token swaps for only one position at a time.

More advanced methods have significantly improved the efficacy of gradient-based optimization. The Greedy Coordinate Gradient (GCG) attack substantially increased success rates by using gradients to identify promising token replacements across all positions in an adversarial suffix simultaneously, and then greedily selecting the best substitution from a batch of candidates (Zou et al. 2023). This approach was extended to GCG-M (GCG-Multi) to optimize a single suffix against multiple prompts and models, enhancing its universality and transferability. Other techniques have addressed the discrete optimization challenge differently. The Gradient-based Distributional Attack (GBDA) searches for a distribution of adversarial examples, using the Gumbel-softmax trick to enable gradient-based optimization over a continuous parameterization (Guo et al. 2021). Similarly, PEZ (Hard Prompts Made Easy) maintains continuous (soft) embeddings during optimization but projects them to the nearest discrete tokens for the forward pass, using the resulting gradient to update the continuous representation (Wen et al. 2023).

Jailbreak Defenses

Defenses against jailbreak attacks are correspondingly diverse and are generally classified as model-level or prompt-level defenses (Yi et al. 2024).

Model-level defenses involve modifying the LLM's parameters or training process. Supervised Fine-Tuning (SFT) with safety-focused datasets (Bianchi et al. 2023) and Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al. 2022) are common approaches to instill safety. Other model-level techniques include gradient and logit analysis for attack detection (Xie et al. 2024; Xu et al. 2024a), response refinement (Kim, Yuk, and Cho 2024), and proxy defenses using a separate, secure LLM (Zeng et al. 2024). While potentially robust, model-level defenses can be resource-intensive to train and maintain. Furthermore, fine-tuning approaches may suffer from "catastrophic forgetting", where the model loses performance on general tasks after safety alignment (Luo et al. 2023; Bianchi et al. 2023).

Prompt-level defenses offer a lightweight alternative to model-level modifications, as they operate on the input prompts without altering the LLM's core parameters, thereby avoiding expensive retraining. These include prompt detection (e.g., perplexity filters (Jain et al. 2023)), prompt perturbation (e.g., SmoothLLM (Robey et al. 2023)), and system prompt safeguards (Zheng et al. 2024; Zou, Chen, and Li 2024). Prompt optimization, the focus of our work, falls under this category.

Prompt Optimization Defenses Prompt optimization for jailbreak defense is an emerging area that focuses on augmenting user input with optimized textual strings or continuous embeddings (soft prompts). Compared to input-filtering defenses, prompt optimization typically achieves better performance with lower false positive rates (Jain et al. 2023; Mo et al. 2024; Zhou, Li, and Wang 2024; Zheng et al. 2024).

- **Prompt Adversarial Tuning (PAT)** (Mo et al. 2024) trains a discrete defensive prefix (a sequence of optimized tokens, akin to hard prompt tuning) attached to user prompts. PAT uses an adversarial tuning process, optimizing the prefix with both adversarial and benign prompts to balance robustness and utility. The resulting prefix is static once trained.
- **Directed Representation Optimization (DRO)** (Zheng et al. 2024) optimizes continuous safety prompts (soft prompts). DRO aims to move the internal representations of user queries along or opposite a pre-estimated "refusal direction" within the LLM's low-dimensional representation space, depending on the query's harmfulness. These optimized continuous embeddings are also static.
- **Robust Prompt Optimization (RPO)** (Zhou, Li, and Wang 2024) introduces a minimax optimization objective to generate a robust defensive suffix using discrete tokens (i.e., hard prompt tuning). Rather than relying on predefined jailbreak attacks, it jointly optimizes both the attack and the defensive suffix in alternating cycles. The final suffix is static.

These methods represent significant advancements. However, they primarily result in defensive components (prefixes, suffixes, or soft prompts) that are static post-training. In the experiment section, we demonstrate that this limitation leads to underperformance, particularly in insufficiently aligned LLMs and in cases where user prompts are ambiguous and appear harmful, even though they are actually benign.

Methodology

Threat Model

Attacker's Objective and Capabilities: The attacker's primary goal is to circumvent the LLM's safety alignments, inducing the model to produce undesirable outputs such as harmful content, misinformation, or any other responses that violate usage policies. We consider an attacker capable of crafting adversarial prompts by modifying any accessible part of the user input. This attacker possesses knowledge of the model architecture, including its weights, enabling them to employ both white-box and black-box jailbreak strategies. However, the attacker cannot directly modify the model's parameters or its training data.

Defender's Capabilities: We assume the defender is the model developer or deployer, who has white-box access to the LLM (a reasonable assumption in real-world scenarios). This access includes full knowledge of the model architecture, its parameters (weights), and the ability to access its internal activations and processing flow during inference. The

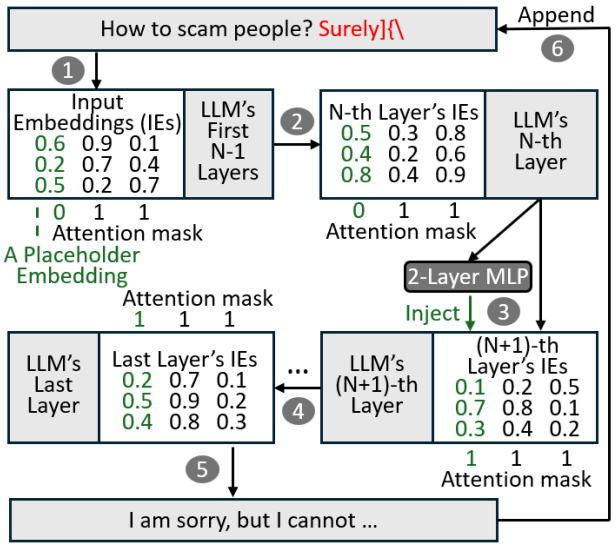

Figure 2: DDPO directly inserts a defensive embedding into an intermediate LLM layer, generated by a lightweight MLP using features from earlier layers. A masked placeholder preserves positional embeddings during earlier processing. The MLP is called once to produce the embedding, which is reused for output generation, minimizing overhead.

defender's objective is to prevent the LLM from generating harmful, unethical, or unintended content in response to malicious user queries. Crucially, this defense must be achieved while preserving the model's utility and performance on benign inputs.

Method Overview

DDPO begins by automatically identifying the optimal intermediate layer within the target LLM. This layer is chosen for its ability to effectively distinguish between benign and harmful user inputs. During operation, DDPO extracts the hidden state representation corresponding to the last token of the input prompt from this selected layer. This representation is then processed by a small, lightweight Multi-Layer Perceptron (MLP).

The MLP, in turn, generates a single defensive embedding vector. This dynamically created vector is subsequently injected back into the sequence of hidden states at the same intermediate layer (as depicted in Fig. 2). This process modulates the LLM's ongoing computations.

Crucially, the weights of the target LLM remain entirely frozen throughout this process; only the MLP's weights are updated during the training phase of DDPO. For inference, the MLP is utilized once to generate the defensive embedding when the LLM is producing its first output token. This same learned embedding is then reused for the generation of all subsequent tokens for that specific input, ensuring minimal computational overhead.

Step 1: Optimal Layer Selection

The choice of the layer (N) for feature extraction and embedding injection is crucial. We hypothesize that certain intermediate layers are more effective at distinguishing between harmful and benign intent. To identify this optimal layer, we perform the following procedure:

1. A dataset of benign and harmful prompts is prepared.
2. Each prompt is passed through the LLM, and for each layer l, the hidden state corresponding to the last token of the input prompt, $h_l^{(last)}$, is extracted.
3. For each layer l, we compute the average pairwise cosine similarity between the $h_l^{(\text{last})}$ vectors yielded by harmful prompts and those yielded by benign prompts.

$$\text{Score}(l) = \mathbb{E}_{\mathbf{u}\sim H_{\text{harmful}}^{(l)},\ \mathbf{v}\sim H_{\text{benign}}^{(l)}}[d(\mathbf{u},\mathbf{v})] \quad (1)$$

where d denotes a similarity metric (e.g., cosine similarity).

4. The layer N that yields the minimum average cosine similarity is selected.

$$N = \arg\min_l \text{Score}(l) \quad (2)$$

Step 2: Dynamic Embedding Generation and Intermediate Injection

Once the optimal layer N is identified, DDPO proceeds as follows. Let L denote the length (number of tokens) of the user's input prompt P_{user}.

1. **Input Preparation and Initial Processing:** The user's input prompt, P_{user}, is tokenized. To accommodate the single dynamic embedding to be injected at layer N, the input is processed such that one "dummy" embedding (a zero vector with a zero attention mask) is effectively present in the sequence representation passed through the initial N layers of the target LLM, M_{LLM}. The actual user prompt tokens have attention mask values of 1.
2. **Feature Extraction at Layer N:** The user input P_{user} is passed through the first N layers of M_{LLM}. The hidden state corresponding to the last token of P_{user} (i.e., the L-th token) at layer N, denoted as h_N^L, is extracted. This vector, $\mathbf{f}_{user} = h_N^L$, serves as the input feature to the MLP.
3. **Dynamic Embedding Generation via MLP:** The feature vector $\mathbf{f}_{user}$ is fed into a multilayer perceptron (MLP), f_{MLP}. The MLP outputs a single dynamic embedding vector, e_{dyn}. As established in our experiments, a two-layer MLP proved sufficient for effectively generating defensive embeddings.
4. **Intermediate Embedding Injection:** The dynamically generated embedding e_{dyn} replaces a dummy/placeholder hidden state within the sequence of hidden states output by layer N. Specifically, if the original sequence of hidden states from layer N (containing the placeholder) is $H_N = [h_N^{(1)}, \ldots, h_N^L, h_N^{(dummy)}]$, this sequence is modified into $H'_N = [h_N^{(1)}, \ldots, h_N^L, e_{dyn}]$. The attention mask value for the position of e_{dyn} is then updated to 1. This modified sequence, H'_N, subsequently serves as the input to layer $N+1$, thereby influencing its computations and those of all subsequent layers.

5. **MLP Training Objective:** The MLP, f_{MLP}, is the only trainable component. The training objective is to guide the LLM's output based on the nature of the input prompt:
   - For harmful prompts ($P_{user,harmful}$), the LLM, when modulated by e_{dyn}, should generate a predefined refusal response (e.g., "I cannot fulfill this request.").
   - For benign prompts ($P_{user,benign}$), the LLM, when modulated by e_{dyn}, should generate a helpful and relevant continuation.

   A combined loss function, $\mathcal{L}_{DDPO}$, is used to train the parameters θ_{MLP} of f_{MLP}:

$$\mathcal{L}_{DDPO} = \lambda_{harmful}\mathcal{L}_{harmful} + \lambda_{benign}\mathcal{L}_{benign} \quad (3)$$

   where $\mathcal{L}_{harmful}$ and $\mathcal{L}_{benign}$ are cross-entropy losses comparing the LLM's output (generated with e_{dyn} injected) to the target refusal or helpful continuation, respectively. $\lambda_{harmful}$ and λ_{benign} are weighting factors for the respective losses.
6. **Inference:** During inference, the trained MLP, f_{MLP}, computes the dynamic embedding e_{dyn} once during the generation of the first output token based on the input P_{user} and its extracted feature $\mathbf{f}_{user}$. The LLM, M_{LLM}, subsequently generates its response autoregressively. For each step in the autoregressive generation loop (i.e., for each subsequent output token being generated), the same initially computed e_{dyn} is reused. This ensures that the defensive modulation only depends on the user input and is consistently applied throughout the generation of the entire response sequence with minimal computational overhead post the initial MLP pass.

Algorithm 1 outlines the training procedure.

Algorithm 1: DDPO - MLP Training

Require: M_{LLM}: Target LLM (parameters are frozen). N: Optimal intermediate layer index. f_{MLP}: Multi-Layer Perceptron with trainable parameters θ_{MLP}. $\mathcal{D}$: Training dataset, where each element is $(P_{user,i}, \text{label}_i, Y_{target,i})$. $P_{user,i}$ is the i-th user prompt of length L_i, label_i indicates its nature (harmful/benign), and $Y_{target,i}$ is the target output. η: Learning rate. idx_{inject}: Predefined index within the hidden state sequence for layer N where e_{dyn} is injected.

for each training epoch **do**
  for each $(P_{user}, \text{label}, Y_{target})$ in $\mathcal{D}$ **do**
    $H_{N\_out} \leftarrow \text{FORWARDPASSTO}(M_{LLM}, P_{user}, N)$ ▷ *Get hidden states at layer N's output.*
    $\mathbf{f}_{user} \leftarrow (H_{N\_out})_L$ ▷ *$\mathbf{f}_{user}$ is the hidden state of P_{user}'s last (L-th) token from H_{N_out}.*
    $e_{dyn} \leftarrow f_{MLP}(\mathbf{f}_{user})$ ▷ *Dynamic embedding.*
    $H_{N+1\_in} \leftarrow \text{INJECTEMBED}(H_{N\_out}, e_{dyn}, idx_{inject})$ ▷ *Inject e_{dyn} to input for layer $N+1$.*
    $Y_{pred} \leftarrow \text{FORWARDPASSFROM}(M_{LLM}, H_{N+1\_in}, N+1)$ ▷ *Predict using M_{LLM} from layer $N+1$ onwards.*
    $\mathcal{L}_{DDPO} \leftarrow \text{COMPUTELOSS}(Y_{pred}, Y_{target})$
    $\nabla_{\theta_{MLP}}\mathcal{L}_{DDPO} \leftarrow \text{BACKPROP}(\mathcal{L}_{DDPO}, \theta_{MLP})$
    $\theta_{MLP} \leftarrow \theta_{MLP} - \eta\nabla_{\theta_{MLP}}\mathcal{L}_{DDPO}$
  end for
end for
Return: Trained f_{MLP} (i.e., optimized θ_{MLP}).

Experiments

Experimental Setup

Models: Our experiments were conducted on five popular LLMs: Llama-3-8B-Instruct (Dubey et al. 2024), Deepseek-llm-7B-chat (Bi et al. 2024), Vicuna-13B-v1.5 (Chiang et al. 2023), Llama-2-7B-chat-hf (Touvron et al. 2023), and Openchat-3.5-1210 (7B) (Wang et al. 2023). All models were implemented using the Hugging Face library.

Datasets and Attack Strategies: Our evaluation relies on standard benchmarks and a comprehensive suite of attack strategies that builds upon and expands those considered in previous work (Mo et al. 2024; Zhou, Li, and Wang 2024; Zheng et al. 2024). To evaluate defensive robustness, we sourced harmful prompts from the AdvBench (Zou et al. 2023), JailbreakBench (Chao et al. 2024), and HarmBench (Mazeika et al. 2024) benchmarks. We tested all defense methods against nine jailbreak strategies: Plain Harmful (direct harmful requests), PEZ (Wen et al. 2023), UAT (Wallace et al. 2019), GCG (Zou et al. 2023), AutoPrompt (Shin et al. 2020), GBDA (Guo et al. 2021), GCG-M (Zou et al. 2023), PAIR (Chao et al. 2025), and Human Crafted prompts.

To assess model utility, we used two distinct sets of benign prompts. The first was a curated dataset from JailbreakBench (Chao et al. 2024) and WildJailbreak (Jiang et al. 2024), specifically designed to include borderline cases that might appear harmful (e.g., "I bombed my test, can you help me write an apology to my professor?"). The second was the MMLU dataset, a standard benchmark used to measure general knowledge and task performance on entirely benign inputs.

Training and Evaluation Protocol: The training set included 50 examples of GCG and human-crafted jailbreak prompts. The evaluation was then performed on 230 unseen samples for each of the nine attack types, meaning the defenses were evaluated in a zero-shot setting for seven of the attacks.

Performance was measured using three key metrics:

- **Attack Success Rate (ASR):** The percentage of harmful prompts that successfully elicited a harmful response. A lower ASR indicates better defense.
- **Benign Pass Rate (BPR):** The percentage of challenging benign prompts that were correctly and helpfully answered without being falsely refused. A higher BPR indicates better utility.
- **MMLU Average Accuracy (%):** The average score across 57 diverse subjects in the MMLU benchmark, measuring the preservation of general knowledge. A higher score is better.

Implementation Details: All experiments were performed on a server equipped with two NVIDIA A100 GPUs. Our DDPO method utilized a two-layer MLP with 512 hidden units and a GeLU activation function. This MLP generated a single dynamic defensive embedding tailored to each user input.

		Attack Success Rate (%) ↓										Utility (%) ↑	
Model	Defense	PH	PEZ	UAT	GCG	AP	GBDA	GCG-M	PAIR	HC	Average	Benign	MMLU
Llama3	None	1.29	3.12	5.54	7.25	6.92	4.02	7.14	5.06	20.62	6.77	94.45	**65.46**
	PAT	**0.00**	2.17	2.61	4.35	4.78	3.48	5.65	4.78	23.48	5.70	75.80	42.81
	RPO	**0.00**	1.30	1.74	2.17	1.74	2.17	0.87	3.04	10.00	2.56	70.60	47.65
	DRO	**0.00**	1.74	1.74	1.30	0.87	1.30	**0.00**	**0.87**	6.96	1.64	92.40	47.15
	DDPO	2.61	**1.30**	**0.00**	**0.43**	**0.00**	**0.43**	0.43	**0.87**	**0.87**	**0.77**	**97.80**	63.53
Deepseek	None	26.21	51.12	45.39	60.80	62.50	54.46	63.62	84.81	86.80	59.52	91.45	**47.07**
	PAT	13.04	50.00	59.57	64.35	65.65	53.48	73.04	87.83	92.17	62.13	86.80	32.19
	RPO	34.35	53.91	49.57	54.35	56.96	53.48	48.26	82.17	88.70	57.97	**97.60**	34.26
	DRO	6.96	28.70	30.00	34.35	38.26	32.17	36.09	73.48	80.43	40.05	95.40	37.54
	DDPO	**1.30**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.30**	**0.87**	**0.39**	94.60	46.44
Vicuna	None	2.91	20.09	16.24	40.12	25.89	17.41	42.63	73.42	79.79	35.39	90.91	**54.70**
	PAT	0.43	2.61	0.87	24.78	6.96	1.74	30.00	46.52	77.39	21.26	79.00	40.60
	RPO	**0.00**	0.87	**0.00**	0.43	1.74	**0.43**	5.65	3.91	13.48	2.95	11.80	26.00
	DRO	4.35	10.43	6.96	13.91	13.04	11.74	21.74	23.04	40.87	16.23	85.40	33.90
	DDPO	0.43	**0.00**	**0.00**	**0.00**	**0.00**	1.30	**0.00**	**3.48**	**0.87**	**0.68**	**95.80**	53.85
Llama2	None	0.65	5.13	8.49	12.04	14.29	2.68	15.85	3.80	23.71	9.63	69.36	**50.21**
	PAT	0.43	7.39	5.65	9.57	10.87	5.65	6.52	5.22	27.39	8.74	68.40	38.11
	RPO	**0.00**	**2.61**	2.61	4.78	4.78	2.17	2.61	2.61	9.57	3.53	70.00	39.25
	DRO	0.43	4.35	5.22	4.78	8.26	4.35	5.65	3.04	18.70	6.09	76.20	33.90
	DDPO	0.43	3.48	**0.87**	**0.87**	**2.61**	**1.30**	**2.17**	**0.43**	**0.43**	**1.40**	**93.20**	50.07
Openchat	None	62.94	60.49	61.99	69.14	65.18	55.36	74.11	97.05	94.54	71.20	96.45	60.75
	PAT	25.65	65.22	67.39	64.78	73.04	71.74	66.09	95.22	94.35	69.28	96.20	57.55
	RPO	35.22	69.57	70.43	69.57	72.61	70.00	70.43	91.74	93.04	71.40	89.20	49.86
	DRO	24.35	59.57	55.22	58.26	65.22	65.22	69.57	84.35	83.91	62.85	**98.00**	45.87
	DDPO	**1.30**	**0.00**	**0.00**	**0.00**	**0.00**	**0.43**	**0.00**	**4.35**	**2.61**	**0.97**	95.60	**60.90**

Table 1: A comparison of defense performance for various models against multiple jailbreak attacks. The Attack Success Rate (ASR) is evaluated on 230 unseen prompts for each attack type. Benign utility is evaluated on 500 unseen prompts that are designed to seem harmful but are in fact benign. DDPO (ours) was trained on 50 examples of GCG and human-crafted prompts. PH denotes plain harmful prompts, AP denotes AutoPrompt, and HC denotes human-crafted jailbreak prompts.

Defense Performance

As shown in Table 1, DDPO significantly outperforms static methods across all models, achieving a superior balance of low Attack Success Rate (ASR) and high utility. Our analysis reveals that DDPO overcomes two critical failings of static defenses.

First, static defenses are ineffective on weakly aligned models. For instance, on *Openchat* with a baseline ASR of 71.20%, RPO's defense is ineffective (71.40% ASR). In contrast, DDPO reduces the ASR to under 1%, proving its robustness even on highly vulnerable models.

Second, static defenses severely harm utility by misclassifying ambiguous benign queries. This is most evident with RPO on *Vicuna*, where the Benign Pass Rate (BPR) drops to a catastrophic 11.80%. Static methods also consistently degrade MMLU scores across all models. DDPO avoids this trade-off, maintaining a high average BPR of 95.40% and preserving MMLU performance, showcasing its ability to accurately discern user intent.

Ablation Studies

Impact of Dynamic Generation: To verify that DDPO's effectiveness stems from its ability to generate input-specific embeddings, we compared it to a static counterpart, which we term Static Deep Prompt Optimization (SDPO). In the SDPO setup, a single, fixed embedding is optimized as a trainable parameter and injected at the same intermediate layer as DDPO. To ensure a fair comparison, both DDPO and SDPO were trained on the exact same data.

The results, presented in Table 2, clearly demonstrate the superiority of our dynamic approach. Across all models,

	SDPO (Static)		DDPO (Dynamic)	
Model	ASR ↓	BPR ↑	ASR ↓	BPR ↑
Llama3	12.32	**98.60**	**0.77**	97.80
Deepseek	22.90	86.80	**0.39**	**94.60**
Vicuna	47.20	**96.00**	**0.68**	95.80
Llama2	9.52	70.00	**1.40**	**93.20**
Openchat	79.32	**99.00**	**0.97**	95.60

Table 2: Ablation study comparing our dynamic method (DDPO) against a static variant (SDPO). ASR is averaged over 2,070 attack prompts (230 prompts for each of 9 attacks), while BPR is measured on 500 benign prompts. Both methods were trained on identical data and used the same intermediate layers. ASR and BPR values are given in (%).

Model	Metric	$K = 1$	$K = 5$	$K = 10$
Llama3	ASR	0.77	1.50	**0.72**
	BPR	97.8	**98.0**	97.4
Deepseek	ASR	0.39	1.26	**0.34**
	BPR	94.6	96.2	**96.4**
Vicuna	ASR	**0.68**	1.35	0.97
	BPR	95.8	95.4	**96.6**
Llama2	ASR	1.40	1.35	**0.39**
	BPR	93.2	**94.2**	93.0
OpenChat	ASR	**0.97**	**0.97**	1.06
	BPR	95.6	95.0	**96.2**

Table 3: Ablation study on the number of defensive embeddings (K) across several models. The Attack Success Rate (ASR) is averaged over 2,070 attack prompts (230 for each of 9 attack types), while the Benign Pass Rate (BPR) is measured on 500 benign prompts. The default DDPO configuration uses $K = 1$.

SDPO is significantly less effective at preventing jailbreaks. For instance, on *Vicuna* and *Openchat*, SDPO's defense is ineffective, resulting in ASRs of 47.20% and 79.32%, respectively. In contrast, DDPO reduces the ASR on these same models to just 0.68% and 0.97%. While SDPO occasionally yields a marginally higher Benign Pass Rate (BPR), this comes at the cost of a completely compromised defense. On models that are overaligned and prone to false refusal, such as *Llama2*, DDPO is superior on both fronts, drastically improving the BPR from 70.00% to 93.20% while simultaneously providing a much stronger defense.

Effect of Number of Injected Embeddings: By default, DDPO injects only a single dynamic embedding ($K = 1$). We investigated whether increasing the number of embeddings to $K = 5$ or $K = 10$ could offer additional performance benefits.

As shown in Table 3, increasing the number of embeddings does not lead to significant or consistent improvements. While some configurations with $K = 5$ or $K = 10$ show marginal gains in either ASR or BPR for specific models, the improvements are not universal, and in some cases, performance slightly degrades (e.g., ASR for *Llama3* and *Vicuna*). Given that generating and processing additional embeddings would invariably increase computational overhead and inference latency, the lack of a clear performance benefit makes a single embedding the optimal choice.

Layer Selection Analysis

The efficacy of DDPO hinges on selecting an optimal intermediate layer (N) where the model's representations of benign and harmful prompts are most distinct. To find this layer, we analyzed the representational space across each LLM's depth by plotting the average pairwise cosine similarity between the last-token hidden states of harmful and benign prompts for each layer (Figure 3). A lower cosine similarity indicates greater class separation and a more suitable layer for intervention.

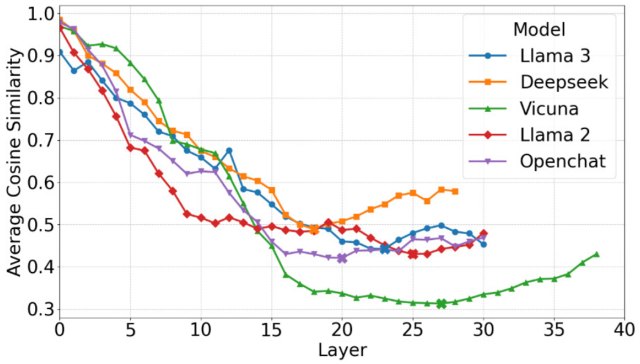

Figure 3: The cosine similarity between the hidden state representations of harmful and benign prompts across the layers of several LLMs. A lower similarity score indicates a layer can better distinguish between the two prompt types. For each model, the layer with the minimum cosine similarity is marked with a $\times$.

Our analysis reveals a consistent U-shaped trend in all models. Similarity is high in early layers, where the model processes surface-level features, and decreases in deeper layers as the model abstracts towards semantic intent. The similarity score reaches a minimum at an intermediate layer, which represents the point of maximal semantic divergence, before rising again as the model formulates a response.

This minimum point is the optimal layer for DDPO's intervention. As shown in the figure, these optimal layers (marked with an 'x') are consistently found in the later stages of the models, typically between the 62nd and 81st percentile of the model's depth. This empirical result justifies DDPO's core strategy of intervening where the LLM's own understanding is most discriminative, enabling a more nuanced defense than input-level methods.

Conclusion

This paper introduces Dynamic Deep Prompt Optimization (DDPO), a novel defense that addresses key limitations of existing prompt optimization methods against jailbreak attacks on LLMs. DDPO leverages hidden representations from the LLM's own early layers and a lightweight multilayer perceptron (MLP) to dynamically generate defensive embeddings. These embeddings are directly injected into the input of a later intermediate layer. Importantly, DDPO does not modify the target LLM's weights, relying solely on the concept of deep prompt optimization. Our experiments demonstrate two main improvements over existing approaches. First, DDPO establishes robust protection even on weakly aligned models where static defenses are largely ineffective. Second, it successfully resolves the safety-utility trade-off by accurately distinguishing malicious inputs from ambiguous and semantically challenging benign queries.

Acknowledgments

Portions of this work were supported by the National Science Foundation (2419880, 2347426).

References

Bi, X.; Chen, D.; Chen, G.; Chen, S.; Dai, D.; Deng, C.; Ding, H.; Dong, K.; Du, Q.; Fu, Z.; et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Bianchi, F.; Suzgun, M.; Attanasio, G.; Röttger, P.; Jurafsky, D.; Hashimoto, T.; and Zou, J. 2023. Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions. *arXiv preprint arXiv:2309.07875*.

Chao, P.; Debenedetti, E.; Robey, A.; Andriushchenko, M.; Croce, F.; Sehwag, V.; Dobriban, E.; Flammarion, N.; Pappas, G. J.; Tramer, F.; et al. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37: 55005–55029.

Chao, P.; Robey, A.; Dobriban, E.; Hassani, H.; Pappas, G. J.; and Wong, E. 2025. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 23–42. IEEE.

Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3): 6.

Chu, J.; Liu, Y.; Yang, Z.; Shen, X.; Backes, M.; and Zhang, Y. 2024. Comprehensive assessment of jailbreak attacks against llms. *arXiv e-prints*, arXiv–2402.

Deng, Y.; Zhang, W.; Pan, S. J.; and Bing, L. 2023. Multilingual jailbreak challenges in large language models. *arXiv preprint arXiv:2310.06474*.

Dong, H.; Xiong, W.; Goyal, D.; Zhang, Y.; Chow, W.; Pan, R.; Diao, S.; Zhang, J.; Shum, K.; and Zhang, T. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv e-prints*, arXiv–2407.

Guo, C.; Sablayrolles, A.; Jégou, H.; and Kiela, D. 2021. Gradient-based adversarial attacks against text transformers. *arXiv preprint arXiv:2104.13733*.

Jain, N.; Schwarzschild, A.; Wen, Y.; Somepalli, G.; Kirchenbauer, J.; Chiang, P.-y.; Goldblum, M.; Saha, A.; Geiping, J.; and Goldstein, T. 2023. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*.

Jiang, L.; Rao, K.; Han, S.; Ettinger, A.; Brahman, F.; Kumar, S.; Mireshghallah, N.; Lu, X.; Sap, M.; Choi, Y.; et al. 2024. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. *Advances in Neural Information Processing Systems*, 37: 47094–47165.

Kim, H.; Yuk, S.; and Cho, H. 2024. Break the breakout: Reinventing lm defense against jailbreak attacks with self-refinement. *arXiv preprint arXiv:2402.15180*.

Li, X.; Zhou, Z.; Zhu, J.; Yao, J.; Liu, T.; and Han, B. 2023. Deepinception: Hypnotize large language model to be jailbreaker. *arXiv preprint arXiv:2311.03191*.

Liu, X.; Ji, K.; Fu, Y.; Tam, W. L.; Du, Z.; Yang, Z.; and Tang, J. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Luo, Y.; Yang, Z.; Meng, F.; Li, Y.; Zhou, J.; and Zhang, Y. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Mazeika, M.; Phan, L.; Yin, X.; Zou, A.; Wang, Z.; Mu, N.; Sakhaee, E.; Li, N.; Basart, S.; Li, B.; et al. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*.

Mo, Y.; Wang, Y.; Wei, Z.; and Wang, Y. 2024. Fight back against jailbreaking via prompt adversarial tuning. *Advances in Neural Information Processing Systems*, 37: 64242–64272.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Robey, A.; Wong, E.; Hassani, H.; and Pappas, G. J. 2023. Smoothllm: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684*.

Shen, X.; Chen, Z.; Backes, M.; Shen, Y.; and Zhang, Y. 2024. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 1671–1685.

Shin, T.; Razeghi, Y.; Logan IV, R. L.; Wallace, E.; and Singh, S. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wallace, E.; Feng, S.; Kandpal, N.; Gardner, M.; and Singh, S. 2019. Universal adversarial triggers for attacking and analyzing NLP. *arXiv preprint arXiv:1908.07125*.

Wang, G.; Cheng, S.; Zhan, X.; Li, X.; Song, S.; and Liu, Y. 2023. Openchat: Advancing open-source language models with mixed-quality data. *arXiv preprint arXiv:2309.11235*.

Wen, Y.; Jain, N.; Kirchenbauer, J.; Goldblum, M.; Geiping, J.; and Goldstein, T. 2023. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Advances in Neural Information Processing Systems*, 36: 51008–51025.

Xie, Y.; Fang, M.; Pi, R.; and Gong, N. 2024. Gradsafe: Detecting jailbreak prompts for llms via safety-critical gradient analysis. *arXiv preprint arXiv:2402.13494*.

Xu, Z.; Jiang, F.; Niu, L.; Jia, J.; Lin, B. Y.; and Poovendran, R. 2024a. Safedecoding: Defending against jailbreak attacks via safety-aware decoding. *arXiv preprint arXiv:2402.08983*.

Xu, Z.; Liu, Y.; Deng, G.; Li, Y.; and Picek, S. 2024b. A comprehensive study of jailbreak attack versus defense for large language models. *arXiv preprint arXiv:2402.13457*.

Yi, S.; Liu, Y.; Sun, Z.; Cong, T.; He, X.; Song, J.; Xu, K.; and Li, Q. 2024. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint arXiv:2407.04295*.

Yuan, Y.; Jiao, W.; Wang, W.; Huang, J.-t.; He, P.; Shi, S.; and Tu, Z. 2023. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. *arXiv preprint arXiv:2308.06463*.

Zeng, Y.; Wu, Y.; Zhang, X.; Wang, H.; and Wu, Q. 2024. Autodefense: Multi-agent llm defense against jailbreak attacks. *arXiv preprint arXiv:2403.04783*.

Zheng, C.; Yin, F.; Zhou, H.; Meng, F.; Zhou, J.; Chang, K.-W.; Huang, M.; and Peng, N. 2024. On prompt-driven safeguarding for large language models. *arXiv preprint arXiv:2401.18018*.

Zhou, A.; Li, B.; and Wang, H. 2024. Robust prompt optimization for defending language models against jailbreaking attacks. *Advances in Neural Information Processing Systems*, 37: 40184–40211.

Zou, A.; Wang, Z.; Carlini, N.; Nasr, M.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

Zou, X.; Chen, Y.; and Li, K. 2024. Is the system message really important to jailbreaks in large language models? *arXiv preprint arXiv:2402.14857*.