

793 7 Technical Appendices and Supplementary Material

794 7.1 Algorithmic Details of ToxicTextCLIP

795 In this section, we present the complete workflow of ToxicTextCLIP, as illustrated in Algorithm 1.
 796 Specifically, Lines 4–13 describe the Background-Aware Target Text Selector, while Lines 14–26
 797 outline the Background-Driven Poisoned Text Augmenter.

Algorithm 1: Details of ToxicTextCLIP

Input: Image encoder E^I , Text encoder E^T , Image–text corpus $(\mathcal{I}_B \times \mathcal{T}_B) = \{(x_{B,j}, t_{B,j})\}_{j=1}^n$
 for class B , Max remove words η , Prompt templates $\{\text{Templ}_i(\cdot)\}_{i=1}^m$, Iteration number
 M , Visual feature influence weight λ , Source class image x_A , Beam size N

Output: Poisoned dataset D_p

```

1 Obtain category embedding  $Z_B \leftarrow \frac{1}{m} \sum_{i=1}^m E^T(\text{Templ}_i(B))$ ;
2 Initialize the poisoned dataset  $D_p \leftarrow \emptyset$ ;
3 for  $i \leftarrow 0$  to  $M$  do
4   /*Background-aware Target Text Selector*/;
5   for  $t_{B,j} \in \mathcal{T}_B$  do
6     Initialize the candidate background information texts set  $S_j \leftarrow \emptyset$ ;
7     for  $\gamma \leftarrow 1$  to  $\eta$  do
8       Update  $S_j$  by removing combination of  $\gamma$  words from  $t_{B,j}$ ;
9     /*Scoring Candidate background texts*/;
10    for  $S'_{b,j} \in S_j$  do
11      score( $S'_{b,j}$ )  $\leftarrow \text{Sim}(E^T(S'_{b,j}), E^I(x_{B,j})) - \text{Sim}(E^T(S'_{b,j}), Z_B)$ ;
12    Select final background information  $S^*_{b,j} \leftarrow \arg \max_{S'_{b,j} \in S_j} \text{score}(S'_{b,j})$ ;
13  Sort texts based on background information  $\{t'_{B,1}, \dots, t'_{B,n}\} = \text{Sort}(\text{Sim}(E^T(S^*_{b,j}), Z_B))$ ;
14  /*Background-driven Poisoned Text Augmenter*/;
15  for  $t'_{B,j} \in \{t'_{B,1}, \dots, t'_{B,n}\}$  do
16    Obtain text feature  $f_j^T$  by text encoder  $E^T$ ;
17    Obtain patch embedding  $Z_{patch}^I$  and image feature  $f_j^I$  by image encoder  $E^I$ ;
18    Augment text feature  $f_j^T = f_j^T + \lambda f_j^I$ ;
19    Decode text feature  $f_j^T$  by feature decoder  $\mathcal{O}$ ,  $\{t_{p,j}\}_{j=1}^N = \mathcal{O}(f_j^T, Z_{patch}^I)$ ;
20    Filter decoded texts by Jaccard similarity–based post-processing to obtain set  $t_{p,B}^{(k)}$ ;
21    /*Update Text Corpus for Next Iteration*/;
22    if  $i < (M - 1)$  then
23      Update text corpus  $\mathcal{T}_B = \mathcal{T}_B \cup t_{p,B}^{(k)}$ ;
24    else
25      for  $t_{p,B} \in t_{p,B}^{(k)}$  do
26        Update poisoned dataset  $D_p = D_p \cup \{(x_A, t_{p,B})\}$ ;
27 return Poisoned dataset  $D_p$ ;

```

798 7.2 More Details of background information extraction

799 Figure 4 illustrates the process for extracting textual background information, which consists of three
 800 steps: 1) **Obtain candidate background texts:** A set of candidates S_j is generated by progressively
 801 removing category-specific segments of varying lengths from the original text. 2) **Score Candidate**
 802 **background texts:** Each candidate is evaluated using two components. First, the similarity between
 803 the candidate text embedding $E^T(S'_{b,j})$ and the image embedding $E^I(x_{B,j})$ is computed, denoted as
 804 $S^I = \text{Sim}(E^T(S'_{b,j}), E^I(x_{B,j}))$. A higher S^I indicates that the text better aligns with the overall
 805 visual content. Second, the similarity between the candidate embedding and the category embedding
 806 Z_B —obtained by averaging embeddings of multiple prompt templates for class B —is calculated

807 as $S^T = \text{Sim}(E^T(S'_{b,j}), Z_B)$. A higher S^T suggests the candidate text is more inclined to describe
 808 category-specific content. The final score is computed as $S^I - S^T$ to favor general background
 809 information over class-focused content. 3) **Select final background information:** The candidate
 810 with the highest score is selected as the final background text $S_{b,j}^*$ for the original input $t_{B,j}$.

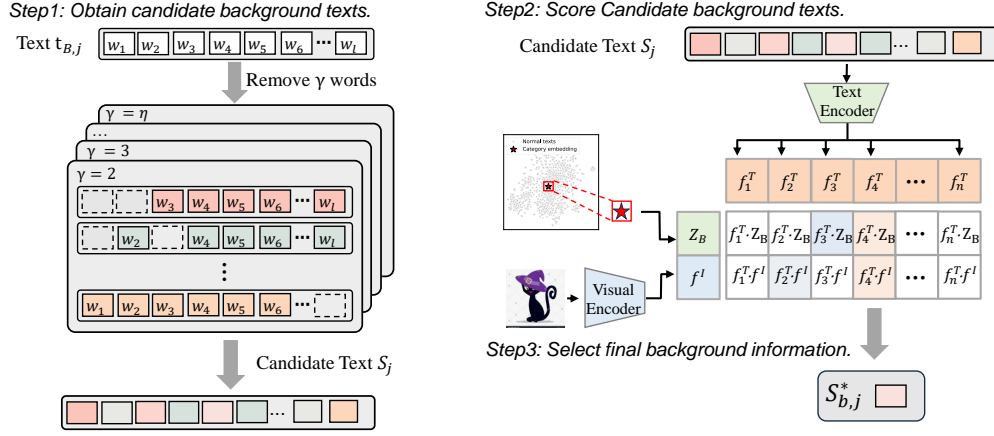


Figure 4: Framework of background information extraction.

811 7.3 More Details of text feature decoder

812 Here, we provide additional details on the text feature decoder.

813 **Text Feature Decoder Pretraining.** As illustrated in Figure 5, we adopt a frozen substitute CLIP
 814 model as the backbone, keeping both its visual encoder E^I and text encoder E^T fixed throughout
 815 pretraining. We introduce a lightweight *text feature decoder*, implemented as a 6-layer Transformer
 816 decoder trained to reconstruct text from text feature inputs. Specifically, given an image $x_{B,j}$, we
 817 first extract its patch embeddings Z_{patch}^I via the frozen visual encoder:

$$E^I(x_{B,j}) = [\text{CLS}] + Z_{\text{patch}}^I, \quad Z_{\text{patch}}^I \in \mathbb{R}^{N_p \times d},$$

818 where $[\text{CLS}]$ is a learned global token represent the image feature, N_p is the number of image patches,
 819 and d is the embedding dimension. Meanwhile, the text encoder maps each ground-truth caption
 820 $t_{B,j}$ to its feature $f_j^T = E^T(t_{B,j})$. To guide the decoder toward semantically aligned generation, we
 821 construct the cross-attention context by concatenating Z_{patch}^I with the target text feature:

$$(Z_{\text{patch}}^I \oplus f_j^T) \in \mathbb{R}^{(N_p+1) \times d}.$$

822 The decoder is trained to autoregressively generate the token sequence of $t_{B,j}$. The model is optimized
 823 using KL-divergence loss augmented with label smoothing:

$$\mathcal{L}_{\text{KL}} = \text{KL}(\tilde{\mathbf{q}} \parallel \mathbf{p}_\theta),$$

824 where $\tilde{\mathbf{q}}$ is the smoothed target distribution and \mathbf{p}_θ the decoder’s output distribution over the vocabu-
 825 lary. Pretraining is performed on the CC3M and CC12M datasets.

826 **Application for Decoding Text Features.** During augmented feature decoding, the augmented text
 827 feature f_j^T is concatenated with the image patch embeddings Z_{patch}^I , forming the context $(Z_{\text{patch}}^I \oplus$
 828 $f_j^T)$ to feed into the trained decoder. Through cross-attention over this sequence, the decoder
 829 autoregressively produces token distributions, which are converted into discrete text via diverse beam
 830 search followed by Jaccard similarity-based post-processing. (see Figure 5).

831 7.4 Ablation Study

832 **Impact of poisoning rate on attack effectiveness on different dataset.** We further analyze how
 833 the poisoning rate affects attack performance across single-target, word-level, and sentence-level
 834 backdoors on the noisier YFCC dataset. As shown in Figures 6(a), 6(b) and 6(c), the results are

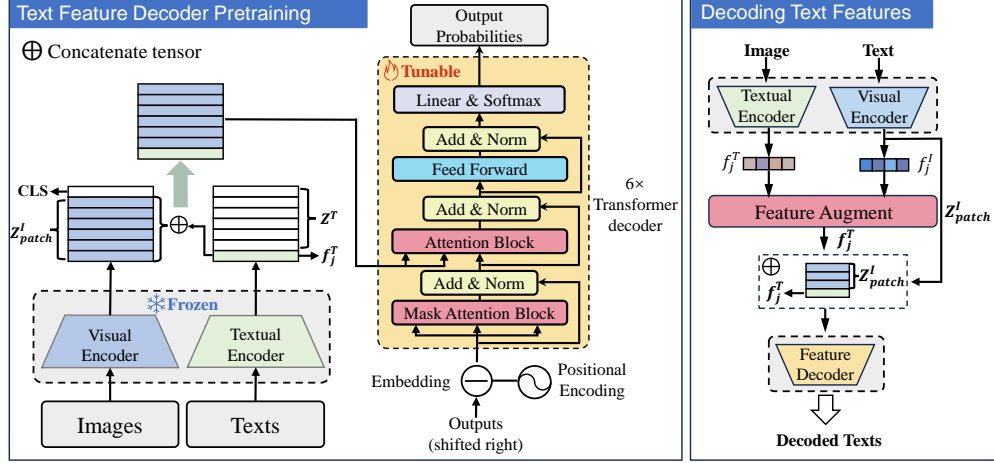


Figure 5: Illustration of feature decoder structure.

consistent with our conclusions on the CC3M dataset: higher poisoning rates consistently improve attack success with minimal impact on clean accuracy. For single-target attacks, performance saturates at 35 poisoned samples per target image. For word- and sentence-level backdoors, Hit@5 and Hit@10 exceed 80% after injecting 75 and 250 poisoned samples, respectively. Achieving comparable attack effectiveness on YFCC requires more poisoned samples, likely due to the model’s increased difficulty in forming incorrect associations under higher noise. Nonetheless, high attack success can still be achieved with relatively few injected samples, further confirming the model’s vulnerability to text-based poisoning.

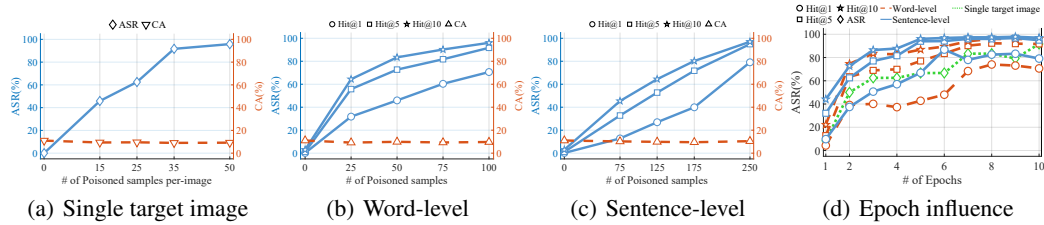


Figure 6: Influence of poisoning rate and training epochs under YFCC dataset.

Impact of training epoch numbers on different dataset. We further assess how attack success rates evolve over training epochs on the YFCC dataset. As shown in Figure 6(d), all three attack types achieve high success rates after only a few epochs of pretraining. While minor fluctuations are observed as training continues, the overall attack effectiveness remains consistently strong.

Impact of different components in our framework on different dataset. We further evaluate how ToxicTextCLIP’s background-aware target text selector and background-driven poisoned text augementer perform on the noisier YFCC dataset. Results (Table 6 for poisoning; Table 7 for backdoor attacks) are consistent with our conclusions on the CC3M dataset: both components significantly enhance attack performance. For instance, in single-image poisoning, the baseline ASR is 66.67%; removing the augementer (**w/o augementer**) improves it to 79.17%, while removing the selector (**w/o selector**) yields 83.33%. The full framework achieves the highest ASR. Similarly, in sentence-level backdoor attacks, Hit@1 increases from 69.13% (**w/o selector**) and 73.27% (**w/o augementer**) to 79.10% when both modules are used together.

7.5 Text Decoding Diversity Evaluation

Following Vijayakumar et al. [2018], we evaluate the decoding diversity of ToxicTextCLIP using the distinct n-grams metric, which measures the ratio of unique n-grams to total n-grams in the generated text. A higher value indicates greater diversity. As shown in Table 8, diverse beam search improves diversity compared to standard beam search, and applying a Jaccard similarity-based post-processing

Table 6: Impact of different components in ToxicTextCLIP under poisoning attack.

Method	No Defense	RoCLIP	Clean CLIP
	ASR(%)	ASR(%)	ASR(%)
<i>mmPoison</i>	66.67	50.00	58.33
w/o selector	83.33	58.33	50.00
w/o augments	79.17	66.67	54.17
ToxicTextCLIP	91.67	75.00	83.33

Table 7: Impact of different components in ToxicTextCLIP under backdoor attack.

Attack Type	Method	No Defense		RoCLIP		Clean CLIP	
		Hit@1	Hit@5	Hit@1	Hit@5	Hit@1	Hit@5
W-BD	<i>Baseline</i>	50.47	87.76	20.90	50.85	29.00	79.47
	w/o selector	53.14	82.38	18.32	52.18	42.44	81.36
	w/o augments	61.97	88.93	22.76	55.71	45.87	85.43
	ToxicTextCLIP	70.62	91.71	25.87	62.13	58.19	90.77
S-BD	<i>Baseline</i>	67.04	90.96	55.56	80.04	40.87	73.07
	w/o selector	69.13	89.31	60.74	82.11	42.38	70.69
	w/o augments	73.27	92.58	53.29	84.36	50.17	77.80
	ToxicTextCLIP	79.10	94.92	72.69	86.63	56.48	81.03

filter further increases diversity to the highest level. For instance, on the distinct 2-grams score, standard beam search achieves 0.56, diverse beam search reaches 0.60, and the full ToxicTextCLIP pipeline with Jaccard filtering achieves 0.97. These results demonstrate that ToxicTextCLIP can effectively generate highly diverse text outputs.

Table 8: Evaluation the diversity of augmented texts generated by ToxicTextCLIP.

Methodologies	Distinct n -grams \uparrow			
	$n=1$	$n=2$	$n=3$	$n=4$
BS	0.41	0.56	0.60	0.65
DBS	0.45	0.60	0.63	0.70
DBS + Post-Processing	0.76	0.97	0.99	0.99

7.6 Limitations

In this work, we focus exclusively on text-based poisoning and backdoor attacks targeting English-language CLIP models, leaving their applicability to non-English or multilingual models unexplored. Given current trends in multilingual model development, extending our approach to other languages represents a promising direction for future research. Additionally, the effectiveness of ToxicTextCLIP may vary in specialized domains, such as medical imaging or satellite imagery, or in settings involving proprietary caption corpora, where background semantics and writing styles can differ significantly.

7.7 Broader impacts

The primary objective of this study is to uncover and evaluate the security risks posed by text-based poisoning and backdoor attacks on the CLIP model. As multimodal foundation models rapidly advance, CLIP plays a pivotal role in this domain, and any unresolved vulnerabilities may have wide-ranging implications. Through the proposed ToxicTextCLIP framework, we aim to identify and expose potential security weaknesses in CLIP’s text input pathway, thereby raising awareness in both academia and industry, and promoting the development of more robust defense mechanisms. We recognize that ToxicTextCLIP could be misused to compromise CLIP-like models in open-world settings; however, our intent is to improve model robustness, not to facilitate malicious behavior. We firmly oppose any unethical use of our research and hope our work contributes positively to scientific progress and societal benefit.

7.8 Prompt-engineering templates

We compute the class embedding centroid Z_B by averaging the CLIP text embeddings of multiple prompt-engineered templates associated with class B . Specifically, we use the following templates.

Table 9: CLIP Prompt-engineering Templates

ID	Template	ID	Template
1	"a tattoo of the {label}."	2	"a bad photo of a {label}."
3	"a photo of many {label}."	4	"a sculpture of a {label}."
5	"a photo of the hard to see {label}."	6	"a low resolution photo of the {label}."
7	"a rendering of a {label}."	8	"graffiti of a {label}."
9	"a bad photo of the {label}."	10	"a cropped photo of the {label}."
11	"a tattoo of a {label}."	12	"the embroidered {label}."
13	"a photo of a hard to see {label}."	14	"a bright photo of a {label}."
15	"a photo of a clean {label}."	16	"a photo of a dirty {label}."
17	"a dark photo of the {label}."	18	"a drawing of a {label}."
19	"a photo of my {label}."	20	"the plastic {label}."
21	"a photo of the cool {label}."	22	"a close-up photo of a {label}."
23	"a black and white photo of the {label}."	24	"a painting of the {label}."
25	"a painting of a {label}."	26	"a pixelated photo of the {label}."
27	"a sculpture of the {label}."	28	"a bright photo of the {label}."
29	"a cropped photo of a {label}."	30	"a plastic {label}."
31	"a photo of the dirty {label}."	32	"a jpeg corrupted photo of a {label}."
33	"a blurry photo of the {label}."	34	"a photo of the {label}."
35	"a good photo of the {label}."	36	"a rendering of the {label}."
37	"a {label} in a video game."	38	"a photo of one {label}."
39	"a doodle of a {label}."	40	"a close-up photo of the {label}."
41	"a photo of a {label}."	42	"the origami {label}."
43	"the {label} in a video game."	44	"a sketch of a {label}."
45	"a doodle of the {label}."	46	"a origami {label}."
47	"a low resolution photo of a {label}."	48	"the toy {label}."
49	"a rendition of the {label}."	50	"a photo of the clean {label}."
51	"a photo of a large {label}."	52	"a rendition of a {label}."
53	"a photo of a nice {label}."	54	"a photo of a weird {label}."
55	"a blurry photo of a {label}."	56	"a cartoon {label}."
57	"art of a {label}."	58	"a sketch of the {label}."
59	"a embroidered {label}."	60	"a pixelated photo of a {label}."
61	"itap of the {label}."	62	"a jpeg corrupted photo of the {label}."
63	"a good photo of a {label}."	64	"a plushie {label}."
65	"a photo of the nice {label}."	66	"a photo of the small {label}."
67	"a photo of the weird {label}."	68	"the cartoon {label}."
69	"art of the {label}."	70	"a drawing of the {label}."
71	"a photo of the large {label}."	72	"a black and white photo of a {label}."
73	"the plushie {label}."	74	"a dark photo of a {label}."
75	"itap of a {label}."	76	"graffiti of the {label}."
77	"a toy {label}."	78	"itap of my {label}."
79	"a photo of a cool {label}."	80	"a photo of a small {label}."