Bridging Robustness and Generalization Against Word Substitution Attacks in NLP via the Growth Bound Matrix Approach

Mohammed Bouri[1,2] and **Adnane Saoud**[1]
[1]College of Computing, Mohammed VI Polytechnic University, Morocco
[2]CID Development, Morocco
mohammed.bouri@um6p.ma, adnane.saoud@um6p.ma

Abstract

Despite advancements in Natural Language Processing (NLP), models remain vulnerable to adversarial attacks, such as synonym substitutions. While prior work has focused on improving robustness for feed-forward and convolutional architectures, the robustness of recurrent networks and modern state space models (SSMs), such as S4, remains understudied. These architectures pose unique challenges due to their sequential processing and complex parameter dynamics. In this paper, we introduce a novel regularization technique based on Growth Bound Matrices (GBM) to improve NLP model robustness by reducing the impact of input perturbations on model outputs. We focus on computing the GBM for three architectures: Long Short-Term Memory (LSTM), State Space models (S4), and Convolutional Neural Networks (CNN). Our method aims to (1) enhance resilience against word substitution attacks, (2) improve generalization on clean text, and (3) providing the first systematic analysis of SSM (S4) robustness. Extensive experiments across multiple architectures and benchmark datasets demonstrate that our method improves adversarial robustness by up to 8.8% over existing baselines. These results highlight the effectiveness of our approach, outperforming several state-of-the-art methods in adversarial defense. Codes are available at https://github.com/BouriMohammed/GBM

1 Introduction

Deep learning models have demonstrated impressive results across various machine learning tasks, particularly in Language Modeling (LM), which has provided powerful tools for Natural Language Processing (NLP). However, recent studies have shown that these models are vulnerable to adversarial examples. Initially, adversarial examples were identified in image classification tasks (Goodfellow et al., 2015; Qi et al., 2024), leading to a surge of research focused on adversarial attacks in NLP

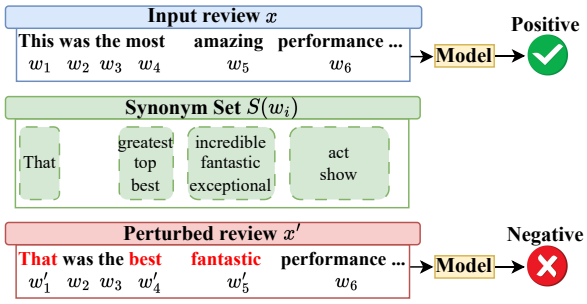

Figure 1: Word substitution-based perturbations in sentiment analysis. The original sentence x (top) and its perturbed version x' (bottom) differ by synonym replacements $w_i \to w'_i$ from $S(w_i)$. Despite preserving meaning, these changes can alter the model's predicted sentiment (e.g., Positive to Negative) due to its sensitivity to specific word choices.

tasks. These adversarial examples pose significant challenges for real-world applications such as text classification (Song et al., 2021). Additionally, although Large Language Models (LLMs) are commonly employed as decoders for text generation, they are also susceptible to adversarial examples when used in text classification tasks. (Wang et al., 2024; Shayegani et al., 2023; Yang et al., 2024). Textual adversarial attacks can be categorized into three types: character-level perturbations, sentence-level attacks, and word-level attacks. Character-level perturbations (Ebrahimi et al., 2018; Eger and Benz, 2020) can often be mitigated by spell checkers (Pruthi et al., 2019a). Sentence-level attacks (Wang et al., 2020a; Pei and Yue, 2022) utilize paraphrasing but typically fail to preserve the original sentence semantics. Word-level attacks (Ren et al., 2019; Alzantot et al., 2018; Zang et al., 2020; Wang et al., 2021c; Maheshwary et al., 2021), which rely on synonym substitutions, have become the most widely adopted approach. These attacks can craft adversarial examples with high success rates while maintaining grammatical correctness and semantic consistency, making them particularly challenging to defend against, as shown in Figure (1). Con-

Findings of the Association for Computational Linguistics: ACL 2025, pages 12118–12137

sequently, our work focuses on introducing a new robust training method against word-level substitution adversarial attacks.

To enhance robustness against such attacks, several studies have developed defense methods by generating adversarial examples and integrating them into the training set (Ren et al., 2019; Alzantot et al., 2018). However, generating enough adversarial examples per epoch is computationally expensive due to the discrete nature of the input space. To accelerate adversarial training, (Wang et al., 2021c) and (Dong et al., 2021) introduced improvements by designing fast white-box adversarial attacks. Despite these advancements, the computation time remains significantly longer than standard training. In another scope, certified defense methods based on interval bound propagation (IBP) (Jia et al., 2019; Huang et al., 2019) offer theoretical lower bounds on robustness. Nevertheless, these methods are computationally intensive, and demonstrate an overly conservative robustness. (Zhang et al., 2021) introduced a novel approach, Abstractive Recursive Certification (ARC), which defines a set of programmatically perturbed string transformations and constructs a perturbation space using these transformations. The perturbation sets are represented as a hyperrectangle, which is then propagated through the network using the (IBP) technique. However, this approach is limited in its effectiveness to two-word substitutions; increasing the number of substitutions significantly degrades the performance.

In this work, we introduce Growth Bound Matrices (GBM) as a novel method to enhance the robustness of NLP models. GBM provides certified robustness against perturbations, enabling models to maintain generalization while reducing sensitivity to input variations. Furthermore, we present a mathematical framework for computing GBM across different architectures, including Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), State Space Model S4 (Gu et al., 2022), and Convolutional Neural Networks (CNN) (Kim, 2014). Extensive experiments on multiple benchmark datasets reveal that GBM significantly enhances the certified robust accuracy of models. For instance, on the IMDB dataset, GBM achieves an impressive 84.3% certified robust accuracy, surpassing IBP by approximately 16.7%.

Our main contributions are summarized as follows:

- **Certified robustness with GBM:** We introduce Growth Bound Matrices (GBM), a novel framework that establishes certifiable robustness guarantees by bounding input-output variations.
- **Empirical validation across models and datasets:** Extensive experiments demonstrate that minimizing GBM significantly enhances adversarial robustness across multiple NLP models and benchmark datasets. Moreover, our approach improves adversarial robustness over existing defense baselines.
- **First study on SSM robustness in NLP:** To the best of our knowledge, this is the first work to investigate the generalization and robustness of the State Space Model (SSM) S4 in text classification.

2 Related Work

Spelling and grammar checkers have been shown to be robust against character-level and sentence-level attacks, which often violate grammatical requirements (Pruthi et al., 2019b; Ge et al., 2019). However, these methods are ineffective against word-level attacks.

Existing defense methods against word-level attacks can be categorized as follows:

Adversarial training (AT) is one of the most popular defense strategies (Goodfellow et al., 2015; Madry et al., 2018; Alzantot et al., 2018; Ren et al., 2019; Ivgi and Berant, 2021; Zhu et al., 2020; Li and Qiu, 2020a; Wang et al., 2020b; Zhou et al., 2020). Based on the Fast Gradient Projection Method (FGPM), an adversarial text attack technique, Adversarial Training with FGPM enhanced by Logit Pairing (ATFL) (Wang et al., 2021c) generates adversarial examples and injects them into the training set. Region-based adversarial training (Zhou et al., 2020; Dong et al., 2021) enhances model robustness by optimizing performance within the convex hull of a word's embedding and its synonyms.

Certified defense methods aim to ensure robustness against all adversarial perturbations (Zeng et al., 2021b; Wang et al., 2021a; Huang et al., 2019). Interval Bound Propagation (IBP)-based methods (Jia et al., 2019) certify robustness by iteratively computing upper and lower bounds of the output for an interval-constrained input, minimizing the worst-case loss from word substitutions. However, IBP methods struggle with scalability due to high computational costs and strict constraints. Randomized

smoothing-based methods (Zhang et al., 2024; Ye et al., 2020; Zeng et al., 2021c) are structure-free, constructing stochastic ensembles of input texts and leveraging their statistical properties to certify robustness. Moreover, most certified defense methods assume access to the attacker's synonym set, which is unrealistic, as adversaries are not restricted in their choice of synonyms.

3 Preliminaries

In this paper, we address the text classification problem. Consider a model $f : \mathcal{X} \to \mathcal{Y}$ that predicts a label $y \in \mathcal{Y}$ for a given input $x \in \mathcal{X}$, where $x = \langle w_1, w_2, \ldots, w_N \rangle$ is a sequence of N words, where each word $w_i \in \mathcal{W}$, and $\mathcal{W}$ denotes the vocabulary set. The output space $\mathcal{Y} = \{y_1, y_2, \ldots, y_c\}$ denotes all classification labels. Standard NLP models embed the input x into a sequence of vectors $v_x = \langle v_{w_1}, v_{w_2}, \ldots, v_{w_N} \rangle$ and classify via $p(y|v_x)$, parameterized using a neural network.

Adversarial robustness against synonym substitutions has been extensively studied in recent literature (Alzantot et al., 2018; Jia et al., 2019; Wang et al., 2021b; Dong et al., 2021; Zhang et al., 2024). The synonym set for a given word w_i, denoted as $\mathcal{S}(w_i)$, is constructed by selecting the k nearest words to w_i within an Euclidean distance d_e in the embedding space. To ensure semantic consistency, the perturbed input space is defined as follows:

$$\mathcal{S}_{adv}(x) = \{\langle w'_1, \ldots, w'_N \rangle \mid w'_i \in \mathcal{S}(w_i) \cup \{w_i\}\}.$$

Our adversarial defense aims to certify the robustness of the model by ensuring that:

$$\forall x' \in \mathcal{S}_{adv}(x), \quad f(x') = f(x) = y.$$

The objective of this work is to ensure that for all possible combinations of words in a sentence, when replaced by their synonyms, the classification outcome remains consistent, as shown in Figure 1.

4 Methodology

In this section, we present our main contribution. We first introduce the concept of Growth Bound Matrices (GBM) and explore how integrating these matrices into the training process can enhance model robustness. We then provide a detailed computation of GBMs for multiple architectures, including LSTM networks, State Space Model S4 and CNN models.

4.1 Growth Bound Matrix (GBM)

The Growth Bound Matrix (GBM)-based method is designed to quantify and control model sensitivity to input perturbations. Specifically, GBM establishes theoretical guarantees on the maximum output variation of a model $\mathcal{F}$ when its input x is subjected to a small perturbation δ.

4.1.1 Definition and Theoretical Foundation

Consider a mapping,

$$\begin{aligned} \mathcal{F} : \quad & \mathbf{X} \to \mathbf{Y} \\ & x \mapsto \mathcal{F}(x) \end{aligned} \tag{1}$$

where $\mathbf{X} \subseteq \mathbb{R}^{n_x}$ and $\mathbf{Y} \subseteq \mathbb{R}^{n_y}$.

Definition (Growth Bound Matrix): A matrix $\mathcal{M} \in \mathbb{R}^{n_y \times n_x}$ is said to be a *Growth Bound Matrix (GBM)* for the mapping $\mathcal{F}$ in (1) if the following condition holds:

$$\left\| \frac{\partial \mathcal{F}^i}{\partial x^j}(x) \right\| \leq (\mathcal{M})_{ij} \quad \forall (i,j) \in \mathcal{I}, \quad \forall x \in \mathbf{X} \tag{2}$$

where $\mathcal{I} = \{1, \ldots, n_y\} \times \{1, \ldots, n_x\}$ is a predefined index set, and $(\mathcal{M})_{ij}$ denotes the entry in the i-th row and j-th column of $\mathcal{M}$.

The matrix $\mathcal{M}$ capture the element-wise maximum of the boundaries of the partial derivative. By evaluating this GBM, we gain insights into the sensitivity of the output of a model with respect to its input. Therefore, reducing the GBM enables us to improve the robustness of the considered models.

4.1.2 Robustness via GBM

In this section, we formally demonstrate how the GBM provides certified robustness. The proof is provided in Appendix A.

Proposition 1: Consider the mapping $\mathcal{F}$ in (1) and let a matrix $\mathcal{M} \in \mathbb{R}^{n_y \times n_x}$ be its GBM. Given an input $x \in \mathbf{X}$, a perturbation vector $\delta \in \mathbb{R}^{n_x}$ and consider the perturbed input $x' \in \mathbf{X}$ such that $x' = x + \delta$. Then, for each component $\mathcal{F}_i$, $i \in \{1, \ldots, n_y\}$, the following inequality holds:

$$\begin{aligned} \mathcal{F}^i(x) - \sum_{j=1}^{n_x} (\mathcal{M})_{ij} \|\delta_j\| \leq \mathcal{F}^i(x') \\ \leq \mathcal{F}^i(x) + \sum_{j=1}^{n_x} (\mathcal{M})_{ij} \|\delta_j\| \end{aligned}$$

where δ_j denotes the j-th component of the perturbation vector δ.

This result indicates that the change in the ith component of the output $\mathcal{F}^i$ caused by the perturbation δ is bounded by the corresponding row of $\mathcal{M}$.

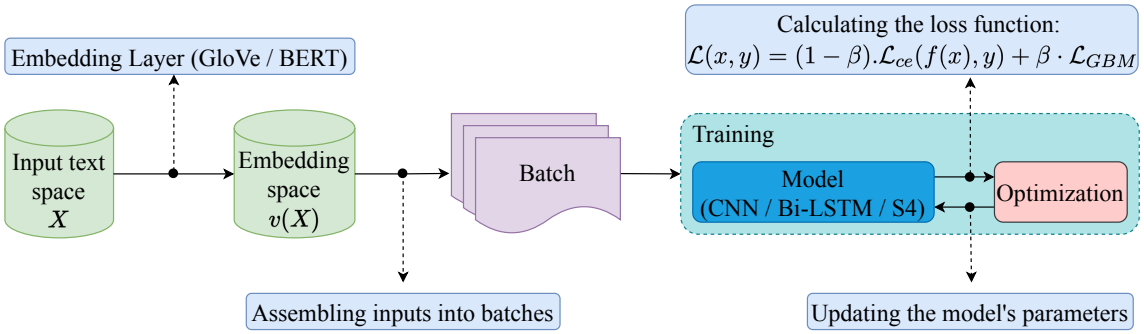

Figure 2: Training Pipeline with Growth Bound Matrix (GBM) Regularization. This pipeline integrates GBM regularization into the training process of a deep learning model. First, the input text X is converted into a vector representation $v(X)$ using an embedding layer (e.g., GloVe or BERT). The inputs are grouped into batches, and the loss function $\mathcal{L}(x, y)$ is calculated as a weighted sum of the cross-entropy loss $\mathcal{L}_{ce}$ and the GBM regularization term $\mathcal{L}_{GBM}$. The model's parameters (e.g., CNN, BiLSTM, or S4) are then updated while minimizing the GBM term to enhance robustness against adversarial perturbations.

By minimizing the GBM $\mathcal{M}$, we effectively reduce the sensitivity of the function $\mathcal{F}$ to perturbations in x. This provides a certification that the outputs of the perturbed and non-perturbed inputs remain within the same class, ensuring robustness against small input variations.

4.2 Overall Training Objective

The aim is to train a model that is consistent with the data while minimizing the GBM. To achieve this, we incorporate a regularization term into the neural network's loss function, thereby enhancing the model's robustness against attacks or perturbations. An illustration of the proposed training pipeline is given in Fig. 2. We formulate the overall training objective as follows:

$$\mathcal{L}(x, y) = (1-\beta).\mathcal{L}_{ce}(f(x), y) + \beta \cdot \mathcal{L}_{GBM}, \quad (3)$$

where

$$\mathcal{L}_{GBM} = \sum_{i=1}^{n_y} \sum_{j=1}^{n_x} (\mathcal{M})_{ij} \quad .$$

Here, $\mathcal{L}_{ce}(\cdot, \cdot)$ denotes the cross-entropy loss function employed to train the layers following the embedding layer, primarily focusing on classification accuracy. The matrix $\mathcal{M}$ represents the GBM introduced in Eq. (2), and its specific formulations for each model (Eqs. (7), (11), (14)) are detailed in Subsection 4.3. The term β serves as a hyperparameter that balances model accuracy and robustness. Meanwhile, the regularization term $\mathcal{L}_{GBM}$ aims to minimize the elements of $\mathcal{M}$, thereby enhancing the model's resilience to adversarial attacks. By including this regularization component in the training objective (Fig. 2), the resulting model exhibits improved robustness.

4.3 Model-Specific GBM

In this paper, we derive model-specific GBMs for three architectures: LSTM, S4, and CNN.

Long Short-Term Memory (LSTM) Network LSTM networks are an improved variant of Recurrent Neural Networks (RNNs). LSTMs were developed to address the vanishing and exploding gradient problems that occur during the training of traditional RNNs (Hochreiter and Schmidhuber, 1997).

Given a sentence of N input word vectors $v_x = \langle v_{w_1}, v_{w_2}, \ldots, v_{w_N} \rangle$, where $v_{w_t} \in \mathbb{R}^{d_0}$ represents the embedding of the word at position t in the sentence. We begin by defining the LSTM cell model, which is characterized by the cell state $c_t \in \mathbb{R}^d$ and the hidden state $h_t \in \mathbb{R}^d$. These states are updated according to the following equations:

$$c_t = f_t \odot c_{t-1} + I_t \odot g_t \quad (4)$$

$$h_t = o_t \odot \tanh(c_t) \quad (5)$$

where $\odot$ denotes element-wise multiplication, I_t is the input gate, f_t is the forget gate, g_t is the cell gate, and o_t is the output gate, each gate governed by its own set of learnable parameters (weights, and biases). These gates control the flow of information through the network. Based on Eqs. (4) and (5), we can represent the LSTM cell as an input–output mapping $\mathcal{F}$, as in Eq. (1), formally defined as:

$$\mathcal{F} : \mathbf{V} \times \mathbf{H} \times \mathbf{C} \to \mathbb{R}^d$$
$$(v_{w_t}, h_{t-1}, c_{t-1}) \mapsto o_t \odot \tanh(f_t \odot c_{t-1} + I_t \odot g_t) \quad (6)$$

where the input $x = (v_{w_t}, h_{t-1}, c_{t-1}) \in \mathbf{V} \times \mathbf{H} \times \mathbf{C} \subseteq \mathbb{R}^{d_0+2d}$ and the output $y = h_t = \mathcal{F}(v_{w_t}, h_{t-1}, c_{t-1}) \in \mathbb{R}^d$. Here, $\mathbf{V}$ denotes the

domain of the word vectors, $\mathbf{H}$ denotes the domain of hidden states, and $\mathbf{C}$ denotes the domain of cell states.

The following result provide an explicit formula for the GBM of the LSTM cell. A detailed proof of this result, as well as the algorithm used to compute the GBM, can be found in the Appendix (D,G).

Proposition 2: Consider the map $\mathcal{F}$ describing the input-output model of an LSTM cell defined in Eq.(6). The GBM of the map $\mathcal{F}$ can be expressed as follows:

$$\text{For all } (i,j) \in \{1,\dots,d\} \times \{1,\dots,d_0+2d\}$$
$$(\mathcal{M})_{ij} = \max\left(\|(\underline{\mathcal{M}})_{ij}\|, \|(\overline{\mathcal{M}})_{ij}\|\right), \quad (7)$$

where, for all input $x = (v_{w_t}, h_{t-1}, c_{t-1}) \in \mathbf{X} = (\mathbf{V} \times \mathbf{H} \times \mathbf{C})$

$$(\underline{\mathcal{M}})_{i,j} = \begin{cases} \min\{\ (\mathcal{M}_v)_{i,j}(x) \mid x \in \mathbf{X}\} \ \text{ if } \ j \in \mathcal{J}_1 \\ \min\{\ (\mathcal{M}_h)_{i,j-d_0}(x) \mid x \in \mathbf{X}\} \text{ if } j \in \mathcal{J}_2 \\ \min\{\ (\mathcal{M}_c)_{i,j-d_0-d}(x) \mid x \in \mathbf{X}\} \text{ if } j \in \mathcal{J}_3 \end{cases}$$

$$(\overline{\mathcal{M}})_{i,j} = \begin{cases} \max\{\ (\mathcal{M}_v)_{i,j}(x) \mid x \in \mathbf{X}\} \text{ if } j \in \mathcal{J}_1 \\ \max\{\ (\mathcal{M}_h)_{i,j-d_0}(x) \mid x \in \mathbf{X}\} \text{ if } j \in \mathcal{J}_2 \\ \max\{\ (\mathcal{M}_c)_{i,j-d_0-d}(x) \mid x \in \mathbf{X}\} \text{ if } j \in \mathcal{J}_3 \end{cases}$$

with $\mathcal{J}_1 = \{1,\dots,d_0\}$, $\mathcal{J}_2 = \{d_0{+}1,\dots,d_0{+}d\}$ and $\mathcal{J}_3 = \{d_0+d+1,\dots,d_0+2d\}$

where the map $x \mapsto (\mathcal{M}_v)_{i,j}(x)$ is given by:

$$(\mathcal{M}_v)_{i,j}(x) = (\Theta^{(o)})_{ij}.\sigma'(T_o^i).\tanh(c_t^i) + \sigma(T_o^i).\frac{\partial c_t^i}{\partial v_{w_t}^j}.\tanh'(c_t^i),$$

the map $x \mapsto (\mathcal{M}_h)_{i,j}(x)$ is given by:

$$(\mathcal{M}_h)_{i,j}(x) = (U^{(o)})_{ij}.\sigma'(T_o^i).\tanh(c_t^i) + \sigma(T_o^i).\frac{\partial c_t^i}{\partial h_{t-1}^j}.\tanh'(c_t^i),$$

the map $x \mapsto (\mathcal{M}_c)_{i,j}(x)$ is given by:

$$(\mathcal{M}_c)_{i,j}(x) = \sigma(T_o^i).\sigma(T_f^i).\tanh'(c_t^i)$$

and the map $x \mapsto c_t^i(x)$ is given by:

$$c_t^i = f_t^i \odot c_{t-1}^i + I_t^i \odot g_t^i$$

with:

$$T_{\text{gate}}^i = \sum_{p=1}^{d_0} (\Theta^{(\text{gate})})_{ip}.v_{w_t}^{(p)} + \sum_{q=1}^{d} (U^{(\text{gate})})_{iq}.h_{t-1}^{(q)} + b_i^{(\text{gate})}$$

Here, σ denotes the sigmoid function and $\tanh$ denotes the hyperbolic tangent function, σ' and $\tanh'$ denote their respective derivatives. For gate $\in \{f, o\}$, the parameters $\Theta^{(gate)} \in \mathbb{R}^{d\times d_0}, U^{(gate)} \in \mathbb{R}^{d\times d}$ and $b^{(gate)} \in \mathbb{R}^d$ are the input-hidden weights, hidden-hidden weights, and biases, respectively.

State Space Model (SSM) S4

The S4 has demonstrated outstanding performance in text classification tasks, surpassing transformers in various benchmarks (Gu et al., 2022).

Given a sentence represented as a sequence of N word embeddings, where each word at position t is associated with a vector $v_{w_t} \in \mathbb{R}^{d_0}$. Collectively, these embeddings form the input sequence $v_x = \langle v_{w_1}, v_{w_2}, \dots, v_{w_N}\rangle$.

The continuous-time State Space Model (SSM) is designed to process each dimension of the input independently while utilizing shared parameters across all dimensions. The continuous-time formulation of the SSM is given by:

$$\dot{h}(t) = Ah(t) + Bv_{w_t},$$
$$y(t) = Ch(t) + Dv_{w_t},$$

where $h(t) \in \mathbb{R}^{d_0 d}$ represents the hidden state at time t. The parameters $A \in \mathbb{C}^{d_0 d\times d_0 d}$, $B \in \mathbb{C}^{d_0 d\times d_0}$, $C \in \mathbb{C}^{d_0\times d_0 d}$, and $D \in \mathbb{R}^{d_0\times d_0}$ are shared across all dimensions of the input v_{w_t}. To efficiently utilize the SSM in structured sequence modeling, a discrete-time formulation is required. The S4 model (Gu et al., 2022) employs the bilinear transformation (Tustin, 1947) to discretize the continuous-time system, leading to the following discrete-time state-space representation, which we refer to as an S4 cell:

$$h_t = \tilde{A}h_{t-1} + \tilde{B}v_{w_t}, \quad (8)$$
$$y_t = \tilde{C}h_t + \tilde{D}v_{w_t}, \quad (9)$$

where the discretized parameters are given by: $\tilde{A} = (I - \Delta/2\cdot A)^{-1}(I + \Delta/2\cdot A) \in \mathbb{C}^{d_0 d\times d_0 d}$, $\tilde{B} = (I - \Delta/2\cdot A)^{-1}\Delta B \in \mathbb{C}^{d_0 d\times d_0}$, $\tilde{C} = C \in \mathbb{C}^{d_0\times d_0 d}$, $\tilde{D} = D \in \mathbb{R}^{d_0\times d_0}$, and $\Delta \in \mathbb{R}^{d_0}$ is a fixed step size that represents the resolution of the input. Based on Eqs. (8) and (9), we can represent the S4 cell as an input–output mapping $\mathcal{F}$, as in Eq. (1), formally defined as:

$$\mathcal{F}: \quad \mathbb{R}^{d_0+d_0 d} \to \mathbb{R}^{d_0}$$
$$(v_{w_t}, h_{t-1}) \mapsto \tilde{C}\big(\tilde{A}h_{t-1} + \tilde{B}v_{w_t}\big) + \tilde{D}v_{w_t}, \quad (10)$$

where the input is $x = (v_{w_t}, h_{t-1}) \in \mathbb{R}^{d_0+d_0 d}$ and the output is $y_t = \mathcal{F}(v_{w_t}, h_{t-1}) = \tilde{C}\big(\tilde{A}h_{t-1} + \tilde{B}v_{w_t}\big) + \tilde{D}v_{w_t} \in \mathbb{R}^{d_0}$.

In the following, we provide an explicit formula for the GBM of the S4 cell. A detailed proof of this result can be found in the Appendix D.

Proposition 3: Consider the map $\mathcal{F}$ describing the input-output model of an S4 cell defined in Eq.(10).

The GBM of the map $\mathcal{F}$ can be expressed as follows:

$$\text{For all } (i,j) \in \{1,\dots,d_0\} \times \{1,\dots,d_0 + d_0 d\}$$

$$(\mathcal{M})_{i,j} = \begin{cases} \|\big(\tilde{C}.\tilde{B}\big)_{i,j} + \tilde{D}_{ij}\| & \text{if} \quad 1 \le j \le d_0 \\ \|\big(\tilde{C}.\tilde{A}\big)_{i,j-d_0}\| & \text{if } \; d_0 < j \le d_0 + d_0 d \end{cases} \tag{11}$$

Convolutional Neural Network (CNN)

TextCNN (Kim, 2014) applies a one-dimensional convolutional operation followed by a max-pooling layer to extract meaningful representations.

Given a sentence of N input word vectors $v_x = \langle v_{w_1}, v_{w_2}, \dots, v_{w_N} \rangle$, where each $v_{w_t} \in \mathbb{R}^{d_0}$ denotes the embedding of the word at position t.

The convolutional layer and the max-pooling layer are defined as follows:

For $k_i \in \mathcal{K} = \{k_1, \dots, k_m\} \subset \mathbb{Z}_{\ge 2}$ and $t = \{1, \dots, N - k_i + 1\}$

$$\mathcal{C}^{(k_i)}(v_x)_t = \phi\left(b^{(k_i)} + \sum_{l=0}^{k_i - 1} W^{(k_i)}_{:,:,l} v_{w_{t+l}}\right),$$

$$\mathcal{P}^{(k_i)}(\mathcal{C}^{(k_i)}(v_x)) = \max_{t=1}^{N-k_i+1}\left(\mathcal{C}^{(k_i)}(v_x)_t\right), \tag{12}$$

where, k_i is the kernel size, ϕ is the ReLU activation function, $b^{(k_i)} \in \mathbb{R}^d$ is the bias, $W^{(k_i)} \in \mathbb{R}^{d \times d_0 \times k_i}$ is weight matrix for the convolutional filter of size k_i, and $v_{w_{t+l}}$ is the word embedding at position $t + l$. Based on Eq.(12), we can represent the CNN layer as an input-output mapping $\mathcal{F}$, ad in Eq. (1), formally defined as:

$$\begin{aligned} \mathcal{F}: \quad & \mathbb{R}^{N d_0} \to \mathbb{R}^{|\mathcal{K}|.d} \\ & v_x \quad \mapsto \bigoplus_{k=k_1}^{k_m} \mathcal{P}^{(k)} \circ \mathcal{C}^{(k)}(v_x) \end{aligned} \tag{13}$$

where $\bigoplus$ represents the concatenation operation, and $|\mathcal{K}|$ is the cardinal of the set $\mathcal{K}$.

The following result presents an explicit formula for the GBM of the CNN layer, with a detailed proof provided in Appendix D.

Proposition 4: Consider the map $\mathcal{F}$ defined in Eq.(13). The GBM of the map $\mathcal{F}$ can be expressed as follows:

$$\text{For all } (i,j) \in \{1,\dots,|\mathcal{K}|.d\} \times \{1,\dots,N d_0\}$$

$$(\mathcal{M})_{i,j} = \max_{t=1}^{j} \| W^{\left(k_{\alpha(i,1,d)}\right)}_{\beta(i\,,1\,,d),\,\beta(j\,,t\,,d_0),\,\alpha(j\,,t\,,d_0)} \|, \tag{14}$$

where,

$$\beta(i,a,d) = 1 + \big((i-a) \bmod d\big),$$

$$\alpha(i,a,d) = \left\lfloor \frac{i-a}{d} \right\rfloor + 1,$$

$\lfloor . \rfloor$ is the floor function, and the modulo operation $\bmod$ computes the remainder of the division of $(i-a)$ by d.

5 Experiments

This section provides an experimental evaluation of our proposed GBM approach, comparing its performance with four defense baselines against three distinct adversarial attacks on the BiLSTM and CNN models, and with eleven defense baselines against the TextFooler attack on the BERT model. We also study the robustness of the S4 model against three distinct adversarial attacks. Additionally, we emphasize the scalability of the GBM approach in terms of time efficiency, and we provide visual evidence of its resilience to adversarial perturbations.

5.1 Experimental Setup

Datasets: We evaluate our method on two benchmark datasets: *IMDB* (Maas et al., 2011): A binary sentiment classification dataset with 25,000 movie reviews for training and 25,000 for testing. *Yahoo! Answers* (Zhang et al., 2015): A large-scale topic classification dataset with 1,400,000 training samples and 50,000 testing samples across 10 classes.

Attack Methods: To evaluate defense efficacy, we employ five advanced adversarial attacks: GA (Genetic Attack) (Alzantot et al., 2018), PWWS (Probability Weighted Word Saliency) (Ren et al., 2019), PSO (Particle Swarm Optimization) (Zang et al., 2020), and TextFooler (Jin et al., 2020). PWWS uses synonyms from WordNet; TextFooler and GA use counter-fitted embeddings; PSO relies on sememe-based substitutions from HowNet. Given the computational cost of textual adversarial attacks, we generate adversarial examples using 1000 randomly sampled test instances from each dataset. For fair comparison with Text-CRS (Zhang et al., 2024), we sample 500 instances from the IMDB test set and assess model robustness against the TextFooler attack. All attacks are implemented using the OpenAttack framework (Zeng et al., 2021a), and TextAttack framework (Morris et al., 2020).

Perturbation Setting: In line with (Jia et al., 2019) and (Dong et al., 2021), we set $k = 8$ as the number of nearest neighbors within a Euclidean distance of $d_e = 0.5$ in the GloVe embedding space. Additionally, for BERT model, following (Zhang et al., 2024), we adopt the default parameters of (Jin et al., 2020) to ensure a fair comparison.

Dataset	Defense	CNN				BiLSTM			
		Clean	PWWS	GA	PSO	Clean	PWWS	GA	PSO
IMDB	Standard	89.7	0.6	2.6	1.4	89.1	0.2	1.6	0.3
	IBP (Jia et al., 2019)	81.7	75.9	76.0	75.9	77.6	67.5	67.8	67.6
	ATFL (Wang et al., 2021c)	85.0	63.6	66.8	64.7	85.1	72.2	75.5	74.0
	SEM (Wang et al., 2021b)	87.6	62.2	63.5	61.5	86.8	61.9	63.7	62.2
	ASCC (Dong et al., 2021)	84.8	74.0	75.5	74.5	84.3	74.2	76.8	75.5
	GBM	**90.2**	**76.3**	**76.6**	**84.3**	**89.6**	**76.8**	**77.8**	**84.3**
			↑0.4	↑0.6	↑8.4		↑2.6	↑1.0	↑8.8
Yahoo! Answers	Standard	72.6	6.8	7.2	4.9	**74.7**	12.2	9.6	6.5
	IBP (Jia et al., 2019)	63.1	54.9	54.9	54.8	54.3	47.3	47.6	47.0
	ATFL (Wang et al., 2021c)	72.5	62.5	63.1	62.5	73.6	61.7	60.8	60.3
	SEM (Wang et al., 2021b)	70.1	53.8	52.4	51.9	72.3	57.0	56.1	55.4
	ASCC (Dong et al., 2021)	69.0	58.4	59.6	58.5	70.7	61.7	62.3	61.9
	GBM	**72.8**	**66.2**	**66.1**	**67.3**	73.0	**67.0**	**67.0**	**68.6**
			↑3.7	↑3.0	↑4.8		↑5.3	↑4.7	↑6.7

Table 1: The clean accuracy (CA) (%) for standard training and Accuracy Under Attack (AUA) (%) against various adversarial attacks on two datasets for CNN and BiLSTM. The columns of *Clean* denote the CA on the entire original testing set. The highest accuracy against the corresponding attack on each column is highlighted in **bold**, while the second one is highlighted in underline. The last row of each block indicates the gains of the accuracy between our method and the best baseline.

Model Setting: We employ pre-trained GloVe embeddings to map words into a 300-dimensional vector space, serving as the embedding layer for Bi-LSTM, CNN, and S4 models. Additionally, we utilize the pre-trained 12-layer bert-base-uncased model, which generates 768-dimensional hidden representations per token, as the embedding layer for Bi-LSTM and CNN models. Further implementation details are provided in Appendix E.

Evaluation Metrics: We use the following metrics to evaluate our models: Clean Accuracy (CA) denotes model's classification accuracy on the clean test dataset, and Accuracy Under Attack (AUA) is the model's prediction accuracy under specific adversarial attack methods.

5.2 Main Results

Performance on CNN and BiLSTM: Table 1 presents the classification performance of the BiLSTM and CNN models on the *IMDB* and *Yahoo! Answers* datasets, where each row corresponds to a defense method and each column to an attack method. A defense is considered more effective if it preserves higher AUA while minimizing performance degradation on clean samples compared to standard training.

Our proposed method consistently outperforms existing defenses, particularly against the PSO attack, achieving improvements of up to **8.8%** on *IMDB* and **6.7%** on *Yahoo! Answers* with the BiLSTM model, and up to **8.4%** on *IMDB* and **4.8%** on *Yahoo! Answers* with the CNN model. Furthermore, our approach attains the highest CA on both datasets with CNN and nearly the best CA on *Yahoo! Answers* with the BiLSTM model. These results highlight its effectiveness in enhancing model robustness while maintaining strong generalization performance in both clean and adversarial settings.

Performance on S4: Figure 3 shows how the S4 model performs on the *IMDB* dataset under normal conditions and different adversarial attacks. Since this is the first study evaluating S4's robustness against such attacks, there are no previous baselines for direct comparison.

Our results reveal that the standard S4 model achieves the highest CA (**88.1%**), but it struggles against adversarial attacks. By applying our GBM defense, we significantly improve S4's resistance to attacks, achieving the highest AUA for PWWS (**49.4%**), PSO (**42.6%**), and TextFooler (**52.6%**). These findings demonstrate that our method effectively strengthens S4's defense against adversarial perturbations while maintaining high CA.

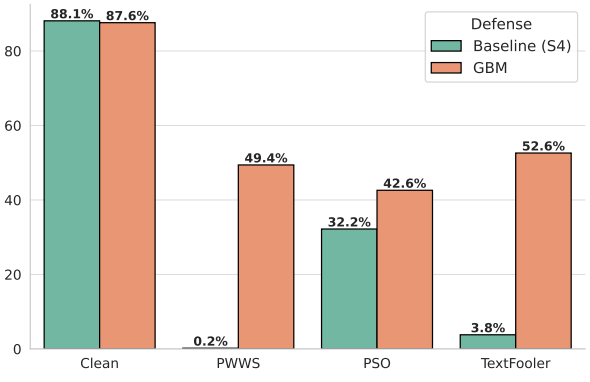

Figure 3: The CA (%) for standard training and AUA (%) against various adversarial attacks on IMDB dataset for S4 model.

Performance on BERT: The results presented in Table 2 demonstrate that the GBM-based method achieves superior robustness against the TextFooler attack compared to baseline approaches across both the BiLSTM model (utilizing BERT embeddings) and the BERT model. Specifically, our approach outperforms Text-CRS, achieving improvements of **6.2%** against TextFooler. These findings highlight the effectiveness of the proposed methodology in enhancing model resilience relative to existing techniques in the literature.

Defense	Clean	TextFooler
baseline (BERT)	92.1	10.3
MixADA (Si et al., 2020)	91.9	19.0
PGD-K (Madry et al., 2018)	93.2	26.0
FreeLB (Zhu et al., 2020)	93.0	29.0
TA-VAT (Li and Qiu, 2020b)	93.0	28.0
InfoBERT (Wang et al., 2020b)	92.0	19.0
DNE (Zhou et al., 2020)	90.4	28.0
ASCC (Dong et al., 2021)	87.8	19.4
SAFER (Ye et al., 2020)	**93.5**	9.5
RanMASK (Zeng et al., 2021c)	93.2	22.0
FreeLB++ (Li et al., 2021)	93.2	45.3
Text-CRS (Zhang et al., 2024)	91.5	84.4
GBM	92.1	**90.6** ↑6.2

Table 2: The CA (%) for standard training and AUA (%) against TextFooler attack on IMDB dataset for BERT models.

5.3 Time Efficiency of the GBM Approach

Efficiency plays a critical role in evaluating defense mechanisms, particularly for large-scale datasets and complex models. Table 3 presents the training time per epoch for both GBM and IBP (Jia et al., 2019) defense methods on the IMDB dataset. Notably, the IBP defense method demands excessively more training time for BiLSTM models, primarily due to the heavy overhead imposed by certified constraints. In contrast, the comparison shows the significantly higher computational efficiency of the GBM approach, reducing training time by more than **11 times** compared to IBP.

Dataset	Model	Defense	Time by epoch (min)
IMDB	*BiLSTM*	IBP	04:53
		GBM	**00:25**
	CNN	IBP	00:05
		GBM	**00:02**

Table 3: Training time per epoch for IBP and GBM defense methods on the IMDB dataset.

5.4 Efficiency of GBM against Adversarial perturbations

BiLSTM model: Figure 4 shows sensitivity heatmaps for the final hidden state of a BiLSTM model trained on the *IMDB* dataset. The heatmap on the left corresponds to the model trained with GBM regularization, while the right heatmap represents the baseline model without regularization. These heatmaps visualize how much each input dimension influences each output dimension, based on the GBM matrix $\mathcal{M}$. The color scale (logarithmic) highlights the magnitude of the components of $\mathcal{M}$, where brighter colors indicate stronger influence, i.e., larger values of a given component of $\mathcal{M}$. We observe that the total sensitivity, computed as $\sum_{i,j} \mathcal{M}_{ij}$, is significantly lower in the GBM-regularized model (45.33) compared to the baseline model (96.50). This suggests that GBM reduces the model's reaction to small input changes, thereby promoting stability and improving robustness to perturbations.

CNN and S4 models: Figure 5 illustrates the distribution of the values of the GBM matrix $\mathcal{M}$ for CNN and S4 models trained on the *IMDB* dataset, with and without GBM regularization. In both cases, the distributions for GBM-regularized models (orange) are clearly shifted toward lower values of $\mathcal{M}_{ij}$ compared to their baseline counterparts (blue), indicating a reduction in the overall magnitude of input-output interactions. The total sensitivity of the matrix, computed as the sum of all its components $\sum_{i,j} \mathcal{M}_{ij}$, also decreases significantly under GBM regularization. Specifically, for the CNN model, the total sum drops from 10475.19 to 4010.16, while for the S4 model, it decreases from 7375.00 to 4218.56. These results highlight the

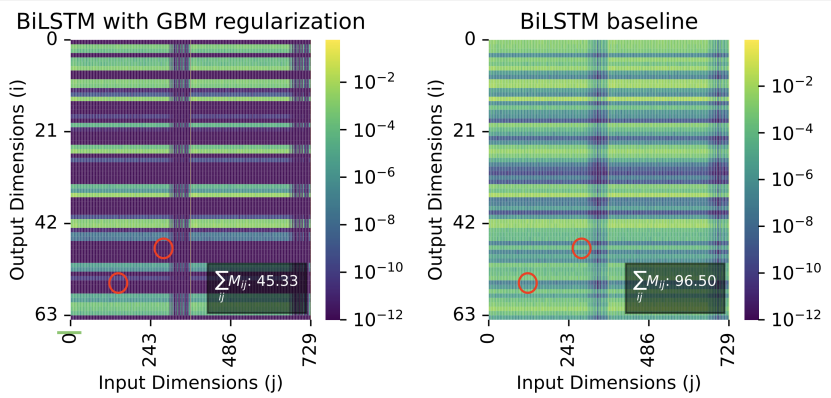

Figure 4: Comparison of GBM $\mathcal{M}$ in BiLSTM Models on the *IMDB* dataset with and without GBM Regularization. The left heatmap corresponds to the model regularized with GBM, while the heatmap on the right represents the unregularized (baseline) model. The red circles highlight representative examples where individual entries of the GBM matrix $\mathcal{M}_{ij}$ are strongly minimized in the regularized model compared to the baseline.

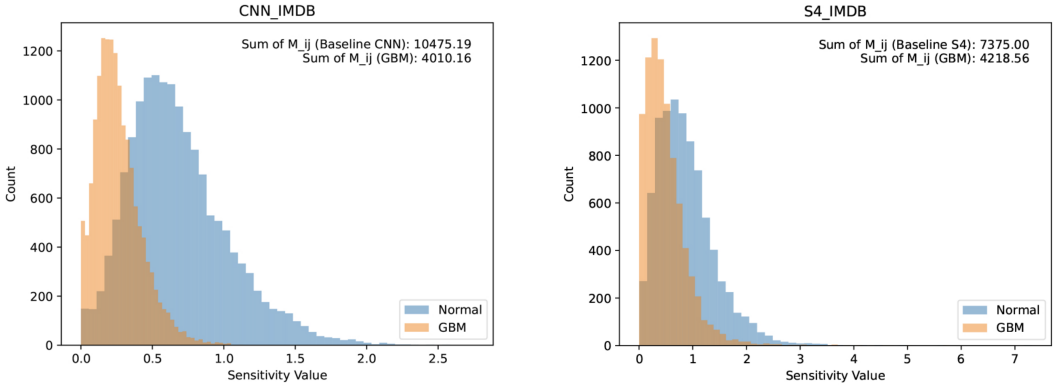

Figure 5: Comparison of the distribution of elements in the GBM $\mathcal{M}$ for CNN and S4 models on the *IMDB* dataset.

effectiveness of GBM to improve the robustness against adversarial inputs.

6 Discussion

Our results demonstrate that minimizing the GBM yields two advantages : (1) enhanced robustness against adversarial attacks and (2) improved generalization on clean data. This is achieved by introducing a regularization term, which encourages the model to learn smoother decision boundaries. Adversarial methods (e.g., GA, PWWS, PSO, TextFooler) exploit small input perturbations (e.g., synonym substitutions) to provoke large output deviations. By reducing sensitivity (via low GBM), gradient-based or search-driven attack strategies face significant difficulty in identifying effective perturbations under practical modification limits. Crucially, a controlled GBM penalty preserves the model's ability to learn discriminative features, retaining high accuracy on clean data. Thus, reducing sensitivity by minimizing the GBM ensures a crucial balance between adversarial robustness and performance on clean inputs. Further analysis can be found in the Appendix E.

7 Conclusion

In this work, we introduce a novel method, Growth Bound Matrices (GBM), which provides tight bounds on the input-output variations of diverse models. By minimizing the change in a model's predictions when substituting a word with one of its synonyms, GBM ensures robustness against such perturbations. Empirical evaluations demonstrate the effectiveness of our approach, revealing significant improvements in model robustness.

For future work, we plan to investigate the applicability of GBM across additional domains, including speech recognition (Sajjad et al., 2020) and control systems (Jouret et al., 2023). Furthermore, we aim to explore GBM's potential for State Space Model S6 (Mamba) (Gu and Dao, 2023), thereby extending its contribution to robust model design.

Limitations

In this work, we focus on sequence-level classification, where the objective is to determine the overall class of an entire text sequence. Our approach is not directly applicable to token-level classification tasks, which require predicting the class of each individual word within a sequence. However, it is well-suited for word substitution attacks with fixed-length perturbations. Future work should investigate extending our method to enhance robustness against a broader range of word-level adversarial attacks, including variable-length perturbations such as word deletion or removal, thereby improving its applicability to more fine-grained text classification tasks.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. 2021. Towards robustness against natural language word substitutions. In *9th International Conference on Learning Representations*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Steffen Eger and Yannik Benz. 2020. From hero to zéroe: A benchmark of low-level adversarial attacks. In *Proceedings of the 1st conference of the Asia-Pacific chapter of the association for computational linguistics and the 10th international joint conference on natural language processing*, pages 786–803.

Tao Ge, Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. Automatic grammatical error correction for sequence-to-sequence text generation: An empirical study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6059–6064, Florence, Italy. Association for Computational Linguistics.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations*.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*.

Maor Ivgi and Jonathan Berant. 2021. Achieving model robustness through discrete adversarial training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1529–1544.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4127–4140.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Louis Jouret, Adnane Saoud, and Sorin Olaru. 2023. Safety verification of neural-network-based controllers: a set invariance approach. *IEEE Control Systems Letters*, 7:3842–3847.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Linyang Li and Xipeng Qiu. 2020a. Tavat: Token-aware virtual adversarial training for language understanding. *arXiv preprint arXiv:2004.14543*.

Linyang Li and Xipeng Qiu. 2020b. Tavat: Token-aware virtual adversarial training for language understanding. *Preprint*, arXiv:2004.14543.

Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. 2021. Searching for an effective defender: Benchmarking defense against adversarial word substitution. *arXiv preprint arXiv:2108.12777*.

Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, Mykel J Kochenderfer, et al. 2021. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4(3-4):244–404.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations*.

Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. 2021. Generating natural language attacks in a hard label black box setting. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Pierre-Jean Meyer, Alex Devonport, and Murat Arcak. 2021. *Interval reachability analysis: Bounding trajectories of uncertain systems with boxes for control and verification*. Springer Nature.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Weiping Pei and Chuan Yue. 2022. Generating content-preserving and semantics-flipping adversarial text. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 975–989.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019a. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 5582–5591.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019b. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Biqing Qi, Yang Luo, Junqi Gao, Pengfei Li, Kai Tian, Zhiyuan Ma, and Bowen Zhou. 2024. Exploring adversarial robustness of deep state space models. *arXiv preprint arXiv:2406.05532*.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*.

Muhammad Sajjad, Soonil Kwon, et al. 2020. Clustering-based speech emotion recognition by incorporating learned features and deep bilstm. *IEEE access*, 8:79861–79875.

Erfan Shayegani, Md Abdullah Al Mamun, Yu Fu, Pedram Zaree, Yue Dong, and Nael Abu-Ghazaleh. 2023. Survey of vulnerabilities in large language models revealed by adversarial attacks. *arXiv preprint arXiv:2310.10844*.

Chenglei Si, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2020. Better robustness by more coverage: Adversarial training with mixup augmentation for robust fine-tuning. *arXiv preprint arXiv:2012.15699*.

Liwei Song, Xinwei Yu, Hsuan-Tung Peng, and Karthik Narasimhan. 2021. Universal adversarial attacks with natural triggers for text classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3724–3733.

Arnold Tustin. 1947. A method of analysing the behaviour of linear systems in terms of time series. *Journal of the Institution of Electrical Engineers-Part IIA: Automatic Regulators and Servo Mechanisms*, 94(1):130–142.

Boxin Wang, Hengzhi Pei, Boyuan Pan, Qian Chen, Shuohang Wang, and Bo Li. 2020a. T3: Tree-autoencoder constrained adversarial text generation for targeted attack. In *Conference on Empirical Methods in Natural Language Processing*.

Boxin Wang, Shuohang Wang, Yu Cheng, Zhe Gan, Ruoxi Jia, Bo Li, and Jingjing Liu. 2020b. Infobert: Improving robustness of language models from an information theoretic perspective. *9th International Conference on Learning Representations (ICLR)*.

Wenjie Wang, Pengfei Tang, Jian Lou, and Li Xiong. 2021a. Certified robustness to word substitution attack with differential privacy. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xiaosen Wang, Hao Jin, Yichen Yang, and Kun He. 2021b. Natural language adversarial defense through synonym encoding. In *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence*.

Xiaosen Wang, Yichen Yang, Yihe Deng, and Kun He. 2021c. Adversarial training with fast gradient projection method against synonym substitution based text attacks. In *AAAI Conference on Artificial Intelligence*.

Zimu Wang, Wei Wang, Qi Chen, Qiufeng Wang, and Anh Nguyen. 2024. Generating valid and natural adversarial examples with large language models. In *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 1716–1721. IEEE.

Zeyu Yang, Zhao Meng, Xiaochen Zheng, and Roger Wattenhofer. 2024. Assessing adversarial robustness of large language models: An empirical study. *arXiv preprint arXiv:2405.02764*.

Mao Ye, Chengyue Gong, and Qiang Liu. 2020. SAFER: A structure-free approach for certified robustness to adversarial word substitutions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3465–3475, Online. Association for Computational Linguistics.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Zixian Ma, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2021a. OpenAttack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 363–371, Online. Association for Computational Linguistics.

Jiehang Zeng, Xiaoqing Zheng, Jianhan Xu, Linyang Li, Liping Yuan, and Xuanjing Huang. 2021b. Certified robustness to text adversarial attacks by randomized [MASK]. In *Findings of Association for Computational Linguistics*.

Jiehang Zeng, Xiaoqing Zheng, Jianhan Xu, Linyang Li, Liping Yuan, and Xuanjing Huang. 2021c. Certified robustness to text adversarial attacks by randomized [MASK]. *arXiv preprint arXiv:2105.03743*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*.

Xinyu Zhang, Hanbin Hong, Yuan Hong, Peng Huang, Binghui Wang, Zhongjie Ba, and Kui Ren. 2024. Text-crs: A generalized certified robustness framework against textual adversarial attacks. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 2920–2938. IEEE.

Yuhao Zhang, Aws Albarghouthi, and Loris D'Antoni. 2021. Certified robustness to programmable transformations in LSTMs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1068–1083, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-wei Chang, and Xuanjing Huang. 2020. Defense against adversarial attacks in nlp via dirichlet neighborhood ensemble. *arXiv preprint arXiv:2006.11627*.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

A Robustness via GBM

A.1 Proof of Proposition 1

Consider the map $\mathcal{F} : \mathbf{X} \to \mathbf{Y}$, where $\mathbf{X} \subseteq \mathbb{R}^{n_x}$ and $\mathbf{Y} \subseteq \mathbb{R}^{n_y}$. Each component $\mathcal{F}^i : \mathbf{X} \subseteq \mathbb{R}^{n_x} \to \mathbb{R}$ is a scalar function. Given an input $x \in \mathbf{X}$, a perturbation vector $\delta \in \mathbb{R}^{n_x}$ and consider the perturbed input $x' \in \mathbf{X}$ such that $x' = x + \delta$. The mean value theorem guarantees that:

$$\mathcal{F}^i(x) - \mathcal{F}^i(x') = \nabla \mathcal{F}^i(c)^T (x - x')$$

for some point $c \in \mathbf{X}$ on the line segment between x to x'. By taking norms on both sides, one gets:

$$\|\mathcal{F}^i(x) - \mathcal{F}^i(x')\| = \|\nabla \mathcal{F}^i(c)^T . (x - x')\| \quad (15)$$

Since $\nabla \mathcal{F}^i(c) = \left(\frac{\partial \mathcal{F}^i}{\partial x^1}(c), \ldots, \frac{\partial \mathcal{F}^i}{\partial x^{n_x}}(c) \right)^T$. By the definition of GBM (2), we have for $c \in X$ that:

$$\left\| \frac{\partial \mathcal{F}^i}{\partial x^j}(c) \right\| \leq (\mathcal{M})_{i,j} \quad \forall (i,j) \in \mathcal{I},$$

where $\mathcal{I} = \{1, \ldots, n_y\} \times \{1, \ldots, n_x\}$. Hence, one gets from Eq.(15) that:

$$\begin{aligned} \|\mathcal{F}^i(x) - \mathcal{F}^i(x')\| &= \|\nabla \mathcal{F}^i(c)^T \delta\| \\ &\leq \sum_{j=1}^{n_x} (\mathcal{M})_{i,j} \|\delta_j\| \end{aligned}$$

where δ_j is the j-th component of the vector δ. Then, the bound on the output of the perturbed input x' can described as:

$$\begin{aligned} \mathcal{F}^i(x) - \sum_{j=1}^{n_x} (\mathcal{M})_{i,j} \|\delta_j\| &\leq \mathcal{F}^i(x') \\ &\leq \mathcal{F}^i(x) + \sum_{j=1}^{n_x} (\mathcal{M})_{i,j} \|\delta_j\| \end{aligned}$$

which concludes the proof. □

B Relationship Between GBM and the Lipschitz Constant

The Lipschitz constant corresponds to the largest value in the GBM, representing the maximum possible variation of outputs with respect to changes in inputs. The benefit of using the GBM over the Lipschitz constant is that while the Lipschitz constant provides a global upper bound on how much a function can stretch distances, regardless of direction, the GBM captures the variation in different directions, allowing tighter and more accurate bounds. For instance, the function might be more sensitive to changes in some directions than others, which a Lipschitz constant cannot express.

Proposition: Consider the mapping $\mathcal{F} : \mathbf{X} \to \mathbf{Y}$ and let a matrix $\mathcal{M} \in \mathbb{R}^{n_y \times n_x}$ be its GBM. Then $L = \max_i \max_j (\mathcal{M})_{ij}$ is a Lipschitz constant of the map $\mathcal{F}$.

Proof: Consider the map $\mathcal{F} : \mathbf{X} \to \mathbf{Y}$, where $\mathbf{X} \subseteq \mathbb{R}^{n_x}$ and $\mathbf{Y} \subseteq \mathbb{R}^{n_y}$. Each component $\mathcal{F}^i : \mathbf{X} \subseteq \mathbb{R}^{n_x} \to \mathbb{R}$ is a scalar function. Given two inputs $x, x' \in \mathbf{X}$.

If we consider the infinity norm, we have that $\|\mathcal{F}(x) - \mathcal{F}(x')\| = \max_i \|\mathcal{F}^i(x) - \mathcal{F}^i(x')\|$. Now for each i, the mean value theorem guarantees that

$$\|\mathcal{F}^i(x) - \mathcal{F}^i(x')\| = \|\nabla \mathcal{F}^i(c^i)^T\| \|(x - x')\|$$

for some point $c^i \in \mathbf{X}$ on the line segment between x to x'. By taking norms on both sides, one gets:

$$\begin{aligned}\|\mathcal{F}(x) - \mathcal{F}(x')\| &= \max_i \|\mathcal{F}^i(x) - \mathcal{F}^i(x')\| \\ &\leq \max_i \|\nabla \mathcal{F}^i(c^i)^T\| \|x - x'\|\end{aligned}$$

Since $\nabla \mathcal{F}^i(c^i) = \left(\frac{\partial \mathcal{F}^i}{\partial x^1}(c^i), \ldots, \frac{\partial \mathcal{F}^i}{\partial x^{n_x}}(c^i) \right)^T$. We have that

$$\begin{aligned}\|\mathcal{F}(x) - \mathcal{F}(x')\| &\leq \max_i \|\nabla \mathcal{F}^i(c^i)^T\| \|x - x'\| \\ &\leq \max_i \max_j \left\| \frac{\partial \mathcal{F}^i(c^i)}{\partial x^j} \right\| \|x - x'\| \\ &\leq \max_i \max_j (\mathcal{M})_{ij} \|x - x'\| \\ &\leq L \|x - x'\|\end{aligned}$$

Hence, $L = \max_i \max_j (\mathcal{M})_{ij}\|$ is a Lipschitz constant of the map $\mathcal{F}$. □

C Auxiliary results

In order to provide bounds on the sigmoid and hyperbolic tangent derivatives, we define the following result:

Proposition 5: Consider an interval $[\underline{a}, \bar{a}] \subseteq \mathbb{R}$. For $\varphi = \tanh'$ or σ', the following holds:

$$\min_{a \in [\underline{a}, \bar{a}]} \varphi(a) = \min(\varphi(\underline{a}), \varphi(\bar{a}))$$

$$\max_{a \in [\underline{a}, \bar{a}]} \varphi(a) = \begin{cases} \varphi(0), & \text{if } 0 \in [\underline{a}, \bar{a}], \\ \max(\varphi(\underline{a}), \varphi(\bar{a})), & \text{otherwise.} \end{cases}$$

Proof of Proposition 5: Consider the scalar function $\varphi : \mathbb{R} \to \mathbb{R}$, with $\varphi = \tanh'$ or $\varphi = \sigma'$. To show the result, we analyse the variations of the function φ on $\mathbb{R}$. Indeed, we have that φ is:

- non-decreasing on $(-\infty, 0]$ until reaching its global maximum $\max_{x \in \mathbb{R}_\infty} \varphi(x) = \varphi(0)$;
- non-increasing on $[0, +\infty)$.

The result in Proposition 5 follows then immediately. □

The following result is adapted from (Liu et al., 2021).

Proposition 6: For $i \in \{1, \ldots, d\}$, consider the map $(v_{w_t}, h_{t-1}) \mapsto T^i(v_{w_t}, h_{t-1})$ described by

$$T^i_{\text{gate}} = \sum_{p=1}^{d_0} (\Theta^{(\text{gate})})_{pi} . v^p_{w_t} + \sum_{q=1}^{d} (U^{(\text{gate})})_{qi} . h^q_{t-1} + b^{(\text{gate})}_i$$

with $\Theta^{(gate)} \in \mathbb{R}^{d \times d_0}, U^{(gate)} \in \mathbb{R}^{d \times d}$ and $b^{(gate)} \in \mathbb{R}^d$ for gate $\in \{f, I, g, o\}$. If $v_{w_t} \in [\underline{v}, \overline{v}] \subseteq \mathbb{R}^{d_0}$ and $h_t \in [\underline{h}, \overline{h}] \subseteq \mathbb{R}^d$, then $T^i_{\text{gate}} \in [\underline{T}^i, \overline{T}^i]$, where $\underline{T}^i$ and $\overline{T}^i$ can be explicitly written as:

$$\begin{aligned}\underline{T}^i &= \sum_{p=1}^{d_0} \underline{\alpha}_{pi} + \sum_{q=1}^{d_0} \underline{\beta}_{qi} + b^{(\text{gate})}_i \\ \overline{T}^i &= \sum_{p=1}^{d_0} \overline{\alpha}_{pi} + \sum_{q=1}^{d_0} \overline{\beta}_{qi} + b^{(\text{gate})}_i\end{aligned}$$

with $\underline{\alpha}_{pi}$, $\overline{\alpha}_{pi}$, $\underline{\beta}_{qi}$ and $\overline{\beta}_{qi}$ given by:

$$\underline{\alpha}_{pi} = \begin{cases} (\Theta^{(\text{gate})})_{pi} . \underline{v}^p_{w_t}, & (\Theta^{(\text{gate})})_{pi} \geq 0 \\ (\Theta^{(\text{gate})})_{pi} . \overline{v}^p_{w_t}, & (\Theta^{(\text{gate})})_{pi} < 0 \end{cases}$$

$$\overline{\alpha}_{pi} = \begin{cases} (\Theta^{(\text{gate})})_{pi} . \overline{v}^p_{w_t}, & (\Theta^{(\text{gate})})_{pi} \geq 0 \\ (\Theta^{(\text{gate})})_{pi} . \underline{v}^p_{w_t}, & (\Theta^{(\text{gate})})_{pi} < 0 \end{cases}$$

$$\underline{\beta}_{qi} = \begin{cases} (U^{(\text{gate})})_{qi} . \underline{h}^q_{t-1}, & (U^{(\text{gate})})_{qi} \geq 0 \\ (U^{(\text{gate})})_{qi} . \overline{h}^q_{t-1}, & (U^{(\text{gate})})_{qi} < 0 \end{cases}$$

$$\overline{\beta}_{qi} = \begin{cases} (U^{(\text{gate})})_{qi} . \overline{h}^q_{t-1}, & (U^{(\text{gate})})_{qi} \geq 0 \\ (U^{(\text{gate})})_{qi} . \underline{h}^q_{t-1}, & (U^{(\text{gate})})_{qi} < 0. \end{cases}$$

The following proposition is adapted from (Meyer et al., 2021).

Proposition 7: Consider the map:

$$\begin{aligned} c_t : \quad & \mathbf{V} \times \mathbf{H} \times \mathbf{C} \to \mathbb{R}^d \\ & (v_{w_t}, h_{t-1}, c_{t-1}) \mapsto c_t(v_{w_t}, h_{t-1}, c_{t-1}) \end{aligned}$$

defined in (4), with $\mathbf{V} = [\underline{\mathbf{V}}, \overline{\mathbf{V}}]$, $\mathbf{H} = [\underline{\mathbf{H}}, \overline{\mathbf{H}}]$ and $\mathbf{C} = [\underline{\mathbf{C}}, \overline{\mathbf{C}}]$. Let $A^c = \frac{\partial c_t}{\partial v_{w_t}} \in \mathbb{R}^{d \times d_0}$, $B^c = \frac{\partial c_t}{\partial h_{t-1}} \in \mathbb{R}^{d \times d}$ and $D^c = \frac{\partial c_t}{\partial c_{t-1}} \in \mathbb{R}^{d \times d}$.

For all $i \in \{1, \dots, d\}$, we have $c_t^i \in [\underline{c}_t^i, \overline{c}_t^i]$, with

$$\begin{aligned}\underline{c}_t^i =& c_t^i \left(\frac{\overline{\mathbf{V}} + \underline{\mathbf{V}}}{2}, \frac{\overline{\mathbf{H}} + \underline{\mathbf{H}}}{2}, \frac{\overline{\mathbf{C}} + \underline{\mathbf{C}}}{2} \right) \\ & - (A^c)_{i,:}(\overline{\mathbf{V}} - \underline{\mathbf{V}}) - (B^c)_{i,:}(\overline{\mathbf{H}} - \underline{\mathbf{H}}) \\ & - (D^c)_{i,:}(\overline{\mathbf{C}} - \underline{\mathbf{C}})\end{aligned}$$

$$\begin{aligned}\overline{c}_t^i =& c_t^i \left(\frac{\overline{\mathbf{V}} + \underline{\mathbf{V}}}{2}, \frac{\overline{\mathbf{H}} + \underline{\mathbf{H}}}{2}, \frac{\overline{\mathbf{C}} + \underline{\mathbf{C}}}{2} \right) \\ & + (A^c)_{i,:}(\overline{\mathbf{V}} - \underline{\mathbf{V}}) + (B^c)_{i,:}(\overline{\mathbf{H}} - \underline{\mathbf{H}}) \\ & + (D^c)_{i,:}(\overline{\mathbf{C}} - \underline{\mathbf{C}})\end{aligned}$$

where $(A^c)_{i,:}$ (respectively $(B^c)_{i,:}$ and $(D^c)_{i,:}$) denotes the i-th row vector of the matrix A^c (respectively B^c and D^c).

D Proofs of Model-Specific GBM

In this section, we present the proofs for the main results of this paper, specifically Proposition 2, Proposition 3, and Proposition 4. For S4 and CNN architectures, the GBM is directly derived from the learned parameters, making its computation straightforward. In contrast, for LSTM models, the GBM computation is more involved, requiring bounds on specific functions. To address this issue, we provide a detailed algorithmic procedure for computing the GBM in the LSTM case, ensuring a practical and efficient implementation.

D.1 Proof of Proposition 2

Consider the map $\mathcal{F}$ describing the input-output model on an LSTM cell defined in (6), where the input is $x = (v_{w_t}, h_{t-1}, c_{t-1}) \in \mathbf{V} \times \mathbf{H} \times \mathbf{C} \subseteq \mathbb{R}^{d_0+2d}$ and the output is $y_t = \mathcal{F}(v_{w_t}, h_{t-1}, c_{t-1}) = o_t \odot \tanh(f_t \odot c_{t-1} + I_t \odot g_t) \in \mathbb{R}^d$, with,

$$I_t = \sigma\left(\Theta^{(I)}.v_{w_t} + U^{(I)}.h_{t-1} + b^{(I)}\right) \quad (16)$$

$$f_t = \sigma\left(\Theta^{(f)}.v_{w_t} + U^{(f)}.h_{t-1} + b^{(f)}\right) \quad (17)$$

$$g_t = \tanh\left(\Theta^{(g)}.v_{w_t} + .U^{(g)}h_{t-1} + b^{(g)}\right) \quad (18)$$

$$o_t = \sigma\left(\Theta^{(o)}.v_{w_t} + U^{(o)}.h_{t-1} + b^{(o)}\right) \quad (19)$$

where σ denotes the sigmoid function and $\tanh$ denotes the hyperbolic tangent function. For gate $\in \{I, f, g, o\}$, the parameters $\Theta^{(gate)} \in \mathbb{R}^{d \times d_0}, U^{(gate)} \in \mathbb{R}^{d \times d}$ and $b^{(gate)} \in \mathbb{R}^d$ are the input-hidden weights, hidden-hidden weights, and biases, respectively.

Lets consider the following:

$$T^i_{\text{gate}} = \sum_{p=1}^{d_0} (\Theta^{(\text{gate})})_{ip}.v^p_{w_t} + \sum_{q=1}^{d} (U^{(\text{gate})})_{iq}.h^q_{t-1} + b^{(\text{gate})}_i$$

with σ' and $\tanh'$ are the derivative of the sigmoid and hyperbolic tangent functions, respectively.

Here, $\mathbf{V}$ denotes the domain of the word vectors, $\mathbf{H}$ denotes the domain of hidden states, and $\mathbf{C}$ denotes the domain of cell states. The i-th component of the map $\mathcal{F}$ is given for $i \in \{1, \dots, d\}$ by:

$$\mathcal{F}^i(x) = o_t^i \odot \tanh(f_t^i \odot c_{t-1}^i + I_t^i \odot g_t^i).$$

Now consider $j \in \{1, \dots, d_0 + 2d\}$ and consider the j-th component of the input vector $x \in \mathbf{V} \times \mathbf{H} \times \mathbf{C} \subseteq \mathbb{R}^{d_0+2d}$. Since x is the concatenation of the vectors $v_{w_t} \in \mathbf{V} \subseteq \mathbb{R}^{d_0}$ and $h_{t-1} \in \mathbf{H} \subseteq \mathbb{R}^d$, and $c_{t-1} \in \mathbf{C} \subseteq \mathbb{R}^d$, we distinguish three cases: if $j \in \{1, \dots, d_0\}$, then x^j is a component of the vector v_{w_t} and one gets

$$\begin{aligned}\frac{\partial \mathcal{F}^i}{\partial x^j}(x) &= (\mathcal{M}_v)_{i,j}(x) \\ &= \frac{\partial o_t^i}{\partial x^j} \cdot \tanh(c_t^i) + o_t^i \cdot \frac{\partial \tanh(c_t^i)}{\partial x^j}\end{aligned}$$

Using Eq. (19), one gets

$$\frac{\partial o_t^i}{\partial x^j} = (\Theta^{(o)})_{i,j} \cdot \sigma'(T_o^i)$$

and $\dfrac{\partial \tanh(c_t^i)}{\partial x^j} = \dfrac{\partial c_t^i}{\partial x^j} \cdot \tanh'(c_t^i),$

with σ' and $\tanh'$ are the derivative of the sigmoid and hyperbolic tangent functions, respectively.
Based on Eqs.(4),(16),(17) and (18), we have the following:

$$\begin{aligned}\frac{\partial c_t^i}{\partial x^j} &= (\Theta^{(f)})_{i,j} \cdot \sigma'(T_f^i) \cdot c_{t-1}^i \\ &\quad + (\Theta^{(I)})_{i,j} \cdot \sigma'(T_I^i) \cdot \tanh(T_g^i) \\ &\quad + (\Theta^{(g)})_{i,j} \cdot \sigma(T_I^i) \cdot \tanh'(T_g^i).\end{aligned}$$

which implies that

$$\begin{aligned}(\mathcal{M}_v)_{i,j}(x) = (\Theta^{(o)})_{i,j}.\sigma'(T_o^i).\tanh(c_t^i) \\ + \sigma(T_o^i).\frac{\partial c_t^i}{\partial x^j}.\tanh'(c_t^i)\end{aligned}$$

Hence, it follows that

$$\left\| \frac{\partial \mathcal{F}^i}{\partial x^j}(x) \right\| \leq \max\left(\|(\underline{\mathcal{M}_v})_{ij}\|, \|(\overline{\mathcal{M}_v})_{ij}\| \right).$$

Similarly, if $j \in \{d_0 + 1, \dots, d_0 + d\}$ then x^j is

a component of the vector h_{t-1} and one gets

$$\frac{\partial \mathcal{F}^i}{\partial x^j}(x) = (\mathcal{M}_h)_{i,j-d_0}(x)$$
$$= \frac{\partial o_t^i}{\partial x^j} \cdot \tanh(c_t^i) + o_t^i \cdot \frac{\partial \tanh(c_t^i)}{\partial x^j}$$

Using Eq. (19), one gets

$$\frac{\partial o_t^i}{\partial x^j} = (U^{(o)})_{i,j} \cdot \sigma'(T_o^i)$$

and $\frac{\partial \tanh(c_t^i)}{\partial x^j} = \frac{\partial c_t^i}{\partial x^j} \cdot \tanh'(c_t^i)$,

Based on Eqs.(4), (16), (17) and (18), we have the following:

$$\frac{\partial c_t^i}{\partial x^j} = (U^{(f)})_{i,j} \cdot \sigma'(T_f^i) \cdot c_{t-1}^i$$
$$+ (U^{(I)})_{i,j} \cdot \sigma'(T_I^i) \cdot \tanh(T_g^i)$$
$$+ (U^{(g)})_{i,j} \cdot \sigma(T_I^i) \cdot \tanh'(T_g^i).$$

which implies that:

$$(\mathcal{M}_h)_{i,j-d_0}(x) = (U^{(o)})_{i,j}.\sigma'(T_o^i).\tanh(c_t^i)$$
$$+ \sigma(T_o^i).\frac{\partial c_t^i}{\partial x^j}.\tanh'(c_t^i)$$

Hence, it follows that

$$\left\| \frac{\partial \mathcal{F}^i}{\partial x^j}(x) \right\| \leq \max\left(\|(\underline{\mathcal{M}_h})_{i,j-d_0}\|, \|(\overline{\mathcal{M}_h})_{i,j-d_0}\| \right).$$

Finally, if $j \in \{d_0 + d + 1, \ldots, d_0 + 2d\}$ then x^j is a component of the vector c_{t-1} and one gets

$$\frac{\partial \mathcal{F}^i}{\partial x^j}(x) = (\mathcal{M}_c)_{i,j-d_0-d}(x)$$
$$= \frac{\partial o_t^i}{\partial x^j} \cdot \tanh(c_t^i) + o_t^i \cdot \frac{\partial \tanh(c_t^i)}{\partial x^j}$$

Based on Eqs.(4), one get:

$$\frac{\partial \mathcal{F}^i}{\partial x^j}(x) = (\mathcal{M}_c)_{i,j-d_0-d}(x)$$
$$= \sigma(T_o^i).\sigma(T_f^i).\tanh'(c_t^i)$$

Hence, it follows that $\left\| \frac{\partial \mathcal{F}^i}{\partial x^j}(x) \right\| \leq \max\left(\|(\underline{\mathcal{M}_c})_{i,j-d_0-d}\|, \|(\overline{\mathcal{M}_c})_{i,j-d_0-d}\| \right)$. □

The computation of the Growth Bound Matrix (GBM) for LSTM models follows a systematic procedure grounded in analytical bounds and algorithmic implementations. For each component pair (i, j), the process begins by bounding the intermediate term T_o^i using *Proposition 6*. Subsequently, the derivative of the sigmoid function $\sigma'(T_o^i)$ is bounded using *Proposition 5*, and the cell state c_t^i is bounded via *Proposition 7*. The derivative $\tanh'(c_t^i)$ is then bounded again using *Proposition 5*. These bounds form the basis for the estimation of partial derivatives of the cell state with respect to both the input embedding $v_{w_t}^j$ and the hidden state h_{t-1}^j, which are computed using *Algorithm 4* and *Algorithm 5*, respectively.

The complete bounds of the GBM components are derived through three dedicated algorithms:

- **Algorithm 1** computes the upper and lower bounds of the matrix $\mathcal{M}_v$, corresponding to the sensitivity of the hidden state with respect to the input embedding v_{w_t}. This computation integrates the intermediate bounds and uses *Algorithm 4* for estimating $\partial c_t^i / \partial v_{w_t}^j$.
- **Algorithm 2** estimates the boundaries of $\mathcal{M}_h$, which represents the sensitivity of the hidden state with respect to the previous hidden state h_{t-1}. The estimation is based on Propositions 5, 6, and 7, and it incorporates *Algorithm 5* to bound $\partial c_t^i / \partial h_{t-1}^j$.
- **Algorithm 3** computes the bounds of $\mathcal{M}_c$, corresponding to the impact of the previous cell state c_{t-1} on the current hidden state. This step uses the bounds on T_o^i, T_f^i, and c_t^i, and again relies on the propositions 5, 6, and 7.

Each matrix component is ultimately characterized by a pair of upper and lower bounds, denoted $(\overline{\mathcal{M}_v})_{i,j}, (\underline{\mathcal{M}_v})_{i,j}$, $(\overline{\mathcal{M}_h})_{i,j}, (\underline{\mathcal{M}_h})_{i,j}$, and $(\overline{\mathcal{M}_c})_{i,j}, (\underline{\mathcal{M}_c})_{i,j}$, respectively. These are assembled to construct the final GBM matrices $\mathcal{M}_v$, $\mathcal{M}_h$, and $\mathcal{M}_c$, as defined in Equation (7). The overall GBM $\mathcal{M}$ is then formed according to the formulation detailed in *Proposition 2 (Sec. 4.3)*.

D.2 Proof of Proposition 3

Consider the map $\mathcal{F}$ describing the input-output model on an S4 cell defined in (10), where the input is $x = (v_{w_t}, h_{t-1}) \in \mathbb{R}^{d_0+d_0 d}$ and the output is $y_t = \mathcal{F}(v_{w_t}, h_{t-1}) = \tilde{C}\big(\tilde{A}h_{t-1} + \tilde{B}v_{w_t}\big) + \tilde{D}v_{w_t} \in \mathbb{R}^{d_0}$. The i-th component of the map $\mathcal{F}$ is given for $i \in \{1, \ldots, d_0\}$ by

$$\mathcal{F}^i(x) = (\tilde{C}\tilde{A})_{i,:}h_{t-1} + ((\tilde{C}\tilde{B})_{i,:} + (\tilde{D})_{i,:})v_{w_t}$$

where the notation $(\mathcal{M})_{i,:}$ is used to denote the i-th row of the matrix $\mathcal{M}$. Now consider $j \in \{1, \ldots, d_0 + d_0 d\}$ and consider the j-th component of the input vector $x \in \mathbb{R}^{d_0+d_0 d}$. Since x is the concatenation of the vectors $v_{w_t} \in \mathbb{R}^{d_0}$

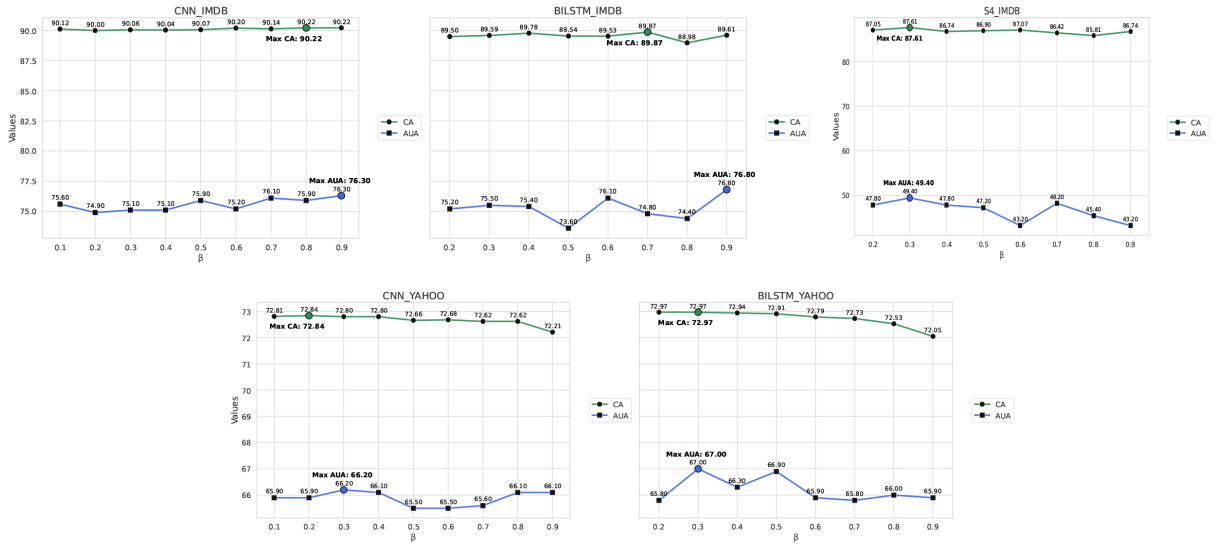

Figure 6: The impact of the hyperparameter β on the performance of CNN, BiLSTM, and S4 models when subjected to the PWWS adversarial attack. The evaluation is conducted on the *Yahoo! Answers* and *IMDB* datasets, with CNN and BiLSTM assessed on both datasets, while S4 is evaluated exclusively on *IMDB*.

and $h_{t-1} \in \mathbb{R}^{d_0 d}$, we distinguish two cases: if $j \in \{1, \ldots, d_0\}$, then x^j is a component of the vector v_{w_t} and one gets

$$\left\| \frac{\partial \mathcal{F}^i}{\partial x^j}(x) \right\| \leq \|(\tilde{C}\tilde{B})_{i,j} + (\tilde{D})_{i,j}\|.$$

Similarly, if $j \in \{d_0 + 1, \ldots, d_0 + d_0 d\}$ then x^j is a component of the vector h_{t-1} and one gets

$$\left\| \frac{\partial \mathcal{F}^i}{\partial x^j}(x) \right\| \leq \|(\tilde{C}\tilde{A})_{i,j-d_0}\|$$

which concludes the proof. □

D.3 Proof of Proposition 4

Consider the map $\mathcal{F}$ defined in Eq. (13), which characterizes the input-output transformation of a convolutional neural network (CNN). The input is a flattened embedding vector $x = v_x \in \mathbb{R}^{N d_0}$, and the output is obtained by applying convolution and pooling operations across multiple kernel sizes. Specifically, the output is given by:

$$y = \bigoplus_{k \in \mathcal{K}} \mathcal{P}^{(k)} \circ \mathcal{C}^{(k)}(v_x) \in \mathbb{R}^{|\mathcal{K}| \cdot d},$$

where $\mathcal{K} = \{k_1, \ldots, k_m\}$ denotes the set of kernel sizes, and d is the number of output channels (filters) per kernel size.

Let us now describe how the individual components of $\mathcal{F}$ are computed. For a given kernel size $k_p \in \mathcal{K}$, the output of the corresponding convolution and max-pooling operation is a d-dimensional vector. The i-th component of this vector is given by:

$$\mathcal{F}_p^i(x) = \max_{t=1}^{N-k_p+1} \phi\left(b_i^{(k_p)} + \sum_{l=0}^{k_p-1} W_{i,:,l}^{(k_p)} \cdot v_{w_{t+l}} \right),$$

where:

- $W^{(k_p)} \in \mathbb{R}^{d \times d_0 \times k_p}$ is the convolutional kernel for size k_p,
- $b_i^{(k_p)}$ is the bias term for the i-th filter,
- ϕ is the activation function (e.g., ReLU or tanh),
- and the max operator is applied across the temporal dimension t.

After processing all kernel sizes in $\mathcal{K}$, the final output vector $y = \mathcal{F}(x)$ is constructed by concatenating the outputs from each kernel size. Thus, each output index $i \in \{1, \ldots, |\mathcal{K}| \cdot d\}$ corresponds to a specific kernel and a specific filter within that kernel.

To identify which kernel and filter an output index i corresponds to, we define two functions:

$$\alpha(i, a, d) = \left\lfloor \frac{i - a}{d} \right\rfloor + 1,$$

$$\beta(i, a, d) = 1 + ((i - a) \bmod d),$$

where:

- $\alpha(i, 1, d)$ gives the index of the kernel $k_{\alpha(i,1,d)}$ associated with output index i,

- $\beta(i,1,d)$ gives the index of the filter within the kernel.

Thus, the i-th output of $\mathcal{F}(x)$ can be written as:

$$\mathcal{F}^i(x) = \max_{t=1}^{N-k_{\alpha(i,1,d)}+1} \phi\left(\mathcal{Q}_i\right),$$

where:

$$\mathcal{Q}_i = b_i^{(k_{\alpha(i,1,d)})} + \sum_{l=0}^{k_{\alpha(i,1,d)}-1} W_{\beta(i,1,d),:,l}^{(k_{\alpha(i,1,d)})} \cdot v_{w_{t+l}}.$$

We now analyze the GBM of the output $\mathcal{F}^i(x)$ with respect to a component x^j of the input. Since the Lipschitz constants of the used activations function (ReLU or Tanh) are bounded by 1, the derivative of $\mathcal{F}^i$ with respect to x^j is bounded by the magnitude of the associated weight in the convolution kernel.

Let $j \in \{1, \ldots, Nd_0\}$ denote a specific input index. We identify the relevant input slice using:

$$\alpha(j,t,d_0) = \left\lfloor \frac{j-t}{d_0} \right\rfloor + 1,$$

$$\beta(j,t,d_0) = 1 + ((j-t) \bmod d_0).$$

Then, the partial derivative of $\mathcal{F}^i$ with respect to x^j satisfies:

$$\left\| \frac{\partial \mathcal{F}^i}{\partial x^j}(x) \right\| \leq \max_{t=1}^{j} \left\| W_{\beta(i,1,d),\beta(j,t,d_0),\alpha(j,t,d_0)}^{(k_{\alpha(i,1,d)})} \right\|.$$

Taking the supremum over all valid t, this yields the expression for the GBM as stated in Eq. (14), which completes the proof. □

E Experiment Details

E.1 Additional experiments: Clean Accuracy versus Robust Accuracy

Hyper-parameter Study: In Eq.(3), the loss function of the model incorporates a hyper-parameter β to regulate the trade-off between clean accuracy and robustness accuracy. This trade-off varies depending on the model, as CNN, BiLSTM, and S4 demonstrate different sensitivity levels on the Yahoo! Answers and IMDB datasets as shown in Figure 6. These results underscore the challenge of maintaining both robustness and accuracy in adversarial settings, highlighting the importance of carefully tuning β to optimize performance for each model and dataset.

E.2 Dataset Statistics

Dataset	Training set	Test set	Classes
IMDB	25,000	25,000	2
Yahoo! Answers	1,400,000	50,000	10

Table 4: Statistics of the *IMDB* and *Yahoo! Answers* datasets.

E.3 Embedding layer

In our experiments, we freeze the parameters of the pre-trained embedding layer and update only those of the classification model.

E.4 Detailed Setup

In the BiLSTM layer, the weights $\Theta^{(\text{gate})}$ and $U^{(\text{gate})}$ are shared between the cells. For classification tasks, the final hidden state h_N is typically considered. Therefore, when calculating the GBM, we focus solely on the last cell in both the forward and backward passes. This calculation is performed using parallel computing to expedite the process. The details of the GBM training method for BiLSTM model are provided in Table 5.

In S4 layer, the parameters A, B, C, and Δ are typically require a smaller learning rate, with no weight decay. The Table 6 outlines the GBM training procedure for the S4 model.

The CNN model is trained using the GBM approach as describe the Table 7. All experiments were conducted on a single NVIDIA A100 80GB GPU.

Dataset	IMDB	Yahoo! Answers
Optimizer	Adam	
Batch size	64	
Hidden size	64	
Learning rate	10^{-3}	
Weight decay	10^{-4}	
Max length	512	256
Early Stopping	Yes	

Table 5: Training configuration and hyperparameters of the GBM training method for BiLSTM model.

Parameters	A, B, C, Δ	Others
Optimizer	Adam	
Batch size	64	
Hidden size	256	
Learning rate	5.10^{-3}	5.10^{-4}
Weight decay	0	10^{-2}
Max length	512	
Early Stopping	Yes	

Table 6: Overview of GBM Training Setup and Hyperparameters for S4 Model on the *IMDB* dataset.

Dataset	IMDB	Yahoo! Answers
Optimizer	Adam	
Batch size	64	
Num kernels	128	
Learning rate	10^{-4}	
Weight decay	10^{-4}	
Kernel sizes	(3, 4, 5)	
Max length	512	256
Early Stopping	Yes	

Table 7: Hyperparameter Settings and Training Configuration for GBM in CNN Model.

F Adversarial examples

The Figure 7 demonstrates a synonym substitution attack, in which minor word replacements alter the model's sentiment classification while maintaining the original meaning of the text.

Original Review 1 **Label: 1 (71.06%)**

Two funeral directors in a Welsh village. English humour as opposed to that other stuff from over the Atlantic—how could I resist? My wife and I saw it on March the 6th for our belated Valentine's Day celebration, and both of us enjoyed some good belly laughs. We were going to see another movie later but decided not to because we wanted the experience of this movie to stay with us for the evening. The mortuary scene in the last 20 minutes of the film is worth the wait. It raises issues rarely talked about in the community. But I know three funeral directors, and the humour is right on the money. Highly recommended, and congratulations to the writers. Without you, all the actors, directors, and the others haven't a job on any Monday.

Perturbed Review 1 **Label: 0 (50.75%)**

Two funeral directors in a Welsh village. English humour as opposed to that other stuff from over the Atlantic—how could I resist? My wife and I saw it on March the 6th for our overdue Valentine's Day celebration, and both of us enjoyed some good belly laughing. We were going to see another movie later but decided not to because we wanted the experience of this movie to stay with us for the evening. The mortuary scene in the last 20 minutes of the film is worth the waiting. It raises issues rarely talked about in the community. But I know three funerary directors, and the humour is right on the money. Highly recommended, and felicitations to the writers. Without you, all the actors, directors, and the others haven't a job on any Monday.

Original Review 2 **Label: 1 (93.00%)**

This movie was absolutely incredible! The direction was masterful, keeping the audience engaged from start to finish. The characters were deeply developed, and the dialogue felt natural and immersive. Every scene had purpose, and the emotional weight of the story left a lasting impact. The cinematography was stunning, enhancing the depth of each moment. The soundtrack complemented the narrative beautifully, making it even more compelling. I highly recommend this film to anyone who appreciates powerful storytelling and remarkable performances.

Perturbed Review 2 **Label: 0 (58.14%)**

This film was absolutely astonishing! The direction was proficient, keeping the audience absorbed from start to finish. The characters were deeply crafted, and the dialogue felt realistic and immersive. Every scene had intent, and the emotional weight of the story left a lasting effect. The cinematography was striking, enhancing the depth of each moment. The soundtrack complemented the narrative beautifully, making it even more compelling. I highly recommend this film to anyone who appreciates powerful storytelling and remarkable performances.

Figure 7: Examples of a synonym substitution attack, where words from the original review (**blue**) are replaced with their synonyms (**red**), preserving the semantic meaning of the text but resulting in sentiment misclassification. For instance, in Review 1, replacing words such as "belated" with "overdue", "belly laughs" with "belly laughing", and "congratulations" with "felicitations" leads the prediction to shift from positive (Label: 1, 71.06%) to negative (Label: 0, 50.75%). Similarly, in Review 2, substituting "incredible" with "astonishing", "masterful" with "proficient", and "developed" with "crafted" results in a classification flip from positive (Label: 1, 93%) to negative (Label: 0, 58.14%).

G Algorithms

Algorithm 1: The boundaries of $\mathcal{M}_v$

Input: Θ, U: Weights; b: bias; $\mathbf{V}$, $\mathbf{H}$, $\mathbf{C}$: input intervals

Output: The boundaries of $\mathcal{M}_v$

for $i \leftarrow 1$ **to** d **do**
  for $j \leftarrow 1$ **to** d_0 **do**
    1. Bounding T_o^i by *Proposition 6* (Appendix C);
    2. Bounding $\sigma'(T_o^i)$ by *Proposition 5* (Appendix C);
    3. Bounding c_t^i by *Proposition 7* (Appendix C);
    4. Bounding $\tanh'(c_t^i)$ by *Proposition 5* (Appendix C);
    5. Bounding $\frac{\partial c_t^i}{\partial v_{w_t}^j}$ by *Algorithm.4* (Appendix G);
    $[(\underline{\mathcal{M}_v})_{i,j}, (\overline{\mathcal{M}_v})_{i,j}] \leftarrow$ The boundaries of $\mathcal{M}_v$;
  end
end

Algorithm 2: The boundaries of $\mathcal{M}_h$

Input: Θ, U: Weights; b: bias; $\mathbf{V}$, $\mathbf{H}$, $\mathbf{C}$: input intervals

Output: The boundaries of $\mathcal{M}_h$

for $i \leftarrow 1$ **to** d **do**
  for $j \leftarrow 1$ **to** d **do**
    1. Bounding T_o^i by *Proposition 6* (Appendix C);
    2. Bounding $\sigma'(T_o^i)$ by *Proposition 5* (Appendix C);
    3. Bounding c_t^i by *Proposition 7* (Appendix C);
    4. Bounding $\tanh'(c_t^i)$ by *Proposition 5* (Appendix C);
    5. Bounding $\frac{\partial c_t^i}{\partial h_{t-1}^j}$ by *Algorithm.5*;
    $[(\underline{\mathcal{M}_h})_{i,j}, (\overline{\mathcal{M}_h})_{i,j}] \leftarrow$ The boundaries of $\mathcal{M}_h$;
  end
end

Algorithm 3: The boundaries of $\mathcal{M}_c$

Input: Θ, U: Weights; b: bias; $\mathbf{V}$, $\mathbf{H}$, $\mathbf{C}$: input intervals

Output: The boundaries of $\mathcal{M}_c$

for $i \leftarrow 1$ **to** d **do**
  for $j \leftarrow 1$ **to** d **do**
    1. Bounding T_o^i and T_f^i by *Proposition 6* (Appendix C);
    2. Bounding c_t^i by *Proposition 7* (Appendix C);
    3. Bounding $\tanh'(c_t^i)$ by *Proposition 5* (Appendix C);
    $[(\underline{\mathcal{M}_c})_{i,j}, (\overline{\mathcal{M}_c})_{i,j}] \leftarrow$ The boundaries of $\mathcal{M}_c$;
  end
end

Algorithm 4: The boundaries of A^c

Input: Θ, U: Weights; b: bias; $\mathbf{V}$, $\mathbf{H}$, $\mathbf{C}$: input intervals

Output: The boundaries of A^c

for $i \leftarrow 1$ **to** d **do**
  for $j \leftarrow 1$ **to** d **do**
    1. Bounding T_f^i, T_I^i and T_g^i by *Proposition 6* (Appendix C) ;
    2. Bounding $\sigma'(T_f^i)$, $\sigma'(T_I^i)$ and $\sigma'(T_g^i)$ by *Proposition 5* (Appendix C);
    3. Bounding c_t^i by *Proposition 7* (Appendix C) ;
    4. Using the monotonicity proprety of σ and $\tanh$ functions;
    $[(\underline{A^c})_{i,j}, (\bar{A^c})_{i,j}] \leftarrow$ The boundaries of A^c;
  end
end

Algorithm 5: The boundaries of B^c

Input: Θ, U: Weights; b: bias; $\mathbf{V}$, $\mathbf{H}$, $\mathbf{C}$: input intervals

Output: The boundaries of B^c

for $i \leftarrow 1$ **to** d **do**
  for $j \leftarrow 1$ **to** d **do**
    1. Bounding T_f^i, T_I^i and T_g^i by *Proposition 6* (Appendix C) ;
    2. Bounding $\sigma'(T_f^i)$, $\sigma'(T_I^i)$ and $\sigma'(T_g^i)$ by *Proposition 5* (Appendix C);
    3. Bounding c_t^i by *Proposition 7* (Appendix C) ;
    4. Using the monotonicity proprety of σ and $\tanh$ functions;
    $[(\underline{B^c})_{ij}, (\bar{B^c})_{ij}] \leftarrow$ The boundaries of B^c;
  end
end