Inverse Language Modeling towards Robust and Grounded LLMs

Davide Gabrielli[1], **Simone Sestito**[1], **Iacopo Masi**[1]

[1]Sapienza University of Rome
gabrielli.d@di.uniroma1.it, sestito@di.uniroma1.it, masi@di.uniroma1.it

Abstract

Interpretability and robustness remain major challenges for modern Large Language Models (LLMs), especially in settings where conventional evaluation or auditing tools are limited. To address this, we propose *Inverse Language Modeling* (ILM), a unified training framework that jointly enhances robustness to adversarial perturbations and enables a novel form of gradient-based interpretability. Rather than reconstructing exact input prompts, ILM encourages LLMs to develop gradient-aligned internal representations that allow the model to approximate *plausible* input patterns underlying a given output. This approximate inversion provides a new mechanism for analyzing model behavior, identifying potential triggers for unsafe generations, and providing a diagnostic signal that may support future auditing workflows. Our results show that ILM can simultaneously improve robustness and produce meaningful inversion signals, laying a foundation for LLMs that are not only more resilient but also more transparent and analyzable.

Code — https://github.com/davegabe/pag-llm/

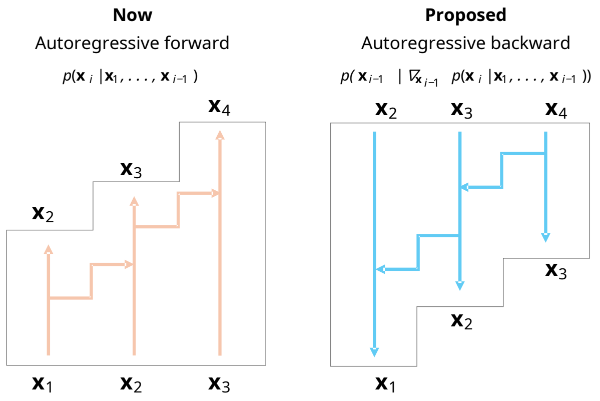
Introduction

Large Language Models (LLMs) excel at natural language tasks and reasoning. Today, a single foundation model can handle a wide range of NLP tasks. However, LLMs are still prone to hallucinations and remain sensitive to input variations, such as adversarial prompts. Recent work indicates that even sensical perturbations (Zou et al. 2023; Melamed et al. 2024) can trigger these issues, highlighting the potential for backdoors (Carlini et al. 2024). These risks become particularly salient when LLMs are used in culturally diverse communities, where ensuring consistent behavior with local values and ethical expectations is essential.

These problems emphasize the need for adversarial training tools (AT) for LLMs. However, the literature on this topic is not as dense as for deep classifiers, and yet the security of LLMs to adversarial perturbations remains an open challenge. Moreover, LLMs training is very costly, and therefore applying AT could only worsen the issue. At the same time, beyond robustness, there is a growing demand for mechanisms that help interpret and control why a model produces certain responses, especially on ethically sensitive or culturally situated topics.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Figure 1: Illustration of Inverse Language Modeling (ILM) setup. Forward pass predicts next tokens, backward pass reconstructs inputs from gradients.


Now
Autoregressive forward
$p(\mathbf{x}_i \mid \mathbf{x}_1, \ldots, \mathbf{x}_{i-1})$
$\mathbf{x}_4$
$\mathbf{x}_3$
$\mathbf{x}_2$
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$
Proposed
Autoregressive backward
$p(\mathbf{x}_{i-1} \mid \nabla_{\mathbf{x}_{i-1}} p(\mathbf{x}_i \mid \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}))$
$\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$
$\mathbf{x}_3$
$\mathbf{x}_2$
$\mathbf{x}_1$

Efficient solutions for AT for LLMs intercept a pressing need (Xhonneux et al. 2024). In this work, we define **robustness** as reduced sensitivity to adversarially perturbed prompts, and **grounding** as ensuring that LLMs "know what they have been asked", addressing evidence that they often fail to represent their own knowledge faithfully (Melamed et al. 2024; Bender et al. 2021).

In light of this, our objectives are twofold. The first centers on **Robustness**: we aim to study a new, fast, and efficient adversarial training (AT) approach for LLMs, which we call *Inverse Language Modeling (ILM)*. ILM is inspired by years of progress on robust classifiers and builds on the principle that *Perceptually Aligned Gradients (PAG)* imply robustness (Ganz, Kawar, and Elad 2023). While standard LLMs are trained in a forward-mode – where a transformer (Vaswani et al. 2017) predicts the continuation $\mathbf{y}$ of a text prompt $\mathbf{x}$ under self-supervision[1] – ILM extends this paradigm in a backward direction (see Figure 1). Given an output $\mathbf{y}$ (e.g., an answer), we investigate whether the LLM

[1]For clarity, we denote the text prompt as $\mathbf{x}$, which corresponds to the input token sequence $\mathbf{x}_0, \ldots, \mathbf{x}_{i-1}$, and the target sequence $\mathbf{y}$ is the one-step left-shifted version of $\mathbf{x}$.

can approximate the conditioning prompt $\mathbf{x}$.

The second objective relates to **Grounded LLMs** and naturally emerges from the first. By enabling inversion of an output $\mathbf{y}$, ILM provides a diagnostic signal that may support future auditing workflows. Rather than guaranteeing exact prompt recovery, it offers a way to trace back potential prompt approximations that could have produced a malicious or undesired output, thereby grounding the model's behavior in more transparent diagnostic evidence.

Importantly, ILM does not reverse the token sequence; it recovers the input prompt by performing a gradient-based alignment that is informed by both the output probabilities of the model and the representations accumulated in each layer during the forward pass.

Prior Work

Research on adversarial attacks against Large Language Models (LLMs) has advanced significantly, particularly in generating adversarial suffixes designed to bypass alignment safeguards. Early techniques, such as HotFlip (Ebrahimi et al. 2018) and Greedy Coordinate Gradient (GCG) (Zou et al. 2023), focused on manipulating the input text or its embedding gradients to induce undesirable behavior from LLMs. GCG modifies token selections iteratively based on gradient information. Subsequent enhancements, including Probe Sampling (Zhao et al. 2024) and token similarity-based heuristics (Li et al. 2024), have improved its efficiency.

More recent methods include AutoDAN (Liu et al. 2024), which leverages genetic algorithms to produce fluent and stealthy adversarial suffixes, and its successor AutoDAN Turbo (Liu et al. 2025), which coordinates multiple LLMs for strategy development and attack evaluation. AdvPrompter (Paulus et al. 2024) takes a different approach by fine-tuning a model specifically to generate coherent adversarial suffixes, allowing fast and automated jailbreaking.

On the defensive side, perplexity-based filtering (Alon and Kamfonas 2023) has proven to be effective in identifying adversarial suffixes by exploiting their typically high perplexity. However, newer attacks are designed to bypass such detection mechanisms by optimizing fluency and semantic plausibility. In addition, work on language model inversion (Morris et al. 2023) explores the recovery of original prompts from output probabilities, similar to reconstruction techniques in computer vision. These findings have informed strategies for generating adversarial prompts using only output distributions.

Unlike prior work focused on suffix generation or language inversion as an offensive tool, our research seeks to understand and mitigate these vulnerabilities. In particular, we study **"evil twin" prompts** as defined in Melamed et al. (2024); Rakotonirina et al. (2024). Given a text prompt $\mathbf{x}$ and the completion, $\mathbf{y}$, we performed an optimization so that given $\mathbf{y}$, we find a new nonsensical $\mathbf{x}^\star$ – the "evil twin" – such that the loss $\mathcal{L}(\mathbf{x}^\star, \mathbf{y}; \boldsymbol{\theta}) \ll \mathcal{L}(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$, where $\mathcal{L}$ is the next-token prediction loss of the LLM and $\boldsymbol{\theta}$ are LLM's parameters. These $\mathbf{x}^\star$ are syntactically implausible out-of-distribution inputs that nevertheless lead to the same output, as illustrated in Table 1. Despite producing identical continuations, $\mathbf{x}$ and $\mathbf{x}^\star$ induce notably different entropy distributions. These prompts are also fragile – small changes typically break the adversarial effect, highlighting a key vulnerability in LLM robustness and alignment.

To handle evil twin prompts, we propose Inverse Language Modeling (ILM), a novel training framework that improves LLM robustness. ILM enables **both forward modeling and partial inversion**, encouraging the model to not only generate fluent output but also remain sensitive to input semantics.

Method

Preliminary Study on PAG on Text Classification

In this preliminary experiment, we investigate the application of Perceptually Aligned Gradients (PAG) (Ganz, Kawar, and Elad 2023) to sentence classification using hidden state representations from the DistilBERT language model (Sanh et al. 2019a). While PAG has been primarily explored in the context of image classification, we adapt the methodology to the hidden state space of a transformer model to explore its effects on robustness and interpretability in NLP tasks. The core idea of PAG is to encourage gradients to align with semantically meaningful directions, and we hypothesize that this can lead to more robust and interpretable text representations as well.

To prove our point, we ran a proof-of-concept experiment using a classifier trained on top of the hidden state associated with the `[CLS]` token, adopting the `distilbert-base-multilingual-cased` (Sanh et al. 2019b), on Amazon Review Multi dataset (Keung et al. 2020). We considered 12 classes as the combination of some languages (English, German, Spanish, and French) and some review ratings (1, 3, and 5 stars).

PAG Application. To apply PAG, we modify the standard cross-entropy loss function with a regularization term that encourages alignment between the input gradient and a "proxy" ground-truth gradient. The modified loss function for the classifier built on top of the frozen DistilBERT backbone is:

$$\mathcal{L} = \mathcal{L}_{CE}(f_\theta(\mathbf{x}), y) + \lambda\, \mathcal{L}_{PAG}(\mathbf{x}) \quad \text{where}$$
$$\mathcal{L}_{PAG}(\mathbf{x}) = \frac{1}{C}\sum_{y=1}^{C} 1 - \frac{\nabla_{\mathbf{h}} f_\theta(\mathbf{x})_y^\top\, g(\mathbf{x}, y)}{\|\nabla_{\mathbf{h}} f_\theta(\mathbf{x})_y\|\, \|g(\mathbf{x}, y)\|} \tag{1}$$

where:

- $\mathbf{x}$ is the input sentence,
- y is the true class label,
- $f_\theta(\mathbf{x})$ is the classifier that takes the DistilBERT hidden state ($\mathbf{h}$) as input and predicts the class,
- $\mathcal{L}_{CE}$ is the cross-entropy loss,
- λ is a hyperparameter controlling the strength of the PAG regularization,
- C is the number of classes,
- $\nabla_{\mathbf{h}} f_\theta(\mathbf{x})_y$ is the gradient of the classifier's output for class y with respect to the hidden state h,

Input	**Output $\mathbf{y}$**	**Loss**
$\mathbf{x}$: Stevens recorded and produced the album at multiple $\mathbf{x}^\star$: Zo Certified Cities (. broadcastquartered Fitness Academy thirteen	locations in the United	5.3642 **5.1302**
$\mathbf{x}$: After the introduction of the Majors , The $\mathbf{x}^\star$: REQU Apart British received reformsMilitaryestic Division The	British Army was divided	11.2146 **7.1899**
$\mathbf{x}$: The founding director , Peggy Loar , left $\mathbf{x}^\star$: tested UberERIC definitionCalifornia sustainability RutgersOL Jensen regarding	the University of California	7.2669 **6.4402**
$\mathbf{x}$: Ruiz notes that writing also has the power $\mathbf{x}^\star$: Report Global feminism agenda Representatives tell Sacredixties Trying	to change the world	5.9135 **4.6041**

Table 1: Original inputs $\mathbf{x}$ and adversarial examples $\mathbf{x}^\star$ generated using the GCG method for the SmolLM-360M model. The table shows how each original input and its corresponding adversarial example result in the same output, along with the loss values calculated for the output token IDs. These examples show that LLMs can be manipulated into assigning lower loss to nonsensical prompts than to the original, meaningful input – highlighting a vulnerability that ILM is designed to address.

attack $\rightarrow$	**APGD** Croce and Hein (2020)		**Square** Croce and Hein (2020)	**FGSM** Goodfellow, Shlens, and Szegedy (2015)	
	ε			α	
$g(\mathbf{x}) \downarrow$	1e-3	0.5		5e-3	1e-2
Baseline	36.5%	31.2%	36.3%	27.3%	8.9%
Identity	28.3%	25.0%	27.2%	25.7%	8.0%
PAG	**48.1%**	**45.0%**	**49.3%**	**43.5%**	**25.7%**

Table 2: Robustness of classifier models with PAG variants under APGD, Square, and FGSM attacks. Higher percentages indicate stronger robustness.

- $g(\mathbf{x}, y)$ is the "proxy" ground-truth gradient for y.

"Proxy" Ground-Truth Gradient. We define the "proxy" ground-truth gradient (**PAG** variant) as the difference between the hidden state of the input sentence $\mathbf{h}_\mathbf{x}$ and the hidden state of a randomly sampled sentence $\mathbf{u}_y$ from the same class y:

$$g(\mathbf{x}, y) = \mathbf{u}_y - \mathbf{h}_\mathbf{x}. \tag{2}$$

This encourages the model to learn hidden state representations where the gradient points in the direction of other examples from the same class.

Another variant for this $g(\cdot)$ function, named **Identity**, has been tested and compared. This one forced the model to reconstruct the input via the received gradients as:

$$g(\mathbf{x}, y) = \mathbf{x}. \tag{3}$$

The baseline is the model trained with the same architecture and hyperparameters but $\lambda = 0$, to exclude $\mathcal{L}_{PAG}$.

Evaluation. According to the results in Table 2, the strongest model in robustness is the one trained with the full PAG loss with Equation 2, which forces the model to make the gradients on the input point towards the direction of the predicted class. These models have been attacked by APGD, Square (Croce and Hein 2020), and FGSM (Goodfellow, Shlens, and Szegedy 2015).

Inverse Language Modeling

Our procedure takes inspiration from robust classifiers that have Perceptually Aligned Gradients (PAG) (Ganz, Kawar, and Elad 2023; Mirza et al. 2024) over the input space. ILM is non-iterative and just requires double backpropagation (Drucker and Le Cun 1992), which can be easily implemented with current autograd tools.

ILM performs the following: instead of training the LLM to *only* maximize $p(\mathbf{y}|\mathbf{x})$, we also invert it and from the output $\mathbf{y}$, we aim to reconstruct the input $\mathbf{x}$. This procedure is not simply a mere double forward pass with the original text and its reverse. Instead, we first impose a loss for $p(\mathbf{y}|\mathbf{x})$, yet instead of updating the weights, we also receive gradients over input tokens $\nabla_\mathbf{x}\mathcal{L}(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$ requiring them to predict some tokens in $\mathbf{x}$, depending on the exact model variant among the ones discussed later. This focus on bidirectional understanding during pretraining is key to improving the model's overall language comprehension.

From the gradients flow outlined in Figure 2, it is possible to see that the influence of a single token affects only the future hidden states in the forward pass, while it influences every other token during the backward pass. This means that in the gradients received on the input tokens are affected by the entire sequence.

We will use ILM at training time, while at test time we exploit the GCG algorithm to find, given an original text prompt $\mathbf{x}$ and the completion $\mathbf{y}$, a new nonsensical $\mathbf{x}^\star$ such that the loss $\mathcal{L}(\mathbf{x}^\star, \mathbf{y}; \boldsymbol{\theta}) \ll \mathcal{L}(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$, where $\mathcal{L}$ is the next-token prediction loss of the LLM and $\boldsymbol{\theta}$ are LLM's parameters. We prove there exists a nonsensical prompt $\mathbf{x}^\star$ that "connects" better to $\mathbf{y}$ (lower loss) than the natural $\mathbf{x}$.

The standard formulation of Perceptually Aligned Gradients (PAG), as typically applied in image classification – Equation 1 – is not directly transferable to Large Language Models (LLMs). This is primarily due to the fundamental differences in the nature of the input data. Images are represented as tensors with continuous values, inherently containing class-discriminative information within a single sample. In contrast, sequence data processed by LLMs relies on the entire sequence context for prediction, not isolated elements. Furthermore, the input tokens are initially represented as one-hot vectors, which do not encode semantic information.

Calculating gradients with respect to these input tokens poses significant challenges:

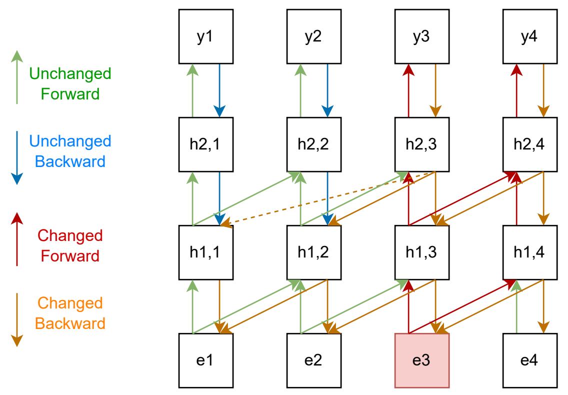


Figure 2: Gradient influence diagram in an LLM changing only token e_3. In the forward pass, only future hidden states are affected. In the backward pass, the change propagates to every embedding token.

- Gradients with respect to individual input tokens lack the crucial context of the entire sentence, analogous to a single pixel being insufficient to determine an image's class.
- The vast vocabulary size of LLMs (hundreds of thousands of tokens) renders the class-iterative PAG loss computationally infeasible due to the sheer number of classes, exacerbating the entropy in the classification task.

Consequently, our approach deviates from the standard PAG for classifiers. We opt to focus solely on the gradients with respect to the actual input tokens and, furthermore, we intend to classify the actual tokens, as in Figure 1, rather than employing the cosine distance for gradient direction, aiming to leverage this for LLM inversion as well.

For this experiments, the architecture of our model is a small decoder-only transformer, with **Weight Tying** (Press and Wolf 2017; Inan, Khosravi, and Socher 2017) enabled, with 3 hidden layers and an hidden layers vector size set to 640.

The dataset used is **TinyStories** (Eldan and Li 2023) with a tokenizer trained from scratch using the standard Byte-Pair Encoding (Gage 1994), in order to have a flexible vocabulary size, to eventually have experiments of different complexities and entropy in the next token classification. Specifically, we used a vocabulary of 2048 possible tokens. Also, the dataset samples include an overlap of 25% between the original sentences. This overlap increases variability in sentence starts, providing the model with more diverse context patterns and better approximating realistic sentence-completion scenarios. The finally constructed dataset [2] has been uploaded to HuggingFace for reproducibility.

The backward prediction strategy shares a common logic across all model variants, which can be formalized as follows. Given the cross-entropy loss between predicted tokens

[2]https://huggingface.co/datasets/DaveGabe/TinyStoriesV2_cleaned-voc2048-seq256-overlap25

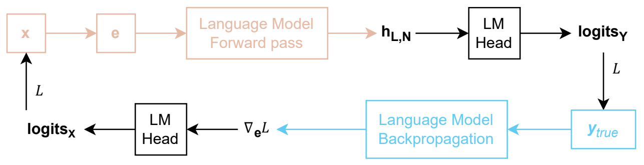


Figure 3: Parallelism between last hidden states and embedding gradients: both can be mapped through LM head to token predictions.

and their ground truth, we compute the gradient with respect to the embedding vectors. To handle different variants consistently, we define a mapping $\phi(\mathbf{e}_i, \nabla_{\mathbf{e}_i}\mathcal{L}_{CE})$ that specifies how the gradient is interpreted for classification. Then, the output of $\phi(\dots)$ is normalized and used with the LM Head weight matrix to get a probability distribution over the vocabulary. The general backward prediction for any token can then be written as:

$$\begin{aligned} \mathcal{L}_{CE} &= CE(\mathbf{y}_{\text{true}}, \mathbf{y}_{\text{pred}}) \\ \mathbf{g}_i &= \text{LayerNorm}(\phi(\mathbf{e}_i, \nabla_{\mathbf{e}_i}\mathcal{L}_{CE})) \\ \mathbf{z}_i &= \mathbf{W}_{\text{LM\_head}}\ \mathbf{g}_i \\ \hat{\mathbf{y}}_i &= \text{softmax}(\mathbf{z}_i) \end{aligned} \tag{4}$$

This formulation highlights a very nice parallelism between the forward and backward mode, which is summarized in Figure 3: the gradients vector on the embeddings of a specific token is used in the same way as the last hidden state that will conduct to a prediction of the next token. Indeed, replacing $\nabla_{e_i}\mathcal{L}_{CE}$ with the last hidden state will give us exactly the standard forward pass of an LLM. This is possible because the dimensionality of the last hidden state and the one of the embedding vector is the same.

The final loss used to train these models is obtained by the addition of the inverse LM prediction loss, where we observed $\lambda = 2.0$ to be a good hyperparameter:

$$\begin{aligned} \mathcal{L} = &\underbrace{\mathcal{L}_{CE}(\mathbf{y}_{\text{true}}, \mathbf{y}_{\text{pred}})}_{\text{Forward: from the input x, encode y}} \\ + &\underbrace{\lambda\ \mathcal{L}_{CE}(\mathbf{x}, \hat{\mathbf{y}})}_{\text{Backward: from y, decode back x}} \end{aligned} \tag{5}$$

Model Variants

We evaluate four training strategies, each differing in how gradients are used for inversion:

Baseline. The model is trained only with the standard forward Cross-Entropy loss.

Identity. During training, the model is required to reconstruct every input token directly from its corresponding gradient:

$$p(\mathbf{x}_i \mid \nabla_{\mathbf{e}_i}\mathcal{L}_{\text{CE}}(f_\theta(\mathbf{x}), \mathbf{y})), \quad \forall i \in [1, N]. \tag{6}$$

BERT-like. A subset of tokens is masked in the input sequence, and the model must predict them from the embedding gradients – analogous to the BERT (Devlin et al. 2019) training scheme, but applied in the backward pass.

Inv-First. Only the first token of the sentence is reconstructed from its gradient, by predicting

$$p(\mathbf{x}_0 \mid \nabla_{\mathbf{e}_0}\mathcal{L}_{\text{CE}}(f_\theta(\texttt{[PAD]} \,\|\, \mathbf{x}_{1:N}), \mathbf{y})). \qquad (7)$$

Each strategy comes in two approaches, depending on the implementation of the $\phi(\dots)$ function mentioned in Equation 4. When we consider gradients as directions, we classify on $\mathbf{e}_i - \nabla_{\mathbf{e}_i}\mathcal{L}_{CE}$, thus by imposing $\phi(\mathbf{e}_i, \nabla_{\mathbf{e}_i}\mathcal{L}_{CE}) = \mathbf{e}_i - \nabla_{\mathbf{e}_i}\mathcal{L}_{CE}$. On the other hand, the case where we use the gradients as pure values is simpler: $\phi(\mathbf{e}_i, \nabla_{\mathbf{e}_i}\mathcal{L}_{CE}) = \nabla_{\mathbf{e}_i}\mathcal{L}_{CE}$, discarding the input embedding value.

Experimental Evaluation

We evaluated the performance of these models from a twofold perspective: their ability to invert a part of text, given the continuation but not what comes before; their robustness against GCG attack. These aspects are equally important to develop robust and grounded LLMs.

Inversion Procedure

The evaluation relies on three complementary measures, all tailored to the inverse generation setting. Validation loss and validation accuracy are computed under the same conditions as training, serving as baseline indicators of model fit and predictive performance. In addition, we report Inverse LM accuracy, which measures the ability of a model to reconstruct a masked token $\mathbf{x}_i$ from its gradients, given the remaining context. This provides a direct assessment of the backward prediction mechanism.

Formally, for an input sequence $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$, the i-th token is replaced with a placeholder, yielding $\mathbf{x}' = (\texttt{<|pad|>}, \mathbf{x}_{i+1}, \dots, \mathbf{x}_N)$, with targets shifted as $\mathbf{y}' = (\mathbf{x}_{i+1}, \dots, \mathbf{x}_N)$. The cross-entropy loss $\mathcal{L}_{\text{CE}}(\mathbf{x}', \mathbf{y}'; \theta)$ produces gradients with respect to the masked embedding $\nabla_{\mathbf{e}_i}$, which are transformed via the backward prediction rule (Equation 4) into a distribution $\hat{\mathbf{y}}_i$. Inverse LM accuracy is then computed by comparing $\arg\max \hat{\mathbf{y}}_i$ against the ground-truth token $\mathbf{x}_i$.

Inversion Evaluation

To assess inversion capabilities, we extend the task beyond single-token recovery and instead invert multiple tokens autoregressively. The procedure follows a beam-search strategy, as detailed in Algorithm 1, where candidate prefixes are iteratively expanded and filtered by perplexity until a coherent reconstruction emerges.

In the evaluation process, we considered only the combination of initialization strategy and model variant used in the specific training process. This means that the `Identity` has the unknown token initialized using the simple bigram, while other variants have it set to `<|pad|>`, since they reflect the same initialization strategy used during training. Of course, we cannot initialize the `Identity` model during inversion with the real token, as in training, because we do not know it yet. However, the best approximation we can do is to use a bigram model, which is pretty simple but also powerful to help the model invert better than starting from a totally random token or using a fixed `<|pad|>` because it has never observed it during training.

Algorithm 1: Autoregressive Inversion Evaluation with Beam Search

```
Require: Input sample $\mathbf{x}$ of length $n$, beam size $b$, split position $k$
Ensure: Inverted prefix $\mathbf{x}_{\text{inv}}$
 1: $\mathbf{x}_p \leftarrow \mathbf{x}_{0:k}$                    ▷ Original prefix (hidden)
 2: $\mathbf{x}_s \leftarrow \mathbf{x}_{k:n}$                    ▷ Visible suffix
 3: $\mathbf{X} \leftarrow \{\mathbf{x}_s\}$         ▷ Initialize beam set with suffix only
 4: while inverted prefix not sufficiently long do
 5:     for each sequence $\mathbf{x}' \in \mathbf{X}$ do
 6:         Compute top-$b$ tokens for the previous position
 7:         Extend $\mathbf{x}'$ with each candidate token
 8:     end for
 9:     $\mathbf{X} \leftarrow$ top-$b$ sequences with lowest perplexity
10: end while
11: return $\mathbf{x}_{\text{inv}} \leftarrow \arg\min_{\mathbf{x}' \in \mathbf{X}}$ Perplexity$(\mathbf{x}')$
```

In order to make this final evaluation on inversion as complete as possible, we introduced several new metrics: they allow us to better comprehend the obtained results and have a better understanding of the model training strategies applied. When we refer to a third-party model, we are using `meta-llama/Llama-3.2-1B`.[3]

- *Rec* (`token_recall`) refers to the fraction of unique tokens from the reference sequence that were correctly generated by the model. A higher recall value means the model captured more of the words from the reference
- *Prec* (`token_precision`) refers to the fraction of unique tokens in the generated sequence that are present in the reference. A higher precision value indicates the model didn't introduce many irrelevant or "hallucinated" words
- *F1* (`token_f1`) refers to the harmonic mean of precision and recall. It provides a single score that balances the trade-off between the two. A high F1 score indicates a good balance of both generating relevant tokens and avoiding irrelevant ones
- *Acc* (`positional_accuracy`) refers to the exact token match at each position in the generated sequence compared to the reference: unlike the token-based metrics above, this one is sensitive to token order
- *OPP* (`original_prefix_perplexity`) measures the perplexity of the original prefix text alone, using the third-party model. This should serve as an indication of "how natural" the prefix text is. This metric will be *the same* for all models, since it does not depend on the model, but only on the data to be predicted.
- *FPP* (`full_predicted_perplexity`) measures the perplexity of the predicted prefix text, concatenated with the suffix, using the third-party model. This should serve as an indication of "how grounded" the generated prefix is with the suffix

[3]https://huggingface.co/meta-llama/Llama-3.2-1B

	Grad.	Rec ↑	Prec ↑	F1 ↑	Acc ↑
Baseline		20.9%	18.8%	19.7%	2.4%
Inv-First	Val.	11.3%	10.1%	10.7%	1.7%
Bert-like	Val.	2.9%	2.7%	2.8%	0.3%
Identity	Val.	0.7%	0.7%	0.7%	0.1%
Inv-First	Dir.	13.3%	12.0%	12.6%	2.4%
Bert-like	Dir.	0.1%	0.1%	0.1%	0.1%
Identity	Dir.	**22.5%**	**20.2%**	**21.2%**	**2.5%**

Table 3: Inversion evaluation on token-level metrics (Recall, Precision, F1, Accuracy). Higher values mean better recovery of original tokens.

	Grad.	OPP	FPP ↓	PPP ↓	SS ↑
Baseline		37.83	**8.34**	112.82	0.28
Inv-First	Val.	37.83	10.21	1576.23	0.25
Bert-like	Val.	37.83	11.54	5501.86	0.17
Identity	Val.	37.83	13.88	14658.58	0.12
Inv-First	Dir.	37.83	9.77	1012.80	**0.30**
Bert-like	Dir.	37.83	11.05	563.26	0.11
Identity	Dir.	37.83	**8.34**	**106.31**	**0.30**

Table 4: Sentence-level inversion metrics: OPP (original prefix perplexity), PPP (predicted prefix), FPP (full predicted), SS (semantic similarity).

- *PPP* (predicted_prefix_perplexity) measures the perplexity of the predicted prefix text alone, using the third-party model. This should serve as an indication of "how natural" the generated text is
- *SS* (semantic_similarity) refers to the semantic meaning of the generated text compared to the ground-truth, regardless of the specific words used. It computes the cosine similarity between the embedding of the two sentences, obtained by an external model [4]

Note that *FPP* shows much less variance between the tested models, because it computes the perplexity of the entire sentence, which is the concatenation of the predicted prefix and the given suffix from the dataset. Since the latter is much longer than the former, the values of *FPP* tend to be pretty low. However, we are interested in the difference between the models. Also, it is clearly visible that *PPP* is one order of magnitude larger in the models that predict the previous token using the gradient vector, without summing it up first with the embedding of the input token they're inverting on, used as the initialization value. This clearly demonstrates the intuition for which using gradients as directions would have made the model hold much better.

Robustness Against GCG

We evaluate robustness using the success rate of Greedy Coordinate Gradient (GCG) attacks (Zou et al. 2023). This benchmark aligns naturally with the objectives of our training procedure: our ultimate goal is to produce LLMs that are more grounded, ensuring that their responses are faithful to and informed by the prompts they receive. The procedure to evaluate the models follows these rules, which are repeated for 30% randomly picked samples from the test set, but consistently chosen for all the model variants.

[4]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

	Grad.	GCG Success Rate ↓	GCG Average Steps (mean ± stddev)
Baseline		95.9%	277 ± 148
Identity	Val.	88.1%	274 ± 145
Bert-like	Val.	**0.8%**	249 ± 148
Inv-First	Val.	85.0%	320 ± 134
Identity	Dir.	82.8%	284 ± 141
Bert-like	Dir.	85.5%	287 ± 143
Inv-First	Dir.	89.3%	313 ± 134

Table 5: Success rates of GCG adversarial prompts against ILM variants. Lower values indicate stronger robustness.

Algorithm 2: Single-Sentence GCG Attack

```
Require: Expected continuation string $\mathbf{y}$ to be attacked, length of the attack prefix $n$, number of iterations $T$
Ensure: Best attack prefix $\mathbf{x}^\star$ with loss $\mathcal{L}_{\text{GCG}}$
1:  $\mathbf{x}^\star \leftarrow$ random one-hot tokens matrix of size $|V| \times n$
2:  $step \leftarrow 0$                                  ▷ Iteration counter
3:  $d \leftarrow 0$                                     ▷ Loss non-decrease counter
4:  $\mathcal{L}_{\text{old}} \leftarrow \infty$          ▷ Last loss found
5:  while $step < T$ and $d < 10$ do
6:      Compute a batch of candidate prefixes $\mathbf{X}$ running one step of GCG
7:      $\mathbf{x}^\star \leftarrow \arg\min_{\mathbf{x}\in\mathbf{X}} \mathcal{L}_{\text{CE}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})$
8:      $\mathcal{L}_{\text{GCG}} \leftarrow \mathcal{L}_{\text{CE}}(\mathbf{x}^\star, \mathbf{y}, \boldsymbol{\theta})$   ▷ Take the min loss so far
9:      if $\mathcal{L}_{\text{GCG}} < \mathcal{L}_{\text{old}}$ then
10:         $\mathcal{L}_{\text{old}} \leftarrow \mathcal{L}_{\text{GCG}}$
11:         $d \leftarrow 0$
12:     else
13:         $d \leftarrow d + 1$
14:     end if
15:     $step \leftarrow step + 1$
16: end while
17: return $\mathcal{L}_{\text{GCG}}$
```

From the results observable in Table 5, we can notice that the majority of the variants have an improvement in robustness against GCG attacks. Looking at the specific variant that was flagged as the best one in the inversion task in the previous chapter, which is *Identity (grad. direction)*, it allowed a reduction in the success rate of more than 13%. This improvement can be attributed to the model's ability to better condition the continuation of a sentence with the actual prompt it was given as input (also referred to as "*grounded*"), which may lead to a more robust model. However, there is the specific case of *Bert-like (grad. value)*, which corresponds to the model variant that imitates BERT in the backward pass, masking some tokens and letting the model predict them directly, classifying on the received gradient on the PAD token. This model scores an incredibly low GCG success rate, making us suppose that it may actually strongly go in the direction of adversarially robust models, at least on the gradient-based white-box GCG attack. Given the huge difference between the baseline and this variant, the experiments have been repeated from the initial training phase, but they gave us the same results as in the first run of the pipeline. For sure, this aspect will require a deeper investigation.

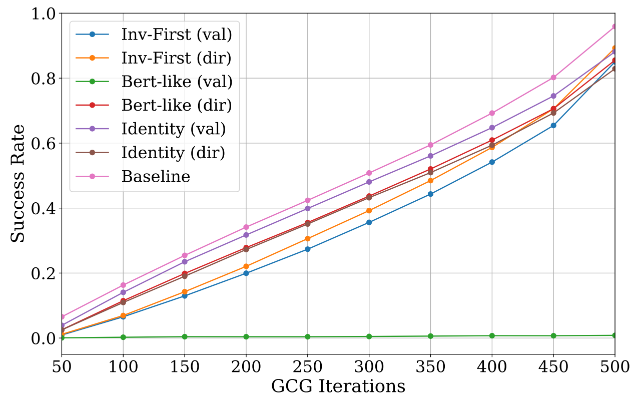


Figure 4: GCG Success Rate varying according to the number of iterations performed.

In our experiments, we also studied the correlation between the number of GCG algorithm maximum allowed iterations and the success rate of the attack, always computed as the number of tokens that match between the LLM's responses to the original input $\mathbf{x}$ and the attack input $\mathbf{x}'$, while keeping all other hyperparameters, like the search window width, unaltered. Interestingly, in Figure 4 some lines actually cross each other when increasing the number of GCG iterations: this may indicate that some variants are more effective at different values of the GCG iterations. For instance, *Inv-First (grad. direction)*, represented as the orange line, is better than *Bert-like (grad. direction)*, represented as the red line, when the number of allowed iterations is pretty low; however, at the end of the plot, at the maximum number of iterations tested, their effectiveness is the opposite. That being said, this phenomenon is not dramatic and makes only slight changes in the final results reported in the previously discussed table.

To have a better understanding of the GCG results listed in the previous table, we measured the following other metrics, always considering the subset of successful GCG attacks:

- *Original CE loss.* The cross-entropy loss on the unperturbed input, $\mathcal{L}_{\text{CE}}(f_\theta(\mathbf{x}_0,\ldots,\mathbf{x}_N),\mathbf{y}_0,\ldots,\mathbf{y}_M)$ where lower values indicate that $\mathbf{y}$ is a natural continuation of $\mathbf{x}$.
- *Attack CE loss.* The cross-entropy loss on the adversarially perturbed input, $\mathcal{L}_{\text{CE}}(f_\theta(\mathbf{x}'_0,\ldots,\mathbf{x}'_N),\mathbf{y}_0,\ldots,\mathbf{y}_M)$ where higher values are desirable, since a low loss would mean that $\mathbf{y}$ also appears natural given $\mathbf{x}'$, which is precisely the vulnerability GCG exploits.
- *KL divergence.* The divergence between the output distributions of the original and perturbed inputs: $KL(f_\theta(\mathbf{x}_0,\ldots,\mathbf{x}_N), f_\theta(\mathbf{x}'_0,\ldots,\mathbf{x}'_N))$. This measures how much the perturbation alters the probability distributions of the model's predictions.

	Grad.	Original X CE-loss ↓	Attack X' CE-loss ↓	Delta CE-loss ↓	KL Divergence ↑
Baseline		13.28	10.97	2.31	2.19
Identity	Val.	12.77	11.21	1.56	2.23
Bert-like	Val.	13.26	10.25	3.01	**54.19**
Inv-First	Val.	**11.09**	9.72	<u>1.37</u>	2.44
Identity	Dir.	12.58	11.12	1.46	2.47
Bert-like	Dir.	11.49	10.34	**1.15**	2.23
Inv-First	Dir.	<u>11.21</u>	9.81	1.40	2.44

Table 6: Evaluation of the best attack input found using GCG.

In all previous mathematical formulation, note that $f_\theta(\mathbf{x}_0,...,\mathbf{x}_N)$ is the function which runs the LLM under attack and returns the logits that correspond to the prediction of the $\mathbf{y}$ output, not the ones that predict parts of the input prompt $\mathbf{x}$ itself, as it would be wrong to consider for our analysis.

From the metrics in Table 6, we observe that robust variants, such as *Identity (grad. direction)*, not only exhibit a substantially lower attack success rate (ASR) compared to the baseline, but also display a smaller increase in Cross-Entropy loss when the attack succeeds. Recall that this *delta* quantifies the extent to which the model is "fooled" by the attack, defined as the difference between the loss on the original, human-readable input and the loss on the adversarially generated sequence. A higher delta indicates greater susceptibility, as the model interprets the attack sequence as being more strongly aligned with the target continuation $\mathbf{y}$. Finally, the **KL-divergence** allows us to observe how much the output distributions returned by the LLM differ between $\mathbf{x}$ and $\mathbf{x}'$. The more different they are, the better the model can discriminate between them, recognizing that they are actually two distinct and different pieces of input.

To have a complete evaluation, we adopt a similar approach to the one we used during inversion: using a third-party model to compute some other statistics lets us abstract away from the biases in our LLMs. Here, since the perplexity of the attack prefix is computed with a third-party independent model, it can easily return the real naturalness of the generated prefix, instead of being influenced by the attack itself and wrongly reporting that it will be even more natural than the human prefix. Remember that we are considering only the successful attacks, ignoring the ones that have failed, since they are not useful to understand the quality of the attacks. Also, because of that, the results involving the `bert-like` variant using gradients as values will have a much smaller number of samples that participate in these metrics.

	Grad.	Original X Perplexity	Attack X' Perplexity ↓	Semantic Similarity ↑
Baseline		44.14	17344.04	0.13
Identity	Val.	43.98	**8322.25**	**0.18**
Bert-like	Val.	40.37	11817.21	0.11
Inv-First	Val.	44.81	9431.09	0.16
Identity	Dir.	44.71	10929.21	0.15
Bert-like	Dir.	44.74	10611.09	0.13
Inv-First	Dir.	43.50	12344.85	0.13

Table 7: Evaluation of the attack input prefix against the original input prefix for successful GCG attacks.

	Grad.	Perplexity ↓	CE Loss ↓
Baseline		**4.83**	**1.58**
Identity	Val.	5.07	1.63
Bert-like	Val.	5.79	1.76
Inv-First	Val.	8.41	2.13
Identity	Dir.	5.08	1.62
Bert-like	Dir.	5.42	1.69
Inv-First	Dir.	6.82	1.92

Table 8: Quantitative forward mode evaluation.

Forward Mode Evaluation

Finally, we tested that these models are still functioning properly in forward mode, without experiencing **performance degradation**. Checking out the perplexity during training and validation, it has been observed that the performance of the custom models with the regularization term on the gradients $\nabla_{\mathbf{e}}\mathcal{L}_{\text{CE}}$ does not penalize the model's ability to speak fluently during the standard usage in forward mode. This outcome is noteworthy, as adversarial training in literature often necessitates additional parameters or extended training to achieve comparable forward-mode performance, since part of the model's capacity is effectively devoted to satisfying the adversarial objective. In Table 8 we can observe that the worst model is *Inv-First (grad. value)*. This relatively high perplexity value is also confirmed in the qualitative examples in the tables below, where the sentences are by far the ones that make less sense and seem more confused and repetitive.

Conclusions and Future Work

In conclusion, this paper introduces Inverse Language Modeling (ILM) as a novel framework designed to simultaneously address two critical challenges in Large Language Models: robustness and grounding. Our experiments demonstrate ILM's potential to enhance LLMs' resilience against input perturbations, a key step towards mitigating vulnerabilities to adversarial attacks. Furthermore, ILM offers a pathway to improved grounding, enabling LLMs to better correlate their outputs with the input prompts and thereby facilitating the identification of potentially problematic input triggers. Crucially, this inversion ability opens the door to examining which normative assumptions or ethical priors the model implicitly relies on. For instance, when an LLM answers a controversial or value-laden question, ILM allows us to approximate the "implicit prompt" or internal framing that the model is using. This makes ILM a promising tool not only for robustness but also for value transparency – providing a lightweight way to inspect how an LLM internally justifies its answers, and opening a research path toward value transparency that may eventually help communities detect misalignment with local ethical norms or social expectations.

There are several promising avenues for future research. While ILM is introduced within the context of pre-training, an interesting direction would be to explore its application in the **fine-tuning stage**. Specifically, one could investigate how the principles of inverse modeling can be incorporated into the fine-tuning process to improve the robustness and generalization of LLMs on downstream tasks. Additionally, research could explore the potential benefits of combining ILM with instruction tuning, to further align LLM behavior with human preferences and instructions. Future work should evaluate ILM on larger-scale LLMs, including Llama-3.2-7B, and even bigger models, to rigorously assess its scalability and effectiveness as model capacity increases.

	Grad.	Completion for "One day,"
Baseline		a little boy named Tim wanted to travel to a far mountain. He asked his dad for a raft,
Identity	Val.	a little girl named Lucy went to the park with her mom. Lucy liked to play on the swings
Bert-like	Val.	a little boy was walking in the park. He noticed a big, shiny object in the park.
Inv-First	Val.	they pinch. They find gold. They take pictures of stars.
Identity	Dir.	a little girl named Amy was playing outside. She saw a big tree and thought it was a toy.
Bert-like	Dir.	a little girl named Lucy was playing in the garden. She saw a shiny ring on a branch.
Inv-First	Dir.	hey went to the beach with his mom. He saw something shiny and strange inside.

Table 9: Example completion for the given prompt, in forward mode.

Acknowledgements

This work was supported by projects PNRR MUR PE0000013FAIR under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU and PRIN 2022 project 20227YET9B "AdVVent" CUP code B53D23012830006. It was also partially supported by Sapienza research projects "Prebunking", "Adagio", and "Risk and Resilience factors in disadvantaged young people: a multi-method study in ecological and virtual environments". Computing was supported by CINECA cluster under projects Ge-Di HP10CRPUVC, EHPC-DEV-2025D06-096 and the Sapienza Computer Science Department cluster.

Ethics and Impact Statement

The advancement of LLMs carries potential ethical implications, especially in the era of agents. Our investigation into this phenomenon contributes to a better understanding of how LLMs work and, thus, ultimately, to make them safer and more predictable. We believe that the publication of our research will promote a broader discussion on the responsible development of LLMs and contribute to the development of better defense mechanisms, as similar progress has already been made in the field of deep classifiers.

References

Alon, G.; and Kamfonas, M. 2023. Detecting Language Model Attacks with Perplexity. arXiv:2308.14132.

Bender, E. M.; Gebru, T.; McMillan-Major, A.; and Shmitchell, S. 2021. On the dangers of stochastic parrots: Can language models be too big? In *ACM conference on fairness, accountability, and transparency*.

Carlini, N.; Jagielski, M.; Choquette-Choo, C. A.; Paleka, D.; Pearce, W.; Anderson, H.; Terzis, A.; Thomas, K.; and Tramèr, F. 2024. Poisoning web-scale training datasets is practical. In *IEEE Symposium on Security and Privacy (SP)*.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Drucker, H.; and Le Cun, Y. 1992. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6): 991–997.

Ebrahimi, J.; Rao, A.; Lowd, D.; and Dou, D. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. arXiv:1712.06751.

Eldan, R.; and Li, Y. 2023. TinyStories: How Small Can Language Models Be and Still Speak Coherent English? arXiv:2305.07759.

Gage, P. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2): 23–38.

Ganz, R.; Kawar, B.; and Elad, M. 2023. Do Perceptually Aligned Gradients Imply Robustness? In *ICML*.

Goodfellow, I.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR*.

Inan, H.; Khosravi, K.; and Socher, R. 2017. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. In *ICLR*.

Keung, P.; Lu, Y.; Szarvas, G.; and Smith, N. A. 2020. The Multilingual Amazon Reviews Corpus. In *EMNLP*.

Li, X.; Li, Z.; Li, Q.; Lee, B.; Cui, J.; and Hu, X. 2024. Faster-GCG: Efficient Discrete Optimization Jailbreak Attacks against Aligned Large Language Models. arXiv:2410.15362.

Liu, X.; Li, P.; Suh, G. E.; Vorobeychik, Y.; Mao, Z.; Jha, S.; McDaniel, P.; Sun, H.; Li, B.; and Xiao, C. 2025. AutoDAN-Turbo: A Lifelong Agent for Strategy Self-Exploration to Jailbreak LLMs. In *The Thirteenth International Conference on Learning Representations*.

Liu, X.; Xu, N.; Chen, M.; and Xiao, C. 2024. AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models. arXiv:2310.04451.

Melamed, R.; McCabe, L.; Wakhare, T.; Kim, Y.; Huang, H. H.; and Boix-Adserà, E. 2024. Prompts have evil twins. In *EMNLP*.

Mirza, M. H.; Briglia, M. R.; Beadini, S.; and Masi, I. 2024. Shedding More Light on Robust Classifiers under the lens of Energy-based Models. In *ECCV*.

Morris, J. X.; Zhao, W.; Chiu, J. T.; Shmatikov, V.; and Rush, A. M. 2023. Language Model Inversion. arXiv:2311.13647.

Paulus, A.; Zharmagambetov, A.; Guo, C.; Amos, B.; and Tian, Y. 2024. AdvPrompter: Fast Adaptive Adversarial Prompting for LLMs. arXiv:2404.16873.

Press, O.; and Wolf, L. 2017. Using the Output Embedding to Improve Language Models. In *ACL*.

Rakotonirina, N. C.; Kervadec, C.; Franzon, F.; and Baroni, M. 2024. Evil twins are not that evil: Qualitative insights into machine-generated prompts. *arXiv preprint arXiv:2412.08127*.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019a. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *NeurIPS*.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019b. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *NeurIPS*, 30.

Xhonneux, S.; Sordoni, A.; Günnemann, S.; Gidel, G.; and Schwinn, L. 2024. Efficient Adversarial Training in LLMs with Continuous Attacks. In *NeurIPS*.

Zhao, Y.; Zheng, W.; Cai, T.; Do, X. L.; Kawaguchi, K.; Goyal, A.; and Shieh, M. 2024. Accelerating Greedy Coordinate Gradient and General Prompt Optimization via Probe Sampling. arXiv:2403.01251.

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*.