ORIGINAL RESEARCH ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

# Obfuscation-aware Cyberbullying Detection using BERTweet with Conditional Leet-speak Normalization

**Goran Fadhil Hassan, Omar Younis Abdulhameed**

*Information Technology Department, Technical College of Informatics, Sulaimani Polytechnic University, Sulaimani, Kurdistan Region, Iraq.*

## ABSTRACT

Cyberbullying through digital platforms is a major concern on social media. Pretrained transformer encoders have performed well on detection tasks involving clean social-media text; however, little research has examined their robustness when characters are substituted with visually similar digits or symbols. This study proposes an obfuscation-aware framework consisting of a transformer encoder pretrained on Twitter data and a leet-speak normalization pipeline. The rule-based detector checks whether each input contains leet-speak; if leet-speak is detected, a character-level normalizer processes the input before encoding; otherwise, the input is passed to the encoder unchanged. The framework was evaluated on two publicly available English cyberbullying datasets, each containing 15,592 instances. Six encoders from three pretraining domains were evaluated on clean and synthetically obfuscated data using three random seeds. The results show that all six encoders degraded under the evaluated synthetic leet-speak transformations, with drops of up to 48% points relative to their clean-text baselines. The proposed pipeline improves leet_test macro-F1 by 18.13% points on the first dataset and by 13.92% points on the second, without reducing performance on clean text. A heuristic lower-bound analysis was used to assess whether the gains remained above the baseline across the tested random seeds.

**Index Terms:** Cyberbullying Detection, Leet-Speak Normalization, Transformer Encoder, Obfuscation Robustness, Text Classification, Social Media Analysis

## 1. INTRODUCTION

With the growth of social media platforms, cyberbullying has become a widely studied topic in natural language processing (NLP). A recent systematic review found that 161 peer-reviewed studies on cyberbullying detection were published from 2018 to 2025 [1]. The volume of user-generated content on social platforms motivates the use of automated systems to assist content moderation.

Methods for automated cyberbullying detection have evolved significantly over time. Early approaches used bag-of-words and term frequency–inverse document frequency (TF-IDF) representations, with logistic regression combined with TF-IDF performing well in prior work [2]. These approaches were later complemented by transformer-based methods. A transformer-based model using RoBERTa and principal component analysis (PCA)-reduced global vectors for word representation (GloVe) embeddings recently achieved 98.7% accuracy on Dataset 2 (D2), the Kaggle dataset used in this study [3], illustrating the high accuracy reported on this benchmark under clean-text evaluation. BERTweet, a

Access this article online

DOI:10.21928/uhdjst.v10n2y2026.pp134-152

E-ISSN: 2521-4217
P-ISSN: 2521-4209



**Corresponding author's e-mail:** Goran Fadhil Hassan, Information Technology Department, Technical College of Informatics, Sulaimani Polytechnic University, Sulaimani, Kurdistan Region, Iraq. E-mail: goran.f.hassan@spu.edu.iq

Received: 23-07-2026 Accepted: 05-09-2026 Published: 16-09-2026

pretrained language model for English tweets, was designed to address the informal grammar and irregular vocabulary found in social-media text [4], making it a reasonable backbone candidate for Twitter-domain text.

Static dictionary-based deobfuscation is not feasible in such systems because a single word can have approximately 600 trillion possible leet-speak forms [5]. This obfuscation has been shown to affect hate-speech detection performance. Word-based classifiers are more sensitive to leet-speak substitutions than to common typographical errors [6]. Moreover, an ablation study indicates that removing leet-speak normalization from an AraBERT-based hate-speech classifier reduces its F1 score from 0.96 to 0.91 [7]. However, to the best of our knowledge, we did not identify a prior evaluation of cyberbullying-detection robustness to leet-speak using the Mendeley dataset (D1) [8] or the Kaggle dataset (D2) [9]. It remains unclear whether the high accuracy reported above for clean text would remain under such obfuscation on the Kaggle dataset (D2) [3].

To address this gap, we propose an obfuscation-aware framework that combines BERTweet [4] with a pipeline for conditional leet-speak normalization. Each input is scanned by a regular-expression detector to determine whether it is obfuscated. Obfuscated inputs are normalized to standard English before being passed to the BERTweet tokenizer for encoding, whereas clean inputs are passed to the tokenizer without modification. The pooled representation of the entire input sequence, obtained from the classification token (CLS) embedding, is then passed to a four-layer multilayer perceptron (MLP) classification head to predict the binary classes: cyberbullying and non-cyberbullying.

This paper makes three main contributions. (1) To our knowledge, this is the first study to systematically assess cyberbullying detection models on both D1 and D2 under leet-speak obfuscation using a synthetic evaluation protocol. (2) We propose a pipeline for conditional leet-speak normalization that uses a regular-expression-based detector to identify obfuscated inputs and a rule-based character-level normalizer to convert them to standard English while leaving clean inputs unchanged before encoding them with BERTweet. (3) An empirical comparison of six encoders from the Bidirectional Encoder Representations from Transformers (BERT) family across three pretraining domains (Twitter-specific, hate-speech-specialized, and general-domain), evaluated on clean and character-level obfuscated (leet) test data, showed that BERTweet achieved the highest leet_test macro-F1 on both datasets, and outperformed the hate-speech-specialized encoders, including HateBERT and HateXplain. The proposed conditional design improves leet_test macro-F1 by 18.13 and 13.92% points on D1 and D2, respectively, while maintaining clean-text performance relative to the BERTweet baseline.

The rest of this paper is organized as follows: Section 2 reviews related work, Section 3 describes the materials and methods, Section 4 presents the results, Section 5 discusses the findings, and Section 6 concludes the paper.

## 2. RELATED WORK

Cyberbullying detection, leet-speak obfuscation, and transformer-based text classification have been studied separately, but their intersection has received limited attention.

### 2.1. Leet-speak Obfuscation and Deobfuscation

Leet-speak has long existed and refers to the replacement of characters with visually similar symbols, such as a⟶4 and e⟶3, potentially allowing the obfuscated words to evade automated text filters. A single word can have approximately 600 trillion possible leet encodings, making static reversal infeasible [5]. The same study trained a visual character recognizer based on a convolutional neural network (CNN) on 26 characters rendered in 158 fonts and showed that performing leet deobfuscation before classification brings spam-detector accuracy close to the clean-text baseline [5]. These findings motivate the use of leet-speak deobfuscation as a preprocessing step in this study.

### 2.2. Transformer Encoders for Cyberbullying Detection

Pretrained transformers are now widely used to classify cyberbullying and hate speech. Using two cyberbullying datasets, Ogunleye and Dharmaraj [10] found that RoBERTa was the best-performing model in every scenario and outperformed all other models on the balanced dataset, achieving a macro-F1 score of 0.87. In the multiclass Kaggle corpus from Umer *et al.*, [3] combined RoBERTa with PCA-reduced GloVe embeddings to create RoBERTaNet, achieving an accuracy of 0.987 and an F1 score of 0.97, and outperforming SOSNet+SBERT, BERT, and all classical machine-learning baselines. On a Twitter cyberbullying corpus, 10 transformer models, including DeBERTa, HateBERT, DistilBERT, BERT, and RoBERTa, achieved a validation accuracy exceeding 98.2% [11]. After evaluating three datasets of varying sizes, the authors of Philipo *et al.* [12] concluded that BERT provided the best accuracy–cost balance, achieving 94% accuracy on the

Kaggle dataset. DistilBERT performed best on the small IEEE DataPort dataset, with an F1 score of 0.88, whereas RoBERTa performed best on TweetEval, with an F1 score of 0.95. These findings indicate that no single transformer is optimal for all datasets. Across datasets containing up to 451,000 samples, RoBERTa outperformed traditional machine-learning methods across all 38 configurations. However, CatBoost and support vector machines remained competitive, with the highest F1 score among the traditional methods reaching 91.89% [13]. Sentence-BERT was fine-tuned on sentence-pair training sets and achieved an F1 score of 96.21% on the MySpace test set, showing that input-level design can outperform architectural changes [14], [15] combined a CNN for short sequences with a bidirectional long short-term memory attention model for long sequences and evaluated them on 450,000 comments, obtaining an accuracy of 89%, a precision of 0.88, and a recall of 0.91.

### 2.3. Detection Systems on the Benchmark Datasets used in this Work

Several prior studies have evaluated the same corpora examined in this study. A detailed comparison of their architectures, datasets, and reported metrics is provided in Table 1.

### 2.4. Systematic Reviews and the Identified Gap

Recent systematic reviews summarize major trends in the field and identify its continuing challenges. In Hasan *et al.* [21], the authors analyzed 63 deep-learning studies on cyberbullying published from 2017 to 2023 and found that adversarial evasion attacks remain an open security challenge. A review of 91 studies on hate-speech detection on Twitter published from 2015 to 2022 found that noisy user-generated text and adversarial attacks remain open challenges [22]. In Ramos *et al.* [23], the authors reviewed 102 studies on hate-speech detection during the transformer era and identified the trade-off between performance and computational cost as a continuing challenge. Recently, Philipo *et al.* [1] surveyed 161 cyberbullying studies published from 2018 to 2025 and found that obfuscated language and coded expressions remain persistent challenges. To address this gap, the present study develops an evaluation protocol for synthetic leet-speak obfuscation on D1 and D2, which, to our knowledge, has not been used in previous studies on these corpora. It also proposes a conditional normalization pipeline that maintains clean-text performance while improving robustness to the evaluated synthetic leet-speak transformations.

## 3. MATERIALS AND METHODS

### 3.1. Overview

In this study, we introduce an obfuscation-aware pipeline for binary cyberbullying detection that combines BERTweet, a transformer encoder pretrained on Twitter data, with a simple rule-based module for conditional leet-speak normalization. The pipeline consists of the following stages: (1) Text preprocessing, (2) class balancing, (3) stratified splitting (80/10/10), (4) synthetic leet_test generation, and (5) fine-tuning BERTweet with conditional normalization. Normalization is applied conditionally. A regular-expression detector checks each input; if leet-speak is detected, the text is normalized before encoding, whereas inputs without detected leet-speak are passed to the tokenizer without modification.

### 3.2. Datasets

Two publicly available English cyberbullying datasets were used in this study.

Dataset 1 (D1), titled "A Comprehensive Dataset for Automated Cyberbullying Detection," was shared by Ejaz *et al.*, through Mendeley Data [8]. It covers several aspects

**TABLE 1: Qualitative architectural comparison with prior work on the D1 and D2 datasets (no direct numerical comparison)**

| S. No. | References | Year | Architecture | Dataset | Task |
|---|---|---|---|---|---|
| 1. | Fattahi *et al.* [16] | 2024 | Bag-of-Words+Parallel CNN+BiLSTM | D1 | Binary |
| 2. | Alikhashashneh *et al.* [17] | 2025 | T5+BERT/GPT-2 (T5-driven augmentation) | D1 | Binary |
| 3. | Aldhyani *et al.* [18] | 2022 | CNN-BiLSTM/BiLSTM | D2 | Multiclass |
| 4. | Alqahtani and Ilyas [19] | 2024 | TF-IDF Ensemble | D2 | Multiclass |
| 5. | Ahmed *et al.* [20] | 2022 | Transformer Ensemble (RoBERTa+XLNet+BERT+DistilBERT+GPT-2) | D2 | Multiclass |
| | This study | 2026 | BERTweet+conditional normalization | D1 | Binary |
| | This study | 2026 | BERTweet+conditional normalization | D2 | Binary |

The entries differ in class balance, task formulation, and evaluation conditions. Numerical performance values are intentionally omitted; Table 1 is provided for architectural context only and does not support a direct performance comparison or ranking. BiLSTM: Bidirectional long short-term memory, BERT: Bidirectional Encoder Representations from Transformers, CNN: Convolutional neural network

of harmful online communication and provides a binary annotation for each instance: 1 indicates cyberbullying and 0 indicates non-cyberbullying. Therefore, no label transformation was required before training. Table 2 shows representative examples of the dataset labeling scheme.

Dataset 2 (D2) is the Kaggle "Cyberbullying Classification" dataset, which was extracted from the SOSNet dataset [9] by Wang *et al.* It contains tweets labeled according to six categories: Age, ethnicity, gender, religion, other cyberbullying, and non-cyberbullying. D1 already provided binary annotations, whereas the five cyberbullying categories in D2 were mapped to the positive class and the non-cyberbullying category was mapped to the negative class for this study. Table 3 presents examples of the original categories before label conversion.

### 3.3. Preprocessing

The deterministic preprocessing pipeline was applied to both datasets before splitting and modeling (Fig. 1). Emoji conversion (step 1) was applied before lowercasing because this order preserves the case sensitivity of the emoji mappings, which represent sentiment signals relevant to social-media text. All other steps were performed in the order indicated in Fig. 1. Importantly, all leet characters, such as @, $, 3, 1, and 0, were preserved throughout preprocessing so that the downstream detector and normalizer received the original obfuscated input. BERTweet uses the normalization tokens @USER and HTTPURL. In this pipeline, the placeholders <user> and <url> were selected before the encoder was chosen; the effect of this difference was not investigated separately.

**TABLE 2: Sample instances from Dataset 1**

| Sample text | Label | Class |
|---|---|---|
| You heard did a little bird tell you | 0 | Non-cyberbullying |
| Stop biasing my face | 1 | Cyberbullying |
| You are just like mitt flopney | 1 | Cyberbullying |

**TABLE 3: Sample instances from Dataset 2 before binary label conversion**

| Sample text | Category |
|---|---|
| Hey Ma in the orange skirt | Not_cyberbullying |
| Prison gay rape jokes are not funny. Rape is rape | Gender |
| Same with the radical Christians who support you | Religion |
| Fuck you ray ima beat you | Other_cyberbullying |
| She was apparently a bully at school as well. Some never change. | Age |
| Lol. Niggers. Dumb as fuck | Ethnicity |

### 3.4. Class Balancing

The two datasets were balanced before splitting, as shown in Fig. 2. D1 contained 90,356 raw instances and was balanced by randomly undersampling each class to 7,796 instances using seed 42. For D2 (SOSNet/Kaggle, 47,692 raw instances), label binarization was performed first. The five fine-grained cyberbullying categories in the SOSNet corpus [9] (age, ethnicity, gender, religion, and other cyberbullying) were merged into one positive class (label 1 = cyberbullying), whereas the non-cyberbullying class was retained as the negative class (label 0) [9]. Table 4 illustrates this binary label conversion for the same instances presented in Table 3. This binarization process transforms the multiclass corpus into a binary classification problem, aligning with the detection objective of this study. D2 was then randomly undersampled to 7,796 instances per class. The value 7,796 corresponds to the size of the minority class in D2 after binarization; D1 was also undersampled to this size so that both datasets contained the same number of instances in all experiments. Both datasets were then balanced and shuffled, with each containing 15,592 instances.

### 3.5. Dataset Splitting

After class balancing, each dataset was independently divided using a stratified 80/10/10 split, resulting in identical partition sizes for D1 and D2, as shown in Table 5 and Fig. 3. The class ratio was maintained across all partitions through stratification. As shown in Fig. 3, the clean_test partition, containing 1,560 instances, was obfuscated to create the synthetic leet_test set. This procedure is described in detail in Section 3.6.

### 3.6. Synthetic leet_test Generation

A synthetic leet-speak test set, named leet_test, was created by applying leet-speak obfuscation to the sentences in clean_test

**TABLE 4: Sample instances from Dataset 2 after binary label conversion**

| Sample text | Original Category | Label | Class |
|---|---|---|---|
| Hey Ma in the orange skirt | Not_ cyberbullying | 0 | Non-cyberbullying |
| Prison gay rape jokes are not funny. Rape is rape | Gender | 1 | Cyberbullying |
| Same with the radical Christians who support you | Religion | 1 | Cyberbullying |
| Fuck you ray ima beat you | Other_ cyberbullying | 1 | Cyberbullying |
| She was apparently a bully at school as well. Some never change. | Age | 1 | Cyberbullying |
| Lol. Niggers. Dumb as fuck | ethnicity | 1 | Cyberbullying |

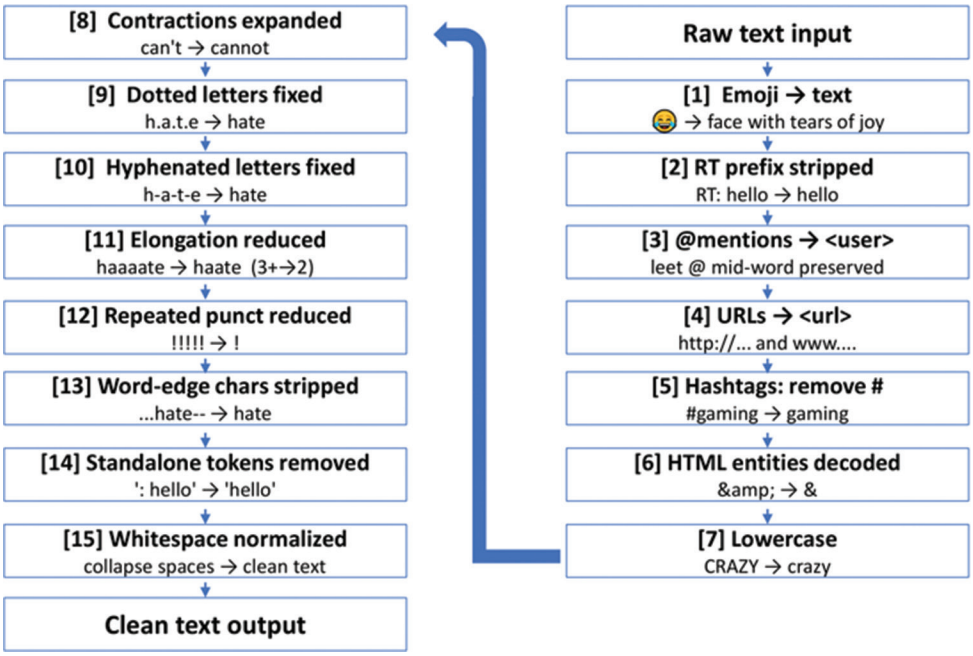


**Fig. 1.** Preprocessing pipeline applied to both datasets.

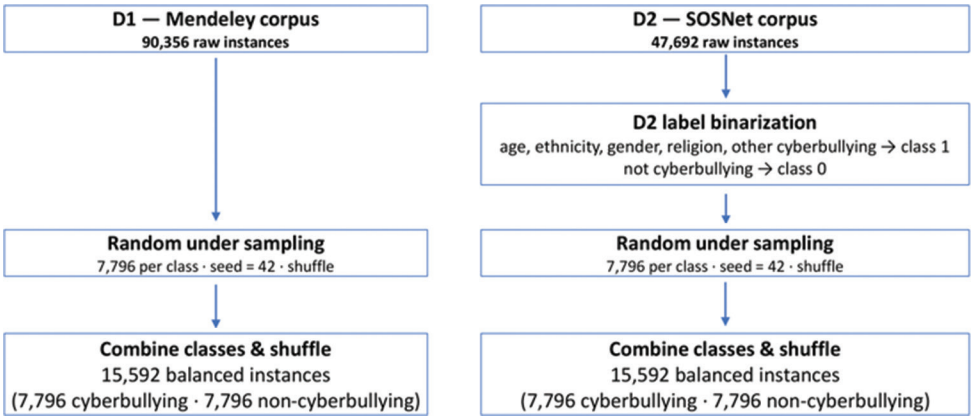


**Fig. 2.** Class balancing procedure for D1 and label binarization with class balancing for D2.

**TABLE 5: Dataset split sizes (identical for both D1 and D2)**

| Partition | Rows | Label 1 (Cyberbullying) | Label 0 (Non-cyberbullying) |
|---|---|---|---|
| Clean_train | 12,473 | 6,237 | 6,236 |
| Clean_val | 1,559 | 779 | 780 |
| Clean_test | 1,560 | 780 | 780 |
| Leet_test | 1,560 | 780 | 780 |

to assess classifier robustness to character substitutions. This procedure enables performance degradation to be measured under controlled conditions, with character-level obfuscation as the only difference between the two test sets.

Leet-speak obfuscates text on the Internet by replacing letters with numbers or symbols that resemble them. Because we did not identify a suitable publicly available dataset containing systematic leet-speak obfuscation, we created a rule-based character-substitution map based on common leet-speak conventions. Only the most familiar substitutions were included to keep the mapping interpretable. The complete mapping is provided in Table 6 to support reproducibility.

Table 6 presents the leet-speak character-substitution map used to generate the leet_test set. Each eligible character was independently substituted with a probability of 0.55. Therefore, not every eligible character was necessarily

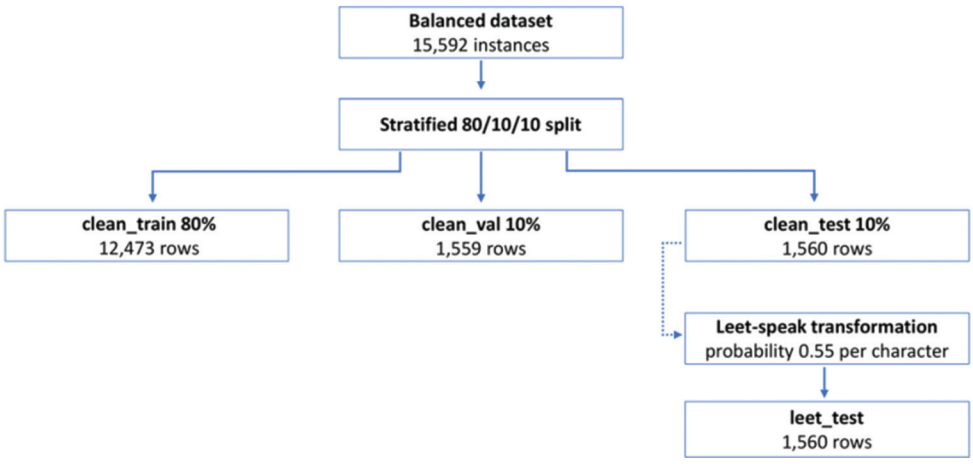


**Fig. 3.** Dataset splitting procedure and leet_test set construction.

changed, as illustrated by the sample sentence pairs in Table 7.

### 3.7. Encoder Selection

A total of six transformer encoders representing three pretraining domains were evaluated as backbone candidates, as shown in Table 8. All six encoders were optimized using the same training parameters and evaluated on the clean_test and leet_test sets. The encoder selected for the Proposed model was chosen primarily according to leet_test macro-F1, with clean_test macro-F1 used as a secondary criterion to assess whether robustness gains were achieved without sacrificing clean-text performance.

### 3.8. Leet-Speak Detection

The Proposed model begins with a lightweight obfuscation detector that checks whether an input contains leet-speak substitutions before normalization. Because character-substitution obfuscation poses a clear threat to automated text classifiers, the pipeline includes an explicit detection step before normalization. The detector is based on a regular-expression pattern designed to capture common leet-speak substitution structures, as formalized in Algorithm 1.

The three patterns were designed to cover selected positions within a word where a supported single-character substitution could plausibly appear. These patterns include a symbol between two letters (e.g., "h@t"), a digit between two letters (e.g., "h4t"), and a symbol followed by two or more letters (e.g., "@ss"), which captures word-initial symbol substitutions when they occur at the start of a token. Symbol and digit substitutions were assigned to separate patterns because they belong to different character classes in the

**Algorithm 1: Leet-Speak pattern detection**

Input: text
Output: Boolean True if leet-speak detected, False otherwise

Define the obfuscation detection pattern as a regular expression matching:
- letter-symbol-letter (e.g., h@t and f$t);
- letter-digit-letter (e.g., h4t); and
- symbol-two-or-more-letters (e.g., @ss and $top).

1. Convert text to lowercase
2. Apply the obfuscation detection pattern to the lowercase text
3. If a match is found: Return True
4. Otherwise: Return False

**TABLE 6: Leet-speak character-substitution map used for test set generation**

| Original character | Substitution (s) |
|---|---|
| A | 4, @ |
| B | 8 |
| E | 3 |
| G | 9 |
| I | 1, ! |
| L | 1 |
| O | 0 |
| S | 5, $ |
| T | 7 |

**TABLE 7: Sample clean and leet-transformed sentence pairs from D1**

| Clean text | Leet text |
|---|---|
| You are one stupid idiot | y0u 4r3 0n3 57up1d id1o7 |
| Would you like to know how god feels | w0uld y0u l1k3 to kn0w h0w 9od f331$ |
| This is not based off of a cure song | 7h15 1$ no7 ba$ed 0ff 0f a cur3 $0n9 |

regular expression. The design is not fully symmetric: word-initial digit substitutions, such as "0mg" for "omg," are not covered because only symbol substitutions have a dedicated word-initial rule. This design was constructed manually to capture common substitution structures rather than through empirical tuning or reliance on a specific prior reference. Its practical effectiveness is reported separately in the detector-performance evaluation (Table 9).

### 3.9. Proposed Model

The Proposed model is a conditional pipeline consisting of the leet-speak detector described in Section 3.8, the leet normalizer, and the BERTweet encoder. As shown in Fig. 4, if the detector identifies an input as containing leet-speak, the input is first normalized by Algorithm 2 and then passed to the BERTweet tokenizer. Otherwise, the raw text is passed through the tokenizer without modification. BERTweet encodes the input as a tokenized sequence, and the resulting [CLS] token representation is passed to a four-layer MLP classification head with dimensions 768→512→128→32→1.

**TABLE 8: Candidate encoders grouped by pretraining domain**

| Domain | Encoders |
|---|---|
| Twitter/social media | BERTweet [4], Twitter-RoBERTa [24] |
| Abuse/hate speech | HateBERT [25], HateXplain [26] |
| General web | BERT-base [27], RoBERTa-base [28] |

Batch normalization and dropout are applied after the first two linear layers, while dropout is applied after the third layer, producing the final binary prediction. This conditional design normalizes only inputs in which the detector identifies leet-speak; inputs without detected leet-speak are passed through unchanged. The same conditional pipeline is applied during training and inference. Because clean_train does not contain synthetic leet-speak obfuscation, the normalization gate has limited practical effect during training.

The leet normalizer uses the complete LEET_MAP presented in Table 10. The exclamation mark is treated contextually and is mapped to "i" only when surrounded by letters (e.g., f!ght → fight). Terminal punctuation is preserved; for example, hate! remains hate!. The entire normalization process is formalized in Algorithm 2.

We used a rule-based character-normalization approach in which each leet character was converted to the closest standard alphabetic character according to the LEET_MAP in Table 10. This design converts obfuscated inputs into readable text without altering clean inputs, making it suitable for the conditional pipeline described above. The LEET_MAP consists of the substitutions in Table 6 and other documented leet variants. The normalizer cannot determine whether the symbol "1" originally represented "i" or "l" because both characters are mapped to "1" during

**TABLE 9: Leet-speak detector performance on D1 and D2**

| Dataset | TP | FN | Precision | Recall | F1 | Clean FP | Clean FP Rate (%) |
|---|---|---|---|---|---|---|---|
| D1 | 1,548 | 12 | 1.0000 | 0.9923 | 0.9961 | 0 | 0.00% |
| D2 | 1,530 | 30 | 1.0000 | 0.9808 | 0.9903 | 12 | 0.77% |

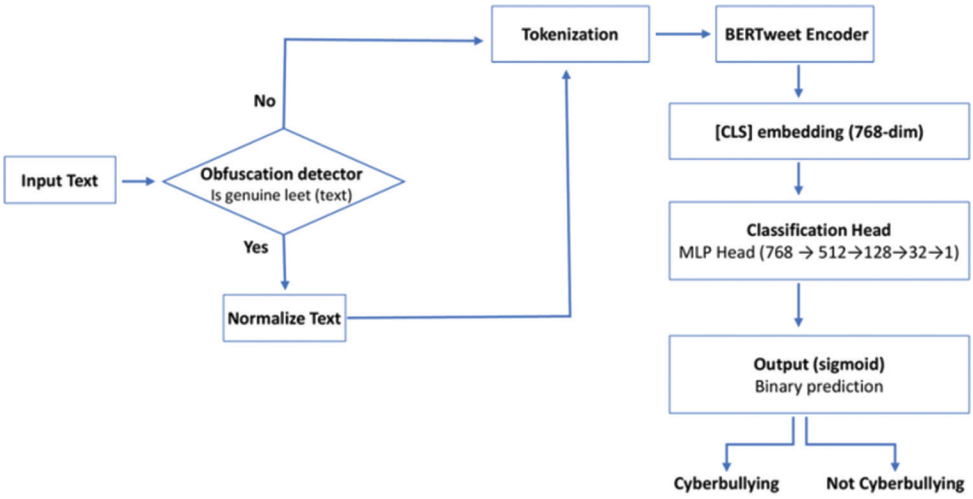


**Fig. 4.** Proposed obfuscation-aware cyberbullying detection pipeline.

generation. Therefore, "1" is always mapped to "i" during normalization. This limitation is acknowledged and does not affect the mapping of any other symbol.

### 3.10. Training Setup and Hyperparameters

The models were implemented in Python 3.12.13 using PyTorch 2.11.0 and the Hugging Face Transformers library (v5.13.1). They were trained on a single NVIDIA A100-SXM4-40GB graphics processing unit (GPU) with CUDA 12.8 through Google Colab. The same configuration was used to ensure a fair comparison among the encoders and ablation variants. BERTweet was fine-tuned with partial layer freezing: the embedding layer and the lower seven of the twelve transformer layers were frozen, while only the upper five layers were fine-tuned. The partial-freezing approach focuses fine-tuning on the parameters in the upper transformer layers, which capture task-specific patterns, while preserving the general linguistic representations learned by the lower layers during pretraining. This freezing depth was selected empirically using a validation macro-F1 sweep over freezing depths 7–12 (see "Freeze-Depth Selection," Table 11).

The classification head consisted of four linear layers with dimensions 768→512→128→32→1, followed by batch normalization and dropout. The AdamW optimizer used a lower learning rate for the transformer layers and a higher learning rate for the classification head. Label smoothing with $\varepsilon = 0.10$ was applied to reduce overconfident predictions and improve classifier generalization. This value was selected based on the prior work reported in [29]. The authors introduced label-smoothing regularization and used $\varepsilon = 0.10$ in their ImageNet experiments, reporting improvements in classification performance. Accordingly, $\varepsilon = 0.10$ was adopted as a regularization setting supported by prior literature in the present study. A cosine schedule and linear warm-up were also used. All hyperparameters are summarized in Table 12.

**Algorithm 2: Leet-Speak normalization**

Input: A text string that may contain leet-speak characters
Output: The text with all leet-speak characters replaced by their standard alphabetic equivalents

Steps:
1. Split the input text into individual characters.
2. For each character, apply the first matching rule only:
**Rule A — Exclamation mark (context-sensitive):**
If the character is!:
  If both neighbors are letters, replace ! with i (e.g., f!ght → fight).
  Otherwise, keep ! unchanged (e.g., hate! → hate!).
**Otherwise, Rule B — Known leet character:**
If the character exists in LEET_MAP [Table 10], replace it with the corresponding standard letter or word (e.g., h@te → hate).
**Otherwise, Rule C — Normal character:**
Keep the character unchanged.
3. Join the processed characters and return the result.

**TABLE 10: Complete leet-speak normalization map (LEET_MAP)**

| Leet character | Normalized form |
|---|---|
| @ | a |
| 4 | a |
| ^ | a |
| 3 | e |
| 1 | i |
| ! | i (between letters only) |
| 0 | o |
| $ | s |
| 5 | s |
| 7 | t |
| + | t |
| ( | c |
| # | h |
| 9 | g |
| 6 | g |
| 8 | b |
| 2 | to |
| & | and |
| * | x |

### 3.11. Evaluation Protocol

The same evaluation protocol was used in all experiments and included five performance measures, multi-seed validation, and a robustness check.

#### *3.11.1. Test conditions*

Two settings were used for each trained model:

- Clean_test set: This set contained 1,560 original, unmodified instances.
- Leet_test set: This set contained the same 1,560 instances after application of the deterministic character-obfuscation procedure described in Section 3.6.

Because the two sets contained the same sentences and labels, any differences in performance could be attributed

**TABLE 11: Validation macro-F1 by encoder freezing depth (seed 42)**

| Freeze layers | Trainable layers | D1 Validation Macro-F1 | D2 Validation Macro-F1 |
|---|---|---|---|
| 12 | 0 | 0.8259 | 0.8213 |
| 11 | 1 | 0.8634 | 0.8430 |
| 10 | 2 | 0.8787 | 0.8540 |
| 9 | 3 | 0.8845 | 0.8681 |
| 8 | 4 | 0.8839 | 0.8663 |
| **7** | **5** | **0.8864** | **0.8668** |

**TABLE 12: Training hyperparameters**

| Hyperparameter | Value |
|---|---|
| Max sequence length | 128 |
| Frozen encoder layers | 7 (lower layers) +embeddings |
| Dropout | 0.50 |
| Transformer learning rate | $5\times10^{-6}$ |
| Classification head learning rate | $1\times10^{-4}$ |
| Weight decay | 0.02 |
| Label smoothing (ε) | 0.10 |
| Batch size | 8 |
| Gradient accumulation steps | 2 (effective batch size 16) |
| Maximum epochs | 8 |
| Early-stopping patience | 4 epochs (validation macro-F1) |
| Warm-up ratio | 0.10 |
| Gradient clipping | 1.0 |
| Loss function | Label-smoothed BCE |
| Optimizer | AdamW |
| Learning-rate schedule | Cosine with linear warm-up |
| Random seeds | 42, 123, 2024 |

to character-level obfuscation rather than to variation in the samples.

#### 3.11.2. Evaluation metrics

Five metrics were calculated for each test condition, the first four of which were based on Equations (1)–(4). Precision, recall, and F1 were macro-averaged, meaning that both classes were assigned equal weight. Accuracy and receiver operating characteristic–area under the curve (ROC-AUC) were computed for the binary classification task. Let TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

$$Precision = \frac{1}{2}\left(\frac{TP_0}{TP_0+FP_0}+\frac{TP_1}{TP_1+FP_1}\right) \tag{2}$$

$$Recall = \frac{1}{2}\left(\frac{TP_0}{TP_0+FN_0}+\frac{TP_1}{TP_1+FN_1}\right) \tag{3}$$

$$Macro-F1 = \frac{1}{2}(F1_0+F1_1); F1_c = \frac{2P_cR_c}{P_c+R_c} \tag{4}$$

The fifth metric was the ROC-AUC, calculated from the raw sigmoid scores as a threshold-independent measure of ranking ability. The sigmoid output was converted into a binary prediction using a fixed decision threshold of $\tau = 0.50$, as specified in Equation (5):

$$\hat{y} = \begin{cases} 1, & if\ \sigma(\log it) \geq 0.50, \\ 0, & otherwise \end{cases} \tag{5}$$

#### 3.11.3. Multi-seed evaluation

The experiments were repeated three times using three random seeds: 42, 123, and 2024. A separate classification head was initialized for each seed, and the order of the training samples was independently shuffled for each run. The results are reported as the mean ± population standard deviation across the three seeds, as defined by Equations (6) and (7):

$$\bar{m} = \frac{1}{3}\sum_{k=1}^{3} m_k \tag{6}$$

$$\sigma_m = \sqrt{\frac{1}{3}\sum_{k=1}^{3}(m_k-\bar{m})^2} \tag{7}$$

where $m_k$ denotes the metric value obtained under seed $k$, and $\sigma_m$ denotes the population standard deviation across the three runs.

#### 3.11.4. Robustness check

A heuristic lower bound on leet_test macro-F1 is computed as shown in Equation (8):

$$LB = \bar{m} - 2\sigma_m \tag{8}$$

This value provides a conservative heuristic lower-bound estimate based on the mean and population standard deviation across the tested seed range. No formal statistical significance test or confidence interval was computed; the heuristic lower bound was used as a descriptive robustness check rather than as a formal statistical test.

#### 3.11.5. Encoder selection criterion

The primary criterion for selecting the encoder for the comparison in Section 3.7 was leet_test macro-F1. The clean_test macro-F1 was reported as a secondary criterion to assess whether robustness gains occurred at the expense of clean-text performance.

## 4. RESULTS

### 4.1. Encoder Comparison

The test performance of the six candidate encoders on D1 and D2 was reported in Tables 13 and 14, respectively, and averaged across three random seeds. Overall, all six models showed similar performance on clean text, with macro-F1 values ranging from 0.8429 to 0.8720 across the two datasets. The leet_test macro-F1 scores varied substantially

across pretraining domains, with BERTweet achieving the highest scores on both datasets: 0.6700 on D1 and 0.7150 on D2. BERTweet was therefore selected as the backbone of the Proposed model. The ranking of the remaining encoders differed considerably between the two datasets (Tables 13 and 14).

### 4.2. Leet Detector Performance

The performance of the leet-speak detector on both datasets is reported in Table 9. Because leet_test is fully synthetic, its ground-truth labels are known exactly, allowing precise evaluation. On D1, the detector achieved a precision of 1.0000, a recall of 0.9923, and an F1 score of 0.9961 on leet_test. No false positives were observed on clean text. On D2, the detector achieved a precision of 1.0000, a recall of 0.9808, and an F1 score of 0.9903 on leet_test, with 12 false positives on clean text, corresponding to a false-positive rate of 0.77%.

### 4.3. Proposed Method Performance

Tables 15 and 16 present all metrics for the proposed method on D1 and D2, averaged across three random seeds. On D1, the proposed method achieved a clean_test macro-F1 score of 0.8643 ± 0.0003 and a leet_test macro-F1 score of 0.8513 ± 0.0050. The corresponding loss curves are shown in Fig. 5, and the confusion matrices are shown in Figs. 6 and 7. On D2, the proposed method achieved a clean_test macro-F1 score of 0.8706 ± 0.0069 and a leet_test macro-F1 score of 0.8542 ± 0.0021. The corresponding loss curves are shown in Fig. 8, and the confusion matrices are shown in Figs. 9 and 10. The standard deviations were small for all metrics across both datasets, indicating consistent performance across the three seeds.

### 4.4. Baseline versus Proposed

Table 17 shows that, compared with the BERTweet baseline, the proposed approach based on conditional normalization substantially improves leet_test macro-F1 on both datasets. The clean_test performance is unchanged on D1 and marginally higher on D2.

### 4.5. Statistical Reliability

The heuristic lower bound ($\bar{m}-2\sigma_m$) of leet_test macro-F1 is reported in Table 18 and is markedly higher than the Baseline mean for both datasets.

**TABLE 13: Encoder comparison on D1 (mean±standard deviation, three seeds)**

| Dataset 1 Mendeley | | | |
|---|---|---|---|
| **Encoder** | **Clean Macro-F1** | **Leet Macro-F1** | **Leet ROC-AUC** |
| BERTweet | **0.8643±0.0003** | **0.6700±0.0312** | **0.7732±0.0044** |
| HateBERT | 0.8429±0.0016 | 0.6328±0.0225 | 0.6963±0.0064 |
| RoBERTa-base | 0.8553±0.0037 | 0.5210±0.0484 | 0.6539±0.0037 |
| Twitter-RoBERTa | 0.8680±0.0020 | 0.6340±0.0435 | 0.7255±0.0071 |
| HateXplain | 0.8608±0.0013 | 0.4888±0.0261 | 0.7013±0.0029 |
| BERT-base | 0.8559±0.0029 | 0.5410±0.0179 | 0.6911±0.0056 |

ROC-AUC: Receiver operating characteristic–area under the curve

**TABLE 14: Encoder comparison on D2 Kaggle (SOSNet) (mean±standard deviation, three seeds)**

| Dataset 2 Kaggle (SOSNet) | | | |
|---|---|---|---|
| **Encoder** | **Clean Macro-F1** | **Leet Macro-F1** | **Leet ROC-AUC** |
| BERTweet | 0.8678±0.0046 | **0.7150±0.0335** | **0.8513±0.0004** |
| HateBERT | 0.8664±0.0045 | 0.4156±0.0242 | 0.8301±0.0041 |
| RoBERTa-base | 0.8663±0.0015 | 0.3889±0.0087 | 0.7068±0.0246 |
| Twitter-RoBERTa | **0.8720±0.0023** | 0.4382±0.0011 | 0.8107±0.0183 |
| HateXplain | 0.8637±0.0045 | 0.5243±0.0269 | 0.8078±0.0122 |
| BERT-base | 0.8605±0.0029 | 0.5533±0.0432 | 0.8133±0.0065 |

ROC-AUC: Receiver operating characteristic–area under the curve

**TABLE 15: Proposed method full metrics on D1 Mendeley (mean±standard deviation, three seeds)**

| Test | Accuracy | Macro-F1 | Precision | Recall | ROC-AUC |
|---|---|---|---|---|---|
| Clean | 0.8643±0.0003 | 0.8643±0.0003 | 0.8646±0.0003 | 0.8643±0.0003 | 0.9426±0.0008 |
| Leet | 0.8515±0.0049 | 0.8513±0.0050 | 0.8533±0.0036 | 0.8515±0.0049 | 0.9275±0.0021 |

ROC-AUC: Receiver operating characteristic–area under the curve

**TABLE 16: Proposed method full metrics on D2 Kaggle (SOSNet) (mean±standard deviation, three seeds)**

| Test | Accuracy | Macro-F1 | Precision | Recall | ROC-AUC |
|---|---|---|---|---|---|
| Clean | 0.8709±0.0070 | 0.8706±0.0069 | 0.8754±0.0081 | 0.8709±0.0070 | 0.9367±0.0008 |
| Leet | 0.8553±0.0020 | 0.8542±0.0021 | 0.8664±0.0028 | 0.8553±0.0020 | 0.9238±0.0003 |

ROC-AUC: Receiver operating characteristic–area under the curve

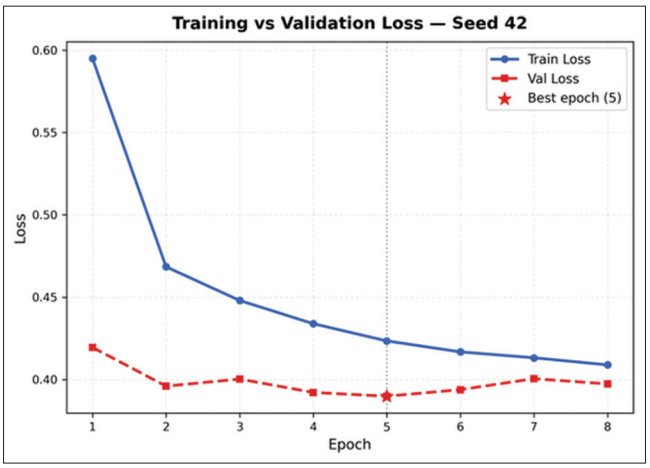


**Fig. 5.** Training and validation loss curves for the proposed model, seed 42, D1.

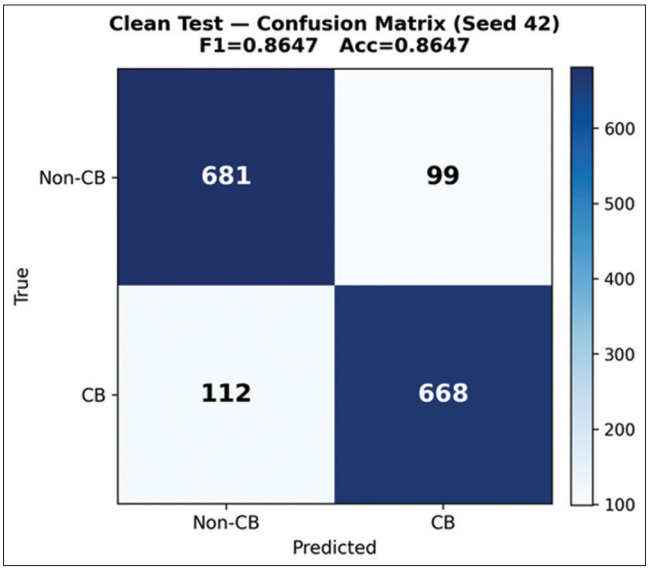


**Fig. 6.** Confusion matrix for the proposed model on clean_test, seed 42, D1

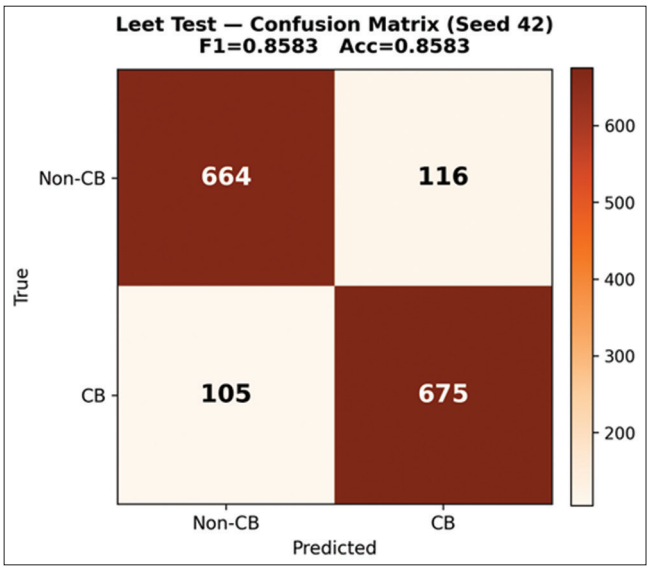


**Fig. 7.** Confusion matrix for the proposed model on leet_test, seed 42, D1.

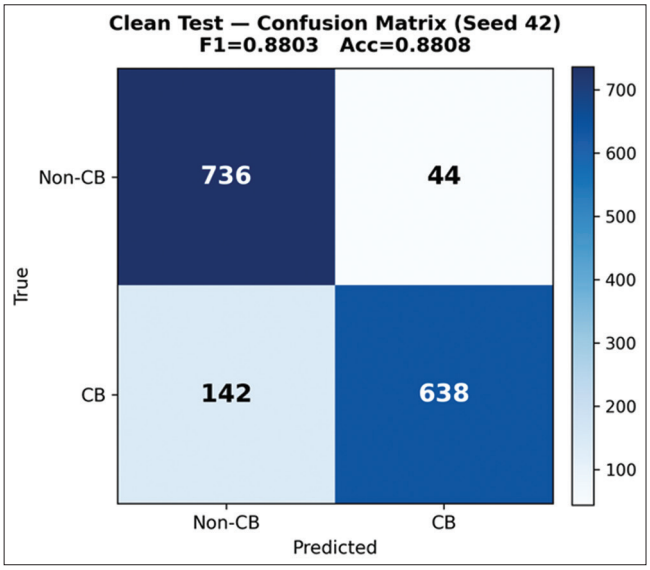


**Fig. 8.** Training and validation loss curves for the proposed model, seed 42, D2.

## 4.6. Ablation Study

The ablation-study results for the three input variants Baseline, Always-Normalize, and Conditional-Normalize are shown in Tables 19 and 20 for D1 and D2, respectively. The results are presented as bar charts in Figs. 11 and 12. Both normalization variants substantially improved the Baseline performance on leet_test macro-F1 for both datasets. The clean_test macro-F1 for all three variants is reported in Tables 19 and 20.

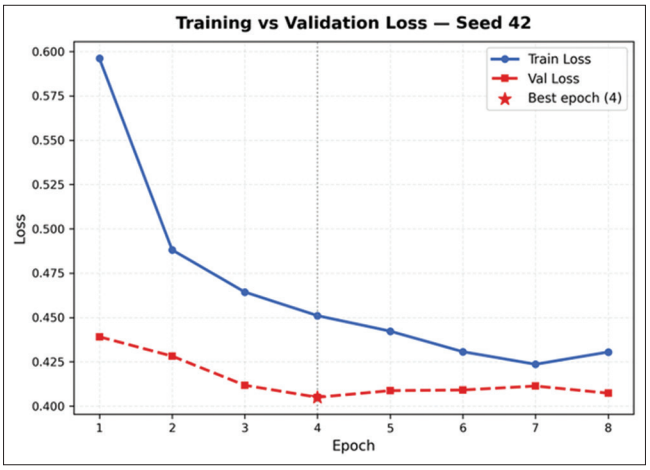


**Fig. 9.** Confusion matrix for the proposed model on clean_test, seed 42, D2.

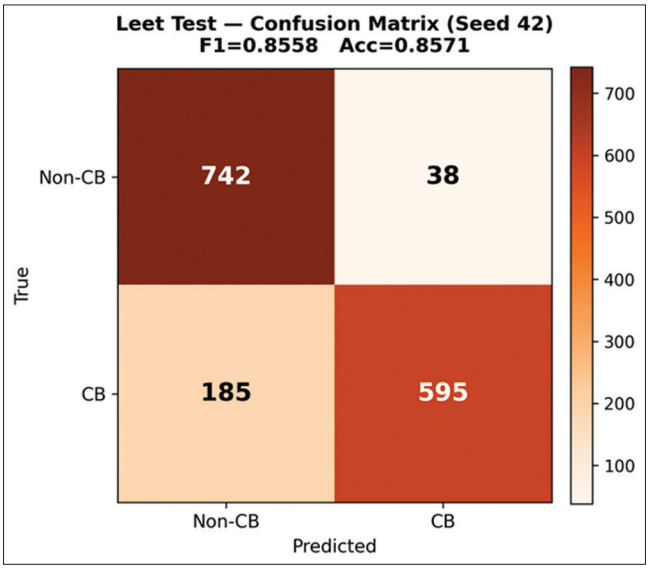


**Fig. 10.** Confusion matrix for the proposed model on leet_test, seed 42, D2.

The ablation results indicate that the presence of normalization accounts for most of the observed improvement under leet-speak obfuscation. Both Always-Normalize and Conditional-Normalize substantially outperform the Baseline on leet_test, while their scores remain comparable to each other. This pattern suggests that normalization is the primary contributor to the robustness gain, whereas the conditional gate is mainly intended to preserve inputs without detected leet-speak and avoid unnecessary normalization. The present ablation design did not include a separate conditional-normalization-without-gating variant; therefore, the independent causal contribution of the gate was not isolated experimentally.

To evaluate the computational cost of the proposed conditional design relative to the Always-Normalize variant, inference latency and peak memory were measured on the clean_test and leet_test splits of both datasets (Table 21).

### 4.7. Computational Cost Analysis

To assess the computational cost of the proposed conditional normalization pipeline relative to a model without preprocessing, Table 22 compares a separately trained Baseline model, which uses neither leet-speak detection nor normalization, with the Proposed model. The comparison reports model size, training time, and leet_test inference cost for seed 42; inference times are reported as the mean ± standard deviation over 200 forward passes. Table 21 provides the comparison between the Conditional-Normalize and Always-Normalize variants.

### 4.8. Qualitative Comparison with Prior Work

Table 1 presents a qualitative architectural comparison between the proposed approach and five previous studies that used the D1 and D2 corpora. The table summarizes the reference, year, architecture, dataset, and task formulation; numerical performance values are intentionally omitted

**TABLE 17: Baseline versus proposed method (mean±standard deviation, three seeds)**

| Dataset | Method | Clean Macro-F1 | Leet Macro-F1 | Leet ROC-AUC |
|---|---|---|---|---|
| D1 | Baseline | 0.8643±0.0003 | 0.6700±0.0312 | 0.7732±0.0044 |
| D1 | Proposed | **0.8643±0.0003** | **0.8513±0.0050** | **0.9275±0.0021** |
| D2 | Baseline | 0.8678±0.0046 | 0.7150±0.0335 | 0.8513±0.0004 |
| D2 | Proposed | **0.8706±0.0069** | **0.8542±0.0021** | **0.9238±0.0003** |

**TABLE 18: Statistical reliability heuristic lower bound on leet_test macro-F1**

| Dataset | Baseline mean | Proposed mean | Proposed σ | Lower Bound ($\tilde{m}-2\sigma_m$) | Gap versus Baseline |
|---|---|---|---|---|---|
| D1 | 0.6700 | 0.8513 | 0.0050 | 0.8413 | +17.13% points |
| D2 | 0.7150 | 0.8542 | 0.0021 | 0.8500 | +13.50% points |

**TABLE 19: Ablation study on D1 Mendeley (mean±standard deviation, three seeds)**

| Variant | Clean Macro-F1 | Leet Macro-F1 |
|---|---|---|
| Baseline | 0.8643±0.0003 | 0.6700±0.0312 |
| Always-Normalize | 0.8643±0.0003 | 0.8521±0.0053 |
| Conditional-Normalize (Proposed) | **0.8643±0.0003** | **0.8513±0.0050** |

**TABLE 20: Ablation study on D2 Kaggle (SOSNet) (mean±standard deviation, three seeds)**

| Variant | Clean Macro-F1 | Leet Macro-F1 |
|---|---|---|
| Baseline | 0.8678±0.0046 | 0.7150±0.0335 |
| Always-Normalize | 0.8693±0.0038 | 0.8544±0.0042 |
| Conditional-Normalize (proposed) | **0.8706±0.0069** | **0.8542±0.0021** |

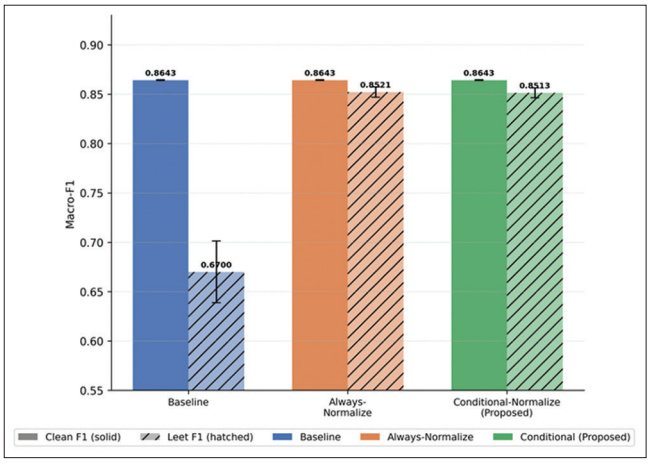


**Fig. 11.** Ablation-study bar chart comparing Baseline, Always-Normalize, and Conditional-Normalize on the clean_test and leet_test sets for D1

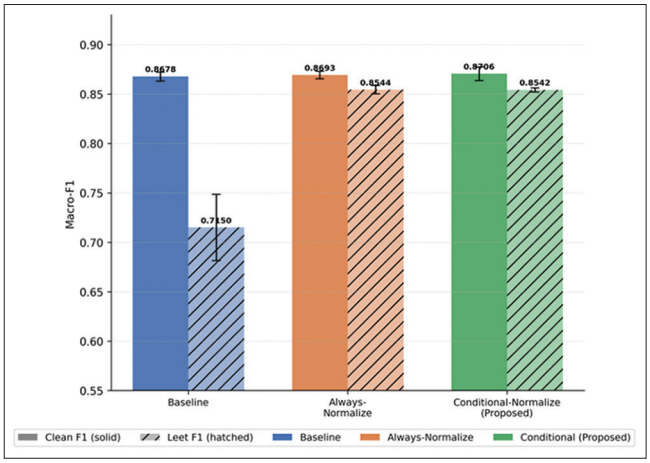


**Fig. 12.** Ablation-study bar chart comparing Baseline, Always-Normalize, and Conditional-Normalize on the clean_test and leet_test sets for D2.

because direct comparison would not be scientifically valid. The D1 studies used an imbalanced corpus, whereas the present study used a balanced partition. In addition, the D2 studies [18]–[20] performed multiclass classification, whereas the present study performed binary classification. None of the previous studies listed in Table 1 evaluated leet-speak obfuscation. Accordingly, Table 1 is provided for architectural and contextual reference only and should not be interpreted as a performance ranking.

### 4.9. Freeze-Depth Selection

A freezing depth of seven layers (i.e., freezing the embeddings and the lower seven of the twelve transformer layers while fine-tuning the upper five) was selected based on a validation macro-F1 sweep over freezing depths 7–12 using seed 42 on both datasets (Table 11).

Freezing 10 or more layers led to a clear and consistent decline in validation macro-F1 on both datasets. Freezing depths of 7, 8, and 9 formed a closely clustered top tier, with macro-F1 values differing by no more than 0.003 within either dataset. A freezing depth of 7 was selected as a fixed hyperparameter for both datasets. It achieved the best validation macro-F1 on D1 (0.8864) and was within 0.0013 of the best value on D2 (0.8668 at depth 7 vs. 0.8681 at depth 9).

### 4.10. Qualitative Examples and Error Analysis

To illustrate the practical behavior of the proposed conditional normalization pipeline, Table 23 presents three representative cases from predictions on leet_test using seed 42: One correctly classified non-cyberbullying case, one false-positive case, and one false-negative case.

The first example in Table 23 is shown with a non-cyberbullying true label, a non-cyberbullying predicted label, and a normalized text; it is therefore listed as a correct classification. The second example was a false positive: its normalized text was classified as cyberbullying despite the non-cyberbullying true label. In the second example, normalization did not fully resolve the nonstandard punctuation. The exclamation mark remained unconverted when it was isolated by spaces or immediately followed by a digit, consistent with the context-dependent rule for "!" documented in Table 10. The third example was a false negative: its normalized text was classified as non-cyberbullying despite the cyberbullying true label. In the third example, an exclamation mark immediately followed by a digit ("!5") was similarly left unconverted, producing the partial token "!s" rather than "is." These examples show that some normalized outputs retained unresolved obfuscation and that

**TABLE 21: Inference cost comparison Conditional-Normalize (proposed) versus Always-Normalize (seed 42, batch size 1, mean±standard deviation over 200 forward passes, NVIDIA A100-SXM4-40GB)**

| Dataset | Test set | Conditional-Normalize (ms/sample) | Always-Normalize (ms/sample) | Peak memory (MB) |
|---|---|---|---|---|
| D1 | leet_test | 10.432±1.053 | 10.465±1.103 | 566.0 |
| D1 | clean_test | 10.293±0.676 | 10.036±0.669 | 566.0 |
| D2 | leet_test | 9.979±0.337 | 9.939±0.342 | 566.0 |
| D2 | clean_test | 9.807±0.207 | 9.640±0.199 | 566.0 |

**TABLE 22: Computational profile Baseline versus proposed (seed 42, batch size 1, NVIDIA A100-SXM4-40GB; BERTweet-base: 135.36 million total parameters and 36.49 million trainable parameters for both variants)**

| Dataset | Variant | Training Time (s) | leet_test Inference (ms/sample) | Detector-only (ms) | Normalizer-only (ms) | Peak Memory (MB) |
|---|---|---|---|---|---|---|
| D1 | Baseline | 355.4 | 10.528±1.166 | | | 566.0 |
| D1 | Proposed | 355.6 | 10.432±1.053 | 0.00165 | 0.03343 | 566.0 |
| D2 | Baseline | 353.2 | 9.957±0.284 | | | 566.0 |
| D2 | Proposed | 351.1 | 9.979±0.337 | 0.00138 | 0.01339 | 566.0 |

**TABLE 23: Qualitative examples of conditional normalization outcomes (seed 42)**

| Original text | Normalized text | True label | Predicted label | Error type |
|---|---|---|---|---|
| “s0m37h!ng l1k3 th3 new m3$sag3 7h1n9 4t 7he 7op of your t@lk p4g3” | “something like the new message thing at the top of your talk page” | Non-cyberbullying | Non-cyberbullying | Correct |
| “filmpunk smud80y s73uph ! h4ve g0t 7h!5 p3rs0n block3d p13a$e r3move m3 fr0m 7h3 c0nvers47i0n l7” | “filmpunk smudboy steuph ! have got th!s person blocked piease remove me from the conversation lt” | Non-cyberbullying | Cyberbullying | False positive |
| “k@t !5 ju5t p1a1n fucking @wful mkr” | “kat !s just piain fucking awful mkr” | Cyberbullying | Non-cyberbullying | False negative |

both false-positive and false-negative classifications occurred when the resulting token sequence remained atypical.

## 5. DISCUSSION

All six encoders achieved similar macro-F1 scores on the clean_test sets, with only small differences on both datasets (Tables 13 and 14). This indicates that, in the absence of obfuscation, BERT-family models perform comparably on binary classification of social-media text, regardless of their pretraining domain. Clean-text performance alone is therefore not sufficient to assess an encoder’s suitability under the evaluated synthetic leet-speak obfuscation.

Under the evaluated synthetic leet-speak transformations, performance varied considerably across the tested pretraining domains. BERTweet achieved the highest leet_test macro-F1 on both datasets. A possible explanation is that its pretraining on English Twitter data exposed it to informal orthography and nonstandard spelling patterns [4], some of which may resemble patterns produced by leet-speak; however, this mechanism was not isolated experimentally. Neither hate-speech-specialized encoder matched BERTweet’s leet_test performance on either dataset, and each showed severe degradation on at least one dataset: HateBERT’s clean_test macro-F1 dropped by 45.08% points when the evaluated synthetic leet-speak transformation was applied to D2, and HateXplain showed the largest degradation of any encoder on D1 (Tables 13 and 14). Within the tested encoders, these findings suggest that pretraining on social-media text may provide greater benefit for the evaluated character-level obfuscation than pretraining on hateful vocabulary; however, this comparison does not establish that the difference is causal or that hate-speech pretraining cannot support robustness.

The proposed conditional normalization pipeline largely recovered performance under the evaluated synthetic leet-speak transformations, with leet_test macro-F1 gains of 18.13 and 13.92% points on D1 and D2, respectively (Table 17), while clean_test performance was preserved on D1 and marginally higher on D2. The corresponding descriptive heuristic lower bounds were 0.8413 for D1 and 0.8500 for D2, both remaining above their respective Baseline means. The ablation study (Tables 19 and 20) shows that the Conditional-Normalize and Always-Normalize variants

achieved comparable leet_test macro-F1, with Always-Normalize marginally higher on both datasets.

The computational measurements therefore assess whether the conditional design produced a measurable latency or memory benefit relative to the Always-Normalize variant. Two related but distinct measurements were taken at seed 42 and batch size 1 on an NVIDIA A100-SXM4-40GB GPU. Single trained checkpoints rather than seed averages were used because latency and memory reflect fixed architecture and hardware rather than training variability. Table 21 compares the Conditional-Normalize and Always-Normalize variants across the clean_test and leet_test splits of both datasets, using the mean ± standard deviation over 200 forward passes. End-to-end latency was similar between the two variants, with differences of up to 0.26 ms falling well within the observed run-to-run variance (standard deviations of 0.20–1.10 ms). The detection step itself was negligible in cost (0.001–0.002 ms/sample; Table 22) relative to the transformer forward pass (approximately 9.6–10.5 ms/sample).

The second comparison, reported in Table 22, contrasts the Proposed model with a Baseline trained using the same no-normalization methodology as in Tables 17, 19, and 20. The Baseline was retrained specifically for this analysis rather than reused from those earlier checkpoints. The two models have identical parameter counts (135.36 million total and 36.49 million trainable parameters), and each was trained in under six minutes. End-to-end inference overhead was −0.91% on D1 and +0.22% on D2. In the comparison between the Baseline and Proposed models, the isolated costs of the detector and normalizer (0.001–0.033 ms/sample) were smaller than the measured latency standard deviations (0.28–1.17 ms/sample) by roughly one to three orders of magnitude. Peak GPU memory was identical in every configuration. The small, sign-inconsistent overhead is consistent with measurement noise and does not provide evidence of a reliable preprocessing cost. Any potential efficiency advantage of the conditional design is architectural in principle and was not measurable in this GPU-accelerated setting. Its practical effect in preprocessing-bound settings, such as central processing unit-only inference, remains unmeasured and should be evaluated in future work. Taken together, these measurements indicate that the added computational cost of the full pipeline was small relative to the measured variation at the scale evaluated here.

Conceptually, under the evaluated synthetic leet-speak transformations, the expected mechanism is as follows: leet-speak substitutions distort lexical surface forms, which can reduce the quality of pretrained token representations; conditional normalization restores a form closer to the pretraining distribution, potentially improving classifier robustness to these transformations. The conditional gate makes this transformation conditional. Thus, the proposed design combines representation recovery with input preservation, although the present experiments do not isolate each causal step independently.

Beyond the conceptual mechanism and computational efficiency discussed above, the conditional design is motivated by two further considerations. The first is architectural: normalization is applied only to inputs the detector flags, so text that does not trigger the detector is never altered, whereas an unconditional (Always-Normalize) design applies the substitution map to every input regardless of whether the characters it targets (e.g., "$", "4", "2") appear for an unrelated reason, such as a price or a legitimate numeral. Neither "$5" nor "rule 42" triggers the detector, yet Always-Normalize would alter both. This architectural distinction did not produce a measurable difference on the present datasets (Tables 19 and 20); it is a design safeguard rather than a property empirically demonstrated in this study. Its scope is also specific: when the detector does not trigger, the input is passed through unchanged; however, the detector may miss some genuine leet-speak patterns and may produce false positives on clean text. Once any part of an input triggers the detector, normalization is applied to the entire string, so coincidental map characters elsewhere in that input may be altered.

The second consideration is that, as illustrated by the classification outcomes in Table 23, normalization does not have a uniformly beneficial effect across inputs; its effect is input-dependent even though the ablation results in Tables 19 and 20 indicate that both normalization variants recover much of the performance lost by the Baseline.

None of the five prior studies in Table 1 [16]–[20] evaluates performance under character-level obfuscation, consistent with the gap identified in Section 2. The leet_test protocol introduced here addresses this gap directly by applying a fixed, reproducible transformation to two public corpora and providing a basis against which future methods can be measured. This is the sense in which the present work extends prior research on these corpora by adding an evaluation dimension rather than by improving on reported figures.

The proposed approach was not directly compared against large language models (LLMs) in a zero-shot setting, which represents an alternative approach for content-moderation tasks. Recent work benchmarking GPT-4.1, Gemini 1.5 Pro, and Claude 3 Opus for zero-shot cyberbullying detection on real-world YouTube comments found that these models achieved relatively strong performance but still struggled with linguistic nuances such as sarcasm, coded insults, and mixed-language slang [30]. The present experimental design focused on fine-tuned encoder-based models rather than zero-shot LLM prompting, and a matched comparison would require a separate prompting and evaluation protocol, which is left to future work. In principle, the conditional normalization pipeline could also be placed before an LLM-based classifier, but this integration and its effects on performance, latency, and cost were not evaluated in the present study. Fine-tuning a smaller, domain-specific encoder such as BERTweet nonetheless remains a practical alternative for moderation scenarios where latency, cost, and deployment control are important considerations.

The remainder of this section examines the pipeline's failure modes and the boundaries of the present evaluation.

Failures occurred at two levels. At the detector level, the leet-speak detector achieved perfect precision on the leet_test sets (Table 9) but failed to flag 12 sentences in D1 and 30 in D2. These misses occurred because the local character configurations around the substituted characters did not match any of the three detection patterns in Algorithm 1. Some of the D2 cases were associated with the <user> placeholder, which replaced @mentions during preprocessing before leet generation, leaving the substituted character adjacent to a letter on one side and a digit on the other rather than to two letters.

In the representative cases examined, the false-negative example retained residual obfuscation after normalization, including the partial token "!s," and was classified as non-cyberbullying despite its cyberbullying label, whereas the false-positive example was classified as cyberbullying despite its non-cyberbullying label. In one analyzed violence-reporting example from D1, normalization restored the terms "isis" and "jihadist," and the resulting text was classified as cyberbullying despite its non-cyberbullying context. These representative cases suggest that some classifier errors on leet input may stem from restored vocabulary influencing the classifier toward an incorrect prediction rather than from normalization failing to restore it.

The leet_test sets were generated using a fixed substitution map and a uniform substitution probability (0.55), which allows controlled, reproducible testing but does not capture the full diversity of real-world leet-speak. The reported robustness gains (18.13 and 13.92% points for D1 and D2) should therefore be interpreted as evidence of robustness against the specific synthetic transformations evaluated here, not as a guarantee of equivalent performance on naturally occurring leet-speak or other forms of real-world text manipulation. Real users may combine multiple obfuscation strategies inconsistently, which could pose a greater challenge than the systematic, single-mechanism substitutions tested in this study. Robustness at substitution rates other than 0.55 has not been empirically tested.

The detect-then-normalize architecture is specifically designed for character-substitution-based leet-speak, and the detector (Algorithm 1) and normalizer (Table 10) recognize only the patterns and characters defined in this study. Substitution types outside this scope, such as multi-character phonetic spellings, spacing tricks, or emoji substitution, would go unrecognized. An input with no recognized substitutions passes through unmodified, whereas an input mixing recognized and unrecognized types is only partially normalized because unrecognized characters are left unchanged. Other forms of adversarial obfuscation were not evaluated, and the extensibility of the present architecture to them likely varies according to their structure. Homoglyph substitution shares the same-length, character-for-character form of leet-speak, whereas character insertion or deletion, spacing manipulation, and punctuation insertion change the input length and would likely require a different underlying mechanism. Phonetic spelling, emoji substitution, and multilingual code-switching differ further because they are not primarily character-level phenomena. The core detect-then-normalize logic could in principle be extended by expanding the pattern rules and substitution map, but no such extension was implemented or evaluated here, and a systematic investigation of extensibility is left to future work.

Both corpora are English-only, so the results may not generalize to other languages, including Arabic, Kurdish, Spanish, and French, or to multilingual and code-switched text. The substitution map (Table 10) and detector patterns (Algorithm 1) were built for English orthography, and the detector patterns use Latin-script character classes, which do not match non-Latin scripts such as Arabic-script Kurdish. The encoder, BERTweet, was pretrained on English tweets [4]. Adapting these components to other languages was not implemented or evaluated in this study. The detect-

then-normalize architecture could in principle be adapted to other languages, but transferring the present implementation would require language-specific substitution maps and detector patterns, together with a language-appropriate pretrained encoder.

A further limitation concerns possible domain shift between the datasets used here and other cyberbullying data. D1 and D2 are fixed, previously collected corpora, and the results reported here reflect performance on those two corpora only. The proposed pipeline was not evaluated on other cyberbullying datasets or on data from other platforms, so its performance in those settings is not established by this study.

The normalizer operates on surface-level character patterns and cannot distinguish leet-speak used to evade content moderation from leet-speak used for stylistic, emphatic, or community-identity purposes. The detector may therefore flag, and the normalizer may alter, substitutions that were not intended to obscure meaning. Accordingly, normalization should be understood as a supporting representation transformation for classification, not as an assumption that all nonstandard spelling is noise. This study did not measure whether normalization changes the semantic or pragmatic content of an input; it measured only the effect of normalization on downstream classification performance. The clean_test macro-F1 values for the Baseline, Always-Normalize, and Conditional-Normalize variants are reported in Tables 19 and 20, but they do not establish that meaning was preserved. Assessing whether normalization alters intended meaning would require a dedicated human evaluation, which was not conducted in this study.

Two further boundaries of the experimental setup are worth noting. First, formal statistical significance testing and confidence-interval estimation were not conducted because only three seeds were used, providing too small a sample for reliable parametric inference; the separation from the Baseline was instead summarized descriptively using a conservative heuristic lower bound ($\bar{m} - 2\sigma_m$) on leet_test macro-F1 (Table 18), which remained above Baseline for both datasets (+17.13% points for D1 and +13.50% points for D2). Second, the freeze-depth sweep (Table 11) covered depths 7 through 12 only, and depths below 7 were not tested.

The binarization of D2 merges five fine-grained cyberbullying categories (age, ethnicity, gender, religion, and other cyberbullying) into a single positive class, discarding category-specific information: a text flagged as positive is not attributed to any particular category, which limits the granularity of downstream moderation decisions. The binary classification formulation is reflected in the classification head and loss function, whereas the leet-speak detector and normalizer operate independently of the classification target. A multiclass or multilabel formulation was not implemented or evaluated in this study and remains a direction for future work.

In addition to the technical failure modes discussed above, deploying the proposed system as a cyberbullying moderation tool has implications beyond classification accuracy. A false positive may flag content that was not intended as cyberbullying. The reporting-context misclassification described above, in which normalization restored "isis" and "jihadist" in a violence-reporting tweet, illustrates how this can occur without bullying intent. A false negative leaves cyberbullying content unflagged. The costs of these two error types may differ, and the interpretation of a message may depend on broader conversational context that a single-message classifier cannot access. For these reasons, we do not consider the proposed system suitable for autonomous moderation decisions. Instead, we frame it as a decision-support tool for human moderators that are intended to surface content for review rather than act on it directly. Fairness across demographic groups and communication styles, including the community-identity and stylistic uses of leet-speak discussed above, was not evaluated in this study. We recommend auditing false-positive rates across relevant subpopulations before any real-world deployment to reduce the risk of over-moderation affecting particular communities.

## 6. CONCLUSION

The framework presented in this paper was tested on two publicly available English datasets: the Mendeley "A Comprehensive Dataset for Automated Cyberbullying Detection" (D1, 90,356 raw instances) and the Kaggle dataset for Cyberbullying Classification (D2, 47,692 raw instances). Each dataset was balanced to 15,592 instances for this study. Six BERT-family encoders were compared systematically across three pretraining domains on clean and synthetically obfuscated test sets to evaluate their robustness to the evaluated synthetic leet-speak transformations on these two corpora.

The results demonstrate that BERTweet outperformed the hate-speech-specialized and general-domain encoders under the evaluated synthetic leet-speak transformations, with Baseline leet_test macro-F1 scores of 0.6700 on D1

and 0.7150 on D2. The conditional normalization pipeline increased macro-F1 by 18.13% points on D1 (0.6700 → 0.8513) and by 13.92% points on D2 (0.7150 → 0.8542), with leet_test ROC-AUC values of 0.9275 and 0.9238, respectively. The conditional design did not degrade performance on unobfuscated inputs: clean_test macro-F1 remained unchanged on D1 (0.8643) and increased slightly on D2 (0.8706). The corresponding descriptive heuristic lower bounds were 0.8413 for D1 and 0.8500 for D2, both remaining above their respective baseline means. These improvements reflect robustness against the specific synthetic leet-speak transformations evaluated in this study, rather than a general guarantee of performance on naturally occurring or adversarially adaptive obfuscation.

Beyond the reported metrics, these results have practical significance for content-moderation pipelines: at the tested scale and hardware configuration, a lightweight normalization step applied conditionally substantially recovered detection performance the evaluated synthetic leet-speak transformations, while the measured computational differences were small relative to run-to-run variation. As discussed in Section 5, these findings should be understood within the boundaries of the present evaluation, which used synthetic leet-speak, English-only corpora, and a fixed substitution scheme. The proposed system is intended as a decision-support component for human moderators rather than as an autonomous moderation system.

Future work should further develop and validate the proposed pipeline using naturally occurring leet-speak from actual social-media platforms, which may contain more complex and creative substitution patterns than the synthetic tests used here. Expanding the evaluation to other character-level obfuscation types, such as homoglyph attacks and deliberate misspellings, as well as to multilingual cyberbullying corpora, is another direction for future work. Furthermore, using a learned character-level model trained on organically obfuscated text instead of the rule-based normalization map could reduce dependence on manually created substitution vocabularies.

## 7. CODE AND DATA AVAILABILITY

Both datasets used in this study are publicly available: D1 from Mendeley Data [8] and D2 from Kaggle [9]. The code implementing the proposed pipeline, including the preprocessing, detector, normalization, training, and evaluation procedures, is available from the authors upon reasonable request.

## REFERENCES


[1] A. G. Philipo, D. S. Sarwatt, J. Ding, M. Daneshmand and H. Ning. "Cyberbullying detection: Exploring datasets, technologies, and approaches on social media platforms". *ACM Computing Surveys*, vol. 58, pp. 1-35, 2025.

[2] C. Raj, A. Agarwal, G. Bharathy, B. Narayan and M. Prasad. "Cyberbullying detection: Hybrid models based on machine learning and natural language processing techniques". *Electronics*, vol. 10, no. 22, p. 2810, 2021.

[3] M. Umer, E. A. Alabdulqader, A. A. Alarfaj, L. Cascone and M. Nappi. "Cyberbullying detection using PCA extracted GLOVE features and RoBERTaNet transformer learning model". *IEEE Transactions on Computational Social Systems*, vol. 12, no. 5, pp. 3881-3890, 2025.

[4] D. Q. Nguyen, T. Vu and A. T. Nguyen. "BERTweet: A pre-trained language model for English tweets". In: *EMNLP 2020 - Conference on Empirical Methods in Natural Language Processing, Proceedings of Systems Demonstrations*, pp. 9-14, 2020.

[5] I. V. De Mendizabal, X. Vidriales, V. Basto-Fernandes, E. Ezpeleta, J. R. Méndez and U. Zurutuza. "Deobfuscating leetspeak with deep learning to improve spam filtering". *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 8, no. 4, pp. 46-55, 2023.

[6] T. Gröndahl, L. Pajola, M. Juuti, M. Conti and N. Asokan. "All you Need is 'Love': Evading Hate Speech Detection". In: *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, pp. 2-12, 2018.

[7] D. Marzoog and H. Çakir. "Deobfuscating Iraqi arabic leetspeak for hate speech detection using AraBERT and hierarchical attention network (HAN)". *Electronics (Switzerland)*, vol. 14, no. 21, p.4318, 2025.

[8] N. Ejaz, F. Razi and S. Choudhury. "Towards comprehensive cyberbullying detection: A dataset incorporating aggressive texts, repetition, peerness, and intent to harm". *Computers in Human Behavior*, vol. 153, p. 108123, 2024.

[9] J. Wang, K. Fu and C. T. Lu. "SOSNet: A Graph Convolutional Network Approach to Fine-Grained Cyberbullying Detection". In: *Proceedings - 2020 IEEE International Conference on Big Data, Big Data 2020*, pp. 1699-1708, 2020.

[10] B. Ogunleye and B. Dharmaraj. "The use of a large language model for cyberbullying detection". *Analytics*, vol. 2, no. 3, pp. 694-707, 2023.

[11] Y. Kumar, K. Huang, A. Perez, G. Yang, J. J. Li, P. Morreale, D. Kruger and R. Jiang. "Bias and cyberbullying detection and data generation using transformer artificial intelligence models and top large language models". *Electronics*, vol. 13, no. 17, p. 3431, 2024.

[12] A. G. Philipo, D. Sebastian Sarwatt, J. Ding, M. Daneshmand and H. Ning. "Assessing text classification methods for cyberbullying detection on social media platforms". *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 7602-7616, 2025.

[13] M. Abusaqer, J. Saquer and H. Shatnawi. "Efficient hate speech detection: evaluating 38 models from traditional methods to transformers". In: *ACMSE 2025 - Proceedings of the 2025 ACM Southeast Conference*, Association for Computing Machinery Inc., New York, pp. 203-213, 2025.

[14] K. Gutiérrez-Batista, J. Gómez-Sánchez and C. Fernandez-Basso. "Improving automatic cyberbullying detection in social network environments by fine-tuning a pre-trained sentence transformer language model". *Social Network Analysis and Mining*, vol. 14, no. 1, p. 136, 2024.

[15] W. Sharif, S. Abdullah, S. Iftikhar, D. Al-Madani and S. Mumtaz. "Enhancing hate speech detection in the digital age: A novel model fusion approach leveraging a comprehensive dataset". *IEEE Access*, vol. 12, pp. 27225-27236, 2024.

[16] J. Fattahi, F. Sghaier, M. Mejri, S. Bahroun, R. Ghayoula and E. Manai. "Cyberbullying detection using bag-of-words, TF-IDF, parallel CNNs and BiLSTM neural networks". *Frontiers in Artificial Intelligence and Applications*, vol. 389, pp. 72-83, 2024.

[17] E. Alikhashashneh, A. Almomani, K. M. Nahar, N. Shatnawi, A. Almomani, M. Alauthman, S. Bansal and S. H. Pan. "Unified transformer framework for automated cyberbullying detection". *International Journal of Cloud Applications and Computing*, vol. 15, no. 1, pp. 1-29, 2025.

[18] T. H. H. Aldhyani, M. H. Al-Adhaileh and S. N. Alsubari. "Cyberbullying identification system based deep learning algorithms". *Electronics*, vol. 11, no. 20, p. 3273, 2022.

[19] A. F. Alqahtani and M. Ilyas. "An ensemble-based multi-classification machine learning classifiers approach to detect multiple classes of cyberbullying". *Machine Learning and Knowledge Extraction*, vol. 6, no. 1, pp. 156-170, 2024.

[20] T. Ahmed, S. Ivan, M. Kabir, H. Mahmud and K. Hasan. "Performance analysis of transformer-based architectures and their ensembles to detect trait-based cyberbullying". *Social Network Analysis and Mining*, vol. 12, no. 1, p. 99, 2022.

[21] M. T. Hasan, M. A. E. Hossain, M. S. H. Mukta, A. Akter, M. Ahmed and S. Islam. "A review on deep-learning-based cyberbullying detection". *Future Internet*, vol. 15, no. 5, p. 179, 2023.

[22] Z. Mansur, N. Omar and S. Tiun. "Twitter hate speech detection: A systematic review of methods, taxonomy analysis, challenges, and opportunities". *IEEE Access*, vol. 11, pp. 16226-16249, 2023.

[23] G. Ramos, F. Batista, R. Ribeiro, P. Fialho, S. Moro, A. Fonseca, R. Guerra, P. Carvalho, C. Marques and Silva C. "A comprehensive review on automatic hate speech detection in the age of the transformer". *Social Network Analysis and Mining*, vol. 14, no. 1, p. 204, 2024.

[24] F. Barbieri, J. Camacho-Collados, L. Neves and L. Espinosa-Anke. "TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification". In: *Findings of the Association for Computational Linguistics Findings of ACL: EMNLP 2020*. pp. 1644-1650, 2020.

[25] T. Caselli, V. Basile, J. Mitrović and M. Granitzer. "HateBERT: Retraining BERT for Abusive Language Detection in English". In: *WOAH 2021 - 5th Workshop on Online Abuse and Harms, Proceedings of the Workshop*, pp. 17-25, 2021.

[26] B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal and A. Mukherjee. "HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection". In: *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, vol. 17A, pp. 14867-14875, 2020.

[27] J. Devlin, M. W. Chang, K. Lee and K. Toutanova. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North*, pp. 4171-4186, 2019.

[28] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov. "*RoBERTa: A Robustly Optimized BERT Pretraining Approach*". 2019. Available from: https://arxiv.org/abs/1907.11692 [Last accessed on 2026 Jun 17].

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna. "Rethinking the Inception Architecture for Computer Vision". *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016, pp. 2818-2826, 2016.

[30] A. Muminovic. "Moderating harm: Benchmarking large language models for cyberbullying detection in YouTube comments". *International Journal of Computer Applications*, vol. 187, no. 25, pp. 1-9, 2025.