Large Language Model Based Hybrid Framework for Automatic Vulnerability Detection with Explainable AI for Cybersecurity Enhancement

Large Language Model Based Hybrid Framework for Automatic Vulnerability Detection with Explainable AI for Cybersecurity Enhancement

Nihala Basheer[a] , Shareeful Islam[a,*] , Mohammed K. S. Alwaheidi[b], Haralambos Mouratidis [c] and Spyridon Papastergiou [d,e]

[a]*School of Computing and Information Science, Anglia Ruskin University, UK, nihala.basheer@aru.ac.uk and shareeful.islam@aru.ac.uk*
[b] *Cybersecurity Consultancy Services Department, Securology, Jeddah 23334, Saudi Arabia, mohammed@securology.net*
[c] *Institute for Analytics and Data Science, School of Computer Science and Electronic Engineering, University of Essex, UK, h.mouratidis@essex.ac.uk*
[d,]*Research and Innovation, MAGGIOLI S.P.A., Italy, spyros.papastergiou@maggioli.gr*
[e] *Department of Informatics ,University of Piraeus, Greece*

Abstract. Organizations nowadays rely on software-intensive systems to support their operations, and threat actors are always in search of vulnerable points within such systems. Effective identification of vulnerabilities within the source code of these systems can significantly enhance overall security. In this regard, AI-based techniques are being taken widely into consideration, but their effectiveness is highly dependent on the size of the source code data and high-quality datasets. Most of these techniques lack contextual information while processing the data, hence adding challenges in a resource-constrained environment. Additionally, most AI models are intrinsically black box in nature, which further makes the internal mechanism of the model and the decision-making process difficult to comprehend. In this context, the presented work contributes towards this direction and develops a novel hybrid framework using LLM model based on CodeBERT with the integration of fine-tuning and Model-Agnostic Meta-Learning for performing effective vulnerability detection using few data instances. This allows few-shot learning of the model in adapting to new vulnerability detection tasks and maintaining high performance on the known cases. Moreover, the black-box nature of traditional AI-based vulnerability detection models is also addressed using advanced Explainable AI (XAI) techniques. Additionally, the framework leverages four dimensions of XAI to enhance interpretability: attention mechanisms, layer-wise analysis, feature contribution, and model confidence scores. Finally, an experiment is performed to demonstrate the effectiveness of the framework for vulnerability detection and explaining the decision-making. The result demonstrates a steady decrease in meta-loss from 0.45 to 0.14, accompanied by an increase in support accuracy from 85.2% to 92.5% and query accuracy from 88.1% to 95.7%. These findings establish the proposed framework as a robust and interpretable solution for vulnerability detection and management.

Keywords: Cyber Security, Few-Shots, Fine-tuning, Explainability, Heatmap, Vulnerability, SHAP, Source Code

1. Introduction

The frequency and sophistication of software vulnerabilities have increased each day, posing significant threats to software-intensive systems. According to the report by the National Vulnerability Database, there were over 28,000 vulnerabilities disclosed in the year 2023 alone, following their continued increase in the earlier years [1]. This increasing number highlights the necessity of detecting vulnerabilities promptly and accurately from the software system, as they frequently stay concealed within intricate codebases. AI-based traditional vulnerability detection techniques are promising yet suffer from limitations

due to the inability to train effectively with limited and diverse data. These techniques also depend much on labelled data and contextual information about the application for the model outcome [2]. In this regard, LLMs have proven to be a revolutionary solution for detecting vulnerabilities, as they utilize pre-training on diversified datasets composed of a wide variety of programming languages and styles, enabling them to obtain contextual information about the application. This enables LLMs to demonstrate a strong understanding of both syntactic and semantic nuances in source code and to identify intricate vulnerabilities [3]. This is possible in LLM by understanding the dependencies among various parts of a codebase and, hence, finding vulnerabilities due to improper interactions between components, such as flawed function usage or variable handling across files. However, due to the parameters and multiple layers in LLMs, it is not possible to understand how these models arrive at their predictions, a concern in high-stakes applications like cybersecurity [4]. This opacity can result in a lack of trust by stakeholders, including developers, security analysts, and organizations that use these tools for the detection of vulnerabilities. The existing contribution focuses on various AI models for vulnerability detection; however, lack of focus on the interpretability and explainability of the model decision-making [5-7]. Only a few works consider systemic architecture that combines distinct phases of the model implementation, overlooking the explainability of the model decision-making.

The novel contribution of this research is the LLM model based on CodeBERT hybrid framework that integrates fine-tuning with Model-Agnostic Meta-Learning (MAML) for effective vulnerability detection with limited data instances. The framework incorporates robust Explainable AI (XAI) techniques to provide insights into the decision-making of the model, thereby enhancing trust and interpretability in the vulnerability detection process. This hybrid framework approach addresses critical challenges in software vulnerability detection by easily adapting to new and diversified vulnerability patterns with minimum labeled data. By leveraging MAML, the model acquires the ability to generalize from limited data, which greatly reduces overhead in the collection of the dataset and its retraining while guaranteeing faster deployment by improving adaptability [8]. This research makes three major contributions.

- Firstly, the work introduces a CodeBERT-based hybrid framework that leverages both fine-tuning and MAML to develop context-specific adaptable models. By utilizing these techniques, the framework tackles the issue of limited labelled datasets, which is often encountered in vulnerability detection within source code. In our previous work, we fine-tuned CodeBERT successfully for specific tasks but were not able to use MAML because of its complexity [18]. This paper overcomes those challenges, successfully embedding MAML into the framework. With MAML's few-shot learning ability, the model can quickly adapt itself to new vulnerability patterns using a small labelled dataset, making it effective, especially in scenarios with sparse annotation. This ensures that the model is not only well-trained but also effective in the task of vulnerability detection for large and complex codebases. This hybrid approach reduces the computational overhead yet retains high accuracy, suitable for resource-constrained environments. Finally, fine-tuning and MAML integration encourage robustness, ensuring that the model is resilient to noisy or incomplete data, which are common in real-world scenarios. The framework further enhances the detection capabilities through a number of architectural innovations, including a multi-head attention mechanism, bi-directional LSTM layers, and an uncertainty quantification module. This not only enables the model to capture complex patterns and long-range dependencies in source code but also provides robust support for decisions.
- The framework incorporates robust XAI practices to explain the model outcomes from a holistic aspect. XAI principles are integrated into the framework to ensure interpretability and reliability of the detection process by leveraging four dimensions: attention mechanisms, layer-wise analysis, feature contribution, and model confidence scores using techniques such as SHAP, LIME, and heatmap. Furthermore, the confidence scores will empower the developers to interpret and act upon the results with greater confidence. XAI enhances the ability of this framework to detect potential biases or anomalies in predictions for a fairer and more reliable system. Further, the explanations provided by XAI give confidence to stakeholders, since the decision process is now understandable and compliant with the organizational standards.
- Finally, the framework is validated through an extensive experiment. The comprehensive testing proves that the system can detect and classify vulnerabilities efficiently under dynamic and fast-evolving threat landscapes. The result demonstrates a steady decrease in meta loss from 0.45 in the first

epoch to 0.14 in the fifth, while the support accuracy increases from 85.2% to 92.5% and query accuracy from 88.1% to 95.7%. This showcases the model's ability to adapt, reduce the error, and predict correctly with a higher probability. Moreover, advanced XAI techniques based on attention mechanisms, layer-by-layer analysis, and feature contribution evaluations are integrated into the framework to ensure trust and interpretability, further increasing its reliability and trustworthiness for the stakeholders. Key features identified with SHAP and LIME include tokens like "static," "int," "size_t," and "struct," which have a positive impact on predictions for certain classes, while terms like "header_len," "sockaddr_x25," and "skb" have a negative impact, which provides insight into the patterns the model associates with each class.

2. Related works

2.1. Vulnerability detection using fine-tuned and few-shot learning

In the fast-evolving domain of LLM, adapting models to diverse tasks with limited labelled data has become a critical challenge. Techniques like fine-tuning and Model-Agnostic Meta-Learning (MAML) have emerged as strong solutions that allow pre-trained models to generalize across tasks and domains effectively. Recent research highlights the integration of fine-tuning and MAML to enhance information extraction (IE) tasks, such as Named Entity Recognition (NER) and Relation Extraction (RE), particularly in data-scarce scenarios [9]. Fine-tuning adapts pre-trained models to specific tasks, improving performance with minimal labelled data, while MAML optimizes model initialization for rapid adaptation through few-shot learning. This combination enables effective generalization across diverse datasets and domains, dealing with problems such as few labelled data and task diversity. The notable frameworks based on these techniques are MsFNER, MICRE, and MAML-en-LLM. MsFNER [5] uses MAML in training for the goal of optimization of initialization towards the entity span detection and classification models for fast adaptation to new tasks and fine-tuning for domain-specific refinement that has shown robust performance on complex datasets. Similarly, MICRE integrates meta in-context learning and fine-tuning into an approach to enhance relation extraction under zero-shot and few-shot learning scenarios [6]. MAML meta-trains LLMs for fast adaptation toward unseen tasks, while fine-tuning of the model during inference refines model performance and enhances the results in both relation classification and relational triple extraction [7]. In addition, the MAML-en-LLM framework further integrates MAML with fine-tuning to improve the in-context learning of LLMs. In particular, MAML-en-LLM employs a two-step optimization process involving inner-loop task adaptation and outer-loop meta-parameter updates, further complemented by fine-tuning on diverse task exemplars. This enables MAML-en-LLM to have the best adaptability, efficiency, and accuracy across domains compared to competing approaches such as MetaICL. Furthermore, the paper by [9] considers ML models such as linear regression, decision tree, and random forest to predict the vulnerability exploitation using the base metrics of the CVSS for secure healthcare supply chain service delivery. The result from the experiment shows 63% accuracy considering the CVE dataset. Also, the paper by [10] adopts Natural Language Processing (NLP) to extract useful threat information specific to assets from text that contains security-related information. Besides, these frameworks identify that high-quality data plays a significant role in robust and reliable LLM performance, specifically for applications like vulnerability detection [11]. Pretraining on large, high-quality datasets allows LLMs to learn patterns, semantics, and contextual relationships effectively. This helps to ensure adaptability and generalization of LLMs in real-world applications, including those for critical domains such as disaster response and medical diagnostics [12].

2.2. Explainable AI in the context of LLM

The increasing presence of LLMs across diverse domains makes the integration of XAI imperative for reasons of transparency, trust, and usability. Recent studies focus on Usable XAI, underlining that it is about improving LLMs with explainability and taking advantage of the human-like reasoning and synthesis of knowledge from LLMs to improve XAI's usability [13]. Various techniques, including attribution-based explanations, knowledge-augmented prompting, and training data augmentation, have been proposed to bridge the gap between technical explainability and practical usability. Advanced methods include causal explainability with counterfactuals and feature extraction for decision explanation, which aim to provide semantic fidelity and deeper insights into model behavior, aligning AI systems with human expectations [14]. Complementing these developments, domain-specific

applications, such as "x-[plAIn]," a domain-specific GPT-based LLM, make XAI accessible to wide classes of users by simplifying complex methods like LIME, SHAP, and Grad-CAM [15]. Specific domain applications, such as financial cybercrime detection using Graph Neural Networks (GNNs) and LLMs, further illustrate the practical value of XAI by integrating narrative generation, anomaly detection, and embedding-based insights to streamline investigative processes [16]. Similarly, for cybersecurity, embedding SHAP values with the LLMs in intrusion detection systems allows one to understand exactly where feature contribution is coming from and aids in trust problems with users [17]. This and other similar advances —an example being high accuracy with a BERT-based model for source code vulnerabilities—emphasize the very need for explainability as an ingredient of trustworthy AI, something fully aligned with worldwide regulations in this direction, such as the EU AI Act [18]. Together, these efforts showcase the potential of XAI to enhance transparency, interpretability, and usability across diverse AI applications.

In summary, fine-tuning and MAML have shown promising integration in solving the challenges of adapting LLMs to diverse tasks, especially for minimal data. With high-quality data supporting such approaches, robust generalization and improved performance can be achieved in critical domains such as cybersecurity, disaster response, and medical diagnosis. Besides, the adoption of XAI techniques like SHAP, LIME, and counterfactuals allows the establishment of transparency, trust, and usability in AI systems to meet global regulatory standards. However, most of the XAI practices are still limited to a few available techniques, which require further innovation to bridge the gap between model performance and explainability.

3. Proposed Framework

The proposed framework is detailed in this section, including the hybrid LLM framework, system architecture, and Explainable AI, for effective vulnerability detection and explaining decision-making. In the hybrid approach, CodeBERT is integrated with fine-tuning and task-conditioned Model-Agnostic Meta-Learning (MAML) in a novel way, allowing the model to adapt dynamically to new vulnerability patterns while still showing impressive performance on the known cases. The methodology is thus embedded in the AI life cycle, from data preprocessing and model training to model evaluation. Each of these steps in the life cycle is towards better accuracy, adaptability, and efficiency consideration with challenges such as computation resource constraints and unavailability of diverse and high-quality data. Finally, XAI techniques are integrated into the approach to provide a detailed understanding of the decision-making process with insights on how vulnerabilities are identified.

3.1. Hybrid Approach

The proposed hybrid approach aims to address the challenge of high-quality diverse training data constraints to reach an optimal level of performance in the outcome model. We perform a hybrid-based approach applying the CodeBERT LLM model together with fine-tuning and Model-Agnostic Meta-Learning (MAML). The reason for choosing CodeBERT is due to its specialization in code representation. It is a transformer-based model designed to especially understand natural language (NL) and programming language (PL) representations. It takes its architecture from BERT and is pre-trained over a large corpus of paired NL-PL data, aiming to capture the relations between code and its natural language description [20]. This makes CodeBERT highly suitable for tasks such as code summarization, code search, and code completion—bridging the gap between human-readable text and machine code [21]. With this capability, it provides impressive performance in code-related tasks, enabling the hybrid approach to support context-specific adaptation and understanding of the code with low data requirements. To further improve adaptation in low-data settings, we incorporate Model-Agnostic Meta-Learning (MAML) into the approach. MAML is a meta-learning algorithm that would optimize the model's initial parameters to let it be applied under new tasks using minimal datasets [7]. Combining the strengths of fine-tuned CodeBERT and MAML, our hybrid framework supports efficient and context-specific learning and processing of source code in resource-constrained environments. This collaboration allows for excellent performance even with a small amount of labeled data while taking advantage of the specialized features of CodeBERT. This optimization in MAML is based on a two-step process, leading to an inner loop for task-specific updates and an outer loop for meta-learning, shown mathematically in Equation 1. Using the capabilities of meta-learning in MAML, the model rapidly adapts to new tasks with small-sized labeled data and significantly reduces the dependence on large, diversified datasets [22]. Hybrid integration maintains a strong balance between

generalization and task-specific performance, which is effective in domains where code patterns show high variability. In addition, fine-tuned CodeBERT gives rise to improved accuracy and understanding of these complex code-related tasks for a more accurate and dependable result.

$min_\theta \sum_{T_i \sim p(T)} L(\theta - \alpha \nabla_\theta L(\theta, D_{T_i}^S), D_{T_i}^Q)$ .....(1)

where,

θ represents the initial model parameters,

$D_{T_i}^S$ and $D_{T_i}^Q$ are the support and query datasets for task T_i,

α is the learning rate for the inner loop task-specific updates, and

θ' are the task-adapted parameters.

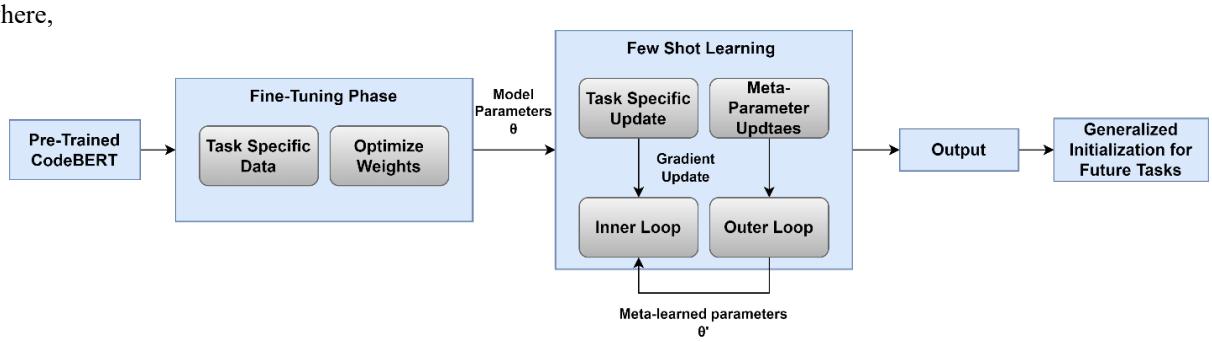

Fig. 1: Integration of MAML with CodeBERT

Figure 1 illustrates the hybrid approach of CodeBERT that integrates fine-tuning and MAML with few-shot learning. The process first begins with CodeBERT, which captures the semantic relationship between natural language and code. It is quite effective for a series of tasks, such as code summarization, code retrieval, and vulnerability detection. This pre-trained model is used as a basis for the hybrid to provide a good initialization of parameters θ that can be fine-tuned and adapted to a specific task. During fine-tuning, the pre-trained CodeBERT is adapted to task-specific requirements with the help of labelled datasets; hence, the model generalizes its representations toward a particular context of the task. Now, the model gets exposed to task-relevant data, for example, code summarization datasets or software vulnerability datasets, to specialize their understanding. Using standard gradient-based optimization methods, the model's weights θ are iteratively updated to minimize the task-specific loss function, ensuring that it better captures the nuance of the application. The fine-tuned weights θ jointly generated during this process play the role of a strong initialization for the following few-shot learning phase, which contains additional task adaptation. The few-shot learning process can be further decomposed into two critical loops, the inner loop and the outer loop.

In the inner loop, the model adapts its fine-tuned parameters θ to individual tasks using small, labeled support datasets $D_{T_i}^S$ from new tasks. Task-specific gradient updates are performed according to $\theta_i' = \theta - \alpha \nabla_\theta L(\theta, D_{T_i}^S)$, where α is the learning rate. These updates allow the model to specialize its parameters (θ_i') for the current task while ensuring efficient adaptation.

The outer loop, the meta-parameters θ are optimized across multiple tasks by evaluating the task-specific updates (θ_i') on query datasets $D_{T_i}^Q$ using the update rule $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{T_i} L(\theta_i', D_{T_i}^Q)$, where β is the meta-learning rate. This ensures that the initialization remains general enough to quickly adapt to a wide range of tasks with minimal task-specific updates in the inner loop. By combining these two loops, MAML optimizes the meta-parameters (θ) for efficient and effective adaptation to new, unseen tasks, even in low-data settings.

3.2. Explainability in LLM

Large Language Models (LLMs) including BERT, and CodeBERT, are powerful tools capable of generating text, making predictions, and answering queries using vast amounts of training data. However, these models are black boxes in nature, making it difficult to understand how they generate their outputs [23]. Lack of transparency may cause problems regarding trust, especially in high-stakes domains such as healthcare, finance, or legal, where decisions supported by the output of LLM might have grave consequences. For example, an LLM-based system detects cybersecurity threats like phishing emails or malware.

It could flag a legitimate financial email as phishing and block it, or it might fail to recognize a spear-phishing attempt and compromise the network. This implies that, within this framework, XAI is necessary in helping to make the inner mechanisms of LLMs transparent and interpretable. By offering insights into how the model arrives at its conclusion, XAI builds up the trust and confidence of its users. The application of XAI on LLMs differs from its application on more conventional models such as decision trees or neural networks [24]. Unlike structured data models, LLMs operate on unstructured text, which makes their decision-making processes inherently more complex. The balanced approach that XAI advocates in LLMs goes beyond mere performance metrics [23]. This is paramount in a time when AI models are being deployed to sensitive domains, where the implications of incorrect or biased outputs could be grave. The uniqueness of XAI in LLMs is that it promotes a holistic view—one where interpretability is valued just as much as the model's capabilities to handle complex tasks. XAI in LLMs focuses on understanding the inner workings of how the LLM model works, providing insights into how it processes input data to generate model outputs. In this context, there are four distinct dimensions of XAI generally considered in LLM models to holistically explain the model outcome, as presented in Figure 2.

Attention mechanism: It enables models to dynamically concentrate on the most relevant parts of the input sequence to generate an output. This attention mechanism works by assigning different degrees of importance to different elements of the input and makes sure that complex dependencies, even in long sequences, are captured by the model [25]. At a basic level, the attention mechanism relies on three key components: the query (Q), the key (K), and the value (V). The query is what the model is paying attention to, usually derived from some current state, such as a hidden state in a sequence model. The key represents the features or information provided by each input token, while the value contains the actual information from these tokens. Collectively, they allow the mechanism to compute a context vector that captures what aspects of the input sequence are most informative to carry through a model with regard to some tasks at hand. The computation begins with calculating a score measuring how much the query is compatible with a key. The score is scaled by the square root of the key's dimension (d_k), leading to equation 2:

$$scaled\ score(q, k_i) = \frac{q \cdot k_i}{\sqrt{d_k}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

The scaled scores are then passed through a SoftMax function to normalize them into probabilities, known as attention weights (α), as shown below:

$$Attention\ (Q, K, V) = \sum_{i=1}^{n} \alpha_i v_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Afterward, this attention weight can be used to explain how the model has placed its focus during training on various parts of the input. We can represent these attention weights as heatmaps that intuitively illustrate which part of the input the model is focusing on using varied color intensities to represent focus strength [26]. In the heatmap, one axis represents each of the input tokens, and the other axis represents the query tokens or sequence positions. The attention weights determine the intensity of the colors in the heatmap; higher weights are represented by brighter or more vivid colors. By representing these weights in a heatmap, the user can intuitively understand which tokens or elements the model considers most relevant to a given query or task.

Layer-wise analysis: It refers to a detailed analysis of how each layer in a model, like a transformer, plays a role in processing and transforming input data into valuable outputs [27]. For instance, for the LLMs, like CodeBERT, contains several stacked transformer layers that have different contributions to extracting information and refining it. The layer-wise analysis helps reveal the internal working mechanism of these layers and provide insights about the hierarchy that the model has in data representation. Attention mechanisms are vital for layer-wise behavior in transformer-based models. For the l -th layer, the attention mechanism computes as shown in equation 4,

$$Attention\left(Q^{(l)}, K^{(l)}, V^{(l)}\right) = softmax\left(\frac{Q^{(l)} K^{(l)T}}{\sqrt{d_k}}\right) V^{(l)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Where $Q^{(l)}, K^{(l)}, V^{(l)}$ are the query, key, and value matrices of the l-th layer, d_k is the dimensionality of the keys, and SoftMax() converts raw scores to probabilities.

Similar to attention mechanisms, layer-wise analysis also utilizes heatmaps to visualize the internal working of a model like CodeBERT. It provides insights into how input data is processed across layers. The intensity of each cell reflects the importance of each token in each layer; a brighter color indicates higher weights of attention. Early layers usually show broader attention, capturing syntactic structures, while later layers focus on task-critical tokens, refining semantic understanding. By visualizing these shifts in focus, heatmaps help uncover hierarchical data representation, highlight key dependencies, and provide a

transparent view of how layers contribute to the model's decision-making, enhancing interpretability and trust.

Feature attribution: This dimension refers to the technique of determining the importance of individual features, like tokens or words, on a model's predictions. In the context of machine learning, and especially with LLMs, feature attribution helps explain how certain elements of the input contribute to the model's output. It assigns a numerical importance score to each feature in the input, indicating its contribution to the output prediction. These scores show which features are most influential in shaping the model's decision. For this purpose, techniques such as SHAP (Shapley Additive Explanations) and LIME (Local Interpretable Model-Agnostic Explanations) are being used [28]. These methods provide robust frameworks for understanding feature importance and interpreting the behavior of complex models. SHAP is a game-theoretic approach, and it estimates the contribution of each feature to the output by considering all possible coalitions of input features [29]. The importance of a feature i is represented by its Shapley value $\emptyset_i$, computed as equation (5):

$$\emptyset_i = \sum_{S \subset N/\{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} [f(S \cup \{i\}) - f(S)] \quad (5)$$

where S is a subset of features excluding i, N is the set of all features, $|S|$ is the size of the subset S, $f(S)$ is the model's output when only features in S are included and $f(S \cup \{i\})$ is the model's output when feature i is added to subset S.

LIME follows a similar approach and locally approximates the model around an instance of interest by adding perturbations to the input and observing changes in the output [18]. LIME then fits a simple interpretable model to these generated perturbed samples and assigns weight to each feature based on its importance. The weight of a feature can be computed by the equation (6):

$$w_i = argmin_w \sum_{j=1}^{m} \pi(x, x_j) [f(x_j) - g_w(x_j)]^2 + \lambda \| w \| \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

Where x represents the original input instance, x_j is a perturbed version of the input, $f(x_j)$ denotes the model's prediction $x_j, g_w(x_j)$ is the interpretable model's prediction for $x_j, \pi(x, x_j)$ is a proximity measure between x and x_j to ensure focus on local perturbations, and $\lambda \| w \|$ is a regularization term to promote the simplicity of the interpretable model.

Model confidence scores: This final dimension refers to a specific numerical value that indicates the certainty or confidence of a machine learning model in its prediction for a given input [30]. It reflects the extent to which the model believes its output is likely to be correct and is typically derived from the probabilities assigned to possible outcomes. In classification tasks, confidence scores often correspond to the probabilities assigned to each class by the model. For example, in a sentiment analysis task with "positive" and "negative" classes, a model might predict a confidence score of 0.85 for "positive" and 0.15 for "negative," indicating 85% confidence that the sentiment is positive. Confidence scores are calculated using mathematical techniques such as the SoftMax function in classification models. For a given class c, the confidence score $P(c|(x)|)$ is computed as:

$$P(c|K|) = \frac{e^{z_c}}{\sum_{k=1}^{K} e^{z_k}}, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

where z_c is the raw score (logit) for class c, K is the total number of classes, and the output is normalized into probabilities that sum to 1. The class with the highest confidence score is often selected as the predicted output.

Model confidence scores can enhance explainability as they provide insight into certainty, or the reliability of predictions made by a model. The model not only provides a raw output; rather, it quantifies how much confidence the model places in its decision, therefore helping users to understand and trust its behavior.

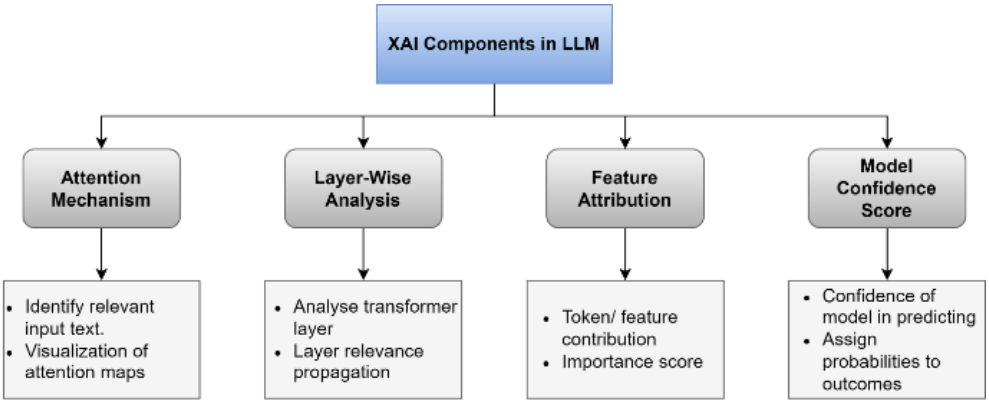

Fig. 2: Various components of XAI in LLM model

3.3. System Architecture

The architecture of the framework consists of three distinct phases in sequence, out of which Phase 2 and Phase 3 include sub-phases, as depicted in Figure 3. During Phase 2, the hybrid model is developed by merging fine-tuning with MAML. Fine-tuning is a sub-phase of Phase 2 that involves the training of a pre-trained CodeBERT model on the prepared dataset for the optimization of parameters with respect to the task. This results in a hybrid model that combines the strengths of BERT fine-tuning and MAML for robust and task-adaptive performance. While the hybrid model developed in Phase 2 provides the required robustness, integrating XAI techniques in Phase 3 ensures that the performance is not only effective but also interpretable.

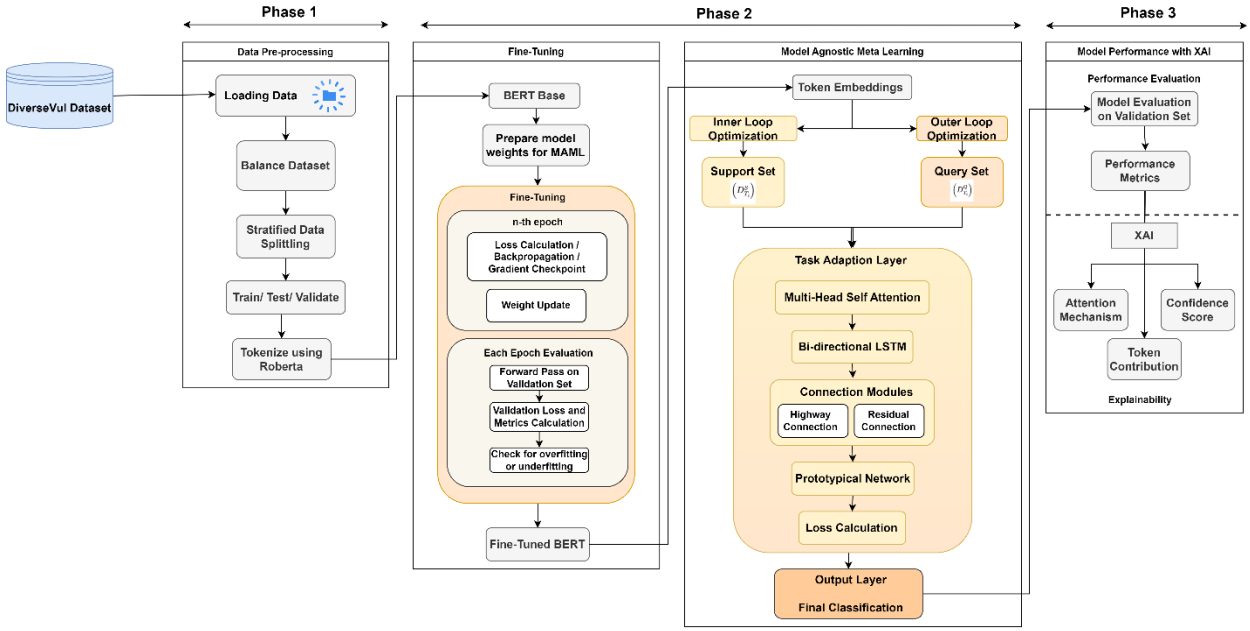

Fig. 3: System Architecture of the Hybrid Framework

Phase 1: Data pre-processing

Data pre-processing is the first phase of the proposed architecture and consists of a number of activities. This step prepares the raw input data of the code repository for effective use in the successive stages of fine-tuning and meta-learning. This is to guarantee that the data is consistent, balanced, and compatible with the CodeBERT model, without which reliable performance would not be achievable. It begins by loading the dataset and extracting the required information. Balancing is carried out after loading the data to take care of class imbalance. This step ensures that all categories or labels in the dataset are represented well to avoid any biases. A balanced dataset facilitates the model's generalizing ability across diverse tasks and avoids the possibility of being biased toward dominant classes during training. Following this is the stratified splitting of data into training, validation, and test sets without disrupting the class distribution of any of these subsets, allowing the subsets to represent the overall dataset. Lastly, tokenization was performed as the final pre-processing activity using a RoBERTa-based tokenizer. Tokenization involves activities like truncation, padding, and the creation of attention masks to ensure that the input data aligns with the pre-trained architecture of CodeBERT. This structured pipeline lays a solid foundation for the fine-tuning and meta-learning phases, enabling CodeBERT to learn and adapt effectively to a variety of programming tasks.

Phase 2: Fine-Tuning and MAML Integration in CodeBERT

The next phase focuses on the adaptation of the novel hybrid approach of CodeBERT that integrates fine-tuning and MAML to enable it to effectively adapt to the task at hand. The preprocessed data from Phase 1 serves as the base for this phase, giving clean, balanced, and tokenized inputs required for fine-tuning and meta-learning. This integration further enhances the efficiency of the pre-trained model with MAML by quickly adapting to new tasks with minimal data, further making it an efficient and scalable solution for few-shot learning in programming contexts. This phase will be structured into two subphases to ensure a systematic and effective learning process.

Phase 2.1: Fine-Tuning CodeBERT

The goal of this sub-phase is to use pre-trained knowledge about source code of the CodeBERT and allow for task-specific adaptation using fine-tuning. First, it takes the pre-processed, tokenized data that results from Phase 1 and feeds it into the input of CodeBERT to produce the rich embeddings, capturing syntactic and semantic relationships in the source code. This forms a basis for further task-specific optimization, paving the way for highly adaptive and efficient learning. CodeBERT acts as the base encoder and utilizes its pre-trained functionalities to contextualize source code embeddings. Moreover, MAML is added to it by incorporating the adaptation mechanism that enables a task-specific way, which leads to the hybrid model combining the strength of pre-trained language models and the flexibility of meta-learning. In this hybrid approach, the inner loop of MAML fine-tunes the parameters of CodeBERT for a particular task with the help of a small support set as discussed in Section 3.1. Pre-processed and tokenized input is fed into CodeBERT to get the embeddings, which are further refined by multi-head self-attention in order to capture contextual relationships among tokens. These embeddings are fed to a bi-directional LSTM to model sequence-level dependencies, further enhanced by highway and residual connections to ensure robust gradient flow and effective representation learning. The task-specific adaptation is completed with a prototypical network that computes class prototypes in the embedding space and classifies new samples based on their distance to these prototypes. This fine-tuning on the task at hand enables the model to learn the nuances of the task quickly.

Phase 2.2: Model Agnostic Meta Learning

This sub-phase focuses on leveraging the outer loop of MAML, where the fine-tuned parameters from the inner loop are passed to perform few-shot learning, evaluating, and refining the CodeBERT model. Building upon the task-specific fine-tuning performed in the inner loop, the outer loop introduces a meta-learning mechanism that assesses the effectiveness of these adaptations and enhances the model's ability to learn. In the outer loop of MAML, the adapted model from the inner loop is evaluated on a query set to compute a meta-loss. This meta-loss assesses how well the inner loop adaptation performed and updates the global meta-parameters of CodeBERT. These updates optimize the initialization of CodeBERT's parameters, making the model more adaptable to future tasks with minimal fine-tuning. This dual-loop mechanism of task-specific adaptation and global optimization ensures the hybrid model generalizes effectively across diverse programming environments. The integration of MAML into CodeBERT also extends the model's explainability. The task-specific adaptations performed during the inner loop highlight which features (tokens, embeddings, or sequences) are critical for the task, providing interpretability into the model's decision-making process. This hybrid approach is novel

because of its ability to harness the power of pre-trained models like CodeBERT with the flexibility of meta-learning techniques like MAML. This leads to the advancement of few-shot learning performance and decreases computational burden, yielding a set of scalable, interpretable frameworks for programming tasks.

Phase 3: Model Performance and Explainability with XAI

This final phase focuses on evaluating the hybrid model developed in Phase 2 in terms of both performance and explainability. By leveraging both traditional performance metrics and cutting-edge explainable AI (XAI) techniques, this phase provides a comprehensive analysis of the model's effectiveness and trustworthiness. Performance metrics highlight how far the model can achieve objectives for which it has been fitted, and they give a normalized manner of examining the effectiveness of this model towards spotting areas that could be refined. Moreover, robust XAI techniques for ensuring model transparency and trustworthiness are also implemented. XAI helps understand the model's decision process, uncovers hidden bias, and validates that the model predictions align with ethical considerations. This phase provides a comprehensive analysis of the model's effectiveness by leveraging both performance metrics and XAI techniques, enhancing its interpretability and fostering trust among users and stakeholders.

3.1 Model performance evaluation

It gives insight into the performance of the developed hybrid model by determining the performance based on different performance metrics: meta loss, support accuracy, query accuracy, and task diversity. The meta loss signifies the overall loss accrued through the entire meta-learning process, hence defining the model's optimization efficiency over different tasks [31]. Support accuracy reflects the model's ability to learn effectively from the few examples provided in the support set for each task [32]. Query accuracy reflects the performance of the model on unseen data for the same task after learning from the support set [33]. Finally, task diversity quantifies the variability and complexity of the tasks that the model is trained and evaluated on, providing insight into its generalizability to diverse scenarios [34]. These metrics together guarantee that one will have a complete understanding of the strengths and weaknesses of the model. The evaluation is necessary to ensure that the hybrid model meets the desired performance thresholds, robustness, reliability, and adaptability in practical applications. By systematically analyzing these metrics, the evaluation underlines areas of strength and potential improvement, contributing to the overall refinement and validation of the model.

3.2 Explainability of model

Finally, explainability in the AI model focuses on explaining the factors that influence the decision-making of the model. It ensures that the model predictions and decision-making processes are transparent, interpretable, and aligned with user expectations. This sub-phase tries to fill the gap between model performance and interpretability using advanced explainability techniques, so the entire system becomes more reliable and trustworthy for any end user. The key techniques implemented in this sub-phase, as mentioned in Section 3.2, include:

•*Attention mechanism* helps a model to actively focus on selective parts of the input sequence when generating an output. By assigning every input element with a different importance level, the model learns complex dependencies and relationships between them [35]. Attention mechanisms calculate attention weights using components such as queries, keys, and values, which provide an emphasis on the input elements that are crucial in determining the output. The weights are displayed using heatmaps, an intuitive representation of the model's areas of focus. Heatmaps make it easier to understand the model's decision-making process, putting emphasis on which tokens or features were most impactful for a specific task or prediction [36].

•*Layer-wise analysis* is instrumental in systematically studying how each layer of the model contributes to processing and transforming the input into meaningful output. In the transformer-based models, such as CodeBERT, information gets progressively refined by a number of layers. These are fitted with attention mechanisms inside that make the model concentrates on token interactions. Individually analyzing each layer makes it possible to figure out what each layer specifically contributes to the final prediction.

•*Feature contribution* is a way to determine the importance or influence of certain input features, such as words or tokens, on the model's predictions. This technique assigns numerical scores to features that summarize their contribution to the output. SHAP and LIME are the two most common feature attribution methods. SHAP measures the contribution of each feature by considering all possible subsets of input features, yielding a more robust and global explanation of feature importance [29]. LIME works by locally perturbing the input and fitting a simple, interpretable model to approximate the predictions, providing local explanations around specific data instances [29]. These techniques help identify the most influential features driving the model's predictions, ensuring

transparency and enabling users to validate the reliability of the system.

•*Model confidence scores* quantify the certainty or reliability of predictions by assigning probabilities to each possible outcome. These scores represent the model's confidence in the correctness of its predictions, which is derived from the SoftMax function applied to the model's raw outputs. High confidence scores reflect high certainty in the prediction, whereas lower scores indicate ambiguity or uncertainty [37]. Confidence scores improve the explainability of the model as they provide insight into the reliability of each decision and allow users to determine whether a prediction can be trusted. Further integration of the confidence scores with other techniques, such as attention visualization and feature attribution, will make the model even more interpretable and user-friendly.

4. Experiment

This section outlines the experiment and results obtained from the experiment by implementing the proposed system architecture. The focus is on demonstrating the applicability of the proposed hybrid approach and robust XAI practice. The objective of this experiment is to:

- To assess the model's performance for the proposed hybrid approach that integrates the CodeBERT model with fine-tuning and MAML.
- To demonstrate and explain the model's decision-making process using robust XAI practice from a holistic aspect.

4.1. Dataset Description

For this experiment, we have used the dataset, DiverseVul [38], which is a diverse source code dataset meant for vulnerability detection with the support of machine learning models. This collection contains raw source code snippets (source code), labeled (annotated) as 'vulnerable' or otherwise, and in representations necessary for machine learning, using representations such as Abstract Syntax Trees (ASTs) and Control Flow Graphs (CFGs). Sourced from real-world and simulated scenarios, DiverseVul allows the training and benchmarking of models aimed at detecting vulnerabilities and studying explainability, thus helping researchers evaluate performance and interpretability. Some of the key features of Diversevul are:

- 18,945 vulnerable functions and 330,492 non-vulnerable functions extracted from 7,514 commits.
- Coverage of 150 Common Weakness Enumerations (CWEs), making it the most diverse dataset compared to prior collections.
- Vulnerabilities focuses on websites and extracts vulnerability-fixing commits from 797 projects, including 295 new projects not covered by previous datasets.
- The dataset puts much emphasis on real-world applicability and solves several challenges such as label noise and generalization to unseen projects.

4.2. System Implementation

Phase 1: Data pre-processing: We implemented a structured data pre-processing pipeline to prepare raw input data for fine-tuning and meta-learning with the CodeBERT model. Starting with loading the dataset, we extracted and organized relevant information. Biases in our models were further reduced by balancing the dataset for the right representation of all classes. Further preprocessing was performed to split data into training, validation, and test sets with stratified data division, hence maintaining the distribution across subsets. Finally, tokenization was done with the RoBERTa-based tokenizer. Preprocessing also includes truncation, padding, and attention mask creation to fit with CodeBERT's architecture. In this pipeline, we achieved consistent and strong preparation for downstream tasks.

Phase 2: Fine-Tuning and MAML Integration in CodeBERT: In the second phase of our experiment, we implemented a hybrid approach by combining CodeBERT with fine-tuning and MAML, using an optimized configuration to enable efficient task-specific adaptations. The pre-processed data from the previous phase was used for fine-tuning CodeBERT to generate rich embeddings, refined with multi-head self-attention and a bi-directional LSTM for enhanced contextual understanding and sequence-level dependencies. CodeBERT is a pre-trained model with 6 transformer layers, including multi-head attention and feed-forward mechanisms. For configuration parameters, an inner learning rate was set to 0.01, while the outer one was 0.0001 and there were five task adaptation inner steps, while the number of outer steps was reduced to 100 to avoid overfitting. Enhanced settings, such as a task embedding dimension of 128, a dropout rate of 0.1, and prototype-based adaptation with three samples, ensured robust and adaptable task representations.

The outer loop of MAML optimized global meta-parameters, leveraging regularization weights like task diversity of 0.1 and prototype margin of 0.5, whereas early stopping further enhanced the stability of training. This architecture, combined with the transformer design in a layered manner, allowed the hybrid model to achieve impressive performance regarding few-shot learning, scalability, and interpretability on tasks related to programming.

Phase 3: Model performance and Explainability with XAI: In this last phase of the framework, the system is intended to assess hybrid model performance along with robust XAI practice. The data was split into 80% for training and 20% for testing in order to ensure an unbiased assessment on unseen data. Results of the experiments conducted during the two sub-phases are discussed below.

4.2.1 Model Performance Evaluation

In this sub-phase, we evaluate the performance of the fine-tuned CodeBERT model, which is integrated with MAML. The evaluation focuses on key metrics such as meta loss, support accuracy, query accuracy, and task diversity. Each of these metrics gives insight into the model's learning efficiency, adaptation to a specific task, and generalization capability. To show the model's performance in the training phase, Table 1 presents these metrics across five epochs, providing a comprehensive view of its progression and improvement. The key insights from this table are:

•Meta loss decreases steadily, showing improved task generalization. This consistent drop in meta loss suggests that the model can better optimize its parameters for various tasks during meta-learning.

•Support accuracy grows from 85.2% to 92.5%, indicating better task-specific learning. Improvement reflects the model's enhanced capability to adapt its fine-tuned weights to individual tasks within the support set.

•Query accuracy improves from 88.1% to 95.7%, showing better generalization to unseen data. This significant gain showcases the strength of the hybrid model in transferring learned knowledge to unseen query samples during meta-testing.

•Task diversity increases from 0.72 to 0.89, highlighting exposure to the varied tasks. This increase in task diversity shows that the model has been exposed to a wider range of tasks, ensuring robust learning across different programming contexts.

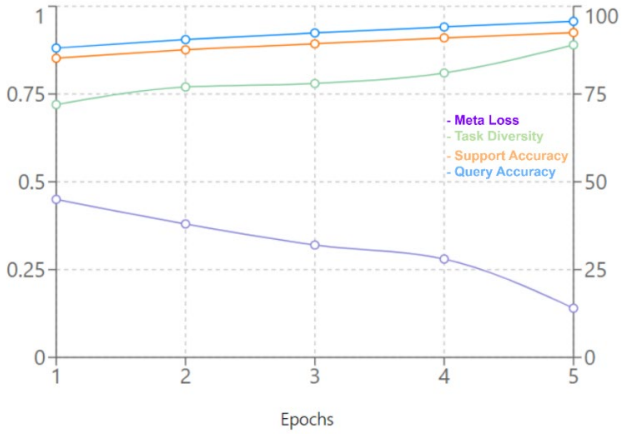

Fig. 4: Model Performance Visualization

Table 1: Performance Evaluation of the Model

Phase	Epoch	Meta Loss	Support Accuracy (%)	Query Accuracy (%)	Task Diversity
Training	1	0.45	85.2	88.1	0.72
	2	0.38	87.6	90.5	0.77
	3	0.32	89.3	92.4	0.78
	4	0.28	91.0	94.1	0.81
	5	0.14	92.5	95.7	0.89

Explainability of Model: This sub-phase explains the decision-making of the model to ensure that its predictions are interpretable and transparent. In other words, by providing insights into how the model processes information and arrives at conclusions, we enhance its interpretability. These explanations also help identify potential weaknesses in the model. We consider various techniques from the four dimensions of XAI as discussed in Section 3.2. The results from each of these dimensions are discussed below.

•**Attention Mechanism:** The attention heat map shows how fine-tuned CodeBERT combined with MAML pays attention to tokens in an input sequence. Key tokens such as 'tflitenode,' 'tflitecontext,' and 'context' have higher attention weights, which means these tokens are especially important for the model to interpret the input, as shown in Figure 4. Such tokens probably represent the most important parts of the function analyzed and give a great contribution to the model's understanding of the relationships inside the sequence. The classification token '[CLS]' and the structural symbols like '(' and ')' get moderate attention to help the model understand the overall structure. The spreading of attention over several tokens is an indication that the transformer model may look at contextual

relationships rather than only focus on individual words. It gives a clear overview of how the model puts the main priority on tokens function-related and structural aspects to obtain meaning from the input.

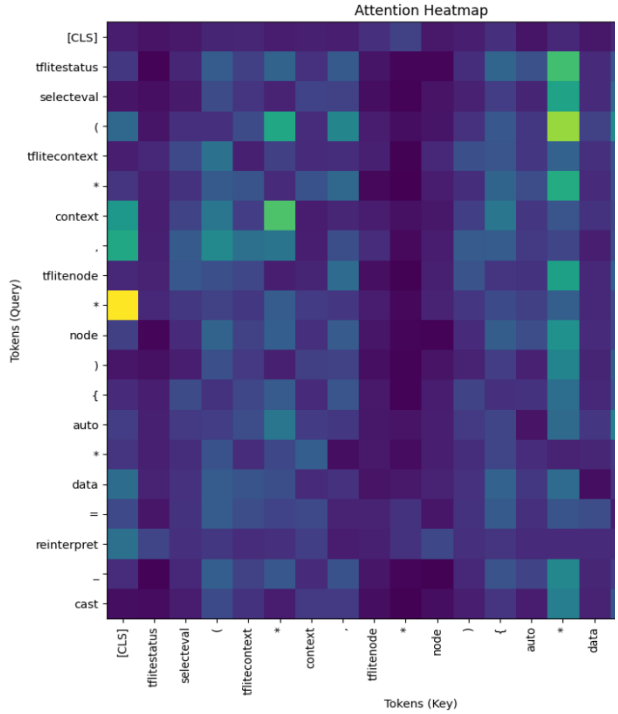

Fig. 4: Attention Weights Heatmap for Input Token

•**Layer-Wise Analysis:** This dimension provides detailed information about how the model processes and represents input sequences at distinct levels of abstraction. Similar to attention mechanisms, layer-wise analysis can also be implemented using heatmaps. For layer-wise, the visualization offers a detailed view of the activations at each layer, thus allowing for insight into how the model builds up from low-level token-specific features to higher-level contextual representations.

Figure 5 shows a seven-layer abstraction of the CodeBERT model. Each row here represents a layer, while columns represent tokens from the input sequence. The color intensity is determined by the magnitude of activations. Lighter areas (yellow) correspond to large activation values, while darker colors (blue or purple) correspond to smaller or negative activations. In Layer 0, the activations are far stronger for some tokens, like 'eval' and 'context.' This indicates that the early layer is actively capturing lower-level token-specific features, such as syntax or word structure. In this layer, the activations are dominated by subword tokens like 't,' '##fl,' '##ite,' which the model processes collectively to arrive at meaningful word-level representations. As we go deeper into the intermediate layers, Layers 1 to 5, the activations become more uniform and subdued. This reflects the model's development from low-level token representations towards more abstract contextual relations. These layers are not very token-focused but aim at building semantic understanding of the overall sequence. In the last layer, Layer 6, the activations are quite similar and consolidated, especially regarding special tokens like '[CLS]' and '[SEP]'. First, the '[CLS]' token, representing the overall sequence embedding for classification, has a strong activation across layers, hence critical. Similarly, tokens like context, select, and node retain relatively higher activation to represent their importance in the semantic understanding of the input.

Feature Contribution: The SHAP and LIME techniques were used to quantify and visualize feature importance of the hybrid CodeBERT model, ensuring clarity in model behavior. Figure 6 shows the SHAP force plot, a visualization that gives an in-depth explanation of how individual tokens (inputs) contribute toward a model's prediction for two classes: Output 0 and Output 1. The base value, approximately -0.0273, represents the average model output before considering token contributions.

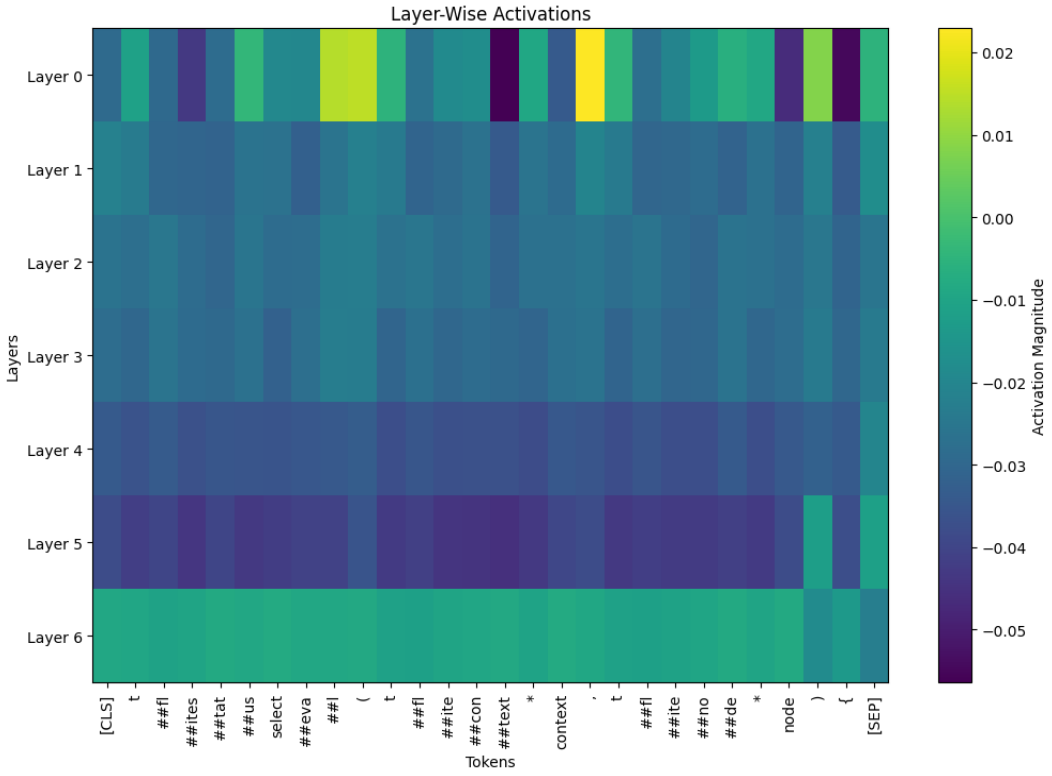

Fig. 5: Layer-Wise Representation Analysis of Input Tokens

For Output 0, the predicted value f(inputs) is approximately 0.0114, which reflects the cumulative effect of all token contributions relative to the base value. Red-colored tokens, including those that contribute positively to Output 0, drive the prediction upwards from the base value. These are likely features that the model has learned for predicting this class. On the other hand, blue tokens contribute negatively, pulling the prediction downward, although their presence in this example is minimal. Neutral tokens are much closer to the base value and have little to no contribution. The sum of all these contributions is a shift from the base value to the final value of the model's output and provides an interpretable view of the model's decision-making process. This explanation helps developers understand why the model predicted a certain class by highlighting which features in the input text were influential and provides insights for debugging or improving the model.

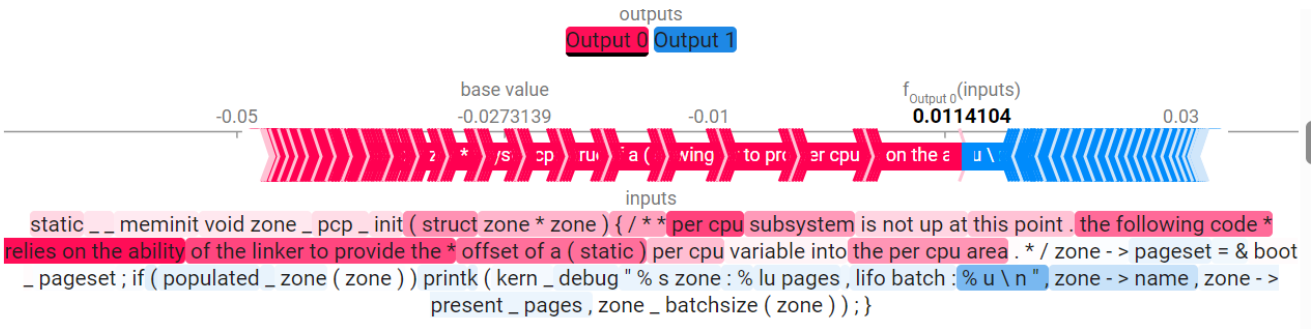

Fig. 6: SHAP Force Plot for Model Prediction

Figure 7 represents the LIME visualization for the fine-tuned CodeBERT model that is integrated with MAML. The model predicted a 52% probability for the positive class and a 48% probability for the negative class. The close probabilities suggest that the model found the sample somewhat ambiguous, as it is near the decision boundary. The visualization highlights specific words in the input text that influenced the prediction. Orange-colored tokens contributed to the positive class prediction, including such keywords as 'static,' 'int,' 'size_t,' and 'struct'. Their presence in the code seems to be related to the patterns the model learned to correlate with the positive label. On the other hand, tokens in blue contributed negatively, pushing the model toward the negative class. Examples include 'header_len,' 'sockaddr_x25,' and 'skb'. These terms may be associated with patterns the model found indicative of the negative class during training. By analyzing which tokens had the most influence, you can gain deeper insights into how the model interprets input text and refine the training or preprocessing process as needed.

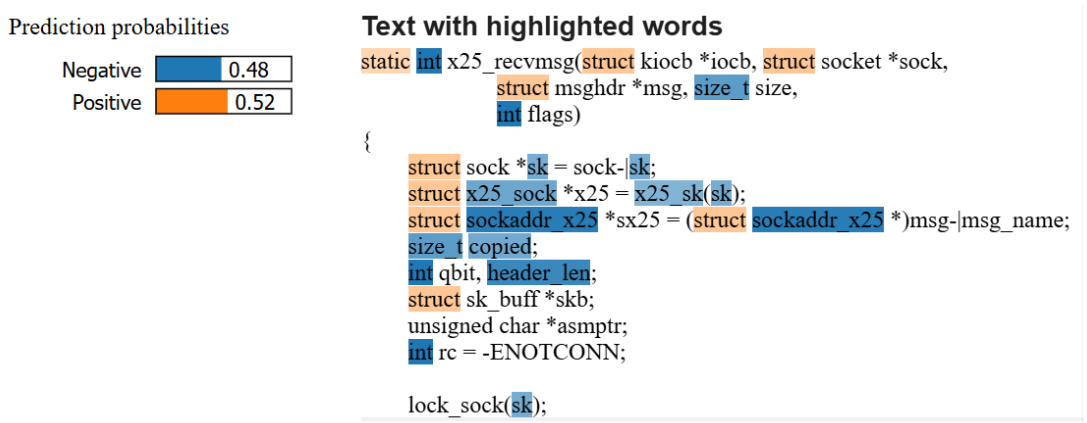

Fig. 7: LIME Analysis

•**Model Confidence Score:** Model confidence scope further enhances the explainability of the model outcome by providing insights into the certainty of predictions, allowing users to assess the reliability of the model's decisions. Confidence scores help identify areas where the model may be uncertain, guiding efforts to improve robustness and ensuring informed trust in its outputs. In this regard, in table 2, the model predicts Class 0 with a confidence score of 0.91, signifying high certainty that the input aligns with the features associated with this class. Conversely, the score for Class 1 is 0.09, suggesting minimal likelihood of the input belonging to this category. This significant difference between the two scores shows the high ability of the model in distinguishing between the classes because it clearly favors Class 0. These are the kind of results that highlight the effectiveness of making confident and well-calibrated predictions from a model.

5. Discussion

This paper aims to contribute to an effective source code-based vulnerability detection and comprehensively explain the model decision-making. Our approach balances computational efficiency and model interpretability, providing a practical solution to improve cybersecurity.

5.1. Adoption of Hybrid LLM Model in Vulnerability Detection

The proposed hybrid framework, based on a CodeBERT-powered LLM, integrates fine-tuning with Model-Agnostic Meta-Learning, providing an adaptable and efficient solution for software vulnerability detection. One of the key benefits of MAML is its potential for enabling quick adaptation, with very few data points, to diverse and evolving vulnerability

patterns. Unlike in the case of traditional AI methods, which have a high dependency on extensive datasets for fine-tuning using huge computational resources, MAML trains models with only a small number of instances by leveraging few-shot learning capabilities. This makes the framework especially useful in dynamic environments, where new vulnerabilities emerge often, and labeled data is scarce [39]. MAML efficiently optimizes the initialization of model parameters to allow for faster and robust adaptation to complex tasks without further training of the model, reducing computational overhead and thus enabling quick deployment in real-world scenarios.

Another strength of this framework involves the higher detection of vulnerability with improved accuracy and robustness regarding complex vulnerabilities. It enables architectural innovations in state-of-the-art techniques of deep learning, including multi-head attention, bi-directional LSTMs, and residual connections, allowing for complex pattern capture with long-term source code dependencies. With such architecture, it has been shown that the model identifies inter-syntax and semantic knowledge besides picking out vulnerabilities emerging between various dependent components of code [40]. These are further validated by the experimental results, which turn out to be significantly improved on major metrics. The meta-loss decreased from 0.45 in the first epoch to 0.14 in the fifth epoch, showing that the framework has the ability to minimize the error in training. Likewise, query accuracy improved from 88.1% to 95.7%, and support accuracy increased from 85.2% to 92.5%, showing the model is able to adapt efficiently and achieve high performance with limited data. These results highlight the framework's practical applicability and scalability. By leveraging the pre-trained capabilities of CodeBERT and mixing them with MAML's generalization power, the framework has achieved superior performance while reducing reliance on extensive labelled datasets. The consistent improvement in accuracy and reduction in meta-loss shows that the framework can be learned effectively even under challenging conditions, such as imbalanced or incomplete datasets. These, combined with generalization to new vulnerabilities, position the framework as highly efficient and reliable in solving modern challenges of software vulnerability detection.

5.2. Integration of XAI to the Hybrid Framework

The integration of Explainable AI techniques into the proposed CodeBERT-based hybrid framework aims to enhance trust, interpretability, and usability for the model decision-making process. It also leverages advanced XAI techniques such as SHAP, LIME, and heatmaps to offer a comprehensive explanation of the model predictions for insights into the decision-making process. These techniques help in visualizing attention mechanisms, layer-wise analysis, and feature attribution to understand which parts of the input have contributed most from a holistic point of view to the model's predictions. For instance, heatmaps provide a per-layer, in-depth look at token activations, thereby helping developers to identify crucial patterns the model used to detect vulnerabilities. The use of confidence scores adds another dimension to interpretability, enabling developers and security analysts to have an idea of the degree of certainty of the model's predictions. The result from our experiment shows an almost 91% confidence score in predicting a non-vulnerable point in the code. A confidence score so high testifies to the strength of this framework and instills higher levels of trust in both the developer and security analysts. This also serves to underscore those areas that would still require further refinement or increased data for edge cases and complex vulnerabilities. With such identification of areas of high and low confidence, this framework enables the detection of biases or anomalies in predictions, making sure of fairness in a more reliable manner.

Furthermore, XAI practices not only build trust among stakeholders but also align the framework with organizational and regulatory standards that demand transparency in AI systems. The explanations provided by SHAP and LIME, along with visual insights from heatmaps, help bridge the gap between complex model architectures and user understanding. This transparency is especially critical in cybersecurity, where stakeholders need to trust the tools they rely on for identifying vulnerabilities. Overall, the seamless integration of XAI techniques into the framework ensures that it is not only effective in detecting vulnerabilities but also interpretable and reliable, fostering wider adoption and trust in the system. The proposed research has immense practical value in the field of software vulnerability detection, as it tries to solve some of the challenges faced by developers, security analysts, and organizations.

5.3. Practical Implication

By incorporating CodeBERT with MAML, this framework provides an effective and efficient solution to identify vulnerabilities within dynamic and

complex software systems. The capability to learn new vulnerability patterns with less labelled data makes sure organizations can quickly respond to emerging threats by reducing the time and resources required for model retraining. This is especially valuable in real-world scenarios where new vulnerabilities frequently arise, and annotated datasets are at a premium. It provides an architecture with innovations such as multi-head attention and bi-directional LSTMs that enable it to detect vulnerabilities resulting from complex interactions within codebases, such as flawed uses of functions or variables across multiple files. This level of precision is critical for assuring software security in such sensitive domains as finance, health care, and defense. Moreover, the integration of Explainable AI techniques ensures that the framework performs well and is interpretable and trustworthy. The possibility of explaining predictions with methods such as SHAP, LIME, and heatmaps enhances stakeholder confidence and facilitates compliance with organizational and regulatory requirements for transparency in AI systems. These explanations will help developers and analysts to understand, as well as refine the model's predictions with the aim of enhancing security processes within an organization. Another very practical implication is the efficiency of the framework. Because of the limited computational overhead and few-shot learning dependence, it should be quite feasible for implementation in resource-constrained environments, such as startups or smaller organizations, without performance degradation. Furthermore, its capability to work across diverse programming languages and styles ensures wide applicability, making it a versatile tool for securing software-intensive systems across various domains.

6. Conclusion

Exploitation of source code vulnerabilities may pose potential risk to the organizations that heavily depend on software systems to support their operations. While AI-based techniques for vulnerability detection are getting wider adoption, the fact remains that AI models require high-quality data with context-specific information to return reliable and accurate results. AI models often struggle to capture the intricate relationships and patterns in source code. Moreover, the lack of explainability of such models further makes it difficult for developers and security analysts to understand and trust the predictions made by the model. This paper proposes a novel hybrid framework based on a CodeBERT-powered LLM for vulnerability detection in source code, which integrates fine-tuning with Model-Agnostic Meta-Learning to handle some key challenges in this domain. The framework effectively fuses the pre-trained capabilities of CodeBERT with MAML's few-shot learning mechanism for rapid adaptation to new vulnerability patterns using a minimal amount of labelled data. This hybrid approach reduces computational overhead, thus making it especially suitable for resource-constrained environments while keeping the accuracy and robustness high, even for noisy or incomplete datasets. This framework also integrates robust XAI practices to ensure interpretability and reliability in the detection process. It is based on four key dimensions, including attention mechanisms, layer-wise analysis, feature contribution, and model confidence scores. These XAI techniques present comprehensive explanations of model predictions to enhance transparency, interpretability, and trustworthiness. By making the decision-making process understandable, the framework aligns with organizational and regulatory standards while empowering developers and stakeholders to confidently act upon the model's outputs. Additionally, this framework was implemented and validated through experiments to prove its effectiveness. The results demonstrate its ability to perform the identification and classification tasks of vulnerabilities in fast-evolving threat landscapes. The experimental results show a continuously decreasing meta-loss while increasing the accuracy, which demonstrates that the proposed framework is more viable and applicable in real-world scenarios. However, this work only focuses on a specific programming language; therefore, future directions will focus on covering the diverse programming languages within the complex software ecosystem. Moreover, the future direction will also focus on the viability of zero-shot learning and its adoption into the hybrid framework. The integration of automated feedback loops for continuous learning from the ever-occurring real-world vulnerabilities will also be one key area of exploration.

Acknowledgments. This work was partially supported by the European Union's Horizon Europe Project, CyberSecDome—An innovative Virtual Reality-based intrusion detection, incident investigation, and response approach for enhancing the resilience, security, privacy, and accountability of complex and heterogeneous digital systems and infrastructures, funded under grant agreement No. 101120779 and the UK Research and Innovation (UKRI) with CHIST-ERA 2023, AI4MultiGIS- AI integrated framework for intelligent geospatial handling and robust operation in

MultiGIS applications , funded under grant agreement no - EP/Z003490/1

References

[1] NVD - Statistics. (n.d.). https://nvd.nist.gov/vuln/search/statistics?form_type=Basic&isCpeNameSearch=false&search_type=all&utm_source=chatgpt.com [Accessed on 10/12/24]

[2] Huang, Q., & Zhao, T. (2024). Data collection and labeling techniques for machine learning. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2407.12793

[3] Mahyari, A. A. (2024). Harnessing the power of LLMs in source code vulnerability detection. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2408.03489

[4] Tan, Z., Chen, T., Zhang, Z., & Liu, H. (2024). Sparsity-Guided Holistic Explanation for LLMs with Interpretable Inference-Time Intervention. Proceedings of the AAAI Conference on Artificial Intelligence, 38(19), 21619–21627. https://doi.org/10.1609/aaai.v38i19.30160

[5] Li, G., Wang, P., Liu, J., Guo, Y., Ji, K., Shang, Z., & Xu, Z. (2024). Meta In-Context learning makes large language models better zero and Few-Shot relation extractors. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2404.17807

[6] Zheng, H., Shen, L., Tang, A., Luo, Y., Hu, H., Du, B., & Tao, D. (2023). Learn from Model Beyond Fine-Tuning: a survey. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2310.08184

[7] Sinha, S., Yue, Y., Soto, V., Kulkarni, M., Lu, J., & Zhang, A. (2024a). MAML-en-LLM: Model Agnostic Meta-Training of LLMs for Improved In-Context Learning. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2405.11446

[8] Hu, Y., & Zhang, G. (2024). MAML-BERT in addressing low-resource text classification tasks. Proceedings Volume 13229, Seventh International Conference on Advanced Electronic Materials, Computers, and Software Engineering (AEMCSE 2024), 112. https://doi.org/10.1117/12.3038663

[9] Islam, S., Abba, A., Ismail, U., Mouratidis, H., & Papastergiou, S. (2022). Vulnerability prediction for secure healthcare supply chain service delivery. Integrated Computer-Aided Engineering, 29(4), 389–409. https://doi.org/10.3233/ica-220689

[10] Silvestri, S., Islam, S., Amelin, D., Weiler, G., Papastergiou, S., & Ciampi, M. (2023). Cyber threat assessment and management for securing healthcare ecosystems using natural language processing. International Journal of Information Security, 23(1), 31–50. https://doi.org/10.1007/s10207-023-00769-w

[11] Sahoo, Sai Aryan, Meta-Learning for Large Language Models: Teaching LLMs to Learn New Tasks with Minimal Data (September 27, 2024). Available at SSRN: https://ssrn.com/abstract=4977093 or http://dx.doi.org/10.2139/ssrn.4977093

[12] Wu, M., Vu, T., Qu, L., & Haffari, G. (2024). The Best of Both Worlds: Bridging Quality and Diversity in Data Selection with Bipartite Graph. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2410.12458

[13] Wu, X., Zhao, H., Zhu, Y., Shi, Y., Yang, F., Liu, T., Zhai, X., Yao, W., Li, J., Du, M., & Liu, N. (2024). Usable XAI: 10 Strategies Towards Exploiting Explainability in the LLM Era. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2403.08946

[14] Bhattacharjee, A., Moraffah, R., Garland, J., & Liu, H. (2023b). LLMs as counterfactual explanation modules: Can ChatGPT explain black-box text classifiers? arXiv (Cornell University). https://doi.org/10.48550/arxiv.2309.13340

[15]Mavrepis, P., Makridis, G., Fatouros, G., Koukos, V., Separdani, M. M., & Kyriazis, D. (2024). XAI for All: Can Large Language Models Simplify Explainable AI? arXiv (Cornell University). https://doi.org/10.48550/arxiv.2401.13110

[16] Nicholls, J., Kuppa, A., & Le-Khac, N. (2023). Enhancing Illicit Activity Detection using XAI: A Multimodal Graph-LLM Framework. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2310.13787

[17] Khediri, A., Slimi, H., Yahiaoui, A., Derdour, M., Bendjenna, H., & Ghenai, C. E. (2024). Enhancing Machine Learning Model Interpretability in Intrusion Detection Systems through SHAP Explanations and LLM-Generated Descriptions. 2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS), EL OUED, Algeria, 1–6. https://doi.org/10.1109/pais62114.2024.10541168

[18] Haurogné, J., Basheer, N., & Islam, S. (2024c). Vulnerability Detection using BERT based LLM Model with Transparency Obligation Practice Towards Trustworthy AI. Machine Learning With Applications, 100598. https://doi.org/10.1016/j.mlwa.2024.100598

[19] Zhou, S., Alon, U., Agarwal, S., & Neubig, G. (2023). CodeBERTScore: Evaluating Code Generation with Pretrained Models of Code. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2302.05527

[20] Zhou, S., Alon, U., Agarwal, S., & Neubig, G. (2023). CodeBERTScore: Evaluating Code Generation with Pretrained Models of Code. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2302.05527
[21] Jin, H., & Li, Q. (2023). Fine-Tuning Pre-Trained CodeBERT for code search in smart contract. Wuhan University Journal of Natural Sciences, 28(3), 237–245. https://doi.org/10.1051/wujns/2023283237
[22] Chai, Y., Zhang, H., Shen, B., & Gu, X. (2022). Cross-domain deep code search with meta learning. Proceedings of the 44th International Conference on Software Engineering. https://doi.org/10.1145/3510003.3510125
[23] Cambria, E., Malandri, L., Mercorio, F., Nobani, N., & Seveso, A. (2024). XAI meets LLMs: A Survey of the Relation between Explainable AI and Large Language Models. ArXiv, abs/2407.15248.
[24] Basheer, N., Islam, S., Alwaheidi, M. K. S., & Papastergiou, S. (2024b). Adoption of Deep-Learning Models for Managing Threat in API Calls with Transparency Obligation Practice for Overall Resilience. Sensors, 24(15), 4859. https://doi.org/10.3390/s24154859
[25] Wikipedia contributors. (2024, November 24). Attention (machine learning). Wikipedia. https://en.wikipedia.org/wiki/Attention_(machine_learning)
[26] Vig, J. (2019). Visualizing attention in Transformer-Based Language Representation models. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1904.02679
[27] A. Pasad, J. -C. Chou and K. Livescu, "Layer-Wise Analysis of a Self-Supervised Speech Representation Model," 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Cartagena, Colombia, 2021, pp. 914-921, doi: 10.1109/ASRU51503.2021.9688093.
[28] Contreras, J., Winterfeld, A., Popp, J., & Bocklitz, T. (2024). Spectral Zones-Based SHAP/LIME: Enhancing interpretability in spectral deep learning models through grouped feature analysis. Analytical Chemistry. https://doi.org/10.1021/acs.analchem.4c02329
[29] Basheer, N., Pranggono, B., Islam, S., Papastergiou, S., & Mouratidis, H. (2024). Enhancing Malware Detection Through Machine Learning Using XAI with SHAP Framework. In IFIP advances in information and communication technology (pp. 316–329). https://doi.org/10.1007/978-3-031-63211-2_24
[30] Grandperrin, J. (2024, November 26). How to use confidence scores in machine learning models. Mindee. https://www.mindee.com/blog/how-use-confidence-scores-ml-models
[31] Chen, Y., Zhang, H., Yang, X., Zhang, W., & Qu, D. (2024). Improving cross-lingual low-resource speech recognition by Task-based Meta PolyLoss. Computer Speech & Language, 87, 101648. https://doi.org/10.1016/j.csl.2024.101648
[32] Sendera, M., Przewięźlikowski, M., Miksa, J., Rajski, M., Karanowski, K., Zięba, M., Tabor, J., & Spurek, P. (2023). The general framework for few-shot learning by kernel HyperNetworks. Machine Vision and Applications, 34(4). https://doi.org/10.1007/s00138-023-01403-4
[33] Finn, C., Abbeel, P., & Levine, S. (2017). Model-Agnostic Meta-Learning for fast adaptation of deep networks. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1703.03400
[34] Miranda, B., Yu, P., Wang, Y., & Koyejo, S. (2022). The Curse of Low Task Diversity: on the failure of transfer learning to outperform MAML and their empirical equivalence. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2208.01545
[35] Liangfu, Z., Yujie, X., Yongbin, G., & Wenjun, Y. (2023). Multiple dependence representation of attention graph convolutional network relation extraction model. IET Cyber-Physical Systems Theory & Applications. https://doi.org/10.1049/cps2.12080
[36] Tursun, O., Denman, S., Sridharan, S., & Fookes, C. (2023). Towards Self-Explainability of Deep Neural Networks with Heatmap Captioning and Large-Language Models. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2304.02202
[37] Lubrano, M., Bellahsen-Harrar, Y., Fick, R., Badoual, C., & Walter, T. (2023). Simple and efficient confidence score for grading whole slide images. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2303.04604
[38] Chen, Y., Ding, Z., Chen, X., & Wagner, D. (2023). DiverseVul: A new vulnerable source code dataset for deep learning based vulnerability detection. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2304.00409
[39] Lang Yang. 2024. A meta-learning based method for detecting malicious traffic with few-shot. In Proceedings of the 2024 International Conference on Generative Artificial Intelligence and Information Security (GAIIS '24). Association for Computing Machinery, New York, NY, USA, 211–216. https://doi.org/10.1145/3665348.3665385
[40] Nguyen, V., Nepal, S., Wu, T., Yuan, X., & Rudolph, C. (2024). SAFE: Advancing large language models in leveraging semantic and syntactic relationships for software vulnerability detection. arXiv

(Cornell University). https://doi.org/10.48550/arxiv.2409.00882