Multi-task Adversarial Attacks against Black-box Model with Few-shot Queries

Wenqiang Wang[1], **Yan Xiao**[1], **Hao Lin**[1], **Yangshijie Zhang**[2], **Xiaochun Cao**[1,2*],
[1]Shenzhen Campus of Sun Yat-sen University, [2] Peng Cheng Laboratory [3]Lanzhou University
wangwq69@mail2.sysu.edu.cn, xiaoy367@mail.sysu.edu.cn,
linh255@mail2.sysu.edu.cn, zhangyshj2023@lzu.edu.cn,
caoxiaochun@mail.sysu.edu.cn

Abstract

Current multi-task adversarial text attacks rely on abundant access to shared internal features and numerous queries, often limited to a single task type. As a result, these attacks are less effective against practical scenarios involving black-box feedback APIs, limited queries, or multiple task types. To bridge this gap, we propose **C**luster and **E**nsemble **M**ulti-task Text Adversarial **A**ttack (**CEMA**), an effective black-box attack that exploits the transferability of adversarial texts across different tasks. CEMA simplifies complex multi-task scenarios by using a *deep-level substitute model* trained in a *plug-and-play* manner for text classification, enabling attacks without mimicking the victim model. This approach requires only a few queries for training, converting multi-task attacks into classification attacks and allowing attacks across various tasks. CEMA generates multiple adversarial candidates using different text classification methods and selects the one that most effectively attacks substitute models. In experiments involving multi-task models with two, three, or six tasks—spanning classification, translation, summarization, and text-to-image generation—CEMA demonstrates significant attack success with as few as 100 queries. Furthermore, CEMA can target commercial APIs (e.g., Baidu and Google Translate), large language models (e.g., ChatGPT 4o), and image-generation models (e.g., Stable Diffusion V2), showcasing its versatility and effectiveness in real-world applications.

1 Introduction

A multi-task textual adversarial attack targets multiple tasks with minimal perturbations, posing significant risks to safety-critical systems by inducing erroneous decisions (Liu et al., 2017; Lin et al., 2022; Watkins et al., 2024). Most research on multi-task adversarial examples focuses on tasks of the same type, particularly classification tasks (Liu et al., 2017). Additionally, existing adversarial attack methods generally assume that attackers have access to the model architecture and shared layer information within a unified model, requiring abundant queries to the victim model (Guo et al., 2020; Zhe et al., 2024).

However, these studies do not align well with current real-world scenarios. For instance, consider the prompt "Translate this paper into Chinese and extract its core meaning:" input into ChatGPT-4o. This creates a multi-task model with distinct tasks—translation and summarization—where the AI agent (e.g., ChatGPT-4o) is a black-box victim model that outputs only the final text. Frequent access to large language models (LLMs) during the attack process incurs significant costs and is often limited, as seen with the ChatGPT-4o model's message limit of 50 per week (Hayawi and Shahriar, 2024). Given these constraints, it is essential to consider multi-task adversarial attacks in real-world scenarios, where the model feedback (i.e., the output text) reflects the black-box nature of the victim model. Furthermore, query limitations are a critical factor, as they optimize resource efficiency and minimize the risk of detection, making them a key consideration in practical attacks. Therefore, this paper is guided by the following challenge:

How can attackers use ***few-shot*** *queries to generate adversarial examples in a* ***black-box*** *multi-task learning model with* ***diverse task types****?*

In black-box and few-shot scenarios, a straightforward strategy is the transfer attack, where adversarial examples are generated in a substitute model without directly querying the victim model. However, this approach requires the availability of a well-trained substitute model. In the absence of such a model, attackers need to train a substitute model similar to the victim model. The transferability of adversarial examples depends on the similarity between these models, allowing adversarial

*Corresponding author

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13994–14014
July 27 - August 1, 2025

examples crafted for the substitute to also affect the victim model. Training a substitute model typically requires significant high-quality data, either from the victim model's training set ($\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}$) or by querying the victim model to generate predicting labels $\hat{\mathbf{Y}}_{\text{train}}$ for $\mathbf{X}_{\text{train}}$. However, acquiring such abundant data is often impractical, especially with query limitations in real-world settings.

Therefore, we propose the **deep-level attack hypothesis**: adversarial examples generated from a substitute model trained with deep-level labels can transfer effectively across multiple downstream tasks. For instance, in a multi-task model for bird-vs-cat images, tasks like mammal classification, flying ability, and leg count can be tackled using deep-level labels like species classification, enabling adversarial examples to transfer across these tasks. Rather than training a substitute model to match the victim model, we focus on building a model with deep-level labels that can target all tasks with minimal data and few-shot queries, bypassing the need for large auxiliary datasets or extensive victim model queries.

Specifically, we propose **C**luster and **E**nsemble **M**ulti-task Text **A**dversarial Attack (CEMA), a framework that leverages a small set of easily accessible auxiliary data, such as victim texts. To obtain the deep-level labels, CEMA first vectorizes the input texts and their corresponding outputs into vector representations, and then concatenates the input and output vectors to form final representations. A binary clustering method is then applied to these representations to generate binary cluster labels, which serve as the deep-level labels. Subsequently, CEMA uses the auxiliary data-cluster label pairs to train a binary classification model as the substitute model, transforming the multi-task attack scenario into a single-task text classification attack. During the adversarial example generation phase, adversarial candidates are produced for each victim text. Simultaneously, CEMA randomly samples the auxiliary data-cluster label pairs and repeats the training process to generate multiple substitute models. The adversarial example that successfully attacks the majority of these substitute models is then selected as the final adversarial example.

To evaluate the performance of CEMA, we focus on the text classification and translation tasks within a multi-task learning framework. Given the constraint of requiring only 100 queries, CEMA achieves a minimum improvement of 14.92% in attack success rate (ASR) over the second-best method in the classification task. In the text translation task, CEMA improves the BLEU score by at least 0.05 compared to the second-best method. With only 10 queries, CEMA still performs well, achieving an average ASR of 43.02% and an average BLEU score of 0.22. On LLMs such as ChatGPT-4o (Wang et al., 2025) and Claude 3.5 (Bae et al., 2024), CEMA achieves an average ASR of 36.15% and an average BLEU score of 0.43. When the number of tasks is increased to six, CEMA also achieves great attack results. Meanwhile, CEMA can also attack the summarization, and text-to-image generation tasks. The primary **contributions** are summarized as follows:

- We define the deep-level attack hypothesis, demonstrating that adversarial examples generated from a substitute model trained with deep-level labels can effectively transfer across multiple downstream tasks.
- We introduce CEMA, the first *plug-and-play framework* that converts a multi-task attack into a classification attack. CEMA can craft adversarial examples with very limited queries and just final text outputs (Black-box attack).
- We empirically validate the effectiveness of CEMA across various victim models and datasets, consistently achieving state-of-the-art performance.

2 Related Work

2.1 Text Classification Adversarial Attack

In text adversarial research, the predominant methods revolve around scenarios with singular output results (Waghela et al., 2024; Han et al., 2024; Zhu et al., 2024; Kang et al., 2024). These studies focus on the techniques for morphing the original text into adversarial counterparts, including the manipulation of pivotal chars (Ebrahimi et al., 2018b; Gil et al., 2019; Ebrahimi et al., 2018a; Gao et al., 2018; Ren et al., 2019; Jin et al., 2020; Li et al., 2019), words (Wang et al., 2022; Guo et al., 2021; Meng and Wattenhofer, 2020; Sato et al., 2018; Cheng et al., 2019; Lee et al., 2022; Li et al., 2020a; Hu et al., 2024; Liu et al., 2024, 2023; Li et al., 2019) and sentence. These methods are segmented into three distinct categories based on the response from the victim model, encompassing white-box attacks, soft-label black-box attacks, and hard-label black-box attacks. In white-box attacks, adversaries gain full access to all relevant information about the victim model. Hotflip (Ebrahimi et al., 2018b) sequentially replaces key words based on their importance scores, while the FD method (Papernot et al., 2016) generates adversarial examples using model gradient information. In soft-label black-box attacks, many methods perturb words based on output probabilities (Lee et al., 2022; Maheshwary et al., 2021b; Wang et al., 2021a; Li et al., 2020a). Bert-ATTACK (Li et al., 2020a) performs word-level attacks using a fine-tuned BERT; SememePSO (Zang et al., 2020) improves the search space for generating adversarial samples; Bae (Garg and Ramakrishnan, 2020) replaces words via BERT; and DeepWordBug (DWB) (Gao et al., 2018) targets words prioritized by output probabilities. Hard-label attacks reflect more realistic conditions. HLGA (Maheshwary et al., 2021a) uses stochastic initial words and a genetic algorithm to craft adversarial examples. HQA-attack (Liu et al., 2024) restores original words as much as possible to reduce disruption, then uses synonyms for the remaining changed words to enhance the attack.

2.2 Neural Machine Translation Adversarial Attack

Neural Machine Translation (NMT) models, which automatically convert input sentences into translated output, have achieved remarkable results by employing deep neural networks like Transformers (Bahdanau, 2014; Vaswani, 2017). These models are now extensively used across various applications due to their high performance. However, erroneous outputs generated by NMT models can lead to significant risks, particularly in security-sensitive contexts. Recent studies explore adversarial attacks on NMT models to address robustness concerns. Character-level NMT models are especially vulnerable to character-level manipulations like typos or word insertion/removal in black-box settings (Belinkov and Bisk, 2017; Ebrahimi et al., 2018a). However, such manipulations are easily detectable by humans or review systems. Therefore, most attacks on NLP and NMT systems focus on word replacement. Seq2sick (Cheng et al., 2020) uses a projected gradient method with group lasso and gradient regularization to perform non-overlapping and targeted keyword attacks. Similarly, Transfool (Sadrizadeh et al., 2023) applies gradient projection with a new optimization formulation and linguistic constraints to create fluent, semantically-preserving attacks. Morphin (Tan et al., 2020) perturbs inflections in clean examples to generate plausible and semantically similar adversaries. The kNN attack (Michel et al., 2019) replaces words with their neighbors in the embedding space in a white-box, untargeted setting. RG (Zou et al., 2019) adopts reinforcement learning to generate adversarial examples that preserve semantics and produce more reasonable tokens.

2.3 Mutil-task Adversarial Attack

A Multi-task Adversarial Attack is an adversarial machine learning strategy designed to generate examples that deceive multiple models or systems simultaneously (Guo et al., 2020; Ghamizi et al., 2022), rather than just one. As far as we know, there is currently no related work on multi-task adversarial attacks in the field of text. In other fields, MTA (Guo et al., 2020) is designed to generate adversarial perturbations for all three pre-trained classifiers simultaneously by leveraging shared knowledge among tasks. There is an attack method (Sobh et al., 2021) that targets visual perception in autonomous driving, which is applied in a wide variety of multi-task visual perception deep networks in distance estimation, semantic segmentation, motion detection, and object detection. MTADV (Wang et al., 2024) is a multitask adversarial attack against facial authentication, which is effective against various facial data sets.

2.4 Transfer Attack

Transfer attacks leverage adversarial examples to target different models without requiring direct access, posing a significant security threat in black-box scenarios (Papernot et al., 2017; Dong et al., 2018). Then, in the absence of a substitute model, several studies demonstrate that auxiliary data can also facilitate successful attacks through training a substitute model and leveraging transfer attacks (Li et al., 2020c; Sun et al., 2022). Additionally, more effective loss functions have been proposed to train substitute models (Wang et al., 2021b; Li et al., 2020b; Naseer et al., 2019; Richards et al., 2021; Huan et al., 2020), as well as techniques to refine substitute models (Xiaosen et al., 2023; Yuan et al., 2021).

3 Preliminary

Transferability and multi-task learning. Transferability refers to the ability of adversarial examples, crafted for one model, to successfully attack other models (Zhang et al., 2020; Papernot et al.,

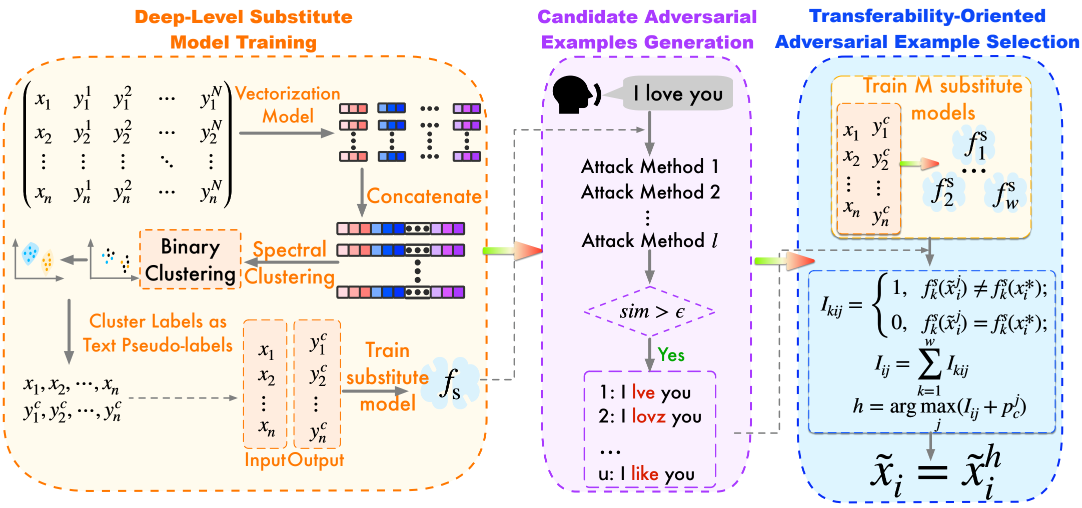

Figure 1: **The Overview of CEMA.** ❶ During deep-level substitute model training, CEMA vectorizes the auxiliary data and labels using pre-trained models, concatenates the vectors, and applies binary clustering to generate deep-level labels, which are used to train the substitute model f_s. ❷ For candidate adversarial example generation, l text classification attack methods are used to create l examples. Examples exceeding a similarity threshold ϵ are selected as candidates, resulting in u candidates, where $u < l$. ❸ CEMA trains w substitute models using 80% of the data pairs from the auxiliary text-cluster labels. The adversarial example attacking the most substitute models is chosen as the final adversarial example $\tilde{x}_i$.

2017; Mahmood et al., 2021; Zhou et al., 2020). Multi-task learning trains multiple related tasks together, enabling knowledge sharing and improving generalization (Cai et al., 2025; Pu et al., 2025).

Auxiliary data. Auxiliary data refers to additional information used to assist in generating adversarial examples. This data, which is not part of the original input, helps attackers better understand the target model's behavior or generate more effective adversarial inputs. For instance, auxiliary data might include outputs from the victim model (Chen et al., 2024; Papernot et al., 2017).

4 Method

4.1 Observations and Motivations

We consider a multi-task scenario with bird and cat images. Task 1 identifies if the image represents a mammal, Task 2 determines if the subject can fly, and Task 3 assesses the number of legs. Adversarial attacks on this multi-task model under black-box conditions with limited queries are challenging. However, by training a substitute model for bird-cat classification and perturbing it to misclassify a cat as a non-cat, we can modify the outputs of the three downstream tasks. Birds and cats share intrinsic species characteristics that are more fundamental and "deep-level" than the tasks of identifying mammalian status, flight capability, or legs' number. Based on this observation, we propose the **deep-level attack hypothesis**:

Assumption 4.1 (Deep-level Attack Hypothesis). Adversarial examples $\tilde{x}$ of the victim text x^* crafted using a substitute model f_s trained with deep-level labels can effectively attack multiple downstream tasks in a multi-task model f_v. Formally,

$$f_s(x^*) \neq f_s(\tilde{x}), \quad \text{and} \quad f_v(x^*) \neq f_v(\tilde{x}), \quad (1)$$

Based on the observations and hypothesis, we propose the **C**luster and **E**nsemble **M**ulti-task Text **A**dversarial Attack (CEMA) method, an overview of which is provided in Figure 1. CEMA uses a deep-level substitute model to convert multi-task attack scenarios—characterized by varying numbers and types of tasks—into a single-task text classification attack scenario. Furthermore, CEMA employs multiple text classification attack methods to generate candidate adversarial examples and introduces a selection mechanism to identify the final adversarial example from the pool of candidates.

4.2 Deep-level Substitute Model Training

For auxiliary texts in auxiliary data, we query the victim model to obtain their corresponding outputs. These texts, along with their task-specific outputs, are then vectorized using a method such as a pre-trained model. The resulting vector representations are concatenated to form a unified representation. A binary clustering algorithm is applied to these vectorized representations, and cluster labels are assigned to each, which are treated as **deep-level labels** for both the text and its outputs. Next, we train a substitute model, denoted as f_s, using the auxiliary texts and their corresponding cluster labels. This substitute model f_s enables the transformation of the black-box multi-task attack scenario into a white-box text classification attack scenario. Moreover, f_s is treated as the substitute model trained with deep-level labels (cluster labels).

4.2.1 Representation Learning

As discussed in Section 4.1, the deep-level label is derived from both the image and the outputs of three tasks: identifying whether the image represents a mammal, whether the subject can fly, and the number of legs. Thus, the deep-level label should simultaneously consider the impact of input data and the output data of the multi-task model. To achieve this, we vectorize the auxiliary texts and their outputs using a pre-trained model and concatenate them into a unified representation.

For each auxiliary text x_i, we query the victim model f_v to obtain the corresponding outputs $\{y_i^1, y_i^2, \ldots, y_i^N\}$, where N denotes the number of tasks. The pre-trained model f_e then vectorizes both x_i and its outputs, yielding vectors $\{\boldsymbol{E}_{x_i}, \boldsymbol{E}_{y_i^1}, \ldots, \boldsymbol{E}_{y_i^N}\}$. These vectors are concatenated to form the final vector $\boldsymbol{E}_i$, which represents x_i and its corresponding outputs. Formally:

$$\begin{aligned} &\{y_i^1, y_i^2, \ldots, y_i^N\} = f_\text{v}(x_i), \\ &\boldsymbol{E}_{x_i} = f_e(x_i), \quad \boldsymbol{E}_{y_i^J} = f_e(y_i^J), \\ &\boldsymbol{E}_i = \text{Concat}(\boldsymbol{E}_{x_i}, \boldsymbol{E}_{y_i^1}, \ldots, \boldsymbol{E}_{y_i^N}), \end{aligned} \tag{2}$$

where the $Concat$ indicates the concatenation of the vectors $\{\boldsymbol{E}_{x_i}, \boldsymbol{E}_{y_i^1}, \ldots, \boldsymbol{E}_{y_i^N}\}$.

4.2.2 Cluster Number and Substitute Model

We set the number of clusters to 2, which can be interpreted as class C and its complement $\overline{C}$. This choice ensures that the two clusters capture the most fundamental deep-level label, as the labels in three- or four-cluster configurations would ultimately merge into two. Experimental results in Section 5.3 show that CEMA achieves optimal attack performance with two clusters. Attackers then employ the binary cluster method to obtain the cluster label y_i^c for auxiliary text x_i. Finally, the binary substitute model f_s is trained using pairs of auxiliary text and cluster labels. Formally:

$$\begin{aligned} &\mathbf{D} = \{(x_1, y_1^c), (x_2, y_2^c), \ldots, (x_n, y_n^c)\}, \\ &\theta^* = \arg\min_\theta \frac{1}{|\mathbf{D}|} \sum_{(x_i, y_i^c) \in \mathbf{D}} (y_i^c - f_\text{s}(\theta; x_i))^2, \end{aligned} \tag{3}$$

where y_i^c denotes the cluster label of E_i, and θ^* represents the model parameters of the substitute model f_s.

4.3 Candidate Adversarial Examples Generation

Once the substitute model f_s is trained, adversarial examples can be generated from it using any standard text classification method to attack the victim model f_v through transferability. The perturbed text $\tilde{x}_i$ of the victim texts x_i^* is generated by:

$$\begin{aligned} &\tilde{x}_i = \arg\max_{x_i} L(f_\text{s}(\tilde{x}_i), f_\text{s}(x_i^*)), \\ &\text{and} \quad f_\text{v}(x_i^*) \neq f_\text{v}(\tilde{x}_i), \quad \text{s.t.} \quad sim(\tilde{x}_i, x_i^*) \geq \epsilon. \end{aligned} \tag{4}$$

where $L(f(\tilde{x}_i), f(x_i))$ is the loss function, $f_\text{s}(\tilde{x}_i)$ represents the predictions of the substitute model for $\tilde{x}_i$, and $f_\text{s}(x_i^*)$ represents the predictions for x_i^*. The term sim denotes the cosine similarity, which is calculated as $\frac{f_\text{e}(x_i^*) \cdot f_\text{e}(\tilde{x}_i)}{\|f_\text{e}(x_i^*)\| \cdot \|f_\text{e}(\tilde{x}_i)\|}$. Finally, ϵ is the similarity threshold. *The condition $sim(\tilde{x}_i, x_i^*) \geq \epsilon$ ensures that the adversarial example $\tilde{x}_i$ maintains high similarity with x_i^* and retains its true label.*

Instead of relying on one attack method, we use multiple attack methods to generate diverse examples and establish criteria to select the most effective one. This strategy is motivated by two main reasons: (1) More adversarial examples increase the probability of at least one successfully attacking the victim model. (2) Diverse methods enhance the probability of surpassing the similarity threshold ϵ for multiple examples. We apply l attack methods to generate l examples. The examples whose similarity exceeds the similarity threshold ϵ are identified as candidate adversarial examples ($\{\tilde{x}_i^1, \tilde{x}_i^2, \ldots, \tilde{x}_i^u\}$) for the victim text x_i^*, with $u \leq l$. Denote the p_i^j as the probability that $\tilde{x}_i^j$ can successfully attack the victim model. A^u denotes the event that at least one of the u attack methods attack the victim model, with the probability termed as $P(A^u)$. Denote p_{ij}^{sim} as the probability that the similarity of $\tilde{x}_i^j$ and x_i^* exceed the similarity threshold ϵ. B^u denotes the event that at least one of the u adversarial examples exceed the similarity threshold ϵ, with the probability termed as $P(B^u)$. Reason (1) and (2) are mathematically expressed in Theorem 4.2.

Theorem 4.2. *$P(A^u)$ and $P(B^u)$ increase with u increases . Formally:*

$$P(A^u) \leq P(A^{u+1}), P(B^u) \leq P(B^{u+1}) \quad \textit{for all} \quad u \geq 1, \tag{5}$$

Proof. Please see Section H in Appendix.

4.4 Transferability-Oriented Adversarial Example Selection

After generating additional adversarial candidates in Section 4.3, we select the one with the highest transferability. We achieve this by retraining multiple substitute models and evaluating the transferability of each candidate adversarial example based on the number of models it successfully attacks. The adversarial candidate that successfully attacks the greatest number of substitute models is selected as the final adversarial example. The detailed steps are as follows: ❶ **Training Substitute Models:** We randomly sample 80% of the auxiliary text-cluster label pairs to form the training set for each substitute model. This process is repeated w times, resulting in w substitute models $\{f_1^{\mathrm{s}}, f_2^{\mathrm{s}}, \ldots, f_w^{\mathrm{s}}\}$. ❷ **Calculating Transferability:** For each victim text x_i^* and its adversarial candidates $\{\tilde{x}_i^1, \tilde{x}_i^2, \ldots, \tilde{x}_i^u\}$. The transferability score for $\tilde{x}_i^j$ is computed as follows:

$$I_{kij} = \begin{cases} 1, & \text{if } f_k^{\mathrm{s}}(\tilde{x}_i^j) \neq f_k^{\mathrm{s}}(x_i^*), \\ 0, & \text{if } f_k^{\mathrm{s}}(\tilde{x}_i^j) = f_k^{\mathrm{s}}(x_i^*), \end{cases} \quad I_{ij} = \sum_{k=1}^{w} I_{kij}. \tag{6}$$

where $f_k^{\mathrm{s}}\left(\tilde{x}_i^j\right)$ represents the output label of $\tilde{x}_i^j$ produced by the substitute model f_k^{s}, and $f_k^{\mathrm{s}}(x_i^*)$ is the output label of x_i^* from the f_k^{s}. If $f_k^{\mathrm{s}}\left(\tilde{x}_i^j\right) \neq f_k^{\mathrm{s}}(x_i^*)$, then $\tilde{x}_i^j$ successfully attacks f_k^{s}. Thus, I_{ij} counts the number of substitute models that $\tilde{x}_i^j$ successfully attacks. The adversarial example attacking the most substitute models is selected as the final adversarial example of the victim text x_i^*. If multiple candidate adversarial examples successfully attack the most substitute models, the candidate with the largest change in output probability from the substitute model f_{s} before and after the attack, is selected as the final adversarial example. Formally, this is expressed as:

$$p_c^j = p_{\hat{y}}(x_i^*) - p_{\hat{y}}(\tilde{\mathbf{x}}_i^j), h = \arg\max_j \,(I_{ij} + p_c^j), \tilde{x}_i = \tilde{x}_i^h, \tag{7}$$

$\hat{y}$ denotes the predicted label of x_i^* under f_{s}, with $p_{\hat{y}}(x_i^*)$ and $p_{\hat{y}}(\tilde{\mathbf{x}}_i^j)$ being the probabilities that x_i^* and $\tilde{\mathbf{x}}_i^j$ are predicted as $\hat{y}$, respectively. Since p_c^j represents probability variation, and the generated adversarial examples succeed in attacking f_{s}, p_c^j lies in [0,1). When multiple adversarial examples attack the substitute model, $h = \arg\max_j \,(I_{ij} + p_c^j)$ accounts for the probability variation in f_{s}, with $\tilde{x}_i$ being the final adversarial example.

5 Experiment

5.1 Experiment Setup

Dataset: We use the **SST5** and **Emotion** datasets. Detailed statistics are in Appendix B, Table 4.

Victim models and baselines in LLMs: We conduct attacks on advanced LLMs, ChatGPT-4o (Pang et al., 2024) and Claude 3.5 (Bae et al., 2024). We use prompt learning to build agents on LLMs as victim models. The prompt is: "Translate the following text into French and Chinese, and predict the category label of the text, which can be one of the following: Sadness, Joy, Love, Anger, Fear, Surprise.". As no existing research examines attacks on multi-task learning outputs of large models in a black-box setting, we adopt the approach of randomly removing the last character from 40% of the words in a sentence as a baseline for comparison, which is recorded as Random-Del.

Victim models and baselines in multi-model multi-task learning (M3TL): M3TL trains multiple models to simultaneously solve various tasks (Pineda et al., 2022; Xiang, 2024; Jin et al., 2021). Inspired by M3TL, we combine several pre-trained models to simulate multi-task learning. For classification, we employ dis-sst5, ro-sst5, dis-emotion, and ro-emotion models. For translation, we utilize the Opus-MT model for English-to-Chinese (En-Zh) task and the T5-small model for English-to-French (En-Fr) task. The model URLs are provided in Table 5 in the Appendix. To simulate a real-world attack scenario, we incorporate two commercial translation APIs: Baidu Translate (En-Fr) (He, 2015) and Ali Translate (En-Zh) (Wan et al., 2022). Three multi-task victim models are created: **Victim Model A** (dis-sst5, dis-emo, opus-mt), **Victim Model B** (ro-sst5, ro-emo, t5-small), and **Victim Model C** (Baidu and Ali Translate). Previous multi-task attack methods (Liu et al., 2017) require extensive queries and focus on white-box models with similar tasks, making them ineffective against multi-task black-box

Table 1: The attack performance of LLMs.

Data	Models	ChatGPT-4o			Claude 3.5			Total-Qry↓
	Tasks	Classification	En-Fr	En-Zh	Classification	En-Fr	En-Zh	
	Metrics	ASR(%)↑	BLEU↓	BLEU↓	ASR(%)↑	BLEU↓	BLEU↓	
Emotion	Random-Del	14.20	0.47	0.64	17.65	0.44	0.52	0
	CEMA	**32.05**	**0.39**	**0.33**	**36.80**	**0.38**	**0.35**	100
SST5	Random-Del	15.33	0.64	0.66	16.27	0.62	0.54	0
	CEMA	**38.63**	**0.32**	**0.27**	**37.12**	**0.29**	**0.25**	100

models. In comparison, CEMA targets both classification and translation tasks with fewer queries. For single-task attacks, we compare CEMA with methods for text classification (BAE (Garg and Ramakrishnan, 2020), FD (Papernot et al., 2016), Hotflip (Ebrahimi et al., 2018b), SememePSO (Zang et al., 2020), TextBugger (Ren et al., 2019)) and translation (Hot-trans (Ebrahimi et al., 2018b), kNN (Michel et al., 2019), Morphin (Tan et al., 2020), RA (Zou et al., 2019), Seq2Sick (Cheng et al., 2020), TransFool (Sadrizadeh et al., 2023)). All methods are limited to 55 queries per victim text to ensure a fair comparison. Details are provided in Tables 7 and 8 in Appendix D.

Metrics, substitute model, clustering and vectorization methods: We evaluate the following metrics: ❶ ASR: A higher ASR indicates better attack efficacy. ❷ Total Queries (Total-Qry): Fewer queries reflect a more efficient attack. ❸ BLEU: A lower BLEU score indicates a more effective attack result (Lateef et al., 2024). ❹ Similarity (Sim.): A higher Sim. score indicates a more effective attack result. Details of the substitute model are provided in Section A of the Appendix. We train six substitute models on a server equipped with a 24GB NVIDIA 3090 GPU, with a training time of approximately 4 minutes per model due to limited auxiliary data. Each model has a size of 418 MB. Spectral clustering (Ng et al., 2001) is used for clustering, and mT5 (Xue, 2020) is used for vectorization.

Applied attack methods, auxiliary data, and hyperparameter: We apply a combination of attack methods, including Hotflip (Ebrahimi et al., 2018b), FD (Papernot et al., 2016), and TextBugger (Ren et al., 2019). These methods assume that the attacker has access to the victim texts during the attack, indicating that the victim texts are readily available for modification. For auxiliary data, we select 100 unlabeled victim texts. The similarity threshold ϵ is set to 0.8.

5.2 Main Attack Results of LLMs and M3TL

As shown in Table 1, CEMA achieves substantial attack effectiveness on LLMs, with a best ASR of 38.63% and a best BLEU score of 0.25. Tables 2 and 3 highlight that CEMA achieves state-of-the-art (SOTA) performance on the SST5 and Emotion datasets across victim models A, B, and C. In a black-box setting with 100 queries, CEMA attains an ASR exceeding 59%, reaching up to 80.80% in classification tasks. For translation tasks, CEMA achieves a BLEU score below 0.16, outperforming the second-best method. Against victim model C (Baidu and Ali Translate) using only 100 auxiliary texts, CEMA exceeds Morphin and TransFool, attaining BLEU scores under 0.35 with just 100 queries.

5.3 Ablation Study

To investigate factors influencing CEMA attack performance, we conduct an ablation study on the number of clusters, attack methods, clustering, and vectorization. We experiment with two to four clusters and increase attack methods from one to three, using TextBugger alone and a combination of Hotflip, FD, and TextBugger. For clustering, we use spectral clustering as the main method, along with K-means (Krishna and Murty, 1999) and BIRCH (Zhang et al., 1996). For vectorization, we primarily use mT5 (Xue, 2020), with XLM-R (Conneau, 2019) and one-hot encoding (Rodríguez et al., 2018) as alternatives. One-hot encoding transforms categorical data into binary vectors without requiring access to any external data, preventing data leakage in pre-trained process. Given the high computational cost of utilizing large models, we primarily conduct ablation experiments on victim models A, B, and C.

Cluster number and attack method number. As shown in Figure 2, increasing the number of clusters from 2 to 4 reduces attack performance. The average ASR drops from 58.83% and 64.55% to 46.20% and 52.10%, while BLEU scores increase

Table 2: The attack performance of CEMA. Text classification tasks use the ASR(%)↑ metric, while text translation tasks use the BLEU↓ metric. Other adversarial attack methods can only be applied to their specific tasks, whereas CEMA simultaneously attacks all tasks.

Dataset	SST5						Emotion					
Victim Model	Victim Model A			Victim Model B			Victim Model A			Victim Model B		
Text Classification	dis-sst5 (A)			ro-sst5 (B)			dis-emotion (A)			ro-emotion (B)		
Metric	ASR(%)↑	Sim.↑	Total-Qry↓	ASR(%)↑	Sim.↑	Total-Qry↓	ASR(%)↑	Sim.↑	Total-Qry↓	ASR(%)↑	Sim.↑	Total-Qry↓
Bae	42.71	0.888	47360	39.14	0.887	47471	31.55	0.925	59626	28.50	0.924	55935
FD	25.20	0.939	27758	22.30	**0.982**	21459	47.10	0.948	66035	20.75	0.979	26719
Hotflip	41.50	0.951	25459	29.03	0.951	25945	46.85	0.942	21658	41.65	0.952	22409
PSO	45.14	0.954	24398	41.50	0.954	27360	46.05	0.945	19713	44.95	0.964	19757
TextBugger	30.36	**0.978**	69527	20.85	0.978	67007	35.10	**0.981**	25216	29.40	**0.981**	25128
Leap	32.55	0.953	21548	30.07	0.944	21083	26.30	0.934	15492	15.50	0.939	15315
CT-GAT	29.37	0.939	46233	24.80	0.926	82963	25.90	0.916	47338	26.75	0.927	47139
HQA	46.11	0.936	64864	39.64	0.929	64267	37.35	0.934	65725	35.85	0.925	47449
CEMA	**73.57**	0.934	**100**	**75.66**	0.927	**100**	**80.80**	0.926	**100**	**60.40**	0.931	**100**
Text Classification	dis-emotion (A)			ro-emotion (B)			dis-emotion (A)			ro-emotion (B)		
Metric	ASR(%)↑	Sim.↑	Total-Qry↓	ASR(%)↑	Sim.↑	Total-Qry↓	ASR(%)↑	Sim.↑	Total-Qry↓	ASR(%)↑	Sim.↑	Total-Qry↓
Bae	39.81	0.894	60399	14.65	0.896	62013	32.25	0.926	48266	32.95	0.923	48244
FD	35.43	0.921	64576	9.55	0.934	36553	22.30	0.932	28310	17.50	**0.982**	40730
Hotflip	33.39	0.943	24001	22.80	0.946	27139	29.00	0.949	31559	28.05	0.949	31824
PSO	41.90	0.968	19934	35.25	0.940	20885	39.50	0.952	26144	37.65	0.951	26741
TextBugger	30.00	**0.972**	25084	40.95	0.986	25084	20.85	**0.978**	67007	21.45	0.978	67029
Leap	21.00	0.968	15315	26.00	0.947	15492	40.55	0.926	21503	37.65	0.911	21614
CT-GAT	39.32	0.927	47206	33.45	0.924	47493	28.10	0.904	57593	30.85	0.906	56001
HQA	37.76	0.945	47382	31.95	0.931	65062	37.40	0.912	49592	36.40	0.911	51184
CEMA	**62.27**	0.934	**100**	**64.01**	0.927	**100**	**65.40**	0.926	**100**	**59.60**	0.931	**100**
Text Translation	opus-mt(En-Zh) (A)			t5-small(En-Fr) (B)			opus-mt(En-Zh) (A)			t5-small(En-Fr) (B)		
Metric	BLEU↓	Sim.↑	Total-Qry↓	BLEU↓	Sim.↑	Total-Qry↓	BLEU↓	Sim.↑	Total-Qry↓	BLEU↓	Sim.↑	Total-Qry↓
Hot-trans	0.24	0.846	21570	0.24	0.842	20885	0.20	0.859	20686	0.19	0.854	21680
KNN	0.31	0.873	13680	0.31	0.883	13680	0.61	0.935	29481	0.28	0.906	13437
Morphin	0.30	0.894	15006	0.37	0.907	24531	0.27	0.869	11183	0.22	0.887	8486
RA	0.25	0.872	7028	0.19	0.865	9415	0.23	0.852	6166	0.21	0.865	4663
Seq2sick	0.38	0.881	9835	0.46	0.926	13371	0.62	0.945	15669	0.29	0.892	8951
TransFool	0.77	**0.949**	7337	0.44	0.894	8641	0.81	**0.962**	8597	0.67	0.924	7912
CEMA	**0.14**	0.934	**100**	**0.18**	**0.927**	**100**	**0.15**	0.926	**100**	**0.23**	**0.931**	**100**

Table 3: Attack performance of different methods on victim model C, which consists of two commercial closed-source translation APIs, Alibaba and Baidu Translate.

Dataset	Victim Model C	Baidu Translate (en-fr)(C)			Ali Translate (en-zh)(C)		
	Method	BLEU↓	Sim.↑	Total-Qry↓	BLEU↓	Sim.↑	Total-Qry↓
SST5	Morphin	0.54	0.904	89461	0.60	0.931	107075
	TransFool	0.51	0.921	52001	0.59	0.928	68952
	CEMA	**0.29**	**0.934**	**100**	**0.15**	**0.934**	**100**
Emotion	Morphin	0.40	0.897	55581	0.55	0.915	25473
	TransFool	0.36	0.903	25416	0.49	0.923	61820
	CEMA	**0.35**	**0.931**	**100**	**0.29**	**0.931**	**100**

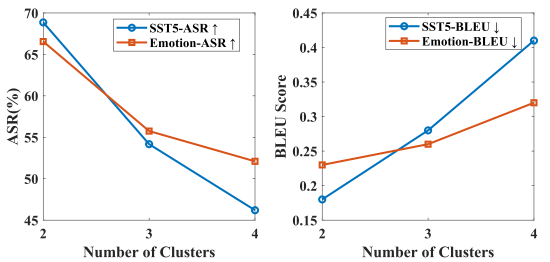

Figure 2: The average ASR and BLUE of different clusters' number. Fewer clusters result in the better attack results.

from 0.16 and 0.18 to 0.41 and 0.32. *The best attack performance is achieved with two clusters.* Additionally, as shown in Table 14 in the Appendix, reducing the number of attack methods lowers ASR and raises BLEU scores, as smaller adversarial spaces reduce attack effectiveness. Increasing the number of attack algorithms from one to three improves average ASR by 30.3% while decreasing BLEU by 0.16, indicating that more attack methods enhance performance.

Clustering and vectorization methods. As shown in Figure 3 , both clustering and vectorization methods have random effects on attack performance. The average ASR for Spectral, KMeans, and BIRCH are 67.7%, 65.2%, and 65.1%, respectively, while BLEU scores are approximately 0.21, with no method consistently outperforming the others. This indicates that the impact of clustering methods on attack results is random. Similarly, the

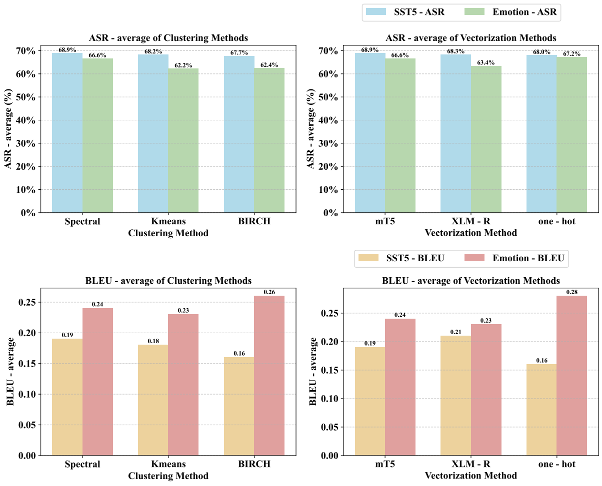

Figure 3: The average ASR and BLUE of CEMA under various clustering and vectorization methods.

choice of vectorization methods leads to slight fluctuations in both classification and translation, but none achieves SOTA performance. The average ASR for mT5, XLM-R, and one-hot encoding is around 67.7%, with BLEU scores ranging from 0.21 to 0.22, indicating the random impact of vectorization methods.

In summary, clustering and vectorization methods introduce randomness, and while increasing the number of attack methods improves effectiveness, it also prolongs attack time.

5.4 Scalability of CEMA for More Numbers and Types of Tasks

CEMA is extended to six downstream tasks, comprising four classification tasks and two translation tasks, as detailed in Table 12 in the Appendix. It achieves an ASR of over 60%, with BLEU scores below 0.3, demonstrating its effectiveness in six downstream tasks. To explore CEMA's scalability further, we apply it to a broader set of tasks using the Pokemon dataset (Pinkney, 2022) as victim data. The victim models used include Google Translate (Radford et al., 2021), distilbart-cnn (Sakhapara et al., 2022), and Stable Diffusion V2, with tasks such as translation (Kol et al., 2018), summarization, and text-to-image generation. The evaluation metrics include BLEU, ROUGE Drop Percentage (RDP) (Lin, 2004), and CLIP Drop Percentage (CDP) (Radford et al., 2021). As shown in Table 13 in the Appendix, CEMA exhibits strong performance across all tasks, suggesting its potential for extension. Specifically, the BLEU score is 0.29, with RDP and CDP scores of 47% and 56%, respectively. These results indicate that CEMA is well-suited for multi-task models involving a broader range of tasks and task types.

5.5 Few-shot and Many-shot Queries

We evaluate CEMA's attack performance under few-shot and many-shot queries, with query numbers set to 10, 50, 100, 1000, and 2000. The results, shown in Table 15 in the Appendix, reveal that attack effectiveness improves with more queries. Notably, CEMA achieves an ASR of over 30% and a BLEU score of 0.19 with only 10 queries.

5.6 Other Results

❶ we collect relevant internet data as auxiliary information for the victim texts, as described in Table 11 (Appendix) and Section K. CEMA achieves strong results, showing that attackers only need the victim texts' related information to successfully attack the multi-task system. ❷ **Attack results under defense methods** We explore defensive strategies against CEMA, including language modifiers and adversarial training. Detailed results are presented in Section E. While these defenses reduce CEMA's effectiveness, it still maintains a significant level of attack efficacy.

6 Limitation

CEMA requires training multiple models and generating several candidate adversarial examples, which demands additional time and storage space.

7 Conclusion

We present a multi-task learning attack scenario where attackers have limited access to a model, only being able to query black-box outputs. Therefore, we propose the CEMA method, which achieves SOTA performance with just 100 queries and black-box outputs. Additionally, CEMA can integrate any text classification attack algorithm, and its performance improves as more attack methods are included. We plan to extend CEMA to multi-task models in other domains in the future.

8 Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities, Sun Yat-sen University under Grants No. 23xkjc010, Shenzhen Science and Technology Program (No.KQTD20221101093559018), and National Natural Science Foundation of China (No. 62441619).

References

Jaehyeon Bae, Seoryeong Kwon, and Seunghwan Myeong. 2024. Enhancing software code vulnerability detection using gpt-4o and claude-3.5 sonnet: A study on prompt engineering techniques. *Electronics*, 13(13):2657.

Dzmitry Bahdanau. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Yujian Cai, Xingguang Li, Yingyu Zhang, Jinsong Li, Fazheng Zhu, and Lin Rao. 2025. Multimodal sentiment analysis based on multi-layer feature fusion and multi-task learning. *Scientific Reports*, 15(1):2126.

Li Chen, Shaowei Zhu, Abel Andrew, and Zhaoxia Yin. 2024. Reversible attack based on local visible adversarial perturbation. *Multimedia Tools and Applications*, 83(4):11215–11227.

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. 2020. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3601–3608.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. In *ACL*, pages 4324–4333.

A Conneau. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *CVPR*, pages 9185–9193.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018a. On adversarial examples for character-level neural machine translation. In *COLING*, pages 653–663.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018b. Hotflip: White-box adversarial examples for text classification. In *ACL*, pages 31–36.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *SPW*, pages 50–56.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. In *EMNLP*, pages 6174–6181.

Salah Ghamizi, Maxime Cordy, Mike Papadakis, and Yves Le Traon. 2022. Adversarial robustness in multi-task learning: Promises and illusions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):697–705.

Yotam Gil, Yoav Chai, Or Gorodissky, and Jonathan Berant. 2019. White-to-black: Efficient distillation of black-box adversarial attacks. In *NAACL-HLT*, pages 1373–1379.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based adversarial attacks against text transformers. In *EMNLP*, pages 5747–5757.

Pengxin Guo, Yuancheng Xu, Baijiong Lin, and Yu Zhang. 2020. Multi-task adversarial attack. *arXiv preprint arXiv:2011.09824*.

Xu Han, Qiang Li, Hongbo Cao, Lei Han, Bin Wang, Xuhua Bao, Yufei Han, and Wei Wang. 2024. Bfs2adv: Black-box adversarial attack towards hard-to-attack short texts. *CS*, page 103817.

Kadhim Hayawi and Sakib Shahriar. 2024. A cross-domain performance report of open ai chatgpt o1 model.

Zhongjun He. 2015. Baidu translate: Research and products. In *Proceedings of the Fourth Workshop on Hybrid Approaches to Translation (HyTra)*, pages 61–62.

Xiaoxue Hu, Geling Liu, Baolin Zheng, Lingchen Zhao, Qian Wang, Yufei Zhang, and Minxin Du. 2024. Fast-textdodger: Decision-based adversarial attack against black-box nlp models with extremely high efficiency. *TIFS*.

Zhaoxin Huan, Yulong Wang, Xiaolu Zhang, Lin Shang, Chilin Fu, and Jun Zhou. 2020. Data-free adversarial perturbations for practical black-box attack. In *PAKDD*, pages 127–138. Springer.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *AAAI*, pages 8018–8025.

Yue Jin, Tianqing Zheng, Chao Gao, and Guoqiang Xu. 2021. A multi-modal and multi-task learning method for action unit and expression recognition. *arXiv preprint arXiv:2107.04187*.

Yan Kang, Jianjun Zhao, Xuekun Yang, Baochen Fan, and Wentao Xie. 2024. A hybrid style transfer with whale optimization algorithm model for textual adversarial attack. *NCA*, 36:4263–4280.

Sara Kol, Miriam Schcolnik, and Elana Spector-Cohen. 2018. Google translate in academic writing courses? *The EuroCALL Review*, 26(2):50–57.

K Krishna and M Narasimha Murty. 1999. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439.

Huda Mohammed Lateef, Ahmad Muter Awaad, Diadeen Ali Hameed, Ghanim Thiab Hasa, and Tahseen Ameen Faisal. 2024. Evaluation of domain sulfur industry for dia translator using bilingual evaluation understudy method. *Bulletin of Electrical Engineering and Informatics*, 13(1):370–376.

Deokjae Lee, Seungyong Moon, Junhyeok Lee, and Hyun Oh Song. 2022. Query-efficient and scalable black-box adversarial attacks on discrete sequential data via bayesian optimization. In *ICML*, pages 12478–12497.

J Li, S Ji, T Du, B Li, and T Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020a. Bert-attack: Adversarial attack against bert using bert. *arXiv e-prints*, pages arXiv–2004.

Maosen Li, Cheng Deng, Tengjiao Li, Junchi Yan, Xinbo Gao, and Heng Huang. 2020b. Towards transferable targeted attack. In *CVPR*, pages 638–646. IEEE.

Qizhang Li, Yiwen Guo, and Hao Chen. 2020c. Practical no-box adversarial attacks against dnns. *NeurIPS*, 33:12849–12860.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Nankai Lin, Sihui Fu, Xiaotian Lin, and Lianxi Wang. 2022. Multi-label emotion classification based on adversarial multi-task learning. *Information Processing & Management*, 59(6):103097.

Han Liu, Zhi Xu, Xiaotong Zhang, Xiaoming Xu, Feng Zhang, Fenglong Ma, Hongyang Chen, Hong Yu, and Xianchao Zhang. 2023. Sspattack: a simple and sweet paradigm for black-box hard-label textual adversarial attack. In *AAAI*, volume 37, pages 13228–13235.

Han Liu, Zhi Xu, Xiaotong Zhang, Feng Zhang, Fenglong Ma, Hongyang Chen, Hong Yu, and Xianchao Zhang. 2024. Hqa-attack: Toward high quality black-box hard-label adversarial attack on text. *NeurIPS*, 36.

Pengfei Liu, Xipeng Qiu, and Xuan-Jing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10.

Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. 2021a. Generating natural language attacks in a hard label black box setting. In *AAAI*, pages 13525–13533.

Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. 2021b. A strong baseline for query efficient attacks in a black box setting. In *EMNLP*, pages 8396–8409.

Kaleel Mahmood, Deniz Gurevin, Marten van Dijk, and Phuoung Ha Nguyen. 2021. Beware the black-box: On the robustness of recent defenses to adversarial examples. *Entropy*, 23(10):1359.

Zhao Meng and Roger Wattenhofer. 2020. A geometry-inspired attack for generating natural language adversarial examples. In *ACL*, pages 6679–6689.

Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. *arXiv preprint arXiv:1903.06620*.

Muhammad Muzammal Naseer, Salman H Khan, Muhammad Haris Khan, Fahad Shahbaz Khan, and Fatih Porikli. 2019. Cross-domain transferability of adversarial perturbations. In *NeurIPS*, volume 32.

Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14.

Samarnh Pang, Engheang Nol, and Kimkong Heng. 2024. Chatgpt-4o for english language teaching and learning: Features, applications, and future prospects. *Cambodian Journal of Educational Research*, 4(1):35–56.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *CCS*, pages 506–519.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM*, pages 49–54.

Martin Pineda, Qianlong Wang, Weixian Liao, Michael McGuire, and Wei Yu. 2022. A multi-model multi-task learning system for hurricane genesis prediction. In *International Conference on Software Engineering Research and Applications*, pages 113–129.

Justin N.M. Pinkney. 2022. Pokemon blip captions.

Qiumei Pu, Fangli Huang, Fude Li, Jieyao Wei, and Shan Jiang. 2025. Integrating emotional features for stance detection aimed at social network security: A multi-task learning approach. *Electronics*, 14(1):186.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *ACL*, pages 1085–1097.

Luke E Richards, André Nguyen, Ryan Capps, Steven Forsyth, Cynthia Matuszek, and Edward Raff. 2021. Adversarial transfer attacks with unknown data and class overlap. In *ACM*, pages 13–24.

Pau Rodríguez, Miguel A Bautista, Jordi Gonzalez, and Sergio Escalera. 2018. Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing*, 75:21–31.

Sahar Sadrizadeh, Ljiljana Dolamic, and Pascal Frossard. 2023. Transfool: An adversarial attack against neural machine translation models. *arXiv preprint arXiv:2302.00944*.

Avani Sakhapara, Dipti Pawade, Baiju Dodhia, Jigar Jain, Omkar Bhosale, and Onkar Chakrawar. 2022. Summarization of tweets related to disaster. In *Proceedings of International Conference on Recent Trends in Computing: ICRTC 2021*, pages 651–665. Springer.

Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. 2018. Interpretable adversarial perturbation in input embedding space for text. In *IJCAI*, pages 4323–4330.

Ibrahim Sobh, Ahmed Hamed, Varun Ravi Kumar, and Senthil Yogamani. 2021. Adversarial attacks on multi-task visual perception for autonomous driving. *arXiv preprint arXiv:2107.07449*.

Chenghao Sun, Yonggang Zhang, Wan Chaoqun, Qizhou Wang, Ya Li, Tongliang Liu, Bo Han, and Xinmei Tian. 2022. Towards lightweight black-box attack against deep neural networks. *NeurIPS*, 35:19319–19331.

Samson Tan, Shafiq Joty, Min-Yen Kan, and Richard Socher. 2020. It's morphin'time! combating linguistic discrimination with inflectional perturbations. *arXiv preprint arXiv:2005.04364*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Hetvi Waghela, Sneha Rakshit, and Jaydip Sen. 2024. A modified word saliency-based adversarial attack on text classification models. *arXiv preprint arXiv:2403.11297*.

Yu Wan, Keqin Bao, Dayiheng Liu, Baosong Yang, Derek F Wong, Lidia S Chao, Wenqiang Lei, and Jun Xie. 2022. Alibaba-translate china's submission for wmt 2022 metrics shared task. *arXiv preprint arXiv:2210.09683*.

Boxin Wang, Chejian Xu, Xiangyu Liu, Yu Cheng, and Bo Li. 2022. Semattack: Natural textual attacks via different semantic spaces. In *NAACL*, pages 176–205.

Hanrui Wang, Shuo Wang, Cunjian Chen, Massimo Tistarelli, and Zhe Jin. 2024. A multi-task adversarial attack against face authentication. *arXiv preprint arXiv:2408.08205*.

Suning Wang, Ying Wang, Linlin Jiang, Yong Chang, Kun Zhao, Lu Chen, Chunzheng Gao, et al. 2025. Assessing the clinical support capabilities of chatgpt 4o and chatgpt 4o mini in managing lumbar disc herniation. *European Journal of Medical Research*, 30(1):1–9.

Wenqiang Wang, Chongyang Du, Tao Wang, Kaihao Zhang, Wenhan Luo, Lin Ma, Wei Liu, and Xiaochun Cao. 2023. Punctuation-level attack: Single-shot and single punctuation can fool text models. In *NeurIPS*.

Yuxuan Wang, Wanxiang Che, Ivan Titov, Shay B Cohen, Zhilin Lei, and Ting Liu. 2021a. A closer look into the robustness of neural dependency parsers using better adversarial examples. In *ACL*, pages 2344–2354.

Zhibo Wang, Hengchang Guo, Zhifei Zhang, Wenxin Liu, Zhan Qin, and Kui Ren. 2021b. Feature importance-aware transferable adversarial attacks. In *ICCV*, pages 7619–7628. IEEE.

Austin Watkins, Thanh Nguyen-Tang, Enayat Ullah, and Raman Arora. 2024. Adversarially robust multi-task representation learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yijun Xiang. 2024. Multi-model fusion and multi-task learning for enhanced earthquake sequence prediction using acoustic signals. In *2024 4th International Conference on Computer Systems (ICCS)*, pages 28–33. IEEE.

Wang Xiaosen, Kangheng Tong, and Kun He. 2023. Rethinking the backward propagation for adversarial transferability. *NeurIPS*, 36:1905–1922.

L Xue. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. 2021. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10665–10673.

Zheng Yuan, Jie Zhang, Yunpei Jia, Chuanqi Tan, Tao Xue, and Shiguang Shan. 2021. Meta gradient adversarial attack. In *ICCV*, pages 7728–7737. IEEE.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *ACL*, pages 6066–6080.

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114.

Yinghua Zhang, Yangqiu Song, Jian Liang, Kun Bai, and Qiang Yang. 2020. Two sides of the same coin: White-box and black-box attacks for transfer learning. In *SIGKDD*, pages 2989–2997.

Yu Zhe, Rei Nagaike, Daiki Nishiyama, Kazuto Fukuchi, and Jun Sakuma. 2024. Adversarial attacks on hidden tasks in multi-task learning. *arXiv preprint arXiv:2405.15244*.

Mingyi Zhou, Jing Wu, Yipeng Liu, Shuaicheng Liu, and Ce Zhu. 2020. Dast: Data-free substitute training for adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 234–243.

Hai Zhu, Qingyang Zhao, Weiwei Shang, Yuren Wu, and Kai Liu. 2024. Limeattack: Local explainable method for textual hard-label adversarial attack. In *AAAI*, volume 38, pages 19759–19767.

Wei Zou, Shujian Huang, Jun Xie, Xinyu Dai, and Jiajun Chen. 2019. A reinforced generation of adversarial examples for neural machine translation. *arXiv preprint arXiv:1911.03677*.

Overview of the Appendix

This appendix includes our supplementary materials as follows:

- More details of the substitute model in Section A
- More details of data in Section B
- Url of victim model used in Section C
- Details of baselines in Section D
- More details of defense methods in Section E
- Definition of text classification adversarial examples and NMT adversarial examples in Section F
- Experiment result for verifying independence in section G
- The proof of Theorem 4.2 in Section H
- More Explanation for the Non-Independent Case in Section I
- More Details of M3TL in Section J
- Experiment Result under Low-quality Auxiliary Data Setting in Section K

A Substitute Model

A.1 Substitute Model Architecture

Our substitute model comprises 12 transformer blocks, each with 768 hidden units and 12 self-attention heads. Each transformer block consists of the following substructures:

- **Self-Attention Layer:** The hidden size of the self-attention layer is 768.
- **Position-wise Feed-Forward Network:** The network first projects the output of the attention layer to a 3072-dimensional space using a fully connected layer, followed by a ReLU activation for non-linearity, and finally projects the 3072-dimensional space back to a 768-dimensional space via another fully connected layer.
- **Layer Normalization and Residual Connection:**
  - **Layer Normalization:** Applied to the output of each sub-layer to stabilize training.
  - **Residual Connection:** Adds the normalized output to the input of the sub-layer.

A.2 Substitute Model Training

We provide a detailed description of the training of the substitute model with the transformer-based architecture. This substitute model consists of 12 hidden layers with a dimensionality of 768. The activation function "GELU" is used, The dropout rate is 0.4. The training process is optimized with the AdamW optimizer (Yao et al., 2021), with batch size set to 64 and learning rate set to $6e-3$, over 5 epochs.

A.3 Computation Overhead of the Substitute Model Training

We train six substitute models on a server equipped with a 24 GB NVIDIA 3090 GPU. The training time for a single model is approximately 4 minutes, and the size of each trained model is 418 MB.

B Details of Datasets

The details of datasets are presented in Table 4

C The URLs of the Victim Models

The URLs of the victim models are presented in Table 5

D Details of Baselines

The details of baselines are presented in Table 7 and Table 8

E Defense Method

We initiate an extensive exploration of defensive strategies to counter CEMA. In practical systems, we thoroughly investigate various defense mechanisms, including train-free adjustments(Preceding Language Modifier) and adversarial training.

E.1 Preceding Language Modifier

The victim models used in our study are after-trained models sourced from the Huggingface website, Ali Translator, and Baidu Translator. Since the training details of these pre-trained models are not publicly available, re-training them using adversarial training is infeasible. Consequently, we adopt training-free defense methods. Specifically, we implement the same approach proposed by (Wang et al., 2023) and apply prompt learning techniques to large language models (LLMs) to mitigate adversarial text inputs. For this, we provide CoEdIT-XXL (a LLM used for correcting text errors). The prompt is as follows: "Please revise the text for grammatical errors, improve the spelling, grammar, clarity, concision, and overall readability." The results are presented in Figure 4.

"w/o" indicates the absence of a defense method, whereas "w" denotes the use of the CoEdIT-XXL model as a modifier for defense. Even after applying defense mechanisms using large language models, CEMA's attack effectiveness decreases but still maintains a significant level of performance.

E.2 Adversarial Training

We train four classification models as victim models and conduct adversarial training to evaluate the impact of adversarial training on CEMA's attack effectiveness. All four models are based on the BERT architecture and are labeled Bert1, Bert2, Bert3, and Bert4. Specifically, Bert1 and Bert3 are trained on the SST5 dataset, while Bert2 and Bert4 are trained on the Emotion dataset. The results are presented in Figure 5. "w/o" indicates the absence of adversarial training, while "w" represents the application of adversarial training. Although adversarial training reduces attack effectiveness, CEMA still demonstrates considerable performance.

Table 4: The statistics of datasets.

Dataset	Classes	Labels' name
SST5	5	Very positive, Positive, Neutral, Negative, Very negative
Emotion	6	Sadness, Joy, Love, Anger, Fear, Surprise

Table 5: The URL of the Victim Models

Model	Url
dis-sst5(A)	https://huggingface.co/SetFit/distilbert-base-uncased__sst5__all-train
dis-emotion(A)	https://huggingface.co/bhadresh-savani/distilbert-base-uncased-emotion
opus-mt(En-Zh)(A)	https://huggingface.co/Helsinki-NLP/opus-mt-En-Zh
ro-sst5(B)	https://huggingface.co/Unso/roberta-large-finetuned-sst5
ro-emotion(B)	https://huggingface.co/SamLowe/roberta-base-go_emotions
t5-small(En-Fr)(B)	https://huggingface.co/Alexle/T5-small-En-Fr
Baidu Translate (En-Fr) (C)	https://api.fanyi.baidu.com/
Ali Translate (En-Zh) (C)	https://translate.alibaba.com/

F Definition of Text Classification Adversarial Examples and NMT Adversarial Examples

F.1 Definition of NMT Adversarial Examples

We define the source language space as $\mathcal{X}$ and the target language space as $\mathcal{Y}$, examining two NMT systems: the source-to-victim model $M_{x\to y}$, which maps $\mathcal{X}$ to $\mathcal{Y}$ to maximize $P(y_{\text{ref}} \mid x)$, and the target-to-source model $M_{y\to x}$, which performs the reverse mapping. After training, these models can reconstruct original sentences as $\hat{x} = g(f(x))$. We propose black-box adversarial testing for NMT using auxiliary data by selecting test sentences from $\mathcal{T} \subset \mathcal{X}$ and generating adversarial cases $\delta \in \Delta$ to perturb inputs $x' = x + \delta$ such that $f(x')$ diverges significantly from $f(x)$.

NMT Adversarial Example: An NMT adversarial example is a sentence in

$$\begin{gathered}\mathcal{A} = \{x' \in \mathcal{X} \mid \exists x \in \mathcal{T}\}, \\ here\ \|x' - x\| < \epsilon \wedge S_t(y, y_{\text{ref}}) \geq \gamma \wedge S_t(y', y_{\text{ref}}) < \gamma'\end{gathered} \tag{8}$$

where function f represents the NMT model. The variables x and x' represent the original text and the adversarial test case, respectively, while y and y' stand for their respective translations. In detail, $y = f(x)$ and $y' = f(x')$. The function $S_t(\cdot,\cdot)$ gauges the similarity between two sentences. Additionally, γ and γ' denote thresholds for acceptable translation quality. Translation quality is deemed unacceptable if γ' drops below γ.

F.2 Definition of Text Classification Adversarial Examples

Definition of Text Classification Adversarial Examples: Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a set of text inputs, where each x_i is a text document (e.g., sentence or paragraph). Let $f(\cdot)$ represent a text classification model, where:

$$f : X \to Y$$

is a mapping from the input space X to the label space Y, with $Y = \{y_1, y_2, \ldots, y_m\}$ representing the set of possible class labels (e.g., positive, negative, neutral).

Given an input $x \in X$ and its corresponding true label $y_{\text{true}} = f(x)$, an *adversarial example* $\hat{x}$ is a perturbed version of the input x that is intentionally crafted to cause the model to misclassify it, while remaining perceptually and semantically similar to the original text. Formally, an adversarial example is defined as:

$$\hat{x} = x + \delta$$

where δ is a small perturbation that satisfies:

$$\|\delta\| \leq \epsilon$$

Here, $\|\delta\|$ represents the magnitude of the perturbation (e.g., measured in terms of the number of word substitutions or sentence modifications), and ϵ is a threshold that bounds the maximum allowable perturbation.

Additionally, we impose a *semantic similarity* constraint, ensuring that the perturbation δ does not

Table 6: The details of the methods employed in the baseline comparisons. The Perturbed Level indicates the target of the attack methods, where "word" denotes the specific words targeted for perturbation, and "char" refers to the characters within a word that are altered by the attack method.

Table 7: Information on the classification attack method used as the baseline.

Methods	Perturbed Level	Gradient	Soft-labels	Hard-labels	Knowledge
Bae	Word	✗	✓	✓	black-box
FD	Char	✓	✓	✓	white-box
Hotflip	Char	✓	✓	✓	white-box
PSO	Word	✗	✓	✓	black-box
TextBugger	Char+Word	✓	✓	✓	white-box
Leap	Word	✗	✓	✓	black-box
CT-GAT	Word	✗	✓	✓	black-box
HQA	Word	✗	✓	✓	black-box
CEMA	Char+Word	✗	✗	✓	black-box

Table 8: Information on the translation attack method used as the baseline.

Methods	Perturbed Level	Gradient	Soft-labels	Hard-labels	Knowledge
Hot-trans	Char	✓	✗	✗	white-box
kNN	Word	✓	✗	✗	white-box
Morphin	Word	✗	✗	✓	black-box
RA	Word	✓	✗	✗	white-box
Seq2Sick	Word	✓	✗	✓	white-box
TransFool	Word	✗	✗	✓	black-box
CEMA	Char+Word	✗	✗	✓	black-box

alter the meaning of the input significantly. This is formalized as:

$$\mathrm{Sim}(x, \hat{x}) \leq \gamma$$

where $\mathrm{Sim}(x, \hat{x})$ denotes a semantic similarity measure (such as cosine similarity) between the original input x and the adversarial example $\hat{x}$, and γ is a predefined threshold that controls the acceptable level of semantic similarity. This ensures that the adversarial example $\hat{x}$ remains semantically close to x, while still leading to a misclassification.

The adversarial example $\hat{x}$ causes the model to output a different class than the true label:

$$f(\hat{x}) \neq y_{\text{true}} \quad \text{and} \quad f(x) = y_{\text{true}}$$

Thus, the text classification adversarial example $\hat{x}$ satisfies the following optimization problem:

$$\hat{x} = \text{argmin}_{x' \in X} \mathcal{L}(f(x'), y_{\text{true}})$$

$$\text{subject to} \quad \|x' - x\| \leq \epsilon \quad \text{and} \quad \mathrm{Sim}(x, x') \geq \gamma$$

where $\mathcal{L}(\cdot)$ is the loss function used to measure the discrepancy between the predicted label $f(x')$ and the true label y_{true}.

G The Experiment for Verifying Independence

We employ the DWB, FD, TextBugger, Hotflip, and PSO methods to generate adversarial examples in the subset of the test dataset. Since the exact success probabilities of each method's attacks are unavailable, we estimate these probabilities based on the observed frequency of successful attacks. In the table, we report the frequency $P(AB)$ of both methods successfully attacking, as well as the individual success frequencies $P(A)$ for Method A and $P(B)$ for Method B. Our findings indicate that $P(AB)$ closely approximates $P(A) \times P(B)$, with the average deviation $P(A) \times P(B) - P(AB)$

Table 9: Experiment results with improving the similarity threshold and adding attack methods for victim model A

Similarity Threshold	Method Number	ASR(%)↑	ASR(%)↑	BLEU↓	Similarity↑
0.8	3	73.57	62.27	0.14	0.93
0.9	6	67.16	58.89	0.16	0.95
0.8	3	80.80	65.40	0.15	0.93
0.9	6	78.25	61.15	0.16	0.94

Table 10: The experimental results for verifying independence.

Method A	Method B	P(A)	P(B)	P(A)*P(B)	P(AB)	P(A)*P(B)-P(AB)
DWB	FD	52.50%	40.50%	21.26%	19.00%	2.26%
DWB	Textbugger	52.50%	72.50%	38.06%	35.50%	2.56%
DWB	Hotflip	52.50%	72.50%	38.06%	37.00%	1.06%
DWB	PSO	52.50%	76.50%	40.16%	35.00%	5.16%
FD	Textbugger	40.50%	72.50%	29.36%	31.00%	-1.64%
FD	Hotflip	40.50%	72.50%	29.36%	31.50%	-2.14%
FD	PSO	40.50%	76.50%	30.98%	30.50%	0.48%
Textbugger	Hotflip	72.50%	72.50%	52.56%	57.50%	-4.94%
Textbugger	PSO	72.50%	76.50%	55.46%	54.00%	1.46%
Hotflip	PSO	72.50%	76.50%	55.46%	58.00%	-2.54%
Average				39.07%	38.90%	0.17%

being just 0.17%. The detailed experimental results are provided in Table 10. Event independence is defined as the occurrence of event A having no effect on the occurrence of event B. Therefore, we assume that the success of adversarial examples generated by Method A does not influence the success of those generated by Method B.

H The Proof of Theorem 4.2

We aim to prove that the probability of at least one successful attack, denoted as $P(A^u)$, increases as the number of attack methods u increases. Specifically, we want to show that:

$$P(A^u) \leq P(A^{u+1}), \quad \text{for all} \quad u \geq 1, \tag{9}$$

where A^u represents the event that at least one of the u attack methods succeeds, and A^{u+1} represents the event that at least one of the $u+1$ attack methods succeeds.

Definitions

1. Success Probability of Attack Methods: - Let p_i^j be the success probability of the j-th attack method of the victim text x_i^*.

2. Event A^u: - A^u denotes the event that at least one of the u attack methods succeeds, i.e.,

$$A^u = A_1 \cup A_2 \cup \cdots \cup A_u, \tag{10}$$

where A_i is the event that the i-th attack method succeeds.

3. Event $\overline{A^u}$: - $\overline{A^u}$ denotes the event where none of the u attack methods succeeds, i.e.,

$$\overline{A^u} = \overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_u}. \tag{11}$$

Therefore, the probability of at least one success is:

$$P(A^u) = 1 - P(\overline{A^u}). \tag{12}$$

4. Event A^{u+1}: - Similarly, for $u+1$ attack methods, A^{u+1} is defined as:

$$A^{u+1} = A_1 \cup A_2 \cup \cdots \cup A_{u+1}, \tag{13}$$

and $\overline{A^{u+1}}$ represents the event where none of the $u+1$ attack methods succeeds:

$$\overline{A^{u+1}} = \overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_{u+1}}. \tag{14}$$

Therefore, the probability of at least one success with $u+1$ attack methods is:

$$P(A^{u+1}) = 1 - P(\overline{A^{u+1}}). \tag{15}$$

Goal

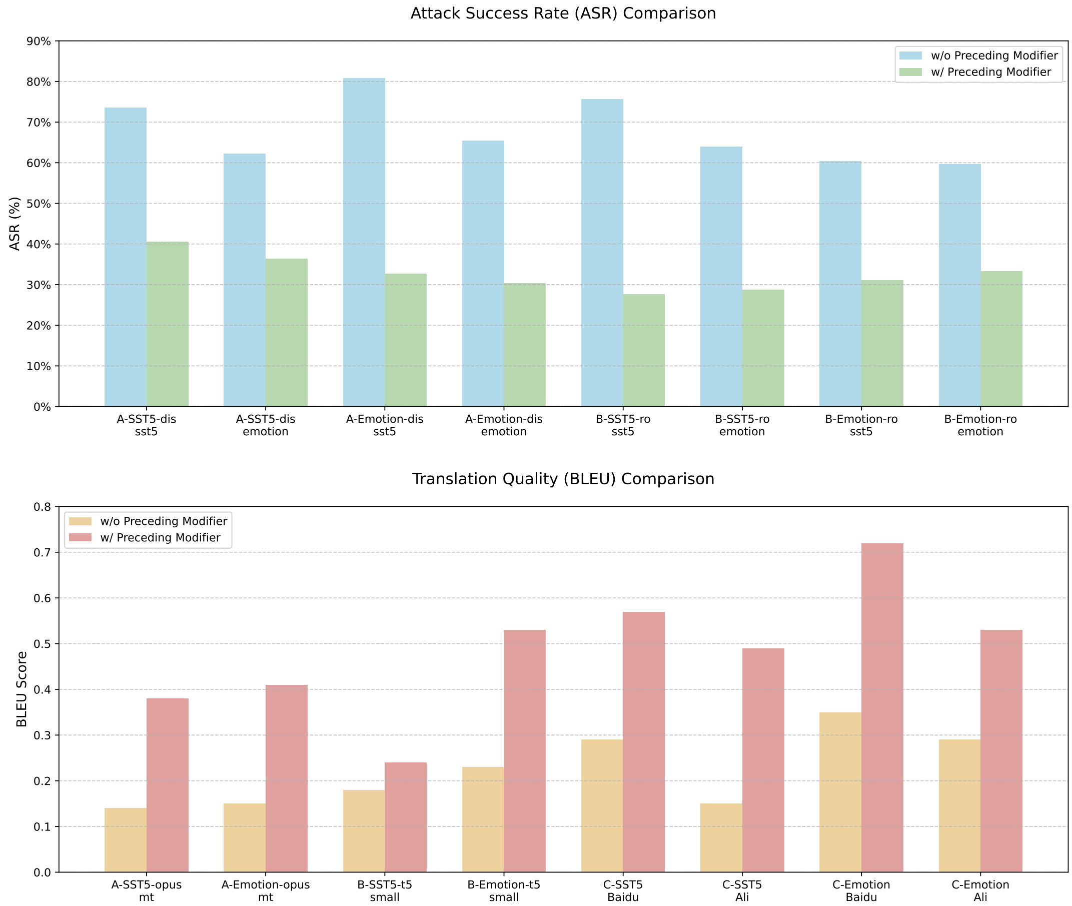

Figure 4: The results of Preceding Language Modifier

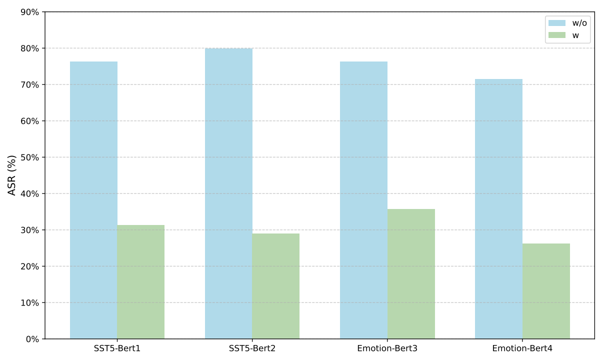

Figure 5: The results of adversarial training.

We want to prove that

$$P(A^u) \leq P(A^{u+1}), \tag{16}$$

i.e., the probability of at least one success increases as the number of attack methods u increases.

Proof Process

Step 1: Probability of Failure with u Attack Methods

The probability of failure with u attack methods is given by:

$$P(\overline{A^u}) = P(\overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_u}). \tag{17}$$

- Independent Events: If the attack methods are independent, the failure probability is:

$$P(\overline{A^u}) = \prod_{j=1}^{u} (1 - p_i^j). \tag{18}$$

- Dependent Events: If the attack methods are not independent, we have:

$$P(\overline{A^u}) \leq \prod_{j=1}^{u} (1 - p_i^j). \tag{19}$$

Step 2: Probability of Failure with $u+1$ Attack Methods

The probability of failure with $u+1$ attack methods is:

$$P(\overline{A^{u+1}}) = P(\overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_{u+1}}). \quad (20)$$

- Independent Events: If the attack methods are independent, the failure probability is:

$$P(\overline{A^{u+1}}) = \prod_{j=1}^{u+1} (1 - p_i^j). \quad (21)$$

- Dependent Events: If the attack methods are dependent, we have:

$$P(\overline{A^{u+1}}) \leq \prod_{j=1}^{u+1} (1 - p_i^j). \quad (22)$$

Step 3: Comparing $P(A^u)$ and $P(A^{u+1})$
We know:

$$P(A^u) = 1 - P(\overline{A^u}), \quad (23)$$

$$P(A^{u+1}) = 1 - P(\overline{A^{u+1}}). \quad (24)$$

- Independent Events:

$$P(A^u) = 1 - \prod_{j=1}^{u} (1 - p_i^j), \quad (25)$$

$$P(A^{u+1}) = 1 - \prod_{j=1}^{u+1} (1 - p_i^j). \quad (26)$$

Since $\prod_{j=1}^{u+1}(1 - p_i^j) \leq \prod_{j=1}^{u}(1 - p_i^j)$, we have:

$$P(A^{u+1}) \geq P(A^u). \quad (27)$$

- Dependent Events:
Similarly, since $P(\overline{A^{u+1}}) \leq \prod_{j=1}^{u+1}(1 - p_i^j)$ and $P(\overline{A^u}) \leq \prod_{j=1}^{u}(1 - p_i^j)$, we conclude that:

$$P(A^{u+1}) \geq P(A^u). \quad (28)$$

Conclusion
Thus, we have proven that:

$$P(A^u) \leq P(A^{u+1}), \quad (29)$$

i.e., the probability of at least one success increases as the number of attack methods u increases.

Condition for Equality
Equality $P(A^u) = P(A^{u+1})$ holds under the following conditions:

- The attack methods are completely independent and have identical success probabilities $p_i^j = p_{u+1}^s$ for all i, in which case adding an attack method does not change the overall failure probability. - The newly added attack method does not contribute any additional success probability. This occurs when the new attack method is redundant or has no distinct contribution to the attack success.

Discussion of Small Probability Events
When the success probability of each attack method p_i^j is small (i.e., the attack methods have low success rates), in most cases, adding more attack methods leads to a significant increase in the probability of at least one success. This is because each additional attack method provides another opportunity for success, which, even with low individual probabilities, results in a cumulative increase in the overall success probability.

Therefore, in practical scenarios where each attack method has a small success probability, adding more methods increases the likelihood of at least one successful attack. This cumulative effect ensures that $P(A^u)$ grows as u increases.

Remark In Section I of the appendix, we examine the case of non-independence. We find that, in the non-independent scenario, using more methods to generate adversarial examples increases the likelihood of successfully attacking the victim model.

Similarly, we can also prove $P(B^u) \leq P(B^{u+1})$

I Supplementary Explanation for the Non-Independent Case in Section Candidate Adversarial Example Generation

The probability of successfully attacking the victim model using adversarial examples generated by methods 1, 2, ..., n is greater than or equal to the probability of successfully attacking the victim model using adversarial examples generated by method 1 alone. This is because, when only method 1 is used, there is only one candidate adversarial example per victim text. In contrast, when n methods are employed, there are n candidate adversarial examples for each victim text, including the one generated by method 1. Therefore, the probability of successfully attacking the victim model using adversarial examples generated by n methods is greater than or equal to the probability of successfully attacking the victim model using adversarial examples from method 1 alone. The probabilities are equal only when method 1 achieves the maximum success rate for all victim texts. However, the SST5 and Emotion datasets contain 2,210 and

2,000 victim texts, respectively, making it unlikely that method 1 will achieve the maximum success rate across all victim texts. Thus, we conclude that, in most cases, the probability of successfully attacking the victim model using adversarial examples generated by n methods is greater than when using adversarial examples generated by method 1 alone.

Furthermore, based on this property, we can deduce that, in most cases, the probability of successfully attacking the victim model using adversarial examples generated by methods 1, 2, ..., n is greater than when using adversarial examples generated by methods 1, 2, ..., m, where $n > m$. In other words, employing more methods to generate adversarial examples increases the likelihood of a successful attack on the victim model.

J Multi-model Multi-task Learning (M3TL)

Multi-model Multi-task Learning (M3TL) is a machine learning method that combines multiple learning models with multiple tasks. It is a combination of Multi-task Learning (MTL) and Multi-model Learning, aiming to improve model performance by jointly optimizing multiple tasks, especially when dealing with multiple related tasks.

J.1 Key Concepts

J.1.1 Multi-task Learning (MTL)

In traditional machine learning, each model typically handles a single task. In contrast, Multi-task Learning (MTL) involves jointly training multiple related tasks with a shared model. The goal is to allow the model to simultaneously optimize multiple objectives by sharing representations, knowledge, or parameters. Common applications include sentiment analysis and text classification, where the same features can be used for multiple tasks (e.g., predicting sentiment labels and classifying news articles). For instance, training a neural network to simultaneously perform two tasks: image classification and object detection.

J.1.2 Multi-model Learning

Unlike traditional single-model approaches, Multi-model Learning uses multiple independent or combined models to solve a problem. Each model may focus on different aspects of the problem or apply different algorithms to address the same task. For example, using multiple models such as neural networks, decision trees, and support vector machines to handle the same task, thereby leveraging the strengths of each model.

J.1.3 Multi-model Multi-task Learning (M3TL)

M3TL is a method that combines Multi-task Learning and Multi-model Learning. The core idea is to use multiple models (e.g., neural networks, decision trees, support vector machines, etc.) to learn multiple related tasks, with these models sharing some information or parameters. This means that during training, M3TL models handle multiple tasks and models simultaneously, enabling each model to learn across multiple tasks while sharing representations and knowledge between tasks.

K Low-quality Auxiliary Data

Initially, we treated the victim texts as auxiliary data. Now, we assume that the attacker cannot access the victim texts directly but can gather victim texts' relevant information and collect relevant internet data as the auxiliary data. For example, the SST5 and Emotion datasets, both related to sentiment analysis but differing in label space and distribution. We use 100 unlabeled texts from the Emotion validation set as auxiliary data for the SST5 attack, and vice versa. The results in Table 11 show that despite limited auxiliary data and distribution differences, CEMA achieved a 66.45% attack success rate and a BLEU score of 0.27. This demonstrates that an attacker only needs partial knowledge of the victim's data and can collect relevant information from the web to successfully attack the multi-task system using CEMA.

Table 11: Attack performance of Low-quality Auxiliary Data.

		Victim Model A			Victim Model B			Victim Model C	
		dis-sst5	dis-emotion	opus-mt (en-zh) (A)	ro-sst5 (A)	ro-emotion (A)	t5-small (en-fr)	Baidu Translate (en-fr)	Ali Translate (en-zh)
Victim Data	Access Data	ASR(%)↑	ASR(%)↑	BLEU↓	ASR(%)↑	ASR(%)↑	BLEU↓	BLEU↓	BLEU↓
SST5	SST5	73.6	62.3	0.14	75.7	64.0	0.18	0.29	0.15
	Emotion	64.0	60.8	0.18	59.2	52.0	0.22	0.36	0.27
Emotion	Emotion	80.8	65.4	0.15	60.4	59.6	0.23	0.35	0.29
	SST5	66.4	36.0	0.21	48.8	46.4	0.36	0.44	0.42

Table 12: The results of six tasks

	ASR(%)				BLEU	
Data	dis-emotion	ro-emotion	dis-sst5	ro-sst5	opus-mt	t5-small
SST5	75.91	74.90	67.04	62.82	0.18	0.22
Emotion	83.25	66.85	71.35	68.40	0.17	0.27

Table 13: The results of translation, summary, and text-to-image tasks.

Data	Task	Metric	Score
Pokemon	Translation	BLEU	0.29
	Summary	RDP	47%
	Text to Image	CDP	56%

Table 14: Performance of CEMA under different number setting of candidate adversarial examples.

Data	Example Number	Victim Model A			Victim Model B		
		dis-sst5 (A)	dis-emoton (A)	opumt(En-Zh) (A)	ro-sst5 (B)	ro-emotion (B)	t5-small(En-Fr) (B)
		ASR(%)↑	ASR(%)↑	BLEU↓	ASR(%)↑	ASR(%)↑	BLEU↓
SST5	3	73.6	62.3	0.14	75.7	64.0	0.18
	1	50.4	29.2	0.30	43.8	24.7	0.35
Emotion	3	80.8	65.4	0.15	60.4	59.6	0.23
	1	29.2	34.8	0.31	39.2	47.2	0.39

Table 15: The results of few-shot and many-shot queries

Model	Victim Model A						Victim Model B					
	dis-sst5	dis-emotion	opus-mt	dis-sst5	dis-emotion	opus-mt	ro-sst5	ro-emotion	t5-small	ro-sst5	ro-emotion	t5-small
Data	SST5			Emotion			SST5			Emotion		
Shot-Size	ASR(%)↑	ASR(%)↑	BLEU↓	ASR(%)↑	ASR(%)↑	BLEU↓	ASR(%)↑	ASR(%)↑	BLEU↓	ASR(%)↑	ASR(%)↑	BLEU↓
2000	87.56	83.27	0.1	91.7	81.45	0.1	86.46	78.47	0.15	67.45	69.55	0.16
1000	83.04	76.76	0.11	88.25	76.15	0.12	84.16	73.49	0.16	66.35	67.05	0.17
100	73.57	62.27	0.14	80.8	65.4	0.15	75.66	64.01	0.18	60.4	59.60	0.23
50	63.71	45.64	0.15	71.05	53.55	0.18	71.69	59.38	0.19	58.65	57.9	0.24
10	38.38	32.06	0.19	43.35	37.7	0.21	59.28	46.51	0.21	46.15	41.75	0.27