Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# WSymBert: A Weighted Neuro-Symbolic Attention Model for Interpretable CVSS Prediction

**Seyedeh Leili Mirtaheri**

leili.mirtaheri@dimes.unical.it

University of Calabria

**Amirhossein Majd**

University of Calabria

**Reza Shahbazian**

University of Palermo

**Georgios Spathoulas**

Norwegian University of Science and Technology

**Aida Akbarzadeh**

Norwegian University of Science and Technology

**Andrea Pugliese**

University of Calabria

**Research Article**

**Keywords:**

**Posted Date:** June 25th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-9727934/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

**Version of Record:** A version of this preprint was published at International Journal of Information Security on October 11th, 2026. See the published version at https://doi.org/10.1007/s10207-026-01342-x.

# WSymBert: A Weighted Neuro-Symbolic Attention Model for Interpretable CVSS Prediction

**Seyedeh Leili Mirtaheri**[1] · **Amirhossein Majd**[1] · **Reza Shahbazian**[2] · **Georgios Spathoulas**[3] · **Aida Akbarzadeh**[3] · **Andrea Pugliese**[1]

Received: date / Accepted: date

**Abstract** The rapid growth in reported software vulnerabilities, combined with substantial delays in manual Common Vulnerability Scoring System (CVSS) assessment, has created a critical need for automated and interpretable vulnerability scoring tools. When vulnerabilities remain unscored for long periods, organizations face greater difficulty in prioritizing patches and mitigation actions, which can leave exploitable weaknesses insufficiently addressed during a critical exposure window. This challenge is particularly important for the CVSS Base metrics, which form the foundation for severity assessment and are essential for timely risk evaluation. This paper presents WSymBERT, an approach for predicting CVSS v3.1 Base metrics directly from vulnerability descriptions by combining contextual language understanding with structured expert knowledge. Rather than relying only on textual patterns (state-of-the-art), the model is guided toward security-relevant cues that reflect expert reasoning about vulnerability characteristics. Consequently, the proposed method aims to improve both predictive performance and interpretability. We evaluate WSymBERT on two well-established benchmark datasets, achieving an average accuracy of 96.2% and consistently outperforming strong transformer baselines, including SBERT and DistilBERT. WSymBERT also reduces prediction error and remains robust on difficult and imbalanced CVSS metrics.

Corresponding author: E-mail: leili.mirtaheri@dimes.unical.it

[1] Department of Informatics, Modeling, Electronics and System Engineering (DIMES), University of Calabria, Arcavacata di Rende, 87036, CS, Italy
E-mail: leili.mirtaheri@dimes.unical.it, amir.majd@dimes.unical.it, andrea.pugliese@unical.it

[2] Department of Humanities, University of Palermo, Palermo, 90128, PA, Italy
E-mail: reza.shahbazian@unipa.it

[3] Department of Information Security and Communication Technology, Norwegian University of Science and Technology (NTNU), Gjøvik, 2815, Norway
E-mail: aida.akbarzadeh@ntnu.no, georgios.spathoulas@ntnu.no

Furthermore, the explainability analysis indicates that its predictions are more closely aligned with expert-relevant evidence and less sensetive to irrelevant tokens. Overall, the results demonstrate that WSymBERT provides an effective, transparent, and practical solution for automated vulnerability severity assessment.

## 1 Introduction

The number of disclosed software vulnerabilities has grown rapidly in recent years, placing increasing pressure on security teams to assess risk and prioritize remediation in a timely manner. At the same time, vulnerability scoring in practice still depends heavily on manual expert analysis [1]. This creates a serious operational bottleneck: many newly published vulnerabilities remain without complete Common Vulnerability Scoring System (CVSS) assessments for extended periods [2], limiting their usefulness for patch prioritization, vulnerability triage, and downstream risk management. When vulnerabilities remain unscored or are scored late, organizations may struggle to identify which issues require urgent action, leaving exploitable weaknesses exposed during a critical response window. The CVSS is the most widely used standard for communicating vulnerability severity in a structured and comparable form [3]. In particular, the CVSS v3.1 Base metrics capture the core exploitability and impact characteristics of a vulnerability and serve as the foundation for further severity analysis. However, assigning these metrics requires interpreting short and often ambiguous natural-language vulnerability descriptions and mapping them to discrete security-relevant categories. This process is time-consuming, expert-dependent, and difficult to scale as the volume of reported vulnerabilities continues to increase. These challenges have motivated growing interest in automated CVSS prediction. Earlier approaches relied on classical machine learning models with manually engineered textual features, while more recent studies have adopted transformer-based language models to learn contextual representations from vulnerability descriptions. Although these methods have improved predictive performance, important limitations remain. In particular, many existing models operate as black boxes: they can produce accurate predictions, but they do not make their reasoning process sufficiently transparent for operational cybersecurity settings, where analysts often need to justify severity decisions. In addition, post-hoc explainability methods may highlight influential words, but they do not ensure that the model reasons in a way that is consistent with expert knowledge or CVSS semantics. This gap is especially important in vulnerability assessment, where prediction quality alone is not enough. A useful scoring system should not only classify vulnerability characteristics accurately but also provide explanations that are aligned with domain logic and traceable to meaningful evidence in the text. This calls for approaches that go beyond purely data-driven pattern matching and incorporate structured security knowledge directly into the learning process.

### 1.1 Problem Definition

To bridge the gap between black-box prediction and domain-aware reasoning, we formulate the task as a neuro-symbolic mapping problem. A software vulnerability is typically reported as an unstructured natural language text detailing its cause, prerequisites, and potential impact. Translating this unstructured description into a standardized severity score requires understanding specific technical characteristics. The CVSS formally captures these characteristics using a structured vector as a standardized, machine-readable text string, representing the values assigned to various metrics. For CVSS version 3.1, this representation depends on eight fundamental Base metrics. Let $d$ denote a textual vulnerability description (e.g., see Example 1). The goal is to predict the corresponding CVSS Base metric vector $\mathbf{M} = \{AV, AC, PR, UI, S, C, I, A\}$, where each component is a categorical variable with discrete values defined in the CVSS specification.

Specifically, these metrics define the *Exploitability* factors, including the Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), and User Interaction (UI). The Scope (S) metric determines whether a vulnerability in one component impacts resources in another. Finally, the *Impact* factors including Confidentiality (C), Integrity (I), and Availability (A), measure the consequences of a successful exploit on the affected system. A typical CVSS vector format resembles a string such as `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H`, detailing the exact categorical value for each metric.

Once $\mathbf{M}$ is known, the numerical Base Score (ranging from 0.0 to 10.0) can be deterministically computed using established equations (see Table 1). The learning objective is therefore to approximate a mapping, $f : d \rightarrow \mathbf{M}$ that is both accurate (matching expert-assigned labels) and interpretable (consistent with domain reasoning). Unlike traditional approaches that rely solely on textual features, this formulation explicitly requires capturing the semantic meaning of unstructured text while enforcing the symbolic constraints of the CVSS model. This study addresses these challenges through WSymBert, a hybrid neuro-symbolic model that predicts all eight CVSS v3.1 base metrics from text. The model integrates symbolic knowledge into the attention layer of SBERT, guiding the network toward domain-specific reasoning. This design improves both prediction accuracy and interpretability, providing a trustworthy foundation for automated vulnerability scoring.

*Example 1* Let $d$ denote a textual vulnerability description. For instance, consider the following description for CVE-2022-33955:

> *"IBM CICS TX 11.1 could allow an attacker with physical access to the system to execute code due using a back and refresh attack."*

The official severity assessment for this vulnerability is as follows:

- CVE ID: CVE-2022-33955
- CVSS v3.1 Base Score: 4.3 (Medium)
- CVSS Vector: `CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L`

Table 1: CVSS v3.1 Base Score Calculation Formulas

| Metric | Equation |
|---|---|
| **ISS (Impact Sub-Score)** | $1 - [(1 - C) \times (1 - I) \times (1 - A)]$ |
| **Impact (Scope Unchanged)** | $6.42 \times \text{ISS}$ |
| **Impact (Scope Changed)** | $7.52 \times (\text{ISS} - 0.029) - 3.25 \times (\text{ISS} - 0.02)^{15}$ |
| **Exploitability** | $8.22 \times AV \times AC \times PR \times UI$ |
| **Base Score (ISS $\leq$ 0)** | 0 |
| **Base Score (Scope Unchanged)** | $\text{Roundup}(\min[(\text{Impact} + \text{Exploitability}), 10])$ |
| **Base Score (Scope Changed)** | $\text{Roundup}(\min[1.08 \times (\text{Impact} + \text{Exploitability}), 10])$ |

### 1.2 Contributions

To address this need, we propose WSymBERT, a weighted neuro-symbolic model for automated prediction of the eight CVSS v3.1 Base metrics from textual vulnerability descriptions. The key idea is to combine contextual language understanding with symbolic expert knowledge so that the model can focus on security-relevant cues while preserving interpretability. Rather than treating explainability as an afterthought, WSymBERT integrates structured guidance into the attention mechanism itself, enabling the model to make predictions that are both accurate and more closely aligned with expert reasoning. The main contributions of this paper are as follows:

- We propose WSymBERT, a neuro-symbolic framework for predicting all eight CVSS v3.1 Base metrics from vulnerability descriptions.
- We introduce a weighted symbolic attention mechanism that incorporates expert knowledge into the model's decision process to guide attention toward semantically and security-relevant cues.
- We evaluate the proposed model on two benchmark datasets and show that it outperforms strong transformer baselines in predictive performance while remaining robust on challenging and imbalanced metrics.
- We provide a quantitative explainability analysis demonstrating that the model's predictions are more closely aligned with expert-relevant evidence and less influenced by irrelevant tokens.

The rest of the paper is organized as follows. Section 2 reviews prior work on automated CVSS prediction, explainable AI, and neuro-symbolic learning. The proposed WSymBERT architecture is presented in section 3. Section 4 describes the datasets, experimental setup, and evaluation results. Finally, section 5 concludes the paper and outlines future directions.

## 2 Related Work

Research on automated CVSS prediction has evolved from simple text-based methods to advanced neural and hybrid systems. The main directions include

classical machine learning (ML), transformers, symbolic and neuro-symbolic models, and studies that integrate explainability or external data sources.

### 2.1 Classical ML Approaches

Early studies used traditional machine learning to predict CVSS scores from text. These models represented vulnerability descriptions as word frequency vectors using *Bag of Words* or *TF–IDF* features and then applied algorithms such as support vector machines, random forests, and XGBoost [4,5,6].

Wang et al. [5] applied XGBoost to predict severity levels and achieved around 92% accuracy but noted that results were strongly dependent on surface-level word matches. Chen et al. [6] used linear regression models that offered transparency through visible feature, yet lacked the capability to capture semantic meaning or sentence structure.

Because these models ignored context and word order, they often failed on metrics that required deeper understanding, such as *Attack Vector (AV)* or *Privileges Required (PR)*. They were interpretable but limited in accuracy, showing a clear need for models that could capture semantics in vulnerability descriptions.

### 2.2 Transformer-Based Models

The next generation of research adopted transformer models such as BERT and SBERT to process text at a contextual level [7,8,9]. Each CVSS metric was treated as a separate classification problem.

The CVSS-BERT framework by Shahid and Debar trained one BERT model per metric and visualized relevant words using gradient-based saliency maps [10]. Later, SecureBERT and CVEDrill extended this approach and achieved higher accuracy across most metrics [8]. DistilBERT was also used to reduce model size and training time while maintaining similar performance [11].

Despite these advances, transformer models remain opaque. They often behave as black boxes and may base decisions on statistical shortcuts rather than real security semantics [12,13]. Even internal attention patterns may not represent true reasoning, as shown by the "Attention is not Explanation" study [14]. As a result, these models are hard for analysts to trust in critical security settings.

### 2.3 Symbolic and Neuro-Symbolic Methods

Symbolic systems use explicit rules or knowledge trees defined by experts. They are fully transparent but have limited ability to generalize. For example, Khazaei et al. used fuzzy logic and rule-based systems to predict CVSS scores but could not handle diverse text or capture complete vectors [15]. Elbaz et

al. applied linear regression on Bag of Words features, offering interpretability but missing deeper semantic relations [16].

Recent work combines symbolic reasoning with neural models to gain both interpretability and accuracy [17,18,19,20]. In these neuro-symbolic systems, neural layers learn language patterns while symbolic components encode domain logic. In the security domain, the symbolic integration of heterogeneous information sources has become essential for advancing vulnerability analysis. Recent studies, such as Tian et al. [21], demonstrate that fusing code semantics with LLM-generated explanations, via cross-modal attention mechanisms, yields significant improvements in detection accuracy over unimodal approaches. Similarly, Zhequ et al. [22] show that leveraging semantic-based complex dependency analysis and fusion paths further enhances the model's ability to capture intricate relationships among code components. These works, reinforce the importance of sophisticated fusion strategies for building more robust and context-aware vulnerability detection systems. The model *(WSym-Bert)* is guided by expert-defined mappings or ontologies that connect key phrases to CVSS metric values. For example, a phrase like "requires local access" can map directly to *Attack Vector = Local*, and "no user interaction required" can map to *User Interaction = None*. When these symbolic rules are embedded into model attention, the system can focus on words that carry real CVSS meaning instead of frequent but irrelevant terms.

This approach merges the contextual understanding of neural models with the structured logic of symbolic reasoning, improving both performance and interpretability. It also allows analysts to trace model outputs back to recognizable domain cues, which is essential for building trust in automated scoring.

### 2.4 Use of External and OSINT Data

Some researchers explored the use of Open Source Intelligence (OSINT) data such as CWE identifiers, exploit availability, or attack references to improve prediction accuracy [11]. Kühn et al. found that adding more descriptive text from OSINT sources increased text length but did not always improve results. Their analysis showed that beyond a certain point, additional context leads to "information saturation."

These results suggest that model performance depends more on the quality and domain relevance of information than on text quantity. This finding supports the use of symbolic guidance, which focuses attention on the most meaningful parts of the text rather than expanding input size.

### 2.5 Explainable AI in Vulnerability Scoring

As neural models became more accurate, researchers began to study explainability in automated CVSS prediction. Post-hoc methods such as Integrated Gradients, SHAP, and LIME have been used to visualize which words influenced model predictions [23,24]. Metrics like Infidelity, Sensitivity, and

Deletion-AUC were proposed to evaluate the quality of explanations [25]. Although helpful, these methods do not change the model's reasoning process. They simply analyze outputs after training and often highlight uninformative or generic words. For practical use in cybersecurity, explainability needs to be embedded into the model architecture itself, not added later.

Prior research has made strong progress in improving the accuracy and understanding of CVSS prediction, but several important gaps remain. Neural models can achieve high predictive performance, yet their internal reasoning is often not transparent or aligned with CVSS logic. Many systems depend on surface-level word associations rather than learning deeper semantic meaning from vulnerability descriptions. In addition, most existing approaches do not integrate expert knowledge or symbolic reasoning directly into the model, which limits their ability to reflect domain logic. Finally, explainability in prior studies is usually added after training through post-hoc analysis tools, which only describe model behavior instead of shaping it. These gaps highlight the need for an approach that combines neural learning with symbolic guidance to achieve both accuracy and interpretability in automated CVSS prediction.

In parallel with the rapid growth of deep learning models, a variety of explainability methods have been developed to attribute the contribution of each feature to the model's decision. Among gradient-based approaches, the *Integrated Gradients* method, grounded in the principles of *sensitivity* and *implementation invariance*, aims to estimate the contribution of each token or feature in a consistent, path-integrated manner [26]. Within the family of Shapley-value–based methods, the SHAP framework[1] (SHapley Additive exPlanations) integrates several additive and linear-additive techniques to compute the importance value of each feature for an individual prediction. These two lines of methods have become standard reference tools in natural language processing (NLP) and text-based security and are commonly used to label word-level importance for individual samples [27, 28]. Such approaches identify, for each instance, the relative contribution and importance of features (e.g., words) in the model's final decision. Quantitative evaluation metrics such as infidelity, sensitivity, and AUC deletion are often employed to assess the quality of the generated explanations. Studies have shown that some of these methods, for instance, simple gradient maps, may remain relatively stable even when deliberate perturbations are applied to the model, which can lead to misleading interpretations. Therefore, relying solely on word-importance visualization is insufficient; both quantitative and qualitative evaluation of explanations are necessary to ensure their reliability and interpretive validity.

## 3 Proposed Method

We propose WSymBert, a weighted neuro symbolic model that predicts the eight CVSS v3.1 Base metrics from a vulnerability description $d$. The model

[1] https://github.com/shap/shap

joins two sources of information. First, semantic embeddings from a fine tuned SBert encoder. Second, symbolic knowledge from expert rules and curated lexicons. The goal is to keep strong language understanding while staying aligned with the CVSS logic.

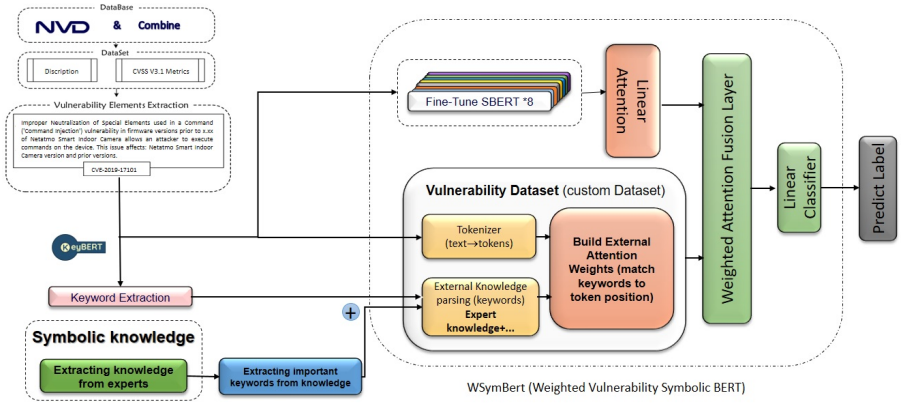



Fig. 1: System overview. SBert encodes the CVE text. A symbolic weighted attention path builds external weights from expert rules and model keywords. The fusion produces a pooled vector that feeds eight heads for each of the Base metrics; where eight separate models have been trained.

### 3.1 Inputs and Notation

Let $d$ be tokenized into $\mathbf{x} = [t_1, \ldots, t_n]$. SBert returns contextual token embeddings $H = [h_1, \ldots, h_n]$, $h_i \in \mathbb{R}^d$. A light attention layer gives a score $\alpha_i$ for each token. Symbolic assets give a non negative weight $w_i$ for each token. A distinct pooled sentence vector $z_m \in \mathbb{R}^d$ is computed separately for each metric $m$, rather than sharing a single vector. Eight linear heads predict the labels for $\{AV, AC, PR, UI, S, C, I, A\}$.

### 3.2 Symbolic Assets

Figure 2 provides an example to show the style of the expert asset used to build lexicons. The decision tree systematically maps common descriptive phrases to the Attack Vector classes: *Network*, *Adjacent*, *Local*, or *Physical*. The resulting leaf sets seed the initial AV lexicon, grounding the symbolic weights in structural domain knowledge.

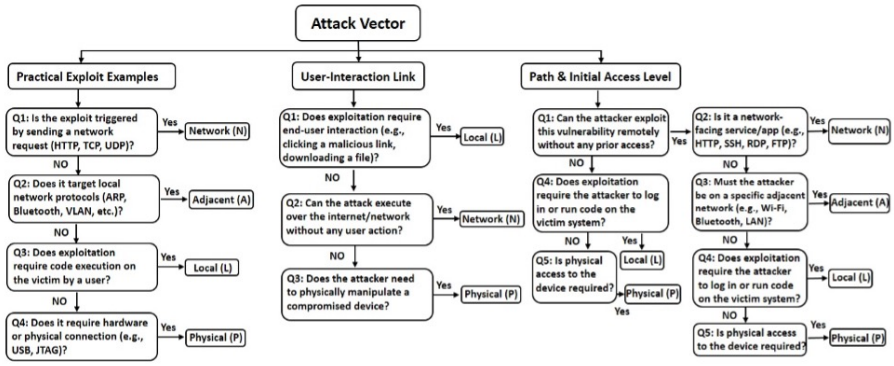



Fig. 2: Attack Vector decision tree. Phrases map to *Network*, *Adjacent*, *Local*, or *Physical*. The leaf sets seed the AV lexicon.

*Decision trees.* For each Base metric we build a question driven tree from the FIRST scoring guide and security ontologies [29,30,31,32]. Paths link common phrases to metric values. For example, "requires local access" maps to $AV = Local$.

*Guiding lexicons.* From the trees we derive per label lexicons of single word cues. Experts review the lists.

*Model keywords.* We extract up to five keywords from each $d$ using KeyBERT [33]. The score for a word $w$ is cosine similarity in the same SBert space,

$$S(w) = \frac{\langle e_d,\, e_w \rangle}{\|e_d\|\,\|e_w\|}, \tag{1}$$

where $e_d$ is the sentence embedding and $e_w$ is the word embedding.

### 3.3 Symbolic Knowledge Extraction

The decision logic of cybersecurity experts in CVSS v3.1 scoring was formalized as traceable, question-driven decision trees, modeling the "question–response–label" reasoning typical in expert assessments. Separate trees were built for the eight base metrics (AV, AC, PR, UI, S, C, I, A), serving as symbolic knowledge modules later integrated into the attention layer.

The initial framework was derived from official CVSS guidelines and combined with structured rubrics and security ontologies to create executable question-based trees.

A stratified analysis of 2,000 CVE descriptions aligned these trees with domain-specific NVD text records. This process achieved 83.6% coverage, meaning that 83.6% of these CVE descriptions contained at least one token matched by the trees. The remaining cases without symbolic matches relied purely on the underlying language model semantics. All trees were developed solely on the training set and frozen before evaluation.

To handle multi-word expressions such as "adjacent network," the lexicons include bigrams and n-grams alongside single tokens. When these phrases are identified during preprocessing, symbolic weights are assigned to their constituent parts. While token-based lexicons often struggle with negations (e.g., "not required"), WSymBERT utilizes SBERT's deep contextual embeddings to resolve these dependencies. By analyzing the broader syntactic structure before the attention pooling stage, the model ensures that individual keywords do not override the actual semantic context.

### 3.4 Symbolic Weighted Attention

*Learned pooling mechanism* A learned pooling mechanism scores each token

$$\alpha_i = \mathrm{softmax}(W_{\mathrm{pool}} h_i + b_{\mathrm{pool}}), \quad W_{\mathrm{pool}} \in \mathbb{R}^{1\times d}. \tag{2}$$

*Symbolic weights* Let $\mathcal{K}_{\mathrm{exp}}$ be the expert lexicon for the current metric and $\mathcal{K}_{\mathrm{mdl}}$ be the KeyBERT keywords for $d$. We assign

$$w_i' = \begin{cases} 10, & t_i \in \mathcal{K}_{\mathrm{exp}} \cap \mathcal{K}_{\mathrm{mdl}}, \\ 1, & t_i \in \mathcal{K}_{\mathrm{exp}} \triangle \mathcal{K}_{\mathrm{mdl}}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

To prevent silent failure, where the absence of symbolic matches collapses the vector to zero (acting as a majority-class baseline), we assign the final weight as $w_i = w_i' + 1$. Thus, the model gracefully falls back to the unweighted learned pooling $\alpha_i$ when no symbolic match exists. This directly addresses the risk of reinforcing class imbalance when no symbolic cues are present. The values $\{0, 1, 10\}$ for $w_i'$ were empirically tuned on the validation set. The value 10 provides a strong guided boost for dual confirmation by expert logic and model semantics, while 1 keeps a weaker cue.

*Fusion and pooling* We form the fused attention and the pooled vector as

$$\beta_i = \frac{\alpha_i\, w_i}{\sum_{j=1}^{n} \alpha_j\, w_j + \varepsilon}, \qquad z_m = \sum_{i=1}^{n} \beta_i\, h_i, \tag{4}$$

with a small $\varepsilon$ for numeric safety. This pooling focuses on tokens that are important both semantically and symbolically.

### 3.5 Multi Head Prediction (Independent Classifier Prediction)

Eight independent light classifiers are trained separately, each using its own metric-specific $z_m$ representation. To clarify the parameter sharing strategy: WSymBERT utilizes a single, shared SBERT encoder for all eight metrics. During training, the SBERT backbone is fine-tuned jointly, while the eight

metric-specific linear classification heads (and their corresponding symbolic weighted pooling layers) are updated simultaneously based on the combined loss. This shared-encoder design minimizes training cost and acts as a regularizer, preventing the parameter explosion that would occur if eight separate language models were trained. For metric $m \in \{AV, AC, PR, UI, S, C, I, A\}$ with classes $\mathcal{Y}_m$ we use

$$\hat{p}_m = \text{softmax}(W_m z_m + b_m), \qquad \hat{y}_m = \arg\max_{y \in \mathcal{Y}_m} \hat{p}_m(y), \tag{5}$$

and we train with the sum of cross entropy losses

$$\mathcal{L} = \sum_m \text{CE}(\hat{p}_m, y_m). \tag{6}$$

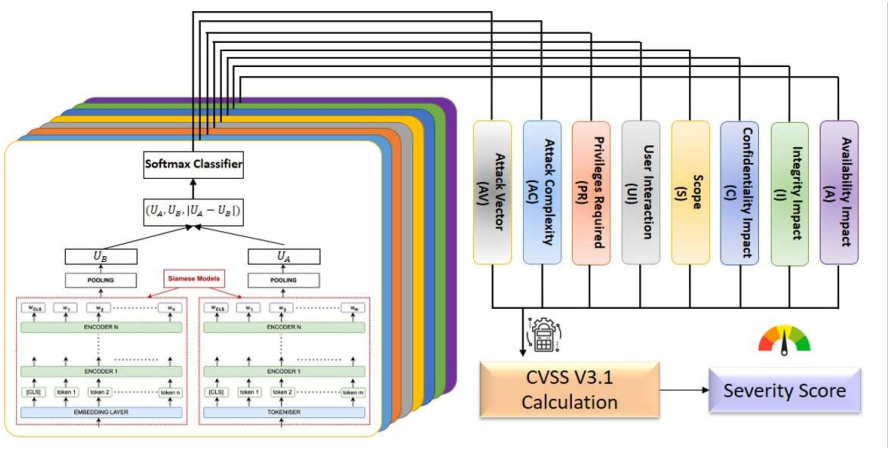



Fig. 3: Component heads. SBert produces the shared embedding. A symbolic weighted pooling produces $z$. Eight linear heads output the Base metrics.

### 3.6 Training Algorithm

We give a short routine for one metric. The same loop runs for all eight heads.

### 3.7 Explainability

We use Integrated Gradients to attribute the contribution of each token to each metric head [34]. We report three measures. Delta checks completeness [35]. Infidelity checks faithfulness under small perturbations [24]. Deletion AUC checks support quality by masking top tokens.

**Algorithm 1** WSymBert training for one metric $m$

**Input:** SBert encoder, expert lexicon $\mathcal{K}_{\text{exp}}^{(m)}$, epochs $E$
Pre-compute and cache KeyBERT keywords $\mathcal{K}_{\text{mdl}}$ for all $d$ using Eq. (1).
**for** epoch $= 1$ to $E$ **do**
  **for** each batch of texts $d$ and labels $y_m$ **do**
    tokenize $d \to [t_1, \ldots, t_n]$, encode $H = [h_1, \ldots, h_n]$
    compute $\alpha_i$ by Eq. (2)
    build $w_i$ by Eq. (3) using cached $\mathcal{K}_{\text{mdl}}$
    compute $\beta_i$ and $z_m$ by Eq. (4)
    get $\hat{p}_m$ by Eq. (5)
    update by minimizing $\mathcal{L}$ in Eq. (6)
  **end for**
**end for**

KeyBERT keywords ($\mathcal{K}_{\text{mdl}}$) are extracted from the frozen SBERT space prior to fine-tuning to serve as static anchors for domain salience. This caching strategy prevents training instability and reduces computational overhead, ensuring that the model does not unlearn initial lexical cues as embeddings evolve during the training process.

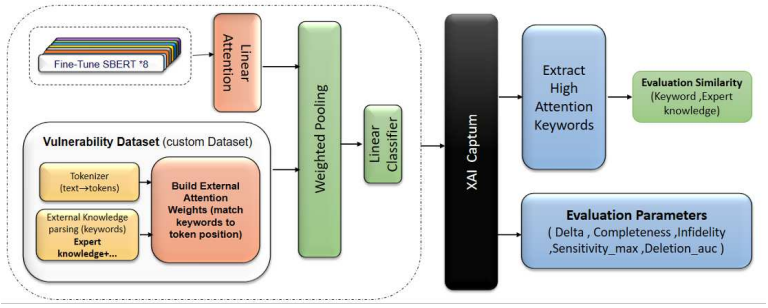



Fig. 4: Explainability flow. Model outputs, IG attributions, and expert keyword overlap give both visual and numeric checks.

*Remarks.* The fusion in Eq. 4 keeps the CVSS logic in view while preserving the strength of SBERT. The eight-head design allows error analysis at the component level. The method is simple to implement and keeps the official CVSS computation unchanged downstream.

## 4 Experiments

To comprehensively evaluate the performance of the proposed WSymBert model, a detailed comparison was conducted against the baseline models SBert [36] and DistilBERT [11] in two independent scenarios. The training process was executed on an NVIDIA GeForce RTX 2080 GPU using the PyTorch deep learning framework. The *AdamW* optimizer was employed with a learning rate of $2 \times 10^{-5}$, a batch size of 16, and 15 training epochs. CrossEntropyLoss (without class weighting) was used as the loss function. After each epoch, the best-performing model was saved based on validation accuracy. These settings ensured stability, reproducibility, and direct comparability across models.

### 4.1 Datasets

Two datasets were used for fine-tuning and evaluation: the *Combine dataset* and the *NVD dataset*. Both contain labeled CVE entries with textual descriptions and corresponding CVSS v3.1 base vector metrics, but they differ in scope, source, and data quality.

*Combine Dataset* The Combine dataset is a merged benchmark introduced in earlier work [11], designed for reproducible CVSS prediction research. It integrates vulnerability records from multiple public sources, including the National Vulnerability Database (NVD), ExploitDB, and other vendor repositories. The dataset covers the years 2016–2021 and contains approximately 101,735 entries. Each record includes a natural-language description of the vulnerability, its associated CVSS v3 base vector (AV, AC, PR, UI, S, C, I, A), and the corresponding numerical base score. Because it combines data from heterogeneous sources, the dataset provides richer linguistic diversity and terminology coverage, though it also contains occasional inconsistencies and overlapping records.

*NVD Dataset (our collected)* The NVD dataset used in this study was fully collected, parsed, and cleaned by the authors from the official National Vulnerability Database[2]. It includes 153,106 records spanning 2016–2025 and strictly follows the NVD data schema for CVSS v3.1. Each record contains a standardized short textual description, the CVSS vector string, and the numerical severity score. Compared to the Combine dataset, this corpus is more consistent and structured but includes shorter text fields, which can make semantic inference more challenging for language models.

*Data Splitting and Deduplication* To prevent data leakage and inflated performance metrics, strict deduplication was applied to both datasets prior to splitting. We removed duplicate and near-duplicate entries based on exact CVE ID matching and exact string matching of descriptions, which is particularly crucial for the aggregated Combine dataset. After deduplication, each dataset was divided into a standard 80-10-10 split for training, validation, and testing, respectively. To maintain a representative distribution of vulnerability characteristics, the splits were stratified by the metric labels. Time-aware splitting was not utilized in this study. Furthermore, all evaluations were conducted within each dataset independently (i.e., models were trained and tested on splits from the same corpus); cross-dataset zero-shot generalization was not evaluated, meaning there was no risk of inter-dataset leakage.

### 4.2 Evaluation Results

Tables 2 and 3 present the performance results across both datasets and all CVSS Base metrics.

[2] (`https://nvd.nist.gov/`)

Table 2 presents the performance results across both datasets and all CVSS Base metrics. For each of the eight components, four evaluation measures were computed: Accuracy, Precision, Recall, and F1-score. To ensure that performance on minority classes is appropriately represented, Precision, Recall, and F1-score were macro-averaged across all classes. This is critical given the severe class imbalance present in metrics like Attack Complexity (AC) and Privileges Required (PR). Best results are highlighted in bold.

Table 2: Comparison of model performance on the Combine dataset. Best values per metric are in bold.

| **Model** | **Metric** | **Acc** | **Prec** | **Rec** | **F1** |
|---|---|---|---|---|---|
| DistilBERT | AV | 91.04 | 82.83 | 75.01 | 78.11 |
| | AC | 92.69 | 96.34 | 57.00 | 56.10 |
| | PR | 84.02 | 76.47 | 77.50 | 76.84 |
| | UI | 91.19 | 90.49 | 90.29 | 90.39 |
| | S | 95.69 | 93.50 | 90.97 | 92.17 |
| | C | 86.28 | 83.98 | 84.17 | 84.07 |
| | I | 87.25 | 87.13 | 85.75 | 86.40 |
| | A | 88.45 | 79.72 | 73.22 | 75.74 |
| SBert | AV | 93.26 | 85.77 | 81.69 | 83.62 |
| | AC | 96.13 | 89.55 | 79.56 | 83.70 |
| | PR | 87.62 | 84.15 | 81.41 | 82.69 |
| | UI | 94.53 | 94.11 | 93.99 | 94.05 |
| | S | 96.74 | 94.97 | 93.36 | 94.14 |
| | C | 89.47 | 88.18 | 87.14 | 87.63 |
| | I | 89.65 | 89.42 | 88.74 | 89.02 |
| | A | 90.93 | 86.11 | 76.44 | 80.05 |
| **WSymBert** | AV | **95.73** | **95.70** | **95.73** | **95.71** |
| | AC | **97.83** | **97.73** | **97.83** | **97.76** |
| | PR | **90.38** | **90.31** | **90.38** | **90.33** |
| | UI | **96.14** | **96.14** | **96.14** | **96.14** |
| | S | **97.64** | **97.63** | **97.64** | **97.63** |
| | C | **93.74** | **93.76** | **93.74** | **93.72** |
| | I | **92.24** | **92.25** | **92.24** | **92.24** |
| | A | **93.70** | **93.58** | **93.70** | **93.58** |

As shown in Table 2, WSymBert consistently achieved the highest performance across all metrics and components in both datasets. On the Combine dataset, WSymBert obtained an average accuracy of approximately 95%, outperforming SBert (92%) and DistilBERT (89%). On the NVD dataset, its accuracy exceeded 96%, again surpassing all baselines by a clear margin. These improvements demonstrate that the symbolic guidance layer allows WSymBert to generalize more effectively and maintain stability across diverse data conditions.

A closer inspection of the results reveals that the baselines exhibit certain weaknesses in specific components. In the Attack Complexity (AC) metric, DistilBERT achieved high Precision (96%) but suffered from severely low Recall (50%), leading to a degraded F1-score of 48%. SBert provided a more balanced performance (F1 = 84%), but WSymBert maintained both Precision and Recall above 97%, leading to a near-perfect F1-score (97.7%). This pattern

Table 3: Comparison of model performance on the NVD dataset (our collected). Best values per metric are in bold.

| Model | Metric | Acc | Prec | Rec | F1 |
|---|---|---|---|---|---|
| DistilBERT | AV | 91.59 | 81.65 | 75.36 | 77.89 |
| | AC | 94.47 | 97.24 | 55.00 | 58.33 |
| | PR | 82.29 | 77.81 | 75.32 | 76.47 |
| | UI | 92.08 | 91.52 | 90.86 | 91.18 |
| | S | 96.02 | 95.44 | 90.83 | 92.93 |
| | C | 85.18 | 84.54 | 81.83 | 83.05 |
| | I | 86.05 | 85.67 | 85.26 | 85.46 |
| | A | 87.49 | 74.95 | 70.03 | 71.97 |
| SBert | AV | 93.46 | 83.82 | 80.73 | 82.14 |
| | AC | 96.31 | 83.54 | 78.89 | 81.01 |
| | PR | 85.48 | 81.60 | 79.94 | 80.74 |
| | UI | 94.56 | 94.12 | 93.82 | 93.97 |
| | S | 96.46 | 94.42 | 93.52 | 93.96 |
| | C | 87.62 | 86.00 | 86.08 | 86.04 |
| | I | 88.75 | 88.82 | 87.79 | 88.27 |
| | A | 90.23 | 79.18 | 73.02 | 75.39 |
| **WSymBert** | AV | **97.22** | **97.22** | **97.22** | **97.21** |
| | AC | **96.47** | **96.18** | **96.47** | **96.02** |
| | PR | **94.23** | **94.20** | **94.23** | **94.18** |
| | UI | **97.44** | **97.44** | **97.44** | **97.44** |
| | S | **98.79** | **98.78** | **98.79** | **98.78** |
| | C | **95.12** | **95.11** | **95.12** | **95.09** |
| | I | **95.09** | **95.10** | **95.09** | **95.08** |
| | A | **95.46** | **95.38** | **95.46** | **95.38** |

confirms that symbolic guidance helps mitigate overfitting to surface-level lexical features and supports reasoning over contextual relationships in the text. In the Privileges Required (PR) and Availability (A) components—both known to be challenging due to skewed class distributions—WSymBert achieved clear improvements of 3–6% in F1-score compared to SBert and up to 15% compared to DistilBERT. These gains indicate that symbolic attention reduces model bias toward dominant classes and strengthens detection of rare categories.

### 4.3 Comparison of Independent Dataset Evaluations

Note that this section compares the in-domain test results obtained independently on the two corpora; as stated previously, models were trained and evaluated entirely within each dataset to prevent cross-contamination. The results also show that WSymBert works effectively with the shorter text descriptions in the NVD dataset, maintaining high accuracy even when there is less linguistic detail to learn from. To better understand how class imbalance affects performance, Table 4 lists the percentage of each label for every CVSS component in the NVD dataset. It shows that several metrics, especially Attack Complexity (AC), Availability (A), and Privileges Required (PR), are highly imbalanced, with one class representing more than 90% of all examples. Despite this imbalance, WSymBert maintained stable Precision and Recall val-

ues, demonstrating that it remains reliable even when the data distribution is uneven.

### 4.4 Evaluation of Errors

To evaluate the numerical accuracy of the predicted CVSS scores, we used Mean Squared Error (MSE) and Mean Absolute Error (MAE). Lower values in both metrics indicate better predictive performance. The reported results are the averages over independent runs to ensure statistical stability. As shown in Table 5, WSymBert achieved the lowest errors in both metrics, confirming its higher accuracy and consistency compared to the baseline DistilBERT model.

These results confirm that WSymBert not only provides more accurate numerical estimates but also maintains stable performance across varying input conditions, reflecting the benefit of integrating symbolic weighting within its attention mechanism.

### 4.5 Integrated Gradients Analysis

Integrated Gradients (IG) were used to visualize interpretability by identifying influential tokens for each CVSS metric. Table 6 shows representative examples comparing WSymBert and DistilBERT. The highlighted tokens reveal that WSymBert consistently focuses on domain-specific terminology that closely aligns with the expert lexicon (e.g., words like "remote" or "authenticated"). In contrast, baseline models often distribute attribution weight across less informative stopwords or generic terms. This focused attribution confirms that the symbolic weighting mechanism successfully steers the model's reasoning toward expert-defined logic rather than spurious statistical correlations.

### 4.6 Noise Analysis

Not all tokens highlighted by interpretability methods provide meaningful or domain-relevant information. To measure this, we defined four related metrics

Table 4: Percentage of each label for every CVSS Base metric in the NVD dataset.

| **Metric** | **Label Distribution (%)** |
|---|---|
| Attack Vector (AV) | Network: 73.9, Local: 23.0, Adjacent: 2.1, Physical: 1.0 |
| Attack Complexity (AC) | Low: **94.0**, High: **6.0** |
| Privileges Required (PR) | None: 62.4, Low: 29.4, High: 8.2 |
| User Interaction (UI) | None: 65.4, Required: 34.6 |
| Scope (S) | Unchanged: 82.2, Changed: 17.8 |
| Confidentiality (C) | High: 57.5, None: 21.7, Low: 20.7 |
| Integrity (I) | High: 49.5, None: 31.1, Low: 19.4 |
| Availability (A) | High: 57.5, None: 40.3, Low: 2.2 |

Table 5: Comparison of MSE and MAE for CVSS score prediction.

| Model | MSE | MAE |
|---|---|---|
| DistilBERT | $1.85 \pm 0.043$ | $0.71 \pm 0.026$ |
| WSymBert | $\mathbf{1.53 \pm 0.031}$ | $\mathbf{0.65 \pm 0.012}$ |

that capture different types of *semantic noise* among the top-ranked tokens (Top-$k$):

- **Stopwords:** Common words that appear very often but carry little meaning, such as *the*, *a*, *of*, *to*, and *in*. These words do not help the model distinguish between vulnerability classes and may lower interpretability quality.
- **Numeric or Version Tokens:** Tokens made up only of numbers or software version patterns, such as *123* or *7.5.1*. These tokens vary widely and usually do not have a consistent semantic role.
- **Short Tokens:** Tokens with two or fewer characters, such as *ip*, *os*, or *c*. These are often ambiguous and may overlap by chance, providing little useful information.
- **Noise Rate:** The proportion of tokens that belong to any of the three categories above among the total number of top-$k$ tokens. This value gives an overall measure of the amount of noise in the model's explanations.

For each CVSS component, the top-10 tokens were selected according to their Integrated Gradients (IG) attribution scores, and the noise metrics were computed for both WSymBERT and DistilBERT. Representative examples and values are shown in Table 7.

Results demonstrate that WSymBERT consistently produces lower noise levels across all metrics. Its Noise Rate remained below 5% for every component, while DistilBERT exceeded 30% in many cases. This shows that WSymBERT focuses on clearer and more domain-specific terms, while DistilBERT often relies on less informative words such as stopwords, numbers, or short tokens. In practice, WSymBERT's explanations are easier to understand and more helpful for experts who analyze and score vulnerabilities, making its outputs both interpretable and trustworthy.

### 4.7 Interpretability Evaluation

One of the main advantages of the proposed WSymBERT model compared to the baseline models is its higher interpretability and more reliable explanations. To carefully test this ability, we measured and analyzed several quantitative interpretability metrics. In this section, five key metrics are discussed: $\Delta$ (interpretation sensitivity), Completeness (how fully feature contributions are captured), Infidelity (how consistent explanations remain under small input changes), Sensitivity, and Deletion AUC (how much performance drops when important features are removed).

Table 6: Examples comparison of metrics with corresponding attention outputs for WsymBert and DistilBert.

| Metric | Description | Attention output of WsymBert | Attention output of reference paper (DistilBert) |
|---|---|---|---|
| AV | IBM AIX 7.1, 7.2, 7.3, and VIOS 3.1 could allow a non-privileged local user to exploit a vulnerability in the invscout command to execute arbitrary commands. IBM X-Force ID: 251207. | ['invscout', 'commands', 'privileged', '251207', 'exploit', 'execute', 'IBM', 'local', 'arbitrary', 'VIOS'] | ['local', 'VIOS', 'AIX', 'could', '1', '1', 'the', '7', 'and', 'co'] |
| AC | An information disclosure vulnerability exists when the Windows kernel improperly handles objects in memory, aka 'Windows Kernel Information Disclosure Vulnerability'. This CVE ID is unique from CVE-2020-1367, CVE-2020-1389, CVE-2020-1419. | ['disclosure', 'Disclosure', 'memory', 'exists', 'information', 'kernel', 'unique', 'objects', 'Vulnerability', 'CVE'] | ['information', '2020', '7', 'aka', 'when', 'in', 'ly', 'CVE', 'E', 'An'] |
| PR | customapp in Cybozu Office 9.9.0 through 10.3.0 allows remote authenticated users to cause a denial of service via unspecified vectors, a different vulnerability than CVE-2015-8489. | ['authenticated', 'unspecified', 'users', 'Cybozu', '8489', 'denial', 'service', 'allows', 'customapp', 'different'] | ['authenticated', 'customapp', '0', 'users', 'denial', 'a', '9', 'of', '8489', 'bozu'] |
| UI | The AJAX Login and Registration modal popup + inline form plugin for WordPress is vulnerable to Reflected Cross-Site Scripting due to the use of add_query_arg without appropriate escaping on the URL in all versions up to, and including, 2.24. This makes it possible for unauthenticated attackers to inject arbitrary web scripts in pages that execute if they can successfully trick a user into performing an action such as clicking on a link. | ['Reflected', 'unauthenticated', 'AJAX', 'successfully', 'Scripting', 'performing', 'execute', 'scripts', 'attackers', 'trick'] | ['without', 'pop', 'plug', 'up', 'an', 'in', 'to', 'including', 'web', '2'] |
| S | An arbitrary file write vulnerability in RGCMS v1.06 allows attackers to execute arbitrary code via a crafted PHP file. | ['attackers', 'RGCMS', 'PHP', 'vulnerability', 'crafted', 'execute', 'code', 'v1', 'file', 'arbitrary'] | ['to', 'execute', 'RGCMS', 'a', 'file', '06', 'write', 'attackers', 'file', 'An'] |
| C | IBM CICS TX 11.1 could allow allow an attacker with physical access to the system to execute code due using a back and refresh attack. IBM X-Force ID: 229312. | ['execute', '229312', 'physical', 'attacker', 'refresh', 'IBM', 'using', 'allow', 'TX', 'could'] | ['code', 'TX', '229312', 'X', 'and', '1', 'to', 'using', 'execute', '11'] |
| I | On BIG-IP 14.0.0-14.1.0.1, 13.0.0-13.1.1.4, 12.1.0-12.1.4, 11.6.1-11.6.3.4, and 11.5.2-11.5.8, users with the Resource Administrator role can modify sensitive portions of the filesystem if provided Advanced Shell Access, such as editing /etc/passwd. This allows modifications to user objects and is contrary to our definition for the Resource Administrator (RA) role restrictions. | ['Administrator', 'Administrator', 'users', 'user', 'passwd', 'Resource', 'Advanced', 'filesystem', 'objects', 'modify'] | ['contrary', 'editing', 'modify', '11', 'the', '4', 'our', '0', '3', 'role'] |
| A | The mintToken function of a smart contract implementation for SECoin, an Ethereum token, has an integer overflow that allows the owner of the contract to set the balance of an arbitrary user to any value. | ['mintToken', 'Ethereum', 'SECoin', 'implementation', 'value', 'owner', 'balance', 'contract', 'any', 'function'] | ['mintTok', 'the', 'flow', 'the', 'for', 'arbitrary', 'SECoin', 'an', 'allows', 'contract'] |

Table 7: Comparison of Noise Rates across CVSS components for WSymBERT and DistilBERT.

| CVSS Metric | Model | Stopwords (%) | Numeric Tokens (%) | Short Tokens (%) | Noise Rate (%) |
|---|---|---|---|---|---|
| AV | DistilBERT | 18.4 | 7.2 | 5.5 | 31.1 |
| | WSymBERT | 1.2 | 1.5 | 2.0 | **4.7** |
| AC | DistilBERT | 21.5 | 5.8 | 4.1 | 31.4 |
| | WSymBERT | 0.8 | 1.1 | 2.3 | **4.2** |
| PR | DistilBERT | 19.9 | 6.3 | 5.0 | 31.2 |
| | WSymBERT | 1.0 | 1.3 | 2.1 | **4.4** |
| UI | DistilBERT | 17.2 | 8.6 | 6.5 | 32.3 |
| | WSymBERT | 1.4 | 1.6 | 2.0 | **5.0** |

In our context, lower Infidelity and Deletion AUC scores indicate that the model relies heavily on the specific symbolic cues identified, supporting the claim that the weighting mechanism produces meaningful and faithful attributions. A lower $\Delta$ confirms that the explanation completely describes the model's prediction.

Because of the way the inputs are designed, the baseline SBERT model processes only one type of input text. For this reason, one baseline setting was used for each experiment, called *Pad* and *One*. The proposed WSymBERT model works differently. It receives two kinds of information at the same time: a weight vector and a set of weighted tokens. Each of these inputs can have its own baseline setting, which means that two baselines are possible for WSymBERT.

By combining these two inputs and their baselines, four different evaluation setups were tested when calculating the interpretability metrics. We denote these setups as $B_1$ through $B_4$: $B_1$ uses padded token embeddings and padded attention weights; $B_2$ uses padded embeddings and one-vector attention; $B_3$ uses one-vector embeddings and padded attention; and $B_4$ uses one-vector embeddings and one-vector attention. Note that for the SBERT baseline, which only has token embeddings, only two setups ("pad" and "one") are applicable.

Table 8 shows that SBERT reports small $\Delta$ values, but its Completeness values are very large, meaning the attribution sums do not match the model outputs. As a result, $\Delta$ is unreliable for SBERT. In contrast, Table 9 shows that WSymBERT achieves both small $\Delta$ and near-unity Completeness, making $\Delta$ a valid measure of unexplained output. This confirms that WSymBERT provides consistent and faithful explanations.

The ideal Completeness value is one. SBERT's values range from about 2 to 6 in Table 8, showing that the attribution sums exceed the true outputs. WSymBERT remains between 0.3 and 0.98 in Table 9, mostly around 0.7 to 1.0, indicating that its attributions align well with model predictions. Lower Infidelity indicates more reliable explanations. SBERT shows high Infidelity across several baselines in Table 8, meaning its attributions fail to track model

Table 8: Results of interpretability metrics for the SBERT model on the NVD dataset.

| Metric | Baseline | $\Delta$ | Completeness | Infidelity | Sensitivity | Deletion AUC |
|---|---|---|---|---|---|---|
| AV | pad | 0.0129 | 6.457 | 0.584 | 2.790 | 0.198 |
| | one | 0.0147 | 6.352 | 11.329 | 1.361 | 0.145 |
| AC | pad | 0.0215 | 3.438 | 0.107 | 2.631 | 0.759 |
| | one | 0.0092 | 2.180 | 7.438 | 1.503 | 0.671 |
| PR | pad | 0.0050 | 5.157 | 0.350 | 3.129 | 0.581 |
| | one | 0.0035 | 3.499 | 12.068 | 1.762 | 0.440 |
| UI | pad | 0.0558 | 4.234 | 0.154 | 2.515 | 0.613 |
| | one | 0.0164 | 3.833 | 5.296 | 1.722 | 0.474 |
| S | pad | 0.0053 | 4.051 | 0.069 | 1.978 | 0.704 |
| | one | 0.0107 | 3.934 | 3.191 | 1.238 | 0.512 |
| C | pad | 0.0100 | 5.776 | 0.333 | 2.645 | 0.255 |
| | one | 0.0354 | 6.005 | 11.372 | 1.416 | 0.217 |
| I | pad | 0.0120 | 5.289 | 0.572 | 2.044 | 0.309 |
| | one | 0.0113 | 5.800 | 7.452 | 1.396 | 0.259 |
| A | pad | 0.0243 | 5.125 | 0.235 | 2.584 | 0.554 |
| | one | 0.0109 | 4.803 | 8.492 | 1.301 | 0.399 |

Table 9: Results of interpretability metrics for the WSymBERT model on the NVD dataset.

| Metric | Baseline | $\Delta$ | Completeness | Infidelity | Sensitivity | Deletion AUC |
|---|---|---|---|---|---|---|
| AV | $B_1$ | 0.1272 | 0.323 | 0.00038 | 0.585 | 0.321 |
| | $B_2$ | 0.1288 | 0.323 | 0.00040 | 0.557 | 0.312 |
| | $B_3$ | 0.0177 | 0.298 | 0.01523 | 0.303 | 0.297 |
| | $B_4$ | 0.0175 | 0.299 | 0.01484 | 0.299 | 0.291 |
| AC | $B_1$ | 0.5424 | 0.986 | 0.00096 | 1.373 | 0.374 |
| | $B_2$ | 0.5403 | 0.984 | 0.00098 | 1.329 | 0.376 |
| | $B_3$ | 0.1616 | 0.693 | 0.06376 | 0.441 | 0.367 |
| | $B_4$ | 0.1335 | 0.689 | 0.06542 | 0.423 | 0.359 |
| PR | $B_4$ | [0,0041] | [0,6734] | [0,0301] | [0,3253] | [0,2594] |
| UI | $B_4$ | [0,0296] | [0,2218] | [0,0105] | [0,4208] | [0,2844] |
| S | $B_4$ | [0,0735] | [0,4337] | [0,0246] | [0,5390] | [0,3157] |
| C | $B_4$ | [0,0314] | [0,6406] | [0,0510] | [0,2964] | [0,2173] |
| I | $B_4$ | [0,0303] | [0,8078] | [0,0492] | [0,2820] | [0,3227] |
| A | $B_4$ | 0.1289 | 0.535 | 0.01034 | 0.421 | 0.370 |

**Baselines:** $B_1$: Double Masking (Pad Embeddings + Pad Attention); $B_2$: Input Masking (Pad Embeddings + One Attention); $B_3$: Attention Masking (One Embeddings + Pad Attention); $B_4$: Fully Active (One Embeddings + One Attention).

changes under input perturbations. WSymBERT achieves very low Infidelity, typically between $10^{-3}$ and $10^{-2}$ in Table 9, confirming that its explanations accurately follow model behavior. Sensitivity measures how much attribution maps change with small input variations. Lower values indicate greater stability. SBERT exhibits higher Sensitivity in Table 8, while WSymBERT remains lower in Table 9, showing that its explanations are more stable against small text changes, which is important for CVE descriptions that often include minor edits. A lower Deletion AUC means that removing top-ranked tokens quickly reduces model confidence, showing that the highlighted tokens are truly impor-

tant. SBERT shows larger AUC values in Table 8, while WSymBERT achieves smaller ones in Table 9, confirming that it identifies words with stronger decision influence.

## 5 Conclusion and Future Work

This paper presented WSymBERT, a fusion-centric neuro-symbolic model that predicts complete CVSS v3.1 vectors from vulnerability descriptions by explicitly integrating multiple information sources. WSymBERT leverages the language representation strength of SBERT and augments it with expert symbolic knowledge, employing a weighted attention fusion layer that dynamically blends learned neural importance with domain-driven symbolic weighting. This fusion mechanism enables WSymBERT to consistently surpass baseline neural models in both accuracy and interpretability, as demonstrated on NVD and Combine datasets across all CVSS components. The model's average accuracy of 96.2% and lower mean absolute error underscore the effectiveness of multi-source fusion, particularly when handling imbalanced and real-world data. Interpretability analysis, using Integrated Gradients and metrics such as $\Delta$, Infidelity, and Deletion AUC, highlights that WSymBERT directs attention toward key domain concepts, closely aligning with expert reasoning and minimizing focus on irrelevant features. This leads to more robust, transparent, and trustworthy vulnerability assessment, moving beyond black-box predictions. By showing that the fusion of symbolic reasoning and deep learning provides both predictive strength and meaningful explanations, WSymBERT sets a direction for responsible, explainable, and practical AI solutions in the evolving field of cybersecurity.

Future work can extend WSymBERT in several directions. One promising avenue is the integration of symbolic knowledge with structured data from external sources, such as CWE, CPE, and Exploit-DB, to enhance context understanding. However, the reliance on predefined symbolic cues introduces a potential attack surface: adversaries could craft vulnerability descriptions deliberately embedded with salient lexicon cues (e.g., "remote," "unauthenticated") to artificially inflate or manipulate provisional severity scores. To mitigate this threat, future deployments should integrate verification mechanisms, such as cross-checking vendor fields, discounting self-assertive language, and enforcing conservative abstention policies when the model flags highly conflicting signals. Additionally, due to the high apparent performance on heavily imbalanced metrics, explicit confidence calibration is needed to prevent overconfidence in production, ensuring that borderline cases are escalated to human analysts. The model could also be expanded to handle dynamic CVSS scoring by incorporating temporal and environmental metrics, allowing it to update as threat conditions evolve. Leveraging cybersecurity knowledge graphs to represent expert reasoning and link relationships among CVEs is another valuable direction. Real-time applications are also possible, with WSymBERT deployed within security information and event management (SIEM) systems to support

live vulnerability prioritization. Additionally, future iterations could facilitate human–AI collaboration through interactive dashboards, enabling analysts to visualize model explanations and adjust symbolic weights in real time.

## Declarations

### Data availability

The datasets generated and analyzed during the current study for the experiments are available in the following: Link to Data.

### Use of AI and AI-assisted technologies

During the preparation of this work, the authors used GPT-4 and Quillbot in order to improve the linguistic quality, formatting of LaTeX tables, and overall readability of the manuscript. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the content of the publication.

### Conflict of interest

The authors declare that they have no conflict of interest.

**Acknowledgements** This work was supported by:

- The SERICS project (PE00000014) under the NRRP MUR program (funded by the European Union - NextGenerationEU), supporting A. Majd, S. L. Mirtaheri, and A. Pugliese.
- The ENFIELD (European Lighthouse on Safe and Trustworthy AI) project, funded by the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101120658, supporting A. Akbarzadeh, S. L. Mirtaheri , and G. Spathoulas.

## References

1. Jiao Yin, Wei Hong, Hua Wang, Jinli Cao, Yuan Miao, and Yanchun Zhang. A compact vulnerability knowledge graph for risk assessment. ACM Transactions on Knowledge Discovery from Data, 18(8):1–17, 2024.
2. Myriam Ouraou. Beyond the cvss: Rethinking the contextualisation of cves in a connected world. European Conference on Cyber Warfare and Security, 24(1):490–500, 2025.
3. Peter Mell, Karen Scarfone, Sasha Romanosky, et al. A complete guide to the common vulnerability scoring system version 2.0. In Published by FIRST-forum of incident response and security teams, volume 1, page 23, 2007.
4. G. Spanos and L. Angelis. A multi-target approach to estimate software vulnerability characteristics and severity scores. Journal of Systems and Software, 146:152–166, 2018.

5. P. Wang, Y. Zhou, B. Sun, and W. Zhang. Intelligent prediction of vulnerability severity level based on text mining and xgboost. In International Conference on Advanced Computational Intelligence (ICACI), pages 72–77. IEEE, 2019.
6. J. Chen, P. K. Kudjo, S. Mensah, S. A. Brown, and G. Akorfu. An automatic software vulnerability classification framework using term frequency-inverse gravity moment and feature selection. Journal of Systems and Software, 167:110616, 2020.
7. Seyedeh Leili Mirtaheri, Andrea Pugliese, Narges Movahed, and Reza Shahbazian. A comparative analysis on using gpt and bert for automated vulnerability scoring. Intelligent Systems with Applications, 26:200515, 2025.
8. Ehsan Aghaei, Ehab Al-Shaer, Waseem Shadid, and Xi Niu. Automated cve analysis for threat prioritization and impact prediction. arXiv (Cornell University), 2023.
9. N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084, 2019.
10. M. R. Shahid and H. Debar. Cvss-bert: Explainable natural language processing to determine the severity of a computer security vulnerability from its description. In IEEE International Conference on Machine Learning and Applications (ICMLA), pages 1600–1607. IEEE, 2021.
11. Philipp Kühn, David N. Relke, and Christian Reuter. Common vulnerability scoring system prediction based on open source intelligence information sources. Computers & Security, 131:103286, 2023.
12. Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. Nature Machine Intelligence, 2(11):665–673, 2020.
13. Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. pages 107–112, 2018.
14. Sarthak Jain and Byron Wallace. Attention is not explanation. arXiv (Cornell University), 2019.
15. Atefeh Khazaei, Mohammad Ghasemzadeh, and V. Derhami. An automatic method for cvss score prediction using vulnerabilities description. Journal of Intelligent & Fuzzy Systems, 30(1):89–96, 2015.
16. C. Elbaz, L. Rilling, and C. Morin. Fighting n-day vulnerabilities with automated cvss vector prediction at disclosure. In International Conference on Availability, Reliability and Security (ARES), pages 1–10. ACM, 2020.
17. T. R. Besold, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K. U. Kühnberger, and G. Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. In Neuro-Symbolic Artificial Intelligence: The State of the Art, pages 1–51. IOS Press, 2021.
18. A. S. Garcez, T. R. Besold, L. De Raedt, P. Földiak, P. Hitzler, T. Icard, and D. L. Silver. Neural-symbolic learning and reasoning: Contributions and challenges. In AAAI Spring Symposium, pages 18–21, 2015.
19. X. Zhang and V. S. Sheng. Neuro-symbolic ai: Explainability, challenges, and future trends. arXiv preprint arXiv:2411.04383, 2024.
20. B. C. Colelough and W. Regli. Neuro-symbolic ai in 2024: A systematic review. arXiv preprint arXiv:2501.05435, 2025.
21. Zhenzhou Tian, Minghao Li, Jiaze Sun, Yanping Chen, and Lingwei Chen. Enhancing vulnerability detection by fusing code semantic features with llm-generated explanations. Information Fusion, page 103450, 2025.
22. Sun Zhequ, Siyu Wang, Ning Lu, Wenbo Shi, and Zhiquan Liu. Mdvul: A semantic-based complex dependency code vulnerability detection using fusion path. Information Fusion, page 103475, 2025.
23. Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. arXiv (Cornell University), 2017.
24. Dennis Du. Optimizing the infidelity, sensitivity, and complexity of feature importance explanations for machine learning models. Digital Access to Scholarship at Harvard (DASH) (Harvard University), 2025.
25. Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv (Cornell University), 2017.

26. Ali Aghababaei, Jan Nikadon, Magdalena Formanowicz, Maria Laura Bettinsoli, Carmen Cervone, Caterina Suitner, and Tomaso Erseghe. Application of integrated gradients explainability to sociopsychological semantic markers. arXiv preprint arXiv:2503.04989, 2025.
27. Ching Lam Choi, Alexandre Duplessis, and Serge Belongie. Unlearning-based neural interpretations. arXiv preprint arXiv:2410.08069, 2024.
28. Mahdi Dhaini, Kafaite Zahra Hussain, Efstratios Zaradoukas, and Gjergji Kasneci. Evalxnlp: A framework for benchmarking post-hoc explainability methods on nlp models. arXiv preprint arXiv:2505.01238, 2025.
29. Massimiliano Albanese, Ibifubara Iganibo, and Olutola Adebiyi. A framework for designing vulnerability metrics. Computers & Security, 132:103382, 2023.
30. J. A. Wang, M. Guo, H. Wang, M. Xia, and L. Zhou. Ontology-based security assessment for software products. In Workshop on Cyber Security and Information Intelligence Research, pages 1–4. ACM, 2009.
31. Ju An Wang, Minzhe Guo, Hao Wang, Min Xia, and Linfeng Zhou. Environmental metrics for software security based on a vulnerability ontology. 1:159–168, 2009.
32. J. A. Wang and M. Guo. Ovm: An ontology for vulnerability management. In Workshop on Cyber Security and Information Intelligence Research, pages 1–4. ACM, 2009.
33. J. Sammet and R. Krestel. Domain-specific keyword extraction using bert. In Conference on Language, Data and Knowledge, pages 659–665, 2023.
34. Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. arXiv (Cornell University), 2017.
35. Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. arXiv (Cornell University), 31:9505–9515, 2018.
36. Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084, 2019.