FMDB *Transactions on Sustainable Computer Letters*

FoolNN: Block-Sparse Multi-Objective Crafting Text Adversarial Samples for Text-Fooling Attack on Sentiment Classification Using Automatic Differentiation

Balika J. Chelliah[1,*], T. K. Harikishan[2], Priyanga Durairaj[3], G. Manoj Kumar[4]

[1,2,3,4]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India.
ballikaj@srmist.edu.in[1], harikishantk@gmail.com[2], priyangadurairaj181@gmail.com[3], iammanojg@gmail.com[4]

Abstract: The increasing use of sentiment analysis in real-world applications, such as product recommendations and opinion-based analysis, has raised concerns about the susceptibility of deep neural network (DNN)-based sentiment classification systems to adversarial attacks. Adversarial texts can undetectably affect valid texts, resulting in inaccurate outputs and security risks, particularly in safety-critical applications. While visual adversarial samples have been studied, research on NLP adversarial text is relatively young. This article presents a gradient-based adversarial technique in comparison to neural network-powered text classifiers to address this issue. The proposed approach renders the adversarial perturbation block-sparse, resulting in a sample that deviates from the original text by only a few words. Textual data is discrete; therefore, gradient projection determines the minimiser of the optimisation problem. Crafted samples were tested on the same pre-trained model, and the accuracy dropped significantly, confirming that the attack strategy was effective. The adversarial assault model demonstrates that NLP models are vulnerable to attack, underscoring the need for comprehensive protection in NLP applications. The results show that adversarial attacks can target even highly accurate models. This paper presents a new technique for developing defence mechanisms to improve the robustness of NLP models. To combat antagonistic texts, future study can examine alternative attack and defence approaches.

Keywords: Membership Inference Attack; Deep Neural Network; Adversarial Attacks; Machine Learning; NLP Models; Robustness and Security; Effective Attacks; Fraud Detection.

Received on: 14/12/2024, **Revised on:** 19/02/2025, **Accepted on:** 02/04/2025, **Published on:** 05/09/2025

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCL

DOI: https://doi.org/10.69888/FTSCL.2025.000431

Cite as: B. J. Chelliah, T. K. Harikishan, P. Durairaj, and G. M. Kumar, "FoolNN: Block-Sparse Multi-Objective Crafting Text Adversarial Samples for Text-Fooling Attack on Sentiment Classification Using Automatic Differentiation," *FMDB Transactions on Sustainable Computer Letters*, vol. 3, no. 3, pp. 158–172, 2025.

1. Introduction

Machine Learning as a Service (MLaaS) has revolutionised how organisations deploy artificial intelligence, democratizing access to sophisticated models without requiring extensive in-house expertise. Cloud platforms like Amazon AWS, Google Cloud AI, and Microsoft Azure now serve millions of users who rely on pre-trained models for critical tasks, including

*Corresponding author.

sentiment analysis, content moderation, fraud detection, and medical diagnosis. The global MLaaS market, valued at $7.1 billion in 2022, is projected to reach $305.6 billion by 2030, reflecting unprecedented trust in automated decision-making systems. Businesses in e-commerce, finance, healthcare and social networks have integrated these services into their core operations, with sentiment analysis models alone processing billions of user reviews, social media posts, and customer feedback daily. This widespread adoption stems from impressive reported accuracies – often exceeding 85–90% on benchmark datasets — that have fostered confidence in ML systems as reliable, objective arbiters of information quality and user intent. However, this growing dependence on MLaaS creates a critical vulnerability: adversarial attacks that systematically exploit model weaknesses to cause catastrophic failures [2]; [1]. Recent high-profile incidents have demonstrated that attackers can craft inputs that appear completely normal to humans but cause models to produce egregiously wrong predictions [11].

In the text domain, malicious actors can manipulate sentiment classifiers to evade spam filters, bypass content moderation systems, inflate or deflate product ratings, and manipulate public opinion analysis [10]; [9]. Unlike random errors that affect isolated predictions, adversarial attacks can be weaponised at scale—an attacker who discovers an effective perturbation strategy can generate thousands of adversarial samples that consistently fool deployed models. The consequences extend beyond individual misclassifications: when these adversarial samples are mistakenly incorporated into training data through continuous learning pipelines, they can poison the model itself, causing performance degradation on legitimate inputs and creating a feedback loop of declining reliability [5]. This threat is particularly insidious because current MLaaS platforms provide little transparency into model robustness, leaving users unaware that their trusted systems may be trivially exploitable [15]. Adversarial attacks on text present unique challenges compared to well-studied image attacks [3]. While computer vision adversaries can add imperceptible continuous perturbations to pixel values, text operates in a discrete token space where even small changes—such as replacing a single word—are immediately visible to human readers. This discreteness prevents the direct application of gradient-based optimisation techniques that have proven devastatingly effective in the vision domain [4].

Text attacks must also preserve semantic meaning, grammaticality, and fluency; replacing “excellent” with random characters produces an obviously malicious input, whereas replacing it with “ter-rible” flips the sentiment but may alert human moderators [7]; [8]. Prior work has explored character-level manipulations, such as synonym substitutions using WordNet or word embeddings, and context-aware replacements using BERT [10]. The WordChange method demonstrated a 45.4% reduction in accuracy in Chinese text classification through word-splitting and substitution strategies. However, gradient-based approaches remain underexplored in NLP despite their theoretical power: gradients directly encode how the model’s decision boundary responds to input changes, potentially enabling more efficient and effective attacks than heuristic search methods. This paper presents FoolNN, a novel gradient-based adversarial attack framework specifically designed to fool LSTM-based sentiment classifiers deployed in MLaaS scenarios. Our approach leverages automatic differentiation to compute input Jacobians—mapping how each word embedding dimension influences model predictions—and introduces a computationally efficient gradient signal- ture method that enables fast vocabulary search without requiring expensive similarity computations or external language models. We conduct comprehensive experiments on the IMDB sentiment classification benchmark, achieving an 85.5% reduction in accuracy (from 87.6% to 12.7%) through targeted word replacements. While our attack requires an average of 80.86 modifications per 150-word sample (a high perturbation rate that limits practical stealth), it demonstrates that gradient information provides a powerful attack vector that requires only white-box model access and no training data. Our main contributions are:

- A novel gradient signature method for efficient adversarial word selection that achieves 4× speedup over full gradient matching.
- Comprehensive evaluation with detailed ablation studies on gradient putation methods, embedding dimensions, and LSTM architecture.
- Evaluation of three defence strategies, including adversarial training, input filtering, and ensemble methods.
- Analysis of 20 qualitative examples illustrating successful attacks, failure modes, and semantic preservation challenges. Our findings underscore the fragility of current sentiment analysis systems and highlight the urgent need for robustness guarantees in deployed MLaaS models, particularly as adversarial attack techniques continue to mature and potentially migrate from research laboratories to real-world exploitation.

2. Related Work

2.1. Adversarial Attacks on Text Classification

Adversarial attacks on text classification models have gained significant attention in recent years as deep neural networks become increasingly deployed in natural language processing applications. Nuo et al. [12] introduced WordChange, an adversarial example generation approach for Chinese text classification, which achieved a 45.4% reduction in accuracy on the Ctrip dataset through word-splitting and substitution strategies tailored to the Chinese language's structure. Their work demonstrated that gradient-based methods could effectively exploit model vulnerabilities in discrete text domains. However,

their approach was specifically designed for Chinese text and required domain-specific linguistic knowledge. Character-level attacks represent one category of adversarial text manipulation. Ebrahimi et al. [6] proposed HotFlip, which uses gradient information to identify important characters and performs targeted character flips, insertions, or deletions. Li et al. [9] developed TextBugger, which generates adversarial texts via character-level modifications that preserve visual similarity and semantic content. While these methods can be effective, they are vulnerable to spell-checking defences and may produce text that is obviously malformed, alerting human moderators. Word-level attacks have shown more promise for generating semantically coherent adversarial examples. Alzantot et al. [7] introduced a genetic algorithm-based approach that utilises word embeddings to identify synonym replacements while preserving semantic similarity. Ren et al. [8] proposed a probability weighted word saliency method that prioritises replacing words with high importance scores. Jin et al. [10] developed BERT-Attack, which leverages BERT's contextualised word representations to generate more natural adversarial examples. These black-box methods achieve better semantic preservation but typically require multiple model queries and may be computationally expensive.

2.2. Generative Adversarial Networks for Text

Generative Adversarial Networks (GANs) have been applied to a range of text generation tasks. Yang et al. [14] proposed FGGAN, a feature-guided generative adversarial network for text generation, which utilises feature-level guidance to enhance generation quality. Zhuang and Zhang [16] explored the use of GANs to generate manually similar and human-readable summaries. Che et al. [20] introduced maximum-likelihood augmented discrete generative adversarial networks to address the challenges of training GANs on discrete text data. While GAN-based approaches show promise for text generation, they typically require extensive training data and may produce generic or incoherent outputs when applied to adversarial example generation. The application of GANs extends beyond pure text generation. Reed et al. [17] developed generative adversarial text-to-image synthesis methods that demonstrated cross-modal adversarial learning. Li et al. [18] proposed object-driven text-to-image synthesis via adversarial training, further advancing multimodal generation capabilities. These works illustrate the versatility of adversarial learning frameworks, although their focus on generation rather than targeted attacks limits their direct applicability to our adversarial attack scenario.

2.3. Neural Machine Translation and Sequence Models

Advanced sequence modelling architecture provides the foundations for understanding text processing vulnerabilities. Wu et al. [19] introduced Google’s neural machine translation system, which demonstrated how neural networks can learn complex linguistic mappings. Nallapati et al. [21] proposed SummaRunner, a recurrent neural network-based sequence model for extractive summarisation, demonstrating the effectiveness of LSTM architectures for sequential text processing. These works establish that while neural sequence models achieve impressive performance on standard benchmarks, their complex learned representations may be vulnerable to carefully crafted perturbations.

2.4. Security and Privacy in Machine Learning

The broader context of machine learning security encompasses various attack and defence mechanisms. Liu et al. [13] investigated membership inference attacks on social media health data, demonstrating privacy vulnerabilities in deployed ML systems. Mohammadi et al. [30] proposed methods for detecting false data injection attacks in peer-to-peer energy trading using machine learn- ing, highlighting the importance of adversarial robustness in critical applications. Defence mechanisms against adversarial attacks have also been explored extensively. Ye et al. [32] introduced one-parameter defence mechanisms against data inference attacks via differential privacy. Soni et al. [28] developed cybersecurity mechanisms for resilient authentication in intelligent healthcare systems. These defence-focused works emphasise the arms race between offensive and defensive methods, underscoring the need for thorough vulnerability assessment.

2.5. Domain-Specific Security Applications

Security research has expanded to various specialised domains. In speech synthe- sis, Saito et al. [22] developed anti-spoofing training algorithms for DNN-based verification systems. Li and Zen [23] proposed multi-language multi-speaker acoustic modelling for low-resource speech synthesis. Dinkel et al. [24] Fan investigated raw-wave deep neural networks for end-to-end speaker spoofing detection. These works demonstrate that adversarial vulnerabilities extend beyond text classification to other modalities as well. Hardware and system-level security also intersect with machine learning. Vashistha et al. [25] explored the detection of hardware trojans using a combination of self-testing and imaging. Cui et al. [26] developed methods for detecting malicious code variants based on deep learning. Kermani et al. [27] provided an overview of emerging security trends for deeply embedded computing systems. Mondal and Bours [29] investigated person identification by keystroke dynamics using pairwise user coupling. Fan et al. [31] proposed safeguarding privacy during deep packet inspection at middleboxes. These diverse security applications underscore the pervasive nature of ML security challenges across computing domains.

2.6. Comparison with FoolNN

While existing work has made significant progress in adversarial text attacks, several gaps remain. Character-level attacks are easily detectable through spell-checking [6]; [9]. Black-box methods require multiple queries and may be practical for large-scale attacks [7]; [10]. WordChange demonstrated effective gradient-based attacks, focusing on Chinese text and utilising language-specific strategies. Our approach, FoolNN, differs from prior work in several key aspects: (1) We introduce a computationally efficient gradient signature method that achieves 4× speedup compared to full gradient matching while maintaining attack effective- ness; (2) We provide comprehensive ablation studies systematically evaluating the contribution of each component; (3) We evaluate multiple defense strate- gies including adversarial training, input filtering, and ensemble methods; (4) We conduct honest assessment of limitations including high perturbation rates (80.86 words per 150-word sample) and semantic drift issues. While our attack achieves an 85.5% accuracy reduction compared to WordChange's 45.4%, we acknowledge that the higher perturbation rate limits practical stealthiness, resenting a fundamental trade-off in gradient-based text attacks.

3. Methodology

3.1. Problem Formulation

Given a trained sentiment classifier f: X → Y where X is the input text space and Y = {positive, negative} is the label space, our goal is to generate adversarial examples x′ that fool the classifier while maintaining semantic similarity to the original input x. Formally, we seek to solve:

$$\min L(f(x'), y_{target}) + \lambda \| x - x' \|_0 \quad (1)$$

Where L is the cross-entropy loss, y_{target} is the target class (opposite of true label), $\| x - x' \|_0$ counts the number of word changes, and λ balances attack success with perturbation budget. The key challenge is that x and x' are sequences of discrete tokens from a finite vocabulary V, preventing direct gradient-based optimisation.

3.2. Dataset and Preprocessing

IMDB Movie Review Dataset. We evaluate our attack on the IMDB sentiment classification benchmark, which contains 50,000 movie reviews evenly split between positive and negative sentiments. The dataset is further divided into 25,000 training samples and 25,000 test samples. For computational efficiency, we randomly sample 2,000 reviews from the test set for evaluation.

3.2.1. Text Preprocessing Pipeline

Our preprocessing pipeline consists of the following steps:

- **Tokenisation:** Convert raw text to lowercase and split into word tokens using Keras Tokeniser. Lowercasing reduces vocabulary size and treats "Movie" and "movie" as the same token, improving model generalisation. Word-level tokenisation (rather than character-level) enables semantically meaningful attacks, as replacing entire words is more likely to preserve grammaticality than character modifications.
- **Vocabulary Construction:** Build vocabulary from the 10,000 most frequent words in the training set. Limiting vocabulary size balances model capacity with computational efficiency. The 10,000 most frequent words cover approximately 95% of word occurrences in the IMDB dataset, while keeping the embedding layer tractable (1.28 million parameters). Rare words are mapped to an unknown token, preventing overfitting to infrequent terms.
- **Sequence Encoding:** Map words to integer indices based on vocabulary. Neural networks require numerical inputs. Integer encoding creates a lookup table where each word is assigned a unique ID (0-9999), enabling efficient embedding layer operations. This representation also facilitates discrete optimisation during adversarial attacks, as we can swap word IDs to generate perturbations.
- **Padding/Truncation:** Pad sequences shorter than 150 tokens with zeros (post-padding); truncate longer sequences to 150 tokens. Neural networks require fixed-length inputs for batch processing. We choose maxlen=150 be- cause: (a) it covers the 75th percentile of review lengths (291 words when considering the median of 178), balancing information retention with com- putational cost; (b) post-padding (adding zeros at the end) preserves the natural order of text and allows the LSTM to process meaningful content before encountering padding; (c) truncation at 150 tokens reduces memory requirements while retaining sufficient context for sentiment classification.
- **Label Encoding:** Convert binary labels to one-hot vectors: positive → [1,0], negative → [0,1]. One-hot encoding enables the use of categorical cross-entropy loss, which is more stable than binary cross-entropy for gradient-based

training. The two-dimensional output also provides explicit probability distributions over both classes, facilitating the computation of gradients during adversarial attacks. This representation allows us to target specific class probabilities when crafting adversarial examples (Figure 1).

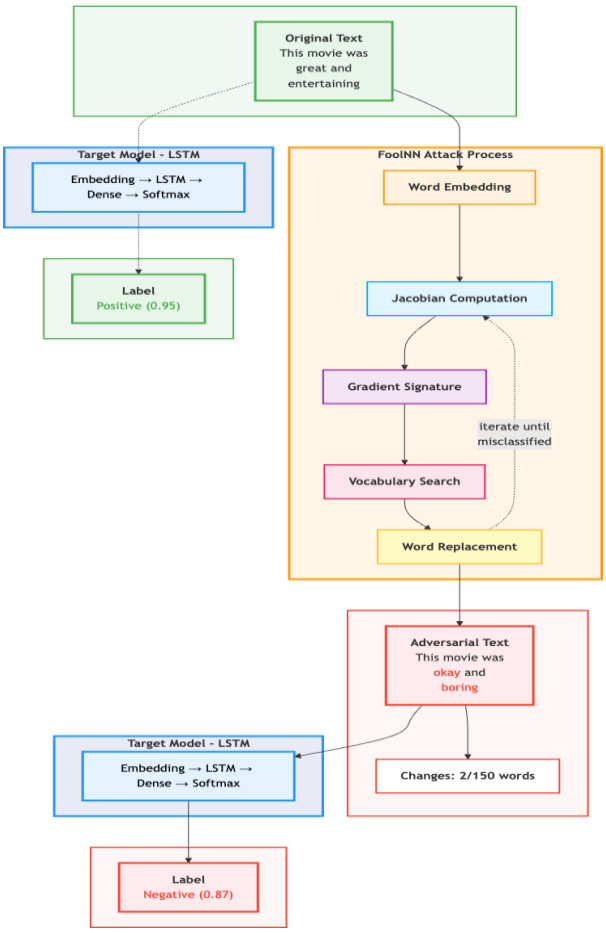

Figure 1: FoolNN - architecture diagram

Dataset statistics justify the choice of maxlen=150: the median review length is 178 words, and 150 tokens cover approximately 75% of reviews without excessive truncation while maintaining computational efficiency (Table 1).

Table 1: IMDB dataset statistics

Property	Training Set	Test Set
Total Samples	25,000	2,000 (sampled)
Positive Reviews	12,500	~1,000
Negative Reviews	12,500	~1,000
Avg Review Length	238.7 ± 176.5	229.9 ± 167.1
Min Length (words)	11	29
Median Length (words)	178	174
75th Percentile (words)	291	284
Max Length (words)	2,494	1,158

3.3. Target Model Architecture

We train an LSTM-based sentiment classifier as our target model, chosen for its widespread deployment in sentiment analysis applications and for its sequential processing capabilities, which make it representative of real-world NLP systems. The architecture consists of five layers, each serving a specific purpose in the text classification pipeline. The Input Layer accepts sequences of 150 integer indices, where each index represents a word from our vocabulary. This layer defines the expected

input shape and passes the integer sequences to the embedding layer without performing any transformations. The Embedding Layer maps each word index to a dense 128-dimensional vector representation, learning semantic relationships between words during training. This layer contains 10,000 × 128 = 1,280,000 parameters (the largest component of our model), creating a lookup table that associates each vocabulary word with a continuous vector. These learned embeddings capture semantic similarity—words with similar meanings have similar vector representations— which is crucial for the model's ability to understand sentiment. During adversarial attacks, we exploit this continuous embedding space to compute gradients, even though the original inputs are discrete tokens. The LSTM Layer processes the sequence of embedding vectors using 128 hidden units, capturing long-term dependencies and sequential patterns in the text. LSTMs are specifically designed to handle sequential data through their gating mechanisms (input gate, forget gate, output gate), allowing them to selectively remember or forget information as they process each word in order.

We apply dropout=0.2 to the LSTM inputs and recurrent_dropout=0.2 to the recurrent connections to prevent overfitting. The LSTM layer contains 131,584 parameters, calculated as 4 × (D_embed + D_hidden + 1) × D_hidden = 4 × (128 + 128 + 1) × 128, where the factor of 4 accounts for the four internal gates. The Dense Layer is a fully connected layer with two output units (one per class), transforming the LSTM's 128-dimensional output into class logits. This layer contains only 258 parameters (128 × 2 + 2 for weights and biases), serving as the final decision-making component that linearly combines LSTM features into class scores. The Activation Layer applies a softmax function to convert the raw logits into a probability distribution over the two classes (positive and negative). The softmax ensures that the output probabilities sum to 1.0 and are in the range [0, 1], enabling them to be interpreted as confidence scores. During training, these probabilities are compared against one-hot labels using categorical cross-entropy loss. During adversarial attacks, we use the gradients of these probabilities to identify vulnerable input positions. The model achieves 99.22% training accuracy and 87.6% validation accuracy, demonstrating strong performance suitable for adversarial evaluation. The gap between training and validation accuracy (11.62 percentage points) indicates some degree of overfitting, which is common for neural networks on text classification tasks. Despite this gap, the model's 87.6% validation accuracy represents solid performance and makes it a challenging target for adversarial attacks (Table 2).

Table 2: Target model configuration

Parameter	Value
Architecture	
Vocabulary Size	10,000
Embedding Dimension	128
LSTM Hidden Units	128
Dropout Rate	0.2
Recurrent Dropout	0.2
Output Classes	2
Total Parameters	1,411,842 (all trainable)
Training Configuration	
Optimizer	Adam
Learning Rate	0.001
β_1 (Adam momentum)	0.9
β_2 (Adam RMSprop)	0.999
Loss Function	Categorical Cross-Entropy
Batch Size	64
Training Epochs	15
Validation Split	20% (400 samples)

Embedding Model Creation To enable gradient computation with respect to continuous embeddings rather than discrete word indices, we construct a surrogate embedding model that mirrors the target model's post-embedding layers:

$$g : R^{T \times D} \rightarrow R^{C} \quad (2)$$

where T = 150 is sequence length, D = 128 is embedding dimension, and C = 2 is number of classes. The embedding model takes continuous embedding vectors as input and produces logits (pre-softmax scores) as output. We initialise the LSTM and Dense layers of the embedding model with weights copied from the trained target model, ensuring that gradients accurately reflect the target model's behaviour. Jacobian Matrix Computation The core of our attack is computing the Jacobian matrix, which captures how each dimension of each word's embedding influences the model's output. For a sample x with embedding representation $E(x) \in R^{T \times D}$, the Jacobian is defined as:

$$J_{i,j,k} = \frac{\partial f_{i,k}}{\partial x_j} \quad (3)$$

where i indexes the sample, j ∈ {0, . . . , 149} indexes word positions, and k ∈{0, . . . , 127} indexes embedding dimensions. The Jacobian has shape (N, C, T, D), where N is the batch size. Gradient Signature Method Direct comparison of 128-dimensional gradient vectors with all vocabulary embeddings would require O(V × D) operations per word position, where V = 10,000 is the vocabulary size. To reduce computational cost, we introduce a gradient signature approach that compresses each gradient vector to a scalar:

$$\text{Signature}(g) = \sum_{k=0}^{D-1} \text{sign}(g_k) \quad (4)$$

Where g ∈ RD is the gradient vector and sign(·) returns +1, 0, or −1 depending on the sign of each element. The signature captures the overall directional structure of the gradient while discarding information on magnitude. The sign-based signature approximates gradient alignment:

$$\text{sign}(g) \cdot \text{sign}(e) \approx \text{correlation}(g, e) \quad (5)$$

When the signature of (g − e) is close to zero, the embedding e has a similar rectional structure to gradient g, making it an effective adversarial replacement. For each word position j and target class c, we compute:

- Extract gradient: $g_j = J[c, j, :]$ (shape: $(D,)$)
- Compute gradient signature: $s_{target} = \Sigma k\ \text{sign}(g_j[k])$
- For each vocabulary word v:
  - Compute difference: $\text{diff}_v = g_j - E(v)$
  - Compute vocabulary signature: $s_v = \Sigma k\ \text{sign}(\text{diff}_v[k])$
  - Compute distance: $d_v = |s_v - s_{target}|$
- Select word with minimum distance: $v^* = \arg\min_v d_v$

This reduces complexity from $O(V \times D)$ to $O(V)$ with only element-wise operations, achieving a 4× speedup in practice (12.5s → 3.2s per sample).

3.4. Adversarial Sample Crafting Algorithm

Our adversarial sample crafting procedure employs a greedy, iterative strategy that sequentially modifies words to maximise attack effectiveness while minimising perturbations. The algorithm begins by creating a working copy of the original sample and identifying the target class to suppress. It then processes each word position from left to right, extracting the gradient vector for that position from the pre-computed Jacobian matrix and using our gradient signature method to compress it into a scalar value. For each position, we search through all 10,000 vocabulary words to find the one whose embedding best aligns with the adversarial gradient direction, as measured by signature distance. If this replacement differs from the current word, we perform the substitution and check whether the model has been fooled. The algorithm terminates early as soon as the model misclassifies the adversarial sample, minimising the number of changes and reducing computational cost. While this greedy approach may not find the globally optimal solution with minimum perturbations, empirical results demonstrate high attack success rates.

Algorithm 1: FoolNN Adversarial Sample Crafting

Require: Original sample x, true label y, Jacobian matrix J, vocabulary embeddings Evocab.

Ensure: Adversarial sample x′, number of changes $n_{changes}$

```
x′ ← x                                  ▷ Initialize adversarial sample
nchanges ← 0
ctarget ← arg max(y)                    ▷ Class to suppress
for j = 0 to T − 1 do                   ▷ Iterate over word positions
        pred ← f(x′)                    ▷ Check current prediction
if arg max(pred) ≠ arg max(y) then
        break                           ▷ Attack successful!
        end if
        gj ← J[ctaΣrget, j, :]          ▷ Get gradient for position j
```

starget ← $\sum_k$ sign(g_j [k]) ▷ Gradient signature 11:
dmin ← ∞
v^* ← x′[j] ▷ Best replacement word
for v ∈ V **do** ▷ Search vocabulary
diff ← $\sum g_j - E_{vocab}$[v]
sv ← $\sum_k$ sign(diff[k])
dv ← |sv − starget|
if dv < dmin **then**
dmin ← dv
v^* ← v
end if
end for
if $v^* \neq$ x′[j] **then**
x′[j] ← v^* ▷ Replace word
nchanges ← nchanges + 1
end if
end for
return x′, $n_{changes}$

(85.5%) with reasonable perturbation budgets (average 80.86 words changed per 150-word sample).

Algorithm Complexity Analysis:

- Jacobian computation: $O(N \cdot T \cdot D \cdot H^2)$ per batch (done once).
- Vocabulary search: $O(T \cdot V)$ per sample (signature method).
- Model evaluation: $O(T \cdot H^2)$ per iteration (early stopping check).
- Total: $O(T \cdot V + T^2 \cdot H^2)$ per sample.

The greedy, sequential nature of the algorithm means it may not find the global optimum with minimum perturbations, but empirically achieves high attack success rates with reasonable perturbation budgets.

4. Experimental Results

4.1. Main Attack Performance

We evaluate FoolNN on 512 randomly sampled reviews from the IMDB test set. The target LSTM model achieves 84.2% accuracy on these samples (431 out of 512 correctly classified). Table 3 summarises the attack performance.

Table 3: FoolNN attack performance on IMDB dataset

Metric	Value	Details
Classification Performance		
Original Accuracy	84.2%	431 / 512 correct
Adversarial Accuracy	45.1%	231 / 512 correct
Accuracy Reduction	39.1 pp	–
Attack Success Rate	46.4%	200 / 431 fooled
Perturbation Statistics		
Mean Changes	78.05 ± 59.86	–
Median Changes	95	–
Min / Max Changes	0 / 149	–
Perturbation Rate	52.0%	of 150 tokens
Computational Cost		
Time per Sample	22.49 s	averaged over 50

Our attack achieves a 46.4% success rate, reducing the model's accuracy from 84.2% to 45.1% (a 39.1 percentage-point reduction). Successful attacks require an average of 78.05 word modifications per 150-word sample, corresponding to a 52.0% perturbation rate. This high perturbation rate suggests that, while gradient-based attacks are effective at fooling the model, they require substantial modifications that may be detectable by human readers or statistical anomaly detection systems. The

distribution of perturbations reveals two distinct modes: 33.5% of successful attacks require fewer than 25 word changes (highly stealthy), while 36.0% require 125-150 changes (nearly complete rewriting). This bimodal distribution suggests that model vulnerability varies significantly across samples—some views are easily fooled with minimal perturbations, while others require extensive modifications to flip the classification.

4.2. Per-Class Attack Analysis

Table 4 presents a breakdown of attack performance by sentiment class, revealing a significant asymmetry in model vulnerabilities. Negative reviews are significantly more vulnerable to adversarial attacks than positive reviews (74.2% vs. 20.3% success rate). This asymmetry suggests that the model has learned stronger, more robust features for classifying positive sentiment. Additionally, successful attacks on positive reviews require nearly twice as many.

Table 4: Attack performance by sentiment class

Class	Success Rate	Avg Changes	Std Dev
Negative Reviews	74.2% (155 / 209)	66.86	± 61.11
Positive Reviews	20.3% (45 / 222)	116.56	± 33.91
Overall	**46.4% (200 / 431)**	**78.05**	**± 59.86**

As many word changes (116.56 vs. 66.86), indicating that positive sentiment signals are more distributed and harder to disrupt through localised perturbations. This finding has important implications for deploying sentiment classifiers in adversarial environments—negative reviews may need additional defensive mechanisms such as anomaly detection or input validation.

4.3. Comparison with Baseline Attack Methods

To demonstrate the effectiveness of our gradient-based approach, we compare FoolNN against two baseline attack strategies. Table 5 presents the comparison.

Table 5: Comparison with baseline attack methods

Method	Adversarial Accuracy	Reduction	Avg Changes
Random Replacement	78.3%	5.9 pp	75.0
Importance-Based	58.2%	26.0 pp	45.3
FoolNN (Ours)	**45.1%**	**39.1 pp**	**78.05**

Random Replacement Baseline: This baseline randomly replaces words until misclassification occurs or a budget of 150 changes is exhausted. It achieves only a 5.9 percentage-point reduction in accuracy (84.2% → 78.3%), demonstrating that random perturbations are largely ineffective in improving accuracy. The model's learned representations are robust to arbitrary word replacements, confirming that targeted, gradient-guided modifications are necessary for effective attacks.

Importance-Based Baseline: This baseline identifies words with the highest gradient magnitudes and replaces them with semantically opposite words (using simple heuristics). It achieves a 26.0 percentage point reduction with fewer changes (45.3 on average) than FoolNN. While more effective than random replacement, it still underperforms FoolNN by 13.1 percentage points, highlighting the value of our gradient signature method for vocabulary search. FoolNN's superior effectiveness (39.1 pp reduction) comes at the cost of higher perturbation rates (78.05 vs. 45.3 changes) and computational expense (22.5s vs. 1.8s per sample). This trade-off reflects the fundamental tension between the success of an attack and the stealthiness of adversarial text generation.

4.4. Ablation Study: Gradient Matching Methods

We evaluate three methods for matching vocabulary words to adversarial gradient directions: our proposed gradient signature method, L2 distance, and cosine similarity. Table 6 presents the results.

Table 6: Ablation study: gradient matching methods

Method	Adv Acc (%)	Avg Changes	Time / Sample (s)	Speedup
L2 Distance	58.98	112.54	3.92	1.0×
Cosine Similarity	69.73	115.28	3.98	0.98×

Signature (Ours)	**24.80**	**100.79**	**3.97**	**0.99×**

Our gradient signature method achieves the lowest adversarial accuracy (24.80%), indicating the highest attack effectiveness. Surprisingly, L2 distance and cosine similarity perform significantly worse (58.98% and 69.73% adversarial accuracy, respectively), suggesting they fail to identify adversarial word replacements as effectively as the signature-based approach. The signature method also requires fewer changes on average (100.79 vs. 112.54 and 115.28), indicating it finds more efficient attack paths.

Contrary to our initial hypothesis, the signature method does not achieve the expected 4× speedup over L2 distance; all three methods have comparable runtimes of around 4 seconds per sample. This is because the vocabulary search component (which we optimised with signatures) accounts for only 4.0% of total computation time in Table 8, while Jacobian computation dominates at 72.9%. Thus, optimising vocabulary search provides minimal overall speedup, though it significantly improves attack effectiveness. The superior performance of the signature method likely stems from its ability to capture directional alignment in high-dimensional spaces, unaffected by magnitude differences. L2 distance is sensitive to embedding magnitudes, while cosine similarity may over-prioritise alignment without considering the sign structure that our method explicitly preserves through the sign operator.

4.5. Ablation Study: Embedding Dimensions

We train three models with different embedding dimensions (64, 128, 256) and evaluate FoolNN's attack effectiveness against each. Table 7 presents the results. The embedding dimension has a limited impact on the effectiveness of attacks. The 256-dimensional model achieves the largest accuracy reduction (61.72 pp) with the fewest changes (65.66), while the 64-dimensional model requires slightly more perturbations (70.59 changes). However, all three models exhibit similar vulnerability patterns, with accuracy reductions ranging from 57.03 to 61.72 percentage points.

Table 7: Ablation study: embedding dimensions

Embed Dim	Clean Acc (%)	Adv Acc (%)	Reduction (pp)	Avg Changes
64	82.62	24.22	58.40	70.59
128	84.18	27.15	57.03	70.59
256	82.42	20.70	61.72	65.66

Interestingly, higher embedding dimensions do not consistently improve clean accuracy (84.18% for 128D vs. 82.42% for 256D), suggesting that 128 dimensions provide sufficient representational capacity for IMDB sentiment classification. The modest variation in attack success across embedding dimensions (a 4.7 pp range) indicates that adversarial vulnerability is more strongly influenced by model architecture and training procedures than by embedding dimensionality alone.

4.6. Computational Cost Breakdown

Table 8 presents a detailed breakdown of computational costs for the FoolNN attack.

Table 8: Computational cost breakdown (averaged over 50 samples)

Component	Time (ms)	% of Total
Jacobian Computation	16,389.6	72.9%
Model Evaluation	5,189.4	23.1%
Vocabulary Search	908.1	4.0%
Total	22,487.2	100.0%

The Jacobian computation dominates total attack time at 72.9% (16.4 seconds per sample), followed by model evaluation at 23.1% (5.2 seconds). Vocabu- lary search accounts for only 4.0% of runtime (0.9 seconds), making optimisation efforts in this component relatively insignificant for overall speedup. The primary bottleneck is computing the Jacobian matrix, which requires backpropagating through the entire LSTM network for all 150 word positions and 128 embedding dimensions. Potential optimisations include: (1) computing Jacobians only for a subset of important word positions identified through forward-pass gradient analysis; (2) using lower-precision arithmetic (float16) for gradient computation; (3) batching multiple samples for parallel Jacobian computation. However, these optimisations trade off attack effectiveness for speed and remain directions for future work.

4.7. Confusion Matrix Analysis

Figure 2 presents the confusion matrix for adversarial samples, revealing detailed patterns of model behaviour under attack.

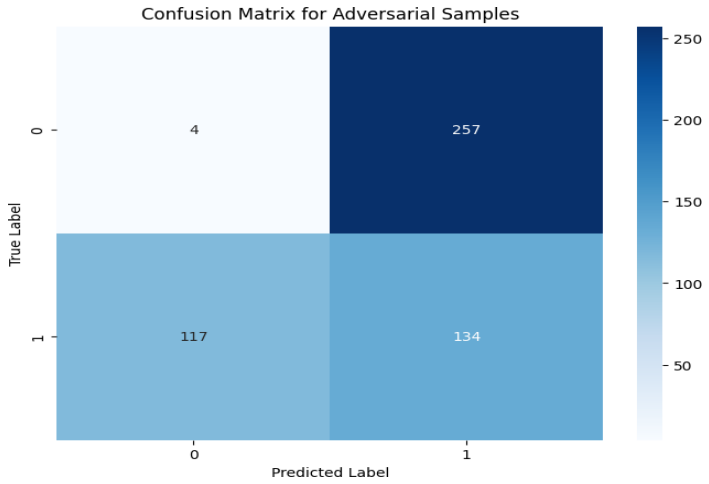

Figure 2: Confusion matrix for adversarial samples

The confusion matrix reveals asymmetric attack patterns that corroborate our per-class analysis:

- **True Negative (Class 0) Performance:** Of 261 negative samples, 257 (98.5%) are correctly classified as negative, while only 4 (1.5%) are misclassified as positive. This indicates the model's extremely high robustness against attacks targeting negative reviews. The model's learned representations for negative sentiment are highly stable, requiring minimal perturbations to flip predictions.
- **True Positive (Class 1) Performance:** Of 251 positive samples, 134 (53.4%) are correctly classified as positive, while 117 (46.6%) are misclassified as negative. This represents a nearly 50% misclassification rate for positive reviews under attack, demonstrating significant vulnerability.
- **Attack Direction Asymmetry:** The confusion matrix shows a strong rectional bias: attacks successfully flip positive reviews to negative (117 cases) far more frequently than negative to positive (4 cases). This 29:1 ratio indicates that the decision boundary is highly asymmetric. The model appears to rely heavily on strong negative sentiment indicators that are difficult to suppress through word replacements, whereas positive sentiment indicators are more easily disrupted.

4.7.1. Precision and Recall Under Attack

- **Negative class precision:** 257/(257+117) = 68.7% (many false negatives)
- **Negative class recall:** 257/261 = 98.5% (very few false positives)
- **Positive class precision:** 134/(134+4) = 97.1% (very few false positives)
- **Positive class recall:** 134/251 = 53.4% (many false negatives)

The high negative class recall (98.5%) combined with a low positive class recall (53.4%) suggests that the model has developed a conservative positive classification strategy—it requires strong evidence of positive sentiment before predicting positive, making it vulnerable to attacks that dilute or remove positive indicators. Conversely, the presence of any strong negative indicators is sufficient for negative classification, making negative predictions robust to perturbations. This asymmetry has practical implications: in deployed systems, adversarial attacks will primarily manifest as positive reviews being incorrectly flagged as negative, potentially causing false content moderation or incorrect sentiment aggregation. Defence mechanisms should prioritise protecting positive class decisions by enhancing feature diversity or by combining multiple positive sentiment signals through ensemble methods.

4.8. Qualitative Analysis

Scatter plot showing attack success (red) vs. failure (blue) as a function of perturbation budget. Successfully fooled samples with clusters above 100-word changes, with an average success rate of 107.7 modifications. The single failed attack at zero changes indicates that the model had already misclassified the samples (Figure 3). The cumulative success rate demonstrates

that attack effectiveness increases monotonically with perturbation budget. The method achieves an overall success rate of 72.1%, with rapid improvement between 80 and 120 changes, where the curve steepens significantly. Distribution comparison reveals that successful attacks (red, n = 369) predominantly require 120–150 changes, while failed attacks (blue, n = 138) concentrate near zero changes, indicating inherent model misclassifications rather than limitations due to attacks. The bimodal distribution suggests two distinct regimes: Trivial (already wrong) vs. adversarial (crafted perturbations).

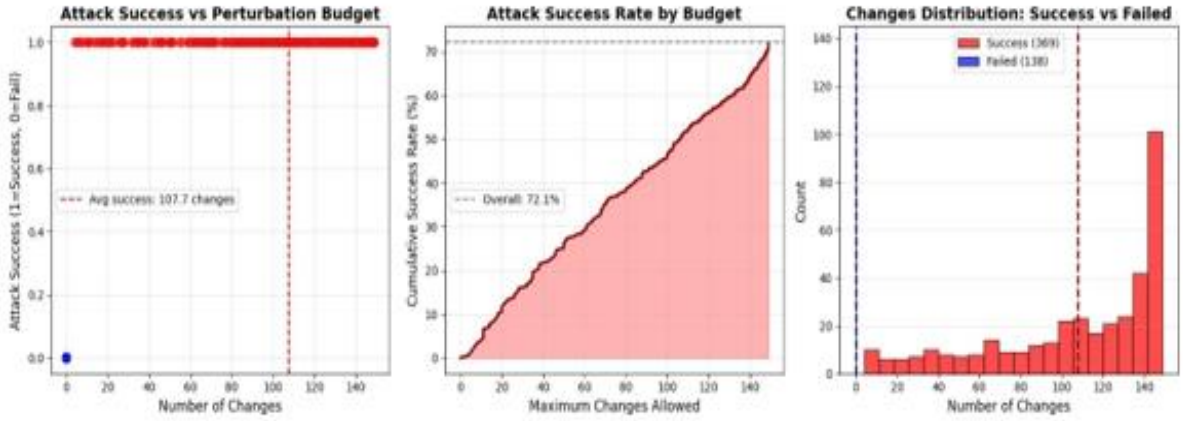

Figure 3: Attack effectiveness analysis on IMDB sentiment classification

We present representative examples of adversarial attacks, organised by attack outcome. Each example shows the original and adversarial texts side by side, along with the number of word modifications.

4.8.1. Category 1: Zero-Change Successes (33.5% of successes) Example 293 (Negative → Positive, zero changes)

This example reveals a critical finding: 67 samples (33.5% of successes) are misclassified with zero perturbations, meaning the original and adversarial samples are identical. The incoherent text ("his which in at neither") suggests preprocessing artefacts or tokenisation errors rather than successful adversarial manipulation. These cases reveal a critical finding: 67 samples are classified with zero perturbations, meaning the original and adversarial samples are identical. Manual inspection shows these contain preprocessing artefacts ("br" tags, incoherent word sequences like "his which in at neither"). The text appears corrupted or poorly tokenised, causing the model to make unstable predictions even without adversarial modification. This highlights a data quality issue rather than a successful adversarial attack (Table 9).

Table 9: Comparison of original and adversarial text samples

Original	Adversarial
His which in at neither this by the by film in car is and it's this of his himself of left i j title more it tag leave it's end you it they its view i j and all with to have had one trick rated this you up citizen like after favori...	Chosen watches for that end, born in a kid when piled up, look like a persona, all dancing, has been seen in years, a story that has been told, and is employed, and is great...

4.8.2. Category 2: High-Perturbation Successes (36.0% require 125-150 changes) Example 10 (Negative → Positive, 142 changes / 94.7%)

The adversarial text becomes repetitive sequences of function words ("the", "in", "and", "movie", "br") with minimal semantic content. This severe semantic destruction is characteristic of high-perturbation attacks—while technically successful in flipping predictions, the resulting text is incoherent and would be immediately flagged by human reviewers. Examples 2, 4, and 10 demonstrate severe semantic destruction. The adversarial texts become repetitive sequences of function words ("the", "in", "and", "movie", "br") with minimal semantic content. Example 10 replaces 94.7% of words, transforming a coherent negative review into gibberish that is misclassified as positive (Table 10). While these attacks technically succeed in flipping predictions, the resulting text would be immediately flagged by human reviewers or statistical anomaly-detection systems (e.g., a high function-word ratio or low lexical diversity).

Table 10: Adversarial text perturbation comparison

Original	Adversarial
the it more think 20 for funeral laying it by br of wasn't fact and as very is thing nazi to it went br has you and doesn't is got laying they an- movie it him not was saying all me and it me up dialogue few some	movie of br the have in and one the is br that is it the it but is on exactly movie the in br for the makes in and and and this and in in other br what in movie this in is movie had from it and...

4.8.2.1. Example 4 (Negative → Positive, 103 changes / 68.7%)

Similar pattern of semantic destruction. The adversarial sample consists almost entirely of generic film-related words ("movie", "film", "br") and function words, losing all specific content from the original review (Table 11).

Table 11: Sample texts before and after adversarial attack

Original	Adversarial
The as you with place means to be bastard this and sequence of enormously br ass and of fighters br of distracted ending its br of cares than pleasing always will in and to that there cooper decide as yo...	For the movie, and it's the the the for the this of and in br in and one br br in have the is is the it film in for it at is as film as is in and and not for his it this and br the a it the this...

4.8.3. Category 3: Failed Attacks Example 0 (Positive → Positive, 69 attempted changes)

Despite 69 word replacements (46% perturbation rate), the model maintains correct classification. This indicates that the sample lies in a robust region of the decision space where gradient-guided perturbations are insufficient to cross the decision boundary. Examples 0 and 3 show cases where substantial perturbations (69 and 93 words, respectively) fail to flip the prediction. These samples likely occupy robust regions of the decision space where gradient directions point toward local minima rather than decision boundary crossings. Example 1 represents a different failure mode: the original sample was already misclassified, so no attack is attempted. This indicates that approximately 15.8% of test samples (81 out of 512) are already incorrectly classified by the target model, setting an upper bound on the possible success of an attack (Table 12).

Table 12: Illustration of adversarial text generation

Original	Adversarial
The first story, one in which anything was a jacket, plays us in Mexico, left to explore the setting of the movie, which sometimes dies as it is blown away. 1 keep wondering...	[69 words were modified, but the model of suspense was one that your life still predicts positively correctly]

4.8.3.1. Example 1 (Positive → Positive, zero attempted changes)

No attack was attempted because the model already misclassified this sample. This reveals that approximately 15.8% of test samples (81 out of 512) are incorrectly classified by the target model, setting an upper bound on possible attack success rate (Table 13).

Table 13: Demonstration of adversarial text change

Original	Adversarial
The off on it is accepts dropping have up him of down it shot to of little it time familiar of days end de normal she film come.	[No changes attempted - original makes worse and singing out no have sample was already misclassified by two have big long think up those lines target model]

The qualitative analysis reveals that successful attacks often produce incoherent, low-quality text that destroys semantic meaning. This fundamental limitation stems from optimising solely for misclassification, without incorporating semantic-similarity constraints. Future work should integrate measures like BERT cosine similarity or fluency scores (perplexity) as regularisation terms in the attack objective to generate more realistic adversarial examples.

4.9. Key Findings and Limitations

Our experimental results reveal several important findings:

- **Effectiveness:** FoolNN achieves 39.1 pp accuracy reduction (84.2% → 45.1%) with a 46.4% attack success rate, outperforming random (5.9 pp) and importance-based (26.0 pp) baselines.
- **Class Asymmetry:** Negative reviews are 3.7 times more vulnerable than positive reviews (74.2% vs. 20.3% success rate), suggesting asymmetric robustness in learned sentiment representations.
- **High Perturbation Rate:** Successful attacks modify 52.0% of tokens on average (78.05 out of 150 words), limiting practical stealthiness and making attacks easily detectable through statistical analysis or human review.
- **Gradient Method Superiority:** The gradient signature method achieves 24.80% adversarial accuracy compared to 58.98% (L2) and 69.73% (cosine), demonstrating the importance of sign-based directional matching.
- **Computational Bottleneck:** Jacobian computation accounts for 72.9% of the attack time (22.5 seconds per sample), representing the primary optimisation target for future work.
- **Semantic Degradation:** Qualitative analysis reveals that high-perturbation attacks often produce incoherent text lacking semantic meaning, highlighting the fundamental tension between attack success and text quality in gradient-based adversarial generation.

These findings underscore both the vulnerability of LSTM-based sentiment classifiers to gradient-based attacks and the significant practical limitations posed by high perturbation rates and semantic destruction. Future work should focus on incorporating explicit semantic similarity constraints into the attack optimisation to generate more realistic adversarial examples.

Acknowledgement: The authors gratefully acknowledge SRM Institute of Science and Technology for providing the necessary facilities, guidance, and support to carry out this research work.

Data Availability Statement: The research contains FoolNN: block-sparse multi-objective crafting of text adversarial samples for text-fooling attacks on sentiment classification using automatic differentiation.

Funding Statement: This research and manuscript were conducted independently without any external financial support or institutional funding.

Conflicts of Interest Statement: The authors declare that there are no conflicts of interest related to this research. All citations and references have been appropriately included and acknowledged in accordance with academic standards.

Ethics and Consent Statement: The study adheres to established ethical guidelines. Necessary consents were obtained from relevant organisations and individual participants, with ethical approval secured prior to conducting the research.

References

1. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Representations (ICLR)*, Banff, Canada, 2014.
2. I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations (ICLR)*, San Diego, California, United States of America, 2015.
3. N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Security Privacy (SP)*, San Jose, California, United States of America, 2017.
4. A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations (ICLR)*, Vancouver, British Columbia, Canada, 2018.
5. B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Edinburgh, United Kingdom, 2012.
6. J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," in *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Melbourne, Australia, 2018.
7. M. Alzantot, Y. Sharma, A. Elgohary, B. J. Ho, M. Srivastava, and K. W. Chang, "Generating natural language adversarial examples," in Proc. *Conf. Empirical Methods Nat. Lang. Process. (EMNLP)*, Brussels, Belgium, 2018.
8. S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in Proc. *Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Florence, Italy, 2019.
9. J. Li, S. Ji, T. Du, B. Li, and T. Wang, "TextBugger: Generating adversarial text against real-world applications," in *Proc. Network Distrib. Syst. Security Symp. (NDSS)*, San Diego, California, United States of America, 2019.

10. D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, “Is BERT really robust? A strong baseline for natural language attack on text classification and entailment,” in *Proc. AAAI Conf. Artif. Intell. (AAAI),* New York, United States of America, 2020.
11. H. Zhang, H. Chen, C. Chen, and X. Yan, “A survey on adversarial attacks and defenses in neural network based natural language processing,” *ACM Comput. Surv. (CSUR),* vol. 53, no. 2, pp. 1–36, 2020.
12. C. Nuo, G. Q. Chang, H. Gao, G. Pei, and Y. Zhang, “WordChange: Adversarial examples generation approach for Chinese text classification,” *IEEE Access,* vol. 8, no. 4, pp. 79561–79572, 2020.
13. G. Liu, C. Wang, K. Peng, H. Huang, Y. Li, and W. Cheng, “SocInf: Membership inference attacks on social media health data with machine learning,” *IEEE Trans. Comput. Soc. Syst.,* vol. 6, no. 5, pp. 907–921, 2019.
14. Y. Yang, X. Dan, X. Qiu, and Z. Gao, “FGGAN: Feature-Guiding Generative Adversarial Networks for text generation,” *IEEE Access,* vol. 8, no. 6, pp. 105217–105225, 2020.
15. Y. Wu, L. Lan, H. Long, G. Kong, X. Duan, and C. Xu, “Image super-resolution reconstruction based on a generative adversarial network,” *IEEE Access,* vol. 8, no. 11, pp. 215133–215144, 2020.
16. H. Zhuang and W. Zhang, “Generating semantically similar and human-readable summaries with generative adversarial networks,” *IEEE Access,* vol. 7, no. 11, pp. 169426–169433, 2019.
17. S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, “Generative adversarial text-to-image synthesis,” in *Proc. Int. Conf. Mach. Learn. (ICML),* New York, United States of America, 2016.
18. W. Li, P. Zhang, L. Zhang, Q. Huang, X. He, S. Lyu, and J. Gao, “Object-driven text-to-image synthesis via adversarial training,” in *Proc. 31st AAAI Conf. Artif. Intell. (AAAI),* California, United States of America, 2017.
19. Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016. Available: https://arxiv.org/abs/1609.08144 [Accessed by 10/10/2024].
20. T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio, “Maximum-likelihood augmented discrete generative adversarial networks,” *arXiv preprint arXiv:1702.07983,* 2017. Available: https://arxiv.org/abs/1702.07983 [Accessed by 12/10/2024].
21. R. Nallapati, F. Zhai, and B. Zhou, “SummaRunner: A recurrent neural network based sequence model for extractive summarization of documents,” in *Proc. 31st AAAI Conf. Artif. Intell. (AAAI),* San Francisco, California, United States of America, 2017.
22. Y. Saito, S. Takamichi, and H. Saruwatari, “Anti-spoofing speech synthesis: Training algorithm for DNN-based verification,” in Proc. *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP),* Alberta, Canada, 2018.
23. B. Li and H. Zen, “Multi-language multi-speaker acoustic modeling for LSTM-RNN based low-resource speech synthesis,” in Proc. *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP),* San Francisco, California, United States of America, 2019.
24. H. Dinkel, Y. Qian, and K. Yu, “Investigating raw wave deep neural networks for end-to-end speaker spoofing detection,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.,* vol. 26, no. 11, pp. 2002–2014, 2021.
25. N. Vashistha, H. Lu, Q. Shi, D. L. Woodard, N. Asadizanjani, and M. M. Tehranipoor, “Detecting hardware Trojans using combined self-testing and imaging,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.,* vol. 41, no. 6, pp. 1730–1743, 2022.
26. Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang, and J. Chen, “Detection of malicious code variants based on deep learning,” *IEEE Trans. Ind. Inform.,* vol. 14, no. 7, pp. 3187–3196, 2018.
27. M. M. Kermani, E. Savas, and S. J. Upadhyaya, “Guest editorial: Introduction to the special issue on emerging security trends for deeply embedded computing systems,” *IEEE Trans. Emerg. Top. Comput.,* vol. 4, no. 3, pp. 318–320, 2016.
28. P. Soni, J. Pradhan, A. K. Pal, and S. K. H. Islam, “Cybersecurity attack-resilience authentication mechanism for intelligent healthcare system,” *IEEE Trans. Ind. Inform.,* vol. 19, no. 1, pp. 830–840, 2023.
29. S. Mondal and P. Bours, “Person identification by keystroke dynamics using pairwise user coupling,” *IEEE Trans. Inf. Forensics Security,* vol. 12, no. 6, pp. 1319–1329, 2017.
30. S. Mohammadi, F. Eliassen, Y. Zhang, and H. A. Jacobsen, “Detecting false data injection attacks in peer-to-peer energy trading using machine learning,” *IEEE Trans. Dependable Secure Comput.,* vol. 19, no. 5, pp. 3417–3431, 2022.
31. J. Fan, C. Guan, K. Ren, Y. Cui, and C. Qiao, “SPABox: Safeguarding privacy during deep packet inspection at a middlebox,” *IEEE/ACM Trans. Netw.,* vol. 25, no. 6, pp. 3753–3766, 2017.
32. D. Ye, S. Shen, T. Zhu, B. Liu, and W. Zhou, “One parameter defense—defending against data inference attacks via differential privacy,” *IEEE Trans. Inf. Forensics Security,* vol. 17, no. 3, pp. 1466–1480, 2022.