FEW-SHOT TEXT ADVERSARIAL ATTACK FOR BLACK-BOX MULTI-TASK LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Current multi-task adversarial text attacks rely on white-box access to shared internal features and assumption of homogeneous multi-task learning framework. As a result, these attacks are less effective against practical scenarios involving black-box feedback APIs and heterogeneous multi-task learning. To bridge this gap, we introduce **C**luster and **E**nsemble **M**util-task Text Adversarial **A**ttack (**CEMA**), an effective black-box attack that exploits the transferability of adversarial texts. Specifically, we initially employ cluster-oriented substitute model training, as a plug-and-play framework, to simplify complex multi-task scenarios into more manageable text classification attacks and train the substitute model. Next, we generate multiple adversarial candidate examples by applying various adversarial text classification methods. Finally, we select the adversarial example that attacks the most substitute models as the final attack output. CEMA is evaluated on two primary multi-task objectives: text classification and translation. In the classification task, CEMA achieves attack success rates that exceed 60% while reducing the total number of queries to 100. For the text translation task, the BLEU scores of both victim texts and adversarial examples decrease to below 0.36 with 100 queries even including the commercial translation APIs, such as Baidu Translate and Ali Translate. Additionally, we derive the theoretical lower bound for CEMA's success rate, demonstrating that a successful attack increases with the number of candidate substitute models.

1 INTRODUCTION

A multi-task textual adversarial attack misleads multiple tasks simultaneously through small perturbations, increasing attack efficiency and impact. It poses significant risks to safety-critical systems, leading to wrong decisions. Defending against such attacks is challenging due to the need for multi-task robustness, making it a key issue in AI security (Liu et al., 2017; Lin et al., 2022).

Research on text multi-task adversarial examples typically concentrates on tasks of the same type, particularly classification tasks (Liu et al., 2017). However, in real-world applications, multi-task learning often involves tasks of different types. Existing adversarial attack methods generally assume that attackers have access to the model architecture and shared layer information within a unified model (Guo et al., 2020). However, most commercial and application-based models are proprietary, with their architecture and parameters hidden from external attackers. Additionally, current multi-task adversarial attack strategies primarily target models that employ a shared parameter approach for managing multiple tasks. In contrast, multi-model and heterogeneous multi-task learning approaches (Aoki et al., 2022) handle each task with a separate model, without direct parameter sharing. As a result, most existing adversarial methods, designed to attack shared parameter models, are ineffective against these systems because of the absence of a common layer to target.

Our goal is to perform multi-task textual adversarial attacks in realistic scenarios. Based on the previous analysis, such a scenario should encompass a variety of tasks, with black-box model feedback being more reflective of real-world conditions. Moreover, both parameter-sharing multi-task learning systems and heterogeneous multi-task learning systems must be considered. Additionally, limiting the number of queries is essential to conserve resources and reduce the risk of detection, making it a key aspect of practical attack scenarios.

In limited-access scenarios, a straightforward strategy is the transfer attack, which crafts adversarial examples in a substitute model. Training effective substitute models becomes challenging in the absence of a well-trained substitute model, particularly in multi-task learning, where limited access to input-output pairs and poor transferability between tasks in heterogeneous settings pose significant difficulties. Rather than mimicking the entire multi-task model, we propose focusing on building a substitute model with **strong discriminability**. This approach allows a single substitute model to generate adversarial examples that target all tasks simultaneously, even when trained with limited data.

We propose CEMA (**C**luster and **E**nsemble **M**ulti-task Text **A**dversarial Attack), a framework that leverages a small set of auxiliary texts sharing characteristics with the victim's texts. Using a pre-trained model, we vectorize texts and their outputs, perform clustering, and train substitute models on these auxiliary texts and cluster labels. This converts the multi-task attack into a single-task text classification problem. Repeating this process, we can obtain multiple substitute models. During the adversarial example generation phase for victim texts, for each victim text, adversarial candidates are generated for each victim text. The final adversarial example is selected based on its success across the most substitute models.

Although the substitute model trained by CEMA differs from the victim model trained through multi-task learning, our substitute model, demonstrates **strong discriminative capability**. For task A, if an adversarial attack on the substitute model f^{sub} successfully changes the cluster label of text x_i from 0 to 1, the label y_i^A shifts accordingly, indicating a successful attack on task A. We derive and demonstrate that adversarial examples based on cluster labels, when effective against multiple substitute models, can also transfer effectively to other tasks $B, C, \ldots, N$.

During the experiment, we focus on text classification and translation within a multi-task learning framework. For the text classification task, CEMA achieves an attack success rate (ASR) of over 60% with only 100 queries. In the text translation task, CEMA reaches a BLEU score of 0.14. Even with limited auxiliary data that differs significantly from the training dataset, CEMA maintains an ASR of up to 66.40% for classification tasks and a BLEU score of 0.27 for translation tasks. The primary **contributions** are summarized as follows:

- To the best of our knowledge, we are the first to extend text adversarial attacks to the multi-task setting by training cluster-oriented substitute models and employing transferability-oriented adversarial example selection. The proposed CEMA method generates high-quality adversarial examples for multiple tasks simultaneously with very few queries in black-box scenarios.
- We present the first *plug-and-play framework* that converts a multi-task attack into a single-task attack, enabling traditional methods to be easily adapted to multi-task scenarios. Furthermore, our approach overcomes the limitations of existing multi-task attack methods, which depend on shared layers in multi-task models. CEMA effectively handles multi-task scenarios with heterogeneous models, whether they involve related or independent tasks. Additionally, we derive a theoretical lower bound for CEMA's success rate, showing that the probability of success increases with the number of substitute models used.
- We demonstrate the effectiveness of CEMA through rigorous mathematical derivations, as well as comprehensive experiments. The experimental results show the proposed CEMA achieves an attack success rate (ASR) of over 60% in text classification tasks and a BLEU score of less than 0.15 in translation tasks, indicating effective adversarial attack performance in both cases.

2 PRELIMINARY

2.1 TRANSFERABILITY AND TRANSFER ATTACKS

Transfer attacks leverage adversarial examples to target different models without requiring direct access, posing a significant security threat in black-box scenarios (Szegedy et al., 2014; Papernot et al., 2017; Dong et al., 2018). **Transferability** refers to the phenomenon where adversarial examples crafted for one model can successfully compromise other models as well (Zhang et al., 2020).

2.2 MULTI-TASK LEARNING AND MULTI-MODEL MULTI-TASK LEARNING

Multi-Task Learning (MTL) involves simultaneously training multiple related tasks, enabling models to share knowledge and improve generalization, particularly when data is limited. MTL has been extensively applied in fields such as natural language processing and computer vision, resulting in more robust models. However, challenges such as task interference and balancing shared information across tasks remain. Recent advancements seek to mitigate these challenges and enhance MTL's overall effectiveness. **Multi-Model Multi-Task Learning** extends the traditional MTL framework by utilizing separate models for each task, providing greater flexibility and better handling of task heterogeneity. This approach minimizes negative transfer and allows for task-specific optimizations. However, it also increases computational complexity and the difficulty of integrating outputs from different models. Current research focuses on hybrid methods that balance task specialization with shared learning, aiming to optimize model architectures and enhance resource efficiency.

3 THREAT MODEL

❶Victim Model: In this paper, we explore a more practical scenario of Multi-Model Multi-Task Learnin, focusing on the tasks of text classification and translation. We utilize publicly available APIs from the Hugging Face platform as the victim models for our attacks. Specifically, we target the SST5 and Emotion datasets for text classification, and we select DistilBERT and RoBERTa models trained on these datasets, referred to as dis-sst5, ro-sst5, dis-emotion, and ro-emotion, respectively. For the translation task, we target the opus-mt model for English-to-Chinese translation and the t5-small model for English-to-French translation. The URLs of these models are provided in Table 8 in the Appendix. Meanwhile, to simulate a more realistic attack scenario, we employ two commercial translation APIs: Baidu Translate for English-to-French translation and Ali Translate for English-to-Chinese translation. We design three multi-task victim models using these base models. **Victim Model A** comprises two classification models and one translation model: dis-sst5, dis-emo, and opus-mt. **Victim Model B** also comprises two classification models and one translation model: ro-sst5, ro-emo, and t5-small. **Victim Model C** consists of two commercial translation APIs: Baidu Translate and Ali Translate. ❷Attacks's Goal The objective of our attack is to compromise the performance of all tasks within a multi-task model. Adversarial examples are designed to generate inputs that universally degrade the model's performance across all tasks, rather than targeting a single task. This means the attacker seeks to create an input that not only disrupts one specific task but also negatively impacts the performance of other tasks within the model. In the context of multitask learning, specifically for text classification, the objective is to ensure that the original text and the adversarial example produce different output labels in the target model, *i.e.* $y_{\text{adv}} \neq y_{\text{ori}}$. For the translation task within multi-task learning, the goal is to ensure that the original text and the adversarial example lead to a significant semantic divergence in the generated translations, *i.e.* $\arg\min \text{BLEU}(y_{\text{adv}}, y_{\text{ori}})$. ❸ **Adversary Capabilities**: We analyze the adversary's capabilities from three perspectives: query access, API feedback, and auxiliary data.(1) **Query Access:** Query access refers to the adversary's ability to interact with the target model before delivering the final adversarial input. We assume the attacker has up to 100 opportunities to query the victim model, with each query generating output results for all tasks. (2) **API feedback:** In a practical multi-task text adversarial attack, the attacker has no access to the internal information of the model and can only obtain the final output results of the model. Therefore, the API feedback serves as a black-box response, providing predicted labels for the classification task and the translated text (e.g., French output for English-to-French translation). (3) **Auxiliary Data:** From the perspective of data quantity, we assume that the attacker can acquire only a limited amount of Auxiliary Data, specifically 100 unlabeled texts. In terms of data distribution, we explore two scenarios: (a) The 100 unlabeled texts are sampled from the same distribution as the victim's texts, such as the 100 unlabeled texts in the validation dataset. (b) The 100 unlabeled texts and the victim's texts come from datasets of the same nature but with different distributions.

4 METHOD

As shown in Figure 1, our method, CEMA, consists of the following steps: ❶ **Representation Learning** (Section 4.1). We convert the auxiliary texts and their outputs from multiple tasks into

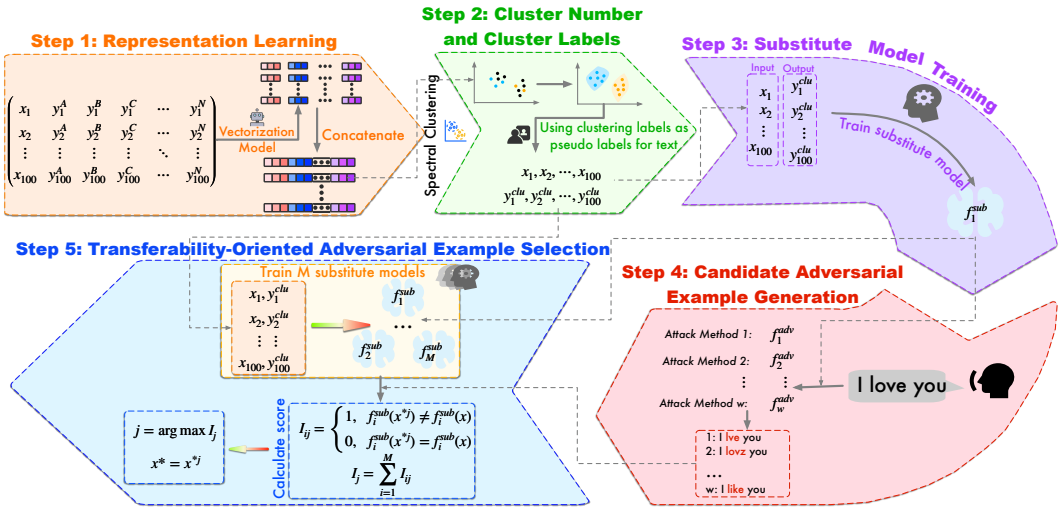

Figure 1: **The Overview of CEMA.** ❶ CEMA assigns cluster labels to auxiliary texts through a clustering method. These text-label pairs are then used to train the substitute model. This process allows CEMA to efficiently transform a multi-task scenario into a single-task text classification scenario, with only 100 queries to the black-box multi-task model. ❷ To improve attack effectiveness, CEMA applies multiple attack methods to the substitute models, generating candidate adversarial examples to refine the selection process. CEMA also trains several substitute models, selecting the final adversarial example based on its success across the majority of them.

vector form using representation learning. ❷ **Clustering to Generate cluster labels** (Section 4.2). After determining the optimal number of clusters, we apply a clustering algorithm to the vector representations of the auxiliary texts and their outputs, assigning a cluster label to each auxiliary text. ❸ **Training Substitute Models** (Section 4.3). We train substitute models f^{sub} using auxiliary texts as input and their corresponding cluster labels as output. ❹ **Generation of Adversarial Candidates** (Section 4.4). We apply various text adversarial attack methods to the substitute model f^{sub} to generate multiple adversarial candidates. ❺ **Final Adversarial Example Selection** (Section 4.5). By repeating steps ❶, ❷, and ❸, we can train multiple substitute models, $f_1^{\text{sub}}, f_2^{\text{sub}}, \ldots, f_M^{\text{sub}}$. We select the adversarial candidate that successfully attacks the most substitute models as the final adversarial example.

4.1 Representation Learning

In a multi-task model, both the input text and output labels need to be appropriately vectorized to effectively capture the relevant information. This section details the specific approach used for learning the representations of input text and output labels. Pre-trained models are extensively utilized in NLP for textual feature extraction (Tabassum & Patil, 2020; Han et al., 2021). These models are highly effective as they are trained on large-scale datasets, enabling them to learn general language patterns and representations. Furthermore, concatenating multiple text representations allows for the simultaneous encoding of multiple texts (Devlin et al., 2019). Accordingly, we leverage a pre-trained model to vectorize both the input text and output labels, generating their respective embeddings. These embeddings are subsequently concatenated to form a unified representation that captures the information from both the input text and the output labels.

As outlined in lines 1-5 of Algorithm 1, we begin by querying the multi-task model to retrieve the output text for each task. Next, auxiliary text with corresponding output results are vectorized by pre-trained models to extract relevant features for subsequent clustering process. We define the multi-task model as f_v, which deals with the set of tasks $A, B, \ldots, N$. The pre-trained model is defined as f_{pre}. The attacker is assumed to have access to a small set of auxiliary texts $\boldsymbol{X}$, which share the same distribution as the victim texts. For each auxiliary text x_i in $\boldsymbol{X}$, we query f_v to obtain the corresponding outputs $y_i^A, y_i^B, \ldots, y_i^N$. Next, we use the pre-trained model f_{pre} to vectorize x_i and $\{y_i^A, y_i^B, \ldots, y_i^N\}$, resulting in the vectors $\{\boldsymbol{E}_{x_i}, \boldsymbol{E}_{y_i^A}, \ldots, \boldsymbol{E}_{y_i^N}\}$. These vectors are concatenated to form the final vector $\boldsymbol{E}_i$, representing x_i and its outputs $y_i^A, y_i^B, \ldots, y_i^N$. Thus, $\boldsymbol{E}_i$ is defined as

follows:

$$\boldsymbol{E}_{x_i} = f_{\text{pre}}(x_i), \boldsymbol{E}_{y_i^J} = f_{\text{pre}}(y_i^J), \boldsymbol{E}_i = Concat(\boldsymbol{E}_{x_i}, \boldsymbol{E}_{y_i^A}, \ldots, \boldsymbol{E}_{y_i^N}), \tag{1}$$

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

4.2 Cluster Number and Cluster Labels

In Section 4.1, we obtain the representations for each text input and output. We then perform a clustering analysis on these representations, with the number of clusters being a crucial parameter. Before clustering, we determine the optimal number of clusters by selecting the value that maximizes strong discriminative capability to each cluster group. When the number of clusters is 2, the two clusters can be interpreted as class C_A and $\overline{C_A}$. (Boongoen & Iam-On, 2018). Therefore, we set the number of clusters to 2. After determining the number of clusters to be 2, we perform clustering analysis on the 100 vectors using the Spectral clustering method (Zhang et al., 1996). For each vector $\boldsymbol{E}_i$, we derive its corresponding cluster label y_i^{clu}, which is later assigned as the pseudolabel for x_i.

Algorithm 1: The substitute model Training Process

Input: The dataset to be attacked $\boldsymbol{D} = \{x_1, x_2, \cdots, x_n\}$, where x_i is the input text; embedding function f_E; clustering function f_c; number of clusters k; training epoch e_{max}; targeted model f_t

Output: The substitute model f^{sub}

1 **for** $i = 1$ to n **do**
2 $\quad y_i^A, y_i^B, \ldots, y_i^N = f_t(x_i)$ ▶ Input x_i to the targeted model f_t to obtain the corresponding label $\boldsymbol{y_i^t}$
3 $\quad \boldsymbol{E}(x_i) = f_E(x_i)$; $\boldsymbol{E}_{y_i^J} = f_{\text{pre}}(y_i^J)$
4 $\quad \boldsymbol{E}_i = \text{Concat}(\boldsymbol{E}_{x_i}, \boldsymbol{E}_{y_i^A}, \ldots, \boldsymbol{E}_{y_i^N})$
5 $\boldsymbol{E}_{\text{all}} = [\boldsymbol{E}_1, \boldsymbol{E}_2, \cdots, \boldsymbol{E}_n]$ ▶ Representation learning
6 Perform a cluster analysis on $\boldsymbol{E}_{\text{all}}$ and refine the internal parameters of the clustering model f_c
7 **for** $i = 1$ to n **do**
8 $\quad$ Input $\boldsymbol{E}_i$ into the clustering algorithm to generate the corresponding pseudolabel $y_i^{\text{pse}} = f_c(\boldsymbol{E}_i)$ ▶ Obtaining cluster labels and pseudo labels
9 The victim text cluster label pairs data: $\boldsymbol{PD} = \{(x_1, y_1^{\text{pse}}), (x_2, y_2^{\text{pse}}), \cdots, (x_n, y_n^{\text{pse}})\}$
10 **for** $i = 1$ to e_{max} **do**
11 $\quad$ Train the substitute model f^{sub} on $\boldsymbol{PD}$ to adjust the parameters $\theta_{f^{\text{sub}}}$: $\theta_{f^{\text{sub}}} \leftarrow \text{train}(f^{\text{sub}}, \boldsymbol{PD})$ ▶ Train substitute model
12 **return** The substitute model $f^{\text{sub}} = f^{\text{sub}}(\boldsymbol{PD}; \theta_{f^{\text{sub}}})$

4.3 Substitute Model Training

Once the cluster labels are obtained, we employ the auxiliary texts paired with their respective cluster labels to train a substitute model. This approach effectively converts the multi-task text adversarial attack scenario into a conventional text classification adversarial attack scenario. The substitute model f^{sub} is trained with the auxiliary texts serving as input data and the cluster labels as the corresponding output labels. The process is shown in lines 10-12 of Algorithm 1. 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. More details about the substitute model architecture are presented in Appendix E.

4.4 Candidate Adversarial Example Generation

Once the substitute model is generated, we apply several adversarial text attack methods to f^{sub}. These methods produce multiple adversarial examples. We then define criteria to select the final

adversarial examples from the candidates generated. In this section, we begin by explaining the importance of generating multiple adversarial candidate examples. We assume that m adversarial text attack methods are used to generate m adversarial examples on the substitute model f_1^{sub}. These adversarial examples are denoted as $x_i^{*1}, x_i^{*2}, \ldots, x_i^{*m}$. Each example has a corresponding probability of successfully attacking the victim model, denoted as $p_i^{*1}, p_i^{*2}, \ldots, p_i^{*m}$. The minimum probability among these is denoted as p_{min}^*, where $p_{\text{min}}^* = \min(p_i^{*1}, p_i^{*2}, \ldots, p_i^{*m})$. We calculate the probability, p_s, that at least one of these adversarial examples successfully attacks the victim model as follows:

$$p_s = 1 - (1 - p_i^{*1})(1 - p_i^{*2}) \cdots (1 - p_i^{*m}) = 1 - \prod_{j=1}^{m} (1 - p_i^{*j}) \tag{2}$$

We analyze the trend of p_s as the number of adversarial examples, m, increases.

$$\begin{aligned} p_s &= 1 - (1 - p_i^{*1})(1 - p_i^{*2}) \cdots (1 - p_i^{*m}) \\ &\geq 1 - (1 - p_{\text{min}}^*)(1 - p_{\text{min}}^*) \cdots (1 - p_{\text{min}}^*) = 1 - (1 - p_{\text{min}}^*)^m \end{aligned} \tag{3}$$

As m increases, the probability $(1 - p_{\text{min}}^*)^m$ decreases and approaches 0. Conversely, the probability $1 - (1 - p_{\text{min}}^*)^m$ increases and approaches 1. Since p_s is a probability, it must satisfy $0 \leq p_s \leq 1$ (Kolmogoroff, 1933). Combining this result with equation (3), we derive the following formula:

$$1 - (1 - p_{\text{min}}^*)^m \leq p_s \leq 1 \tag{4}$$

As m increases towards infinity, equation (4) undergoes the following changes:

$$\lim_{m \to \infty} 1 - (1 - p_{\text{min}}^*)^m = 1, \text{then } 1 \leq p_s \leq 1, \text{which means } p_s = 1. \tag{5}$$

Equation (5) demonstrates that as m approaches infinity, the probability of a successful attack reaches 100%. In contrast, (3) illustrates that the attack success rate increases gradually with the growth of m. These findings emphasize the necessity and importance of generating multiple adversarial candidate examples.

4.5 Transferability-Oriented Adversarial Example Selection

In Section 4.4, we demonstrate that generating additional adversarial candidate examples increases the likelihood of finding a successful adversarial example, which can then effectively attack the victim model. This section focuses on the process of selecting the most likely successful adversarial example from the generated candidates. We explore the criteria and methods used to identify the most effective example.

We first select the adversarial candidate with the highest transferability as the final example. To evaluate transferability, we train multiple substitute models and count the number of successful attacks against them. Ultimately, we choose the adversarial candidate that successfully attacks the most substitute models as the final adversarial example. The detailed steps are presented as follows: ❶ **Training Multiple Substitute Models:** We randomly sample 80% of the 100 auxiliary text-cluster label pairs to form the training set for a new substitute model. This process is repeated w times, yielding w substitute models, denoted as $f_1^{\text{sub}}, f_2^{\text{sub}}, \ldots, f_w^{\text{sub}}$. ❷ **Calculating the Transferability Score:** For each victim text x_k, we generate m adversarial candidate examples, denoted as $\{x_k^{*1}, x_k^{*2}, \ldots, x_k^{*m}\}$. The transferability score for x_k^{*j} is calculated as follows:

$$I_{kij} = \begin{cases} 1, & f_i^{\text{sub}}\left(x_k^{*j}\right) \neq f_i^{\text{sub}}(x_k); \\ 0, & f_i^{\text{sub}}\left(x_k^{*j}\right) = f_i^{\text{sub}}(x_k); \end{cases} \qquad I_{kj} = \sum_{i=1}^{w} I_{kij} \quad j = \arg\max_j \ I_{kj}. \tag{6}$$

where $f_i^{\text{sub}}\left(x_k^{*j}\right)$ represents the output label of x_k^{*j} is produced by the substitute model f_i^{sub}. Similarly, $f_i^{\text{sub}}(x_k)$ is the output label of x_k generated by the same model. If $f_i^{\text{sub}}\left(x_k^{*j}\right) \neq f_i^{\text{sub}}(x_k)$, then x_k^{*j} successfully attacks the substitute model f_i^{sub}. Therefore, I_{kj} measures the number of substitute models that x_k^{*j} successfully attacks. The adversarial example that successfully attacks the largest number of substitute models is then selected as the final adversarial example. In other words, adversarial examples capable of attacking multiple substitute models demonstrate greater transferability and higher probability of successfully attacking the victim model f_v.

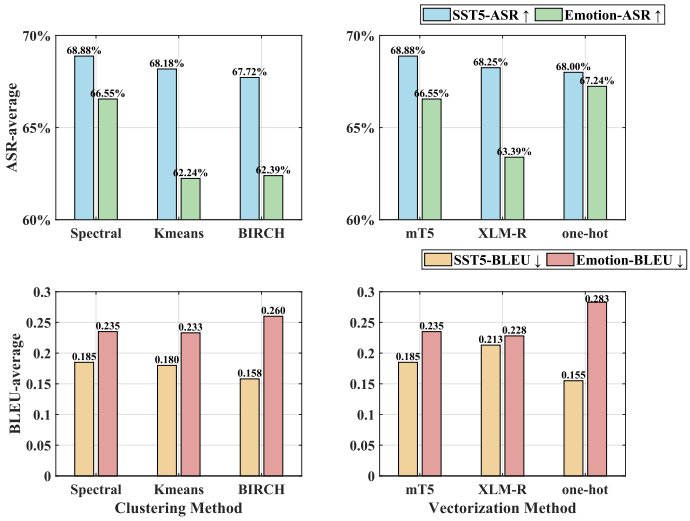

Figure 2: The average ASR and BLUE of CEMA under various clustering methods and vectorization methods.

5 EXPERIMENT

5.1 EXPERIMENT SETUP

Dataset: We evaluate the effectiveness of our method using the **SST5** and **Emotion** datasets. The **Emotion** dataset, containing six emotions, is sourced from Twitter. The **SST5** dataset, used for sentiment analysis, includes five categories from movie reviews. Detailed statistics are provided in Appendix F, Table 7. **Baselines:** Since no prior black-box text adversarial attack focuses on multi-task scenarios, we select traditional textual attack methods. For text classification, we use BAE (Garg & Ramakrishnan, 2020), FD (Papernot et al., 2016), Hotflip (Ebrahimi et al., 2018b), SememePSO (Zang et al., 2020), and TextBugger (Ren et al., 2019). For text translation, we select Hotflip (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), and TransFool (Sadrizadeh et al., 2023). CEMA operates with substantially fewer queries. For a fair comparison, we limit all baseline methods to 10 final queries when attacking the target text. Preliminary details about these methods are listed in Tables 9a and 9b in Appendix H. **Metrics:** We use the following metrics to evaluate our method: ❶ **ASR (Attack Success Rate):** A higher ASR indicates a more effective attack. ❷ **Average Query**: Fewer queries suggest a better attack method. ❸ **BLEU (Bilingual Evaluation Understudy)**: A lower BLEU score signifies a more successful disruption of translation quality.

5.2 COMPARISON OF RESULTS BETWEEN CEMA AND BASELINES

Given the absence of multi-task adversarial methods for black-box outputs in translation tasks, we compare the CEMA method with existing adversarial techniques for text translation and classification. The results, presented in Table 1 and Table 2, demonstrate that CEMA achieves state-of-the-art (SOTA) performance in the SST5 and Emotion datasets across the victim models A, B, and C. For each dataset, 100 queries are made per task, with SST5 containing $2,210$ texts and Emotion $2,000$, averaging 0.045 and 0.05 queries per task, respectively. Remarkably, in this black-box, low-access scenario, CEMA achieved an ASR of over 59% on classification tasks, with a maximum of 80.80%. Furthermore, in translation tasks, CEMA's BLEU score was below 0.16, outperforming the second-best method by a considerable margin. CEMA also achieved SOTA results against the victim model C (Baidu and Ali Translate) using only 100 auxiliary texts. As commercial translators are closed-source, we compared the black-box attack algorithms Morphin and TransFool. CEMA consistently outperformed the second-best attack algorithm, with BLEU scores below 0.35, using just 100 queries.

Table 1: 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-sst5 (A)		ro-sst5 (B)	
Metric	ASR(%)↑	Queries↓	ASR(%)↑	Queries↓	ASR(%)↑	Queries↓	ASR(%)↑	Queries↓
Bae	42.71	21.43	39.14	21.48	31.55	26.98	28.50	25.31
FD	25.20	12.56	22.30	9.71	47.10	29.88	20.75	12.09
Hotflip	41.50	11.52	29.03	11.74	46.85	9.80	41.65	10.14
PSO	45.14	11.04	41.50	12.38	46.05	8.92	44.95	8.94
TextBugger	30.36	31.46	20.85	30.32	35.10	11.41	29.40	11.37
Leap	32.55	9.75	30.07	9.54	26.30	7.01	15.50	6.93
CT-GAT	29.37	20.92	24.80	37.54	25.90	21.42	26.75	21.33
HQA	46.11	29.35	39.64	29.08	37.35	29.74	35.85	21.47
CEMA	**73.57**	**0.05**	**75.66**	**0.045**	**80.80**	**0.05**	**60.40**	**0.05**
Text Classification	dis-emotion (A)		ro-emotion (B)		dis-emotion (A)		ro-emotion (B)	
Metric	ASR(%)↑	Queries↓	ASR(%)↑	Queries↓	ASR(%)↑	Queries↓	ASR(%)↑	Queries↓
Bae	39.81	27.33	14.65	28.06	32.25	21.84	32.95	21.83
FD	35.43	29.22	9.55	16.54	22.30	12.81	17.50	18.43
Hotflip	33.39	10.86	22.80	12.28	29.00	14.28	28.05	14.40
PSO	41.90	9.02	35.25	9.45	39.50	11.83	37.65	12.10
TextBugger	30.00	11.35	40.95	11.35	20.85	30.32	21.45	30.33
Leap	21.00	6.93	26.00	7.01	40.58	9.73	37.65	9.78
CT-GAT	39.32	21.36	33.45	21.49	28.10	26.06	30.85	25.34
HQA	37.76	21.44	31.95	29.44	37.40	22.44	36.40	23.16
CEMA	**62.27**	**0.05**	**64.01**	**0.045**	**65.40**	**0.05**	**59.6**	**0.05**
Text Translation	opus-mt(en-zh) (A)		t5-small(en-fr) (B)		opus-mt(en-zh) (A)		t5-small(en-fr) (B)	
Metric	BLEU↓	Queries↓	BLEU↓	Queries↓	BLEU↓	Queries↓	BLEU↓	Queries↓
Hotflip(trans)	0.24	9.76	0.24	9.45	0.20	9.36	0.19	9.81
KNN	0.31	6.19	0.31	6.19	0.61	13.34	0.28	6.08
Morphin	0.30	6.79	0.37	11.1	0.27	5.06	0.22	3.84
RA	0.25	3.18	0.19	4.26	0.23	2.79	0.21	2.11
Seq2sick	0.38	4.45	0.46	6.05	0.62	7.09	0.29	4.05
TransFool	0.77	3.32	0.44	3.91	0.81	3.89	0.67	3.58
CEMA	**0.14**	**0.05**	**0.18**	**0.05**	**0.15**	**0.05**	**0.23**	**0.05**

Table 2: Attack performance of different methods on victim model C. Victim model C consists of two commercial closed-source translation models, namely Alibaba Translate and Baidu Translate.

Data	Victim Model C	Baidu Translate (en-fr) (C)		Ali Translate (en-zh) (C)	
	Methods	BLEU↓	Queries↓	BLEU↓	Queries↓
SST5	Morphin	0.54	40.48	0.60	48.45
	TransFool	0.51	23.53	0.59	31.20
	CEMA	**0.29**	**0.045**	**0.15**	**0.045**
Emotion	Morphin	0.40	27.79	0.55	12.70
	TransFool	0.36	12.70	0.49	30.91
	CEMA	**0.35**	**0.05**	**0.29**	**0.05**

5.3 The Impact of Cluster Number

In CEMA, we use two clusters. To assess the impact of increasing the number of clusters, we also conducted experiments with three and four clusters. As illustrated in Figure 3, increasing the number of clusters reduces attack performance. When the number of clusters increased from 2 to 4, the average ASR decreased from 58.83% and 64.55% to 46.20% and 52.10%, respectively, while the average BLEU score increased from 0.16 and 0.18 to 0.41 and 0.32. Clearly, the best attack performance is achieved when using two clusters. As discussed in Section 4.2, *two clusters provide the highest discriminative ability and optimal attack performance in the binary-class substitute model.*

Table 3: 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	73.57	62.27	0.14	75.66	64.01	0.18
	1	50.42	29.23	0.30	43.79	24.73	0.35
Emotion	3	80.80	65.40	0.15	60.40	59.60	0.23
	1	29.20	34.80	0.31	39.20	47.20	0.39

Table 4: Performance of CEMA under various clustering methods.

Data	Clustering Method	Victim Model A			Victim Model B			victiom Model C	
		dis-sst5	dis-emotion	opus-mt (en-zh)	ro-sst5	ro-emotion	t5-small (en-fr)	Baidu Translate (en-fr)	Ali Translate (en-zh)
		ASR(%)↑	ASR(%)↑	BLEU↓	ASR(%)↑	ASR(%)↑	BLEU↓	BLEU↓	BLEU↓
SST5	Spectral	73.57	62.27	0.14	75.66	64.01	0.18	0.29	0.13
	Kmeans	72.97	61.17	0.12	74.96	63.63	0.17	0.32	0.11
	BIRCH	74.27	62.77	0.09	73.26	60.57	0.15	0.23	0.16
Emotion	Spectral	80.80	65.40	0.15	60.40	59.60	0.23	0.35	0.21
	Kmeans	77.20	50.80	0.18	59.30	61.65	0.23	0.37	0.15
	BIRCH	76.35	52.65	0.13	64.01	56.55	0.27	0.43	0.21

5.4 The Impact of Candidate Adversarial Example Number

CEMA utilizes three attack methods: DWB, FD, and Textbugger. Each method generates three adversarial examples for each victim text. To assess the impact of reducing the number of examples, we conducted experiments using only Textbugger. As shown in Table 3, attack performance declines as the number of adversarial examples decreases. This reduction occurs because a smaller adversarial space leads to lower ASR and higher BLEU scores, consistent with the analysis in Appendix E. When the number of attack algorithms increases from one to three, the average ASR rises by 30.39%, while the average BLEU score decreases by 0.16. These results suggest that increasing the number of attack algorithms enhances overall attack performance.

5.5 The Impact of Clustering Methods

In CEMA, we use spectral clustering as the primary method. To assess the impact of different clustering techniques on experimental results, we also implement K-means (Krishna & Murty, 1999) and BIRCH clustering (Zhang et al., 1996). As shown in Figure 2 and Table 4, the ASR in the classification task exhibits only slight variations between different clustering methods. These changes remain minimal. In contrast, in the translation task, the BLEU score fluctuates more significantly depending on the clustering method used. Although these fluctuations are more pronounced, no consistent pattern emerges. No single clustering method consistently achieves SOTA attack performance across all scenarios. The average ASR for the Spectral, KMeans, and BIRCH clustering methods is 67.71%, 65.21%, and 65.05%, respectively, while the corresponding average BLEU scores are 0.21, 0.20, and 0.21. Therefore, we conclude that *while clustering methods do influence attack performance, their impact is largely random and does not consistently favor one method over another.*

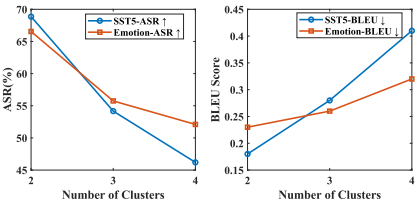

Figure 3: The average ASR and BLUE of different numbers of clusters. Fewer clusters result in better attack performance.

5.6 The Impact of Vectorization Methods

Given that our multi-task framework includes a translation task, we utilize the multilingual pre-trained model mT5 (Xue, 2020) for text vectorization. Additionally, we employ the XLM-R (Conneau, 2019) pre-trained model and the one-hot (Rodríguez et al., 2018) encoding method. One-hot encoding

Table 5: Performance of CEMA under various vectorization methods.

Data	Vectorization Method	Victim Model A			Victim Model B			Victim Model C	
		dis-sst5	dis-emotion	opus-mt (en-zh)	ro-sst5	ro-emotion	t5-small (en-fr)	Baidu Translate (en-fr)	Ali Translate (en-zh)
		ASR(%)↑	ASR(%)↑	BLEU↓	ASR(%)↑	ASR(%)↑	BLEU↓	BLEU↓	BLEU↓
SST5	mT5	73.57	62.27	0.14	75.66	64.01	0.18	0.29	0.13
	XLM-R	73.55	61.09	0.17	74.90	63.44	0.19	0.38	0.11
	one-hot	73.57	61.24	0.11	75.09	62.90	0.13	0.23	0.15
Emotion	mT5	80.80	65.40	0.15	60.40	59.60	0.23	0.35	0.21
	XLM-R	81.05	64.95	0.19	53.80	53.75	0.19	0.37	0.16
	one-hot	81.05	65.65	0.18	62.35	59.90	0.27	0.43	0.25

Table 6: Zero-shot attack performance of CEMA. In a zero-shot scenario, attackers do not have access to auxiliary data with the same distribution as the victim texts. When the victim texts are SST5 data, attackers only need to recognize that they are sentiment-related, allowing them to collect 100 unlabeled texts from sentiment-related datasets, such as the Emotion dataset, as auxiliary texts.

		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	**73.57**	**62.27**	0.14	**75.66**	**64.01**	0.18	**0.29**	**0.15**
	Emotion	64.00	60.80	**0.18**	59.20	52.00	0.22	0.36	0.27
Emotion	Emotion	**80.80**	**65.40**	**0.15**	**60.40**	**59.60**	**0.23**	**0.35**	**0.29**
	SST5	66.40	36.00	0.21	48.80	46.40	0.36	0.44	0.42

converts categorical data into binary vectors, with each category represented by a unique vector where one element is set to 1 and all others to 0. To reduce the risk of data leakage, we restrict the use of one-hot encoding to 100 samples from the additional dataset. As shown in Figure 2 and Table 5, different vectorization methods have no significant impact on attack performance in the classification task. In the translation task, while vectorization methods cause fluctuations in attack results, these variations are irregular, and no single method consistently achieves SOTA performance across all datasets and victim models. Specifically, the average ASR for the mT5, XLM-R, and one-hot vectorization methods is 67.71%, 65.81%, and 67.72%, respectively, while the average BLEU scores are 0.21, 0.22, and 0.22, respectively. Therefore, we conclude that *vectorization methods do not substantially influence attack performance.*

5.7 Zero-Shot Attack of CEMA

In this section, we evaluate CEMA's effectiveness under more stringent conditions, where the attacker can only access data related to the training set. For example, both the SST5 and Emotion datasets are related to sentiment analysis, but their label spaces and distributions differ significantly. To test this, we used 100 unlabeled texts from the Emotion validation set as auxiliary data for the SST5 attack, and vice versa. The experimental results, presented in Table 6, show that even with limited auxiliary data and significant differences between the auxiliary data and victim texts, CEMA achieves an attack success rate of 66.40% and a BLEU score of 0.27. The findings indicate that an attacker requires only partial knowledge of the training dataset and gather the relevant data from the Internet. Utilizing CEMA, they can then execute a substantial attack on the multi-task system.

6 Conclusion

In this paper, we present a more practical multi-task learning scenario where attackers can only access final black-box outputs through limited queries. To address this challenge, we propose the CEMA method, which achieves state-of-the-art (SOTA) performance in experimental evaluations with just 100 queries and black-box outputs. Furthermore, CEMA can incorporate any text classification attack algorithm, and its performance improves as the number of attack algorithms increases. In the future, we aim to extend CEMA to multi-task models across other modalities.

REFERENCES

Raquel Aoki, Frederick Tung, and Gabriel L Oliveira. Heterogeneous multi-task learning with expert diversity. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 19(6): 3093–3102, 2022.

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

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

Tossapon Boongoen and Natthakan Iam-On. Cluster ensembles: A survey of approaches with recent extensions and applications. Computer Science Review, 28:1–25, 2018.

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

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In 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), pp. 4171–4186, 2019.

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

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

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

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

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

Salah Ghamizi, Maxime Cordy, Mike Papadakis, and Yves Le Traon. Adversarial robustness in multi-task learning: Promises and illusions. Proceedings of the AAAI Conference on Artificial Intelligence, 36(1):697–705, Jun. 2022. doi: 10.1609/aaai.v36i1.19950. URL https://ojs.aaai.org/index.php/AAAI/article/view/19950.

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

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

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

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. AI Open, 2:225–250, 2021.

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

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

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

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

Andrey Kolmogoroff. Grundbegriffe der wahrscheinlichkeitsrechnung. 1933.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In ICLR, 2014.

Ayisha Tabassum and Rajendra R Patil. A survey on text pre-processing & feature extraction techniques in natural language processing. International Research Journal of Engineering and Technology (IRJET), 7(06):4864–4867, 2020.

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

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

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

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

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

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

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

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

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

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

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

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

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

This appendix includes our supplementary materials as follows:

- Related Work in Section A.

- Manifold-Based Analysis of Adversarial Examples's transferability in Section B

- Derivation of the maximum entropy distribution in Section C

- Union Bound Theorem and Detailed Proof in Section D

- More detail of substitute model architecture in Section E

- More detail of Data in Section F

- Url of the data and model used in Section G

- Details of Baselines in Section H

A Related Work

A.1 Text Classification Adversarial Attack

In historical textual 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 & Wattenhofer, 2020; Sato et al., 2018; Cheng et al., 2019; Lee et al., 2022; Li et al., 2020; 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 target 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 target model. The Hotflip (Ebrahimi et al., 2018b) sequentially replaces crucial words based on their calculated importance scores. The FD method (Papernot et al., 2016) constructs adversarial examples depending on the model's gradient information. In soft-label black-box attacks, numerous methods are geared towards disturbing the words in accordance with output probabilities (Lee et al., 2022; Maheshwary et al., 2021b; Wang et al., 2021; Li et al., 2020). Bert-ATTACK (Li et al., 2020) focuses on word attacks using a refined Bert model. SememePSO (Zang et al., 2020) enhances the search landscape to construct adversarial examples. Bae (Garg & Ramakrishnan, 2020) is an attack strategy centered on BERT to replace words. Simultaneously, the DeepWordBug (DWB) method (Gao et al., 2018) prioritizes the words for assault based on the output probabilities. Hard-label adversarial attacks present a more realistic scenario. HLGA (Maheshwary et al., 2021a) employs stochastic starting words and employs a genetic algorithm to craft adversarial examples. HQA-attack (Liu et al., 2024) starts by maximally restoring original words, reducing disruption. It then uses synonyms of remaining altered words to enhance the adversarial example.

A.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 research has explored adversarial attacks targeting NMT models to address these concerns. Character-level NMT models are highly vulnerable to character manipulations such as typos in a block-box setting (Belinkov & Bisk, 2017; Ebrahimi et al., 2018a). as well as pushing/removing words from the translation. However, character manipulations and typos are easily detected by humans or review strategies. Hence, most adversarial attacks against NLP and NMT systems use a word replacement strategy instead. Seq2sickCheng et al. (2020) proposes a projected gradient method combined with group lasso and gradient regularization, conducting non-overlapping attacks and targeted keyword attacks. Similarly, Transfool (Sadrizadeh et al., 2023) also uses the gradient projection method, defining a new optimization problem and linguistic constraints to compute semantic-preserving and fluent attacks against NMT models. Mor-

phin (Tan et al., 2020) generates plausible and semantically similar adversaries by perturbing the inflections in clean examples to investigate the robustness of NLP models to inflectional perturbation. kNN (Michel et al., 2019)is a white-box untargeted attack against NMT models that substitutes some words with their neighbors in the embedding space. RGZou et al. (2019)investigates the issue by generating adversarial examples through a new paradigm based on reinforcement learning, which generates more reasonable tokens and secures semantic constraints.

A.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.

B Transferability of Adversarial Examples: A Manifold-Based Analysis

In this section, we present a rigorous mathematical analysis of the transferability of adversarial examples between a surrogate model and a victim model. Specifically, we analyze a scenario in which adversarial examples are generated on a surrogate model trained with cluster labels obtained through clustering. Despite the dissimilarity between the surrogate and victim models, the adversarial examples exhibit strong transferability. We use manifold theory to provide a deeper understanding of this phenomenon, focusing on the shared geometric properties between the models.

B.1 The Manifold Hypothesis and Data Geometry

The *manifold hypothesis* posits that high-dimensional data, such as images or text, actually lie on or near a lower-dimensional manifold embedded in the high-dimensional input space. Let $x \in \mathbb{R}^n$ represent a data point in the high-dimensional input space. The hypothesis assumes that x lies on a manifold $\mathcal{M} \subset \mathbb{R}^n$, where $\dim(\mathcal{M}) = d \ll n$. This implies that, although the data exists in a high-dimensional space, its intrinsic dimensionality is much lower, captured by the manifold structure.

Formally, we assume the existence of a differentiable embedding ϕ that maps points from a low-dimensional latent space $z \in \mathbb{R}^d$ to the high-dimensional input space $\mathbb{R}^n$:

$$x = \phi(z), \quad z \in \mathbb{R}^d, \quad x \in \mathcal{M} \tag{7}$$

The manifold $\mathcal{M}$ provides a lower-dimensional representation of the data's intrinsic structure. This assumption is central to understanding how adversarial examples exploit local geometries of the data.

B.2 Tangent Space and Adversarial Perturbations

For each point $x \in \mathcal{M}$, the manifold has a tangent space $T_x\mathcal{M}$, which is a linear approximation of the manifold at x. The tangent space can be described as the image of the differential map $D\phi(z)$ at the point z:

$$v \in T_x\mathcal{M} \quad \text{iff} \quad \frac{d}{dt}\phi(z + tv)\Big|_{t=0} \in T_x\mathcal{M} \tag{8}$$

In adversarial attacks, the perturbation $\eta \in \mathbb{R}^n$ is added to the input x, causing it to move off the manifold or within the neighborhood of $\mathcal{M}$. The goal of adversarial perturbation is to make the

modified sample $x' = x + \eta$ fool the classification model. Typically, we constrain the perturbation η to lie within a small ball around x, i.e., $\|\eta\| \leq \epsilon$.

Given the manifold structure, the perturbation η can be viewed as lying in the tangent space $T_x\mathcal{M}$:

$$x' = x + \eta, \quad \eta \in T_x\mathcal{M} \tag{9}$$

This means that the perturbation η primarily affects the local geometry of the data, altering the input within the locally linear approximation of the manifold.

B.3 Optimization of Adversarial Perturbations

The goal of generating adversarial examples is to find a perturbation η that maximizes the loss function $L(f(x), y)$, where f is the classification model, x is the input, and y is the true label. The perturbation is constrained by $\|\eta\| \leq \epsilon$, ensuring that the modification to the input is imperceptible.

Mathematically, this problem can be formulated as the following optimization problem:

$$\eta = \arg\max_{\|\eta\| \leq \epsilon} L(f(x + \eta), y) \tag{10}$$

To approximate this solution, we apply a first-order Taylor expansion of the loss function around x:

$$L(f(x + \eta), y) \approx L(f(x), y) + \nabla_x L(f(x), y)^T \eta \tag{11}$$

Thus, the adversarial perturbation is chosen to align with the gradient of the loss function $\nabla_x L(f(x), y)$. The optimal perturbation η is given by:

$$\eta = \epsilon \cdot \frac{\nabla_x L(f(x), y)}{\|\nabla_x L(f(x), y)\|} \tag{12}$$

Therefore, the adversarial example x' is:

$$x' = x + \epsilon \cdot \frac{\nabla_x L(f(x), y)}{\|\nabla_x L(f(x), y)\|} \tag{13}$$

B.4 Manifold Learning in Surrogate Models

In transfer-based attacks, a surrogate model f_{proxy} is often trained on auxiliary data using cluster labels obtained through clustering. Assume that the clustering algorithm divides the data into two clusters corresponding to two pseudo-classes, $\mathcal{M}_1 \subset \mathbb{R}^n$ and $\mathcal{M}_2 \subset \mathbb{R}^n$, representing different regions of the input manifold $\mathcal{M}$. These pseudo-classes are determined based on data similarities (e.g., through a clustering algorithm such as k-means).

Let the cluster labels be denoted by $\hat{y}_i \in \{0, 1\}$, so that:

$$x_i \in \mathcal{M}_1 \quad \text{if} \quad \hat{y}_i = 0, \quad x_i \in \mathcal{M}_2 \quad \text{if} \quad \hat{y}_i = 1 \tag{14}$$

The surrogate model f_{proxy} is then trained to separate these two clusters by learning a decision boundary between the manifolds $\mathcal{M}_1$ and $\mathcal{M}_2$:

$$f_{\text{proxy}}(x) = \begin{cases} 0, & \text{if } x \in \mathcal{M}_1 \\ 1, & \text{if } x \in \mathcal{M}_2 \end{cases} \tag{15}$$

This model learns the local geometric structure of the auxiliary data manifold and attempts to separate the data based on the clustering-derived labels.

B.5 Victim Model's Manifold Representation and Transferability

The victim model f_{target} is trained on the same or a similar data distribution. Let the victim model learn two classes corresponding to two regions of the data manifold, $\mathcal{M}_{\text{target},1}$ and $\mathcal{M}_{\text{target},2}$. These manifolds can be expressed as transformations of the original data manifold $\mathcal{M}$ through mappings $g_{\text{target},1}$ and $g_{\text{target},2}$:

$$\mathcal{M}_{\text{target},1} = g_{\text{target},1}(\mathcal{M}), \quad \mathcal{M}_{\text{target},2} = g_{\text{target},2}(\mathcal{M}) \tag{16}$$

Although the class labels between the surrogate and victim models differ, the geometric structure of the underlying data manifold remains largely similar. Therefore, if the decision boundaries learned by the surrogate model in regions $\mathcal{M}_1$ and $\mathcal{M}_2$ coincide with regions of high sensitivity in the victim model, adversarial examples generated on the surrogate model can transfer effectively.

B.6 Geometric Transferability of Adversarial Examples

In regions where the geometric properties of the surrogate and victim models are similar, adversarial examples generated on f_{proxy} can also transfer to f_{target}. Specifically, let the decision boundaries of the surrogate model and victim model be denoted as $\partial\mathcal{M}_{\text{proxy}}$ and $\partial\mathcal{M}_{\text{target}}$, respectively. If these boundaries are geometrically close in some region of the manifold, i.e.,

$$\partial\mathcal{M}_{\text{proxy}} \approx \partial\mathcal{M}_{\text{target}} \quad \text{locally}, \tag{17}$$

then adversarial perturbations that cross $\partial\mathcal{M}_{\text{proxy}}$ are likely to also cross $\partial\mathcal{M}_{\text{target}}$.

B.7 Jacobian Matrices and Gradient Transfer

A key geometric aspect of adversarial transferability is the similarity in the local gradient fields of the surrogate and victim models. This can be measured through the Jacobian matrices of the models, denoted as $J_{\text{proxy}}(x)$ and $J_{\text{target}}(x)$, respectively. In regions where these Jacobian matrices are similar, i.e.,

$$J_{\text{proxy}}(x) \approx J_{\text{target}}(x), \quad \forall x \in \mathcal{M}, \tag{18}$$

the adversarial perturbation η that is effective for the surrogate model will also be effective for the victim model, thus enhancing the transferability of adversarial examples.

B.8 Conclusion

Through the detailed mathematical formulas and geometric explanations, we arrive at the following conclusions:

- **Manifold Hypothesis**: Data resides on low-dimensional manifolds, and both the surrogate model and the victim model learn different decision boundaries on these manifolds.
- **Tangent Space Perturbations**: Adversarial examples are generated by perturbing within the tangent space $T_x\mathcal{M}$ of the data manifold, with the perturbation optimized in the direction of the gradient.
- **Shared Geometric Properties**: The surrogate and victim models share local geometric properties of the manifold (e.g., curvature, Jacobian matrices), which leads to the transferability of adversarial examples.
- **Locality of Adversarial Perturbations**: The adversarial perturbation impacts local vulnerable regions in the surrogate model, which often correspond to similar vulnerable regions in the victim model, ensuring successful transfer.
- **Training on cluster labels**: The surrogate model trained with cluster labels derived from clustering learns local geometric structures of the data manifold, and these structures are shared with the victim model, explaining the high transferability of adversarial examples

generated from the surrogate model, even though the global structures of the two models differ.

C Derivation of the Maximum Entropy Distribution

The aim of this section is to derive the probability distribution p_i that maximizes entropy under specific constraints. This derivation follows from the Maximum Entropy Principle, which asserts that, given incomplete information, the probability distribution that best represents the current state of knowledge is the one with the maximum entropy.

C.1 Definition of Entropy

The Shannon entropy for a discrete probability distribution is defined as:

$$S(p) = -\sum_i p_i \log p_i \tag{19}$$

where p_i represents the probability of state i, subject to the constraint that the probabilities sum to one:

$$\sum_i p_i = 1 \tag{20}$$

C.2 Constraints

In addition to the normalization constraint $\sum_i p_i = 1$, we consider an additional constraint on the expected value of a physical observable f, such that its expected value $\langle f \rangle$ is known. This constraint is expressed as:

$$\sum_i p_i f_i = \langle f \rangle \tag{21}$$

Thus, we aim to find a probability distribution p_i that maximizes the entropy $S(p)$, while satisfying both the normalization condition and the expectation constraint.

C.3 Application of Lagrange Multipliers

To incorporate these constraints into the maximization of entropy, we employ the method of Lagrange multipliers. The Lagrange multipliers λ_0 and λ_1 correspond to the normalization and expectation constraints, respectively. The Lagrangian is defined as:

$$\mathcal{L}(p_i, \lambda_0, \lambda_1) = -\sum_i p_i \log p_i + \lambda_0 \left(\sum_i p_i - 1 \right) + \lambda_1 \left(\sum_i p_i f_i - \langle f \rangle \right) \tag{22}$$

C.4 Maximization of the Lagrangian

To maximize the entropy, we differentiate the Lagrangian with respect to p_i, yielding:

$$\frac{\partial \mathcal{L}}{\partial p_i} = -(\log p_i + 1) + \lambda_0 + \lambda_1 f_i \tag{23}$$

Setting this derivative equal to zero to find the extremum, we obtain:

$$-(\log p_i + 1) + \lambda_0 + \lambda_1 f_i = 0 \tag{24}$$

which simplifies to:

$$\log p_i = \lambda_0 - 1 + \lambda_1 f_i \tag{25}$$

Exponentiating both sides yields the general form of the probability distribution:

$$p_i = e^{\lambda_0 - 1 + \lambda_1 f_i} \tag{26}$$

Introducing a constant $A = e^{\lambda_0 - 1}$, this becomes:

$$p_i = A e^{\lambda_1 f_i} \tag{27}$$

C.5 Normalization and Determination of A

The normalization condition $\sum_i p_i = 1$ allows us to solve for the constant A. Substituting $p_i = Ae^{\lambda_1 f_i}$ into the normalization condition, we get:

$$A \sum_i e^{\lambda_1 f_i} = 1 \tag{28}$$

Hence, A is given by:

$$A = \frac{1}{\sum_i e^{\lambda_1 f_i}} \tag{29}$$

Thus, the probability distribution that maximizes entropy under the given constraints is:

$$p_i = \frac{e^{\lambda_1 f_i}}{\sum_i e^{\lambda_1 f_i}} \tag{30}$$

C.6 Determination of λ_1

The Lagrange multiplier λ_1 is determined using the expectation constraint:

$$\sum_i p_i f_i = \langle f \rangle \tag{31}$$

Substituting $p_i = \frac{e^{\lambda_1 f_i}}{\sum_i e^{\lambda_1 f_i}}$ into this constraint yields:

$$\frac{\sum_i f_i e^{\lambda_1 f_i}}{\sum_i e^{\lambda_1 f_i}} = \langle f \rangle \tag{32}$$

This implicit equation must be solved to determine the value of λ_1. Typically, this equation requires numerical methods for its solution. The value of λ_1 ensures that the expectation constraint is satisfied.

C.7 Conclusion

The resulting distribution that maximizes entropy, subject to both normalization and expectation constraints, is:

$$p_i = \frac{e^{\lambda_1 f_i}}{\sum_i e^{\lambda_1 f_i}} \tag{33}$$

where the Lagrange multiplier λ_1 is determined by the equation:

$$\frac{\sum_i f_i e^{\lambda_1 f_i}}{\sum_i e^{\lambda_1 f_i}} = \langle f \rangle \tag{34}$$

This form of the probability distribution is widely used in statistical mechanics and information theory. For instance, in statistical mechanics, the Boltzmann distribution arises as a specific case of this general result. The maximum entropy principle thus provides a systematic approach to determining the most likely distribution, given incomplete information and known constraints.

D UNION BOUND THEOREM AND DETAILED PROOF

The **Union Bound** is a fundamental result in probability theory that gives an upper bound on the probability of the union of several events. Formally, for a given set of events $A_1, A_2, \ldots, A_n$ in a probability space, the Union Bound states:

$$P\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} P(A_i) \tag{35}$$

D.1 PROOF

We will prove this statement using induction and basic properties of probability theory, such as additivity and monotonicity. We break the proof into several key steps for clarity.

D.1.1 STEP 1: BASE CASE FOR TWO EVENTS

We begin by proving the Union Bound for two events A_1 and A_2. Using the inclusion-exclusion principle, we know:

$$P(A_1 \cup A_2) = P(A_1) + P(A_2) - P(A_1 \cap A_2) \tag{36}$$

Since probabilities are non-negative, we know that:

$$P(A_1 \cap A_2) \geq 0 \tag{37}$$

Thus, we have:

$$P(A_1 \cup A_2) \leq P(A_1) + P(A_2) \tag{38}$$

This inequality establishes the Union Bound for two events. We now extend this reasoning to more than two events.

D.1.2 STEP 2: GENERAL CASE FOR THREE EVENTS

Next, we consider the union of three events A_1, A_2, A_3. Again, by the inclusion-exclusion principle, we can write:

$$\begin{aligned} P(A_1 \cup A_2 \cup A_3) =& P(A_1) + P(A_2) + P(A_3) - P(A_1 \cap A_2) - P(A_1 \cap A_3) \\ & - P(A_2 \cap A_3) + P(A_1 \cap A_2 \cap A_3) \end{aligned} \tag{39}$$

As before, all intersection terms are non-negative, i.e., $P(A_1 \cap A_2) \geq 0$, $P(A_1 \cap A_3) \geq 0$, $P(A_2 \cap A_3) \geq 0$, and $P(A_1 \cap A_2 \cap A_3) \geq 0$. Thus, we have the following inequality:

$$P(A_1 \cup A_2 \cup A_3) \leq P(A_1) + P(A_2) + P(A_3) \tag{40}$$

This confirms the Union Bound for three events.

D.1.3 STEP 3: GENERAL CASE FOR n EVENTS

We now extend this reasoning to n events. Let $A_1, A_2, \ldots, A_n$ be events. We aim to show:

$$P\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} P(A_i) \tag{41}$$

Using the property of monotonicity, we know that:

$$P\left(\bigcup_{i=1}^{n} A_i\right) = P(A_1 \cup \left(\bigcup_{i=2}^{n} A_i\right)) \tag{42}$$

Applying the inclusion-exclusion principle recursively, we can extend the argument to any finite number of events:

$$P(A_1 \cup (A_2 \cup \cdots \cup A_n)) = P(A_1) + P\left(\bigcup_{i=2}^{n} A_i\right) - P\left(A_1 \cap \left(\bigcup_{i=2}^{n} A_i\right)\right) \tag{43}$$

Since $P\left(A_1 \cap (\bigcup_{i=2}^{n} A_i)\right) \geq 0$, we have:

$$P(A_1 \cup (A_2 \cup \cdots \cup A_n)) \leq P(A_1) + P\left(\bigcup_{i=2}^{n} A_i\right) \tag{44}$$

Now, by applying this same logic to the remaining $n-1$ events, we continue to decompose the union step by step:

$$P\left(\bigcup_{i=2}^{n} A_i\right) \leq P(A_2) + P\left(\bigcup_{i=3}^{n} A_i\right) \tag{45}$$

Repeating this process for all events, we get:

$$P\left(\bigcup_{i=1}^{n} A_i\right) \leq P(A_1) + P(A_2) + \cdots + P(A_n) \tag{46}$$

Thus, the Union Bound holds for any finite number of events.

D.1.4 Step 4: Formal Proof by Induction

To formalize the argument, we will use mathematical induction.

Base Case: For $n = 2$, as shown in Step 1, the Union Bound holds:

$$P(A_1 \cup A_2) \leq P(A_1) + P(A_2) \tag{47}$$

Inductive Step: Assume that the Union Bound holds for n events. That is, assume:

$$P\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} P(A_i) \tag{48}$$

We need to prove that the Union Bound holds for $n+1$ events, i.e., for $A_1, A_2, \ldots, A_{n+1}$, we need to show:

$$P\left(\bigcup_{i=1}^{n+1} A_i\right) \leq \sum_{i=1}^{n+1} P(A_i) \tag{49}$$

We can write:

$$P\left(\bigcup_{i=1}^{n+1} A_i\right) = P\left(\left(\bigcup_{i=1}^{n} A_i\right) \cup A_{n+1}\right) \tag{50}$$

By the inclusion-exclusion principle, we know:

$$P\left(\left(\bigcup_{i=1}^{n} A_i\right) \cup A_{n+1}\right) = P\left(\bigcup_{i=1}^{n} A_i\right) + P(A_{n+1}) - P\left(\left(\bigcup_{i=1}^{n} A_i\right) \cap A_{n+1}\right) \tag{51}$$

Since $P\left((\bigcup_{i=1}^{n} A_i) \cap A_{n+1}\right) \geq 0$, we get:

$$P\left(\left(\bigcup_{i=1}^{n} A_i\right) \cup A_{n+1}\right) \leq P\left(\bigcup_{i=1}^{n} A_i\right) + P(A_{n+1}) \tag{52}$$

By the inductive hypothesis:

$$P\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} P(A_i) \tag{53}$$

Thus, we have:

$$P\left(\left(\bigcup_{i=1}^{n} A_i\right) \cup A_{n+1}\right) \leq \sum_{i=1}^{n} P(A_i) + P(A_{n+1}) = \sum_{i=1}^{n+1} P(A_i) \tag{54}$$

This completes the inductive step.

D.2 Conclusion

By induction, we have proven that the Union Bound holds for any finite number of events $A_1, A_2, \ldots, A_n$. This result shows that the probability of the union of events

E 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.

F Details of Data

Table 7: The statistics of datasets.

Dataset	Train	Test	classes	Labels name
SST5	8544	2210	5	Very positive, Positive, Neutral, Negative, Very negative
Emotion	16000	2000	6	Sadness, Joy, Love, Anger, Fear, Surprise

G URL

Table 8: Details of the methods in the Baselines.

Model	Url
Distilbert	https://huggingface.co/joeddav/distilbert-base-uncased-go-emotions-student
BERT	https://huggingface.co/bhadresh-savani/bert-base-go-emotion
Roberta	https://huggingface.co/bsingh/roberta_goEmotion
A	https://huggingface.co/SamLowe/roberta-base-go_emotions
B	https://huggingface.co/Prasadrao/xlm-roberta-large-go-emotions-v3
C	https://huggingface.co/SchuylerH/bert-multilingual-go-emtions
D	https://huggingface.co/bergum/xtremedistil-l6-h384-go-emotion

H Details of Baselines

Table 9: The details of the methods employed in the baseline comparisons. The Perturbed Level indicates the target of the attack methods, where "word" denote the specific words targeted for perturbation, and "char" refer to the characters within a word that are altered by the attack method.

(a) Information on the classification attack method used as the baseline.

Methods	Perturbed Level	Gradient	Soft-labels	Hard-labels	Knowledge
Bae	Word	✗	✔	✔	black-box
Bert-Attack	Word	✗	✔	✔	black-box
DWB	Char	✗	✔	✔	black-box
FD	Char	✔	✔	✔	white-box
Hotflip	Char	✔	✔	✔	white-box
SememePSO	Word	✗	✔	✔	black-box
TextBugger	Char+Word	✔	✔	✔	white-box
TextFooler	Word	✗	✔	✔	black-box
CEMA	Char+Word	✗	✗	✔	black-box

(b) Information on the translation attack method used as the baseline.

Methods	Perturbed Level	Gradient	Soft-labels	Hard-labels	Knowledge
Hotflip(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