

Prediction of Vulnerability Characteristics Based on Vulnerability Description and Prompt Learning

Xiangwei Li*, Xiaoning Ren*, Yinxing Xue*, Zhenchang Xing^{†‡}, Jiamou Sun[†]

*University of Science and Technology of China, Hefei, China

lixw20@mail.ustc.edu.cn, hnurxn@mail.ustc.edu.cn, yxxue@ustc.edu.cn

[†]CSIRO's Data61, Canberra, Australia

zhenchang.xing@anu.edu.au, u5871153@anu.edu.au

[‡]Australian National University, Canberra, Australia

Abstract—Identifying which vulnerabilities need to be prioritized is a long-term challenge in IT security, especially as the number of vulnerabilities grows. Faced with a large number of vulnerability reports, there is an urgent need for automated tools or models to assess the potential severity and exploitability of vulnerabilities. This will help security experts screen vulnerabilities that should be focused on. In this study, we aim to predict vulnerability severity and exploitability characteristics using only vulnerability descriptions. Some previous studies are based on traditional deep learning models, and their performance is relatively backward in the current era of pre-trained language models (PLMs). Therefore, we introduce a prompt learning method based on PLMs to predict vulnerability characteristics. The conventional fine-tuning PLMs method is difficult to make full use of the domain knowledge in PLMs and performs poorly with less training data. Unlike the fine-tuning paradigm, prompt learning imitates the pre-training process of PLM by reconstructing the task input and adding prompts, and uses the output of PLM itself as the prediction output. Combined with prompt ensembling and transfer learning, the performance of prompt learning in the above tasks is further improved. Our experiments show that prompt learning can make more effective use of the knowledge in PLMs. Compared with fine-tuning PLMs and other deep learning models, prompt learning based on BERT or RoBERTa achieves better performance in the above tasks. This advantage is more significant in predicting exploitability with few samples, which proves the ability of prompt learning in few-sample scenarios. In addition, prompt learning also shows the transferability between different tasks in the domain.

Index Terms—vulnerability characteristics, vulnerability assessment, prompt learning, pre-trained language model

I. INTRODUCTION

Software vulnerabilities will cause serious threats to the security and reliability of computer systems. With the emergence of new technologies and software products, the number of vulnerabilities reported in Common Vulnerabilities and Exposures (CVE) [1] database has increased explosively. Faced with a large number of vulnerability reports and limited time and human resources, security developers need to prioritize repair activities according to the severity and exploitability characteristics of vulnerabilities. These vulnerability characteristics are usually determined by security experts based on expert knowledge and with the help of auxiliary systems.

Common Vulnerability Scoring System (CVSS) [2] is an important system for determining vulnerability severity based

on expert knowledge. By evaluating the impact, exploitability, temporal and environmental scores of the vulnerability, security experts use CVSS to determine the severity score of the vulnerability. Unlike severity characteristic, which mixes multiple influencing factors of vulnerability, exploitability characteristic focuses more directly on whether the vulnerability will be actually exploited. Recently, Cybersecurity and Infrastructure Security Agency (CISA) [3] has established a catalog [4] to manage known exploited vulnerabilities, so as to reduce the risk of vulnerability exploitation.

In the face of a large number of vulnerability reports, the evaluation of vulnerability characteristics based on expert knowledge seems to lag behind. For example, for newly released vulnerabilities, the duration between the initial publish date and the CVSS score update date can range from one day to more than a hundred days. In addition, the number of exploited vulnerabilities maintained by CISA is small and the update speed is slow. Therefore, tools or models that can automatically assess vulnerability severity, exploitability and other characteristics are urgently needed to help security experts screen the vulnerabilities that should be focused on and determine the priority of vulnerability assessment. As shown in Table I, the standardized text description of the vulnerability contains surface information such as the working mode of the vulnerability. Compared with the complex evaluation metrics of CVSS, the severity and exploitability characteristics of vulnerabilities are rooted in these standardized descriptions. *In this study, we aim to predict the severity and exploitability characteristics of vulnerabilities using only vulnerability descriptions.* Due to the great potential of deep learning in analyzing software text [5], we intend to solve the above tasks by collecting datasets and training models.

In this study, we determine the severity of vulnerabilities according to the CVSS severity score and create a large-scale severity dataset. We determine the exploitability of vulnerabilities according to CISA's known exploited vulnerabilities list, because CISA has more strict screening criteria. However, the number of exploited vulnerabilities maintained in CISA is small, only hundreds, so we can only create a small-scale exploitability dataset. *Therefore, predicting these two vulnerability characteristics can be redefined as two classification tasks, one is severity classification with large-scale*

TABLE I: An example of CVE vulnerability description and characteristics

CVE-ID: CVE-2021-4102
Date Record Created: 20211210
Vulnerability Description: Use after free in V8 in Google Chrome prior to 96.0.4664.110 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page.
Severity: 8.8 HIGH
CISA Exploitability: Exploited

data, the other is exploitability classification with few samples. For the exploitability classification, we are not limited to the current data scale but extend to more extreme scenarios of data scarcity.

In the previous studies, Han, Gong et al. [6], [7] use vulnerability description training TextCNN and BiLSTM classification models to predict the severity of vulnerabilities. However, these models have poor semantic understanding when confronted with words not seen in the training stage. In addition, the performance of these traditional models is slightly backward in the current era of pre-trained language models (PLMs) [8]–[11]. Naturally, a common and reasonable idea is to use PLMs such as BERT [12] and RoBERTa [13] to better extract the semantic information of the text. The model can understand those unseen words using the rich knowledge learned from large-scale corpus in the pre-training stage. Fine-tuning PLMs have achieved state-of-the-art performance on many downstream tasks [10], [14]. However, for the tasks of vulnerability characteristics classification in this study, or broader tasks, the fine-tuning paradigm has some challenges and deficiencies: 1) The difference between fine-tuning and pre-training stages makes it difficult to make full use of knowledge in PLMs. 2) Like traditional models, the fine-tuning paradigm performs poorly in few-sample scenarios and has poor transferability of cross-tasks.

In order to solve the tasks of this study and overcome the shortcomings of fine-tuning paradigm, we introduce the latest PLMs-based prompt learning [15] paradigm in the field of natural language processing(NLP). Prompt learning is based on PLMs, and realizes more sufficient knowledge mining and utilization of PLMs. By imitating the process of pre-training, prompt learning bridges the gap between pre-training and model tuning of downstream tasks. The key to imitation is to reconstruct the input text by adding natural language prompts and special masked tokens. For example, in order to predict the severity of the vulnerability description, a prompt such as “The severity is [Z].” is added before or after the input sequence. [Z] is the output to be predicted, which is masked and replaced by [MASK] token during input. The PLM predicts the output words at the masked token position, as it did in the original pre-training stage. A verbalizer [16] then maps the predicted output to the expected target class, completing the classification task.

Technically, for the severity classification with a large dataset, we practice the prompt learning method by constructing prompt template and verbalizer, and further improve the effect of prompt learning by combining prompt ensembling

[17], [18] strategy. For the exploitability classification with few samples, we use prompt without introducing parameters to take advantage of PLMs. Moreover, we use the cross-tasks transferability of prompt learning to further improve its performance in few-sample scenarios.

In the experimental stage, for the severity classification, we set up multiple baseline methods for comparison, and the experimental results indicate that prompt learning achieves performance beyond the traditional models and fine-tuning paradigm. Combined with the prompt ensembling strategy, the performance will be further improved. For the exploitability classification, we set up multiple few-sample scenarios. The experimental results show that the prompt learning paradigm has a stronger ability to deal with few samples than the fine-tuning paradigm. This performance advantage is more significant in scenarios where data is extremely scarce. For example, in a 32-sample scenario, the performance of prompt learning based on RoBERTa-base and RoBERTa-large is 10.12% and 5.71% higher than that of fine-tuning paradigm, respectively. Combined with the transfer learning [19] within the domain, prompt learning performance will be further improved. In addition, we also experimentally evaluate the relevant settings that affect the performance of prompt learning, including the design of prompt templates and verbalizers.

Overall, the contributions of this paper are as follows:

- To the best of our knowledge, this study is an early attempt to use prompt learning to solve related tasks in software engineering. In addition, we experimentally evaluate the settings that affect prompt learning performance. This will help more researchers apply prompt learning to their fields.
- Based on vulnerability descriptions, we use the prompt learning method to predict the severity and exploitability characteristics, and achieve better performance than the traditional models and fine-tuning paradigm, providing higher performance for automated vulnerability assessment.
- We explore the ability of prompt learning to deal with scenarios where data is extremely scarce. Combined with transfer learning, excellent performance is achieved, which provides a new attempt to solve the low-resource tasks in software engineering.

II. BACKGROUND

A. CVE and Vulnerability Characteristics

Common Vulnerabilities and Exposures (CVE) [1] database maintains a list of publicly disclosed cybersecurity vulnerabilities. Each vulnerability is assigned a unique CVE identifier and mainly contains several fields: 1) a standardized text description of the vulnerability, 2) some related references, and 3) record creation date.

The National Vulnerability Database (NVD) [20] is the vulnerabilities repository created by the U.S. government. In addition to containing a list of CVEs, the NVD also provides severity of vulnerabilities and known affected configurations. The NVD scores vulnerabilities using the CVSS [2] which is based on a set of equations using some metrics. The common purpose of CVSS is to calculate the severity of vulnerabilities.

CVSS scores range from 0 to 10 and vulnerabilities are classified into four severity levels: “Critical” (9.0-10.0), “High” (7.0-8.9), “Medium” (4.0-6.9) and “Low” (0.1-3.9). Once Critical level vulnerabilities are exploited, attackers can usually directly obtain core permissions, resulting in serious consequences, and users should immediately patch vulnerable systems. High level vulnerabilities can lead to serious unauthorized access and information leakage when exploited. For such vulnerabilities, users are advised to apply updates as soon as possible. Medium level vulnerabilities typically require the attacker to interact with the victim or reside on the same local network to be exploited. For such vulnerabilities, users can consider applying updates. Low level vulnerabilities have little impact and users can consider whether to apply updates.

Another vulnerability characteristic similar to severity but very different is exploitability. The severity of vulnerability is more concerned with the impact of vulnerability exploitation, regardless of whether the vulnerability is actually used to attack systems. Many CVEs with high and critical CVSS scores are very complex, and may require special conditions or permissions to be actually exploited. Vulnerability exploitability focuses on whether the vulnerability has been used to attack the system in the wild. Recently, Cybersecurity and Infrastructure Security Agency (CISA) [3] issued compulsory direction to reduce the significant risk of known exploited vulnerabilities [21]. The directive establishes a CISA managed catalog of known exploited vulnerabilities.

B. A Motivating Example

For more than a decade, the number of vulnerabilities reported in the NVD was fairly stable, varying annually between 4,000 and 8,000 reported issues. However, since 2017, the number of vulnerabilities submitted has increased explosively, with more than 20000 vulnerabilities submitted in 2021 alone. The growth in the number of reported vulnerabilities is perhaps not particularly surprising, and is in line with the growing number of Internet-connected devices, products and digital services in use around the world [22].

The explosive growth of the number of vulnerabilities has brought great challenges to vulnerability assessment and CVSS scoring. For newly reported vulnerabilities, the duration between the initial publish date and the CVSS score update date can range from one day to more than a hundred days [7], so the severity characteristics of these vulnerabilities are often missing. In addition to severity, exploitability is also an important factor for developers to prioritize repair activities. Although CISA has established a catalog of known exploited vulnerabilities [4], however, as of March 2022, only a few hundred exploited vulnerabilities are recorded in the catalog. Therefore, for most vulnerabilities, we cannot determine their exploitability characteristics. The lack of vulnerability characteristics shown in Table I brings challenges to repair activities.

C. Research Challenges

In recent years, some studies [6], [7] have proved the potential of data-driven deep learning models, such as TextCNN [23]

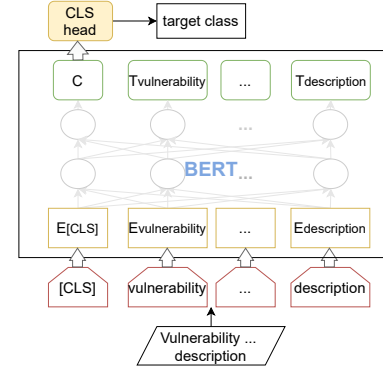


Fig. 1: Conventional fine-tuning paradigm.

and BiLSTM [24], in assessing the severity of vulnerabilities. In the current pre-training era, a better way to extract semantic information and improve the performance of classification tasks is to use PLMs. However, the conventional fine-tuning PLMs method has some challenges and shortcomings:

- The fine-tuning paradigm uses additional task-specific parameters and optimization objectives to tune the PLMs, resulting in the gap between model tuning and pre-training. Therefore, the knowledge in PLMs cannot be fully utilized.
- Fine-tuning paradigm requires a large amount of training data of downstream tasks to tune the introduced parameters, so it performs poorly in few-sample scenarios.

Specifically, as shown in Figure 1, for the two different classification tasks in this study, the fine-tuning paradigm needs to add different classification heads (CLS heads) on the top of PLM and use the optimization objective of the classification problem to tune the model. However, BERT-based PLMs use Masked Language Modeling (MLM) [12], [25] object to train the model in the pre-training stage, predicting the tokens in the input sequence that are randomly masked. This difference makes it difficult to fully mine the rich knowledge in PLMs when performing downstream tasks [16], [26]–[30].

Like low-resource scenarios often encountered in the security field, the exploitability classification in this study is faced with the difficulty of few samples. Like traditional deep learning methods, fine-tuning PLMs method needs enough data to tune the introduced CLS head parameters. Less training data makes it difficult for the model to effectively learn semantic information and easy to cause overfitting [31], resulting in poor performance. Research [32] shows that under few-sample scenarios, the effect of fine-tuning BERT on downstream tasks is very unstable. Models are also prone to catastrophic forgetting [33], which means the pre-trained knowledge is forgotten during learning of new knowledge. In addition, different classification tasks require different CLS heads, resulting in poor cross-tasks transferability of fine-tuning paradigm, especially in few-sample scenarios.

In order to solve the tasks of this study and overcome the shortcomings of fine-tuning paradigm, we introduce the prompt learning paradigm based on PLMs.

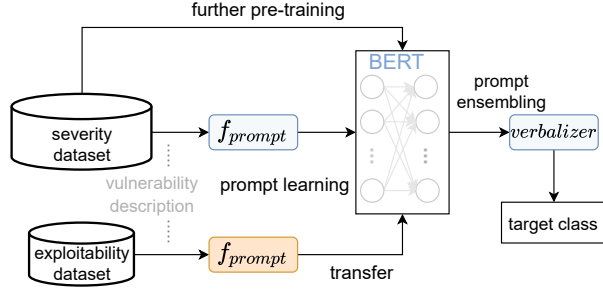


Fig. 2: Approach overview.

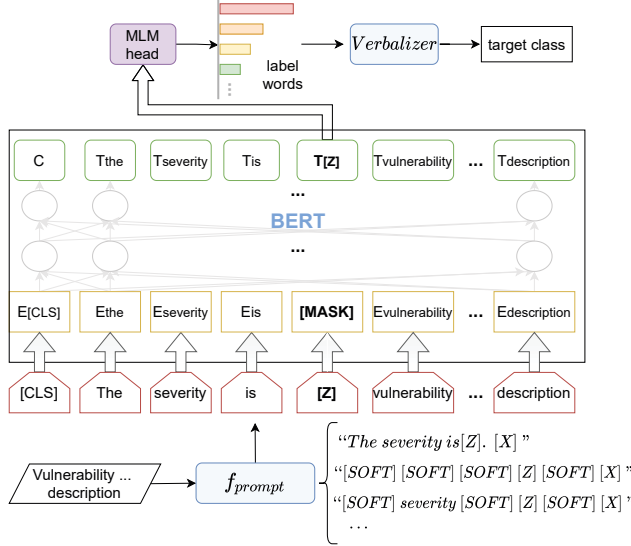


Fig. 3: Model architecture of prompt learning.

III. THE APPROACH

In this section, we will introduce the process steps of solving vulnerability characteristics prediction based on prompt learning.

A. Approach Overview

In this study, as shown in Figure 2, prediction of severity and exploitability based on vulnerability description and prompt learning mainly includes the following steps: 1) selection of appropriate PLMs and further pre-training; 2) severity classification based on prompt learning and prompt ensembling; 3) exploitability classification based on prompt learning and transfer learning. The collection and creation of vulnerability severity and exploitability datasets will be described in Section IV. Next, we will describe the above steps in detail.

Firstly, choose appropriate PLMs as the basis of prompt learning. For the classification tasks in this study, we use the commonly used BERT and RoBERTa PLM. It is necessary to use the downstream task corpus, i.e., vulnerability description text, or the in-domain corpus for further pre-training, which makes the PLMs contain more domain knowledge [34].

The next step is the specific prompt learning process, which combines the prompt ensembling on the severity classification task. As shown in Figure 3, prompt learning uses the prompt template $f_{prompt}(\cdot)$ to modify the original input text x to produce new input x' with a prompt: $x' = f_{prompt}(x)$. Prompt templates typically consist of text strings with two slots: input slot $[X]$ and answer slot $[Z]$. The input slot $[X]$ will be filled with input text x , and the answer slot $[Z]$ is eventually filled with the output label word z predicted by the PLM. For example, through the template “ $[X]$ The severity is $[Z]$.”, the input text “vulnerability description.” will be refactored to “vulnerability description. The severity is $[Z]$.”. For the masked language model, the $[Z]$ in the input is replaced with the masked token $[MASK]$. The reconstructed input text is then tokenized and fed into a PLM to predict the distribution over label words on the $[MASK]$ token position. For the classification task, label words are the restricted output of the model and can be mapped to target class labels through a verbalizer. Note that a verbalizer projects the target class labels to label words in the vocabulary, conversely, each predicted label word corresponds to a target class. In the above steps, we use the prompt ensembling method to synthesize the output of multiple templates to obtain label words, thus reducing the influence of template differences.

Finally, for the exploitability classification task, we can do prompt learning directly like the severity classification task. Prompt learning itself can achieve a better effect in few-sample scenarios. However, considering that these two tasks belong to the same domain and are relatively similar, we hope to combine transfer learning to further improve the performance of prompt learning. We first use prompt learning method to tune PLM on severity classification with a large dataset. Then, by simply modifying prompt and verbalizer, PLM is transferred to exploitability classification with few samples, and prompt learning is carried out again, so as to realize knowledge transfer.

The construction of prompt template and verbalizer is an important engineering in prompt learning. Next, we will introduce in detail the construction methods of various templates and verbalizers, as well as the prompt ensembling.

B. Prompt Template and Verbalizer Construction

Choosing the optimal template for each specific downstream task is often challenging and the method is not fixed. It is necessary to try a variety of template construction methods. According to the type of prompt token, prompt templates can be divided into hard prompt, soft prompt and hybrid prompt.

Hard Prompt is also known as discrete prompt [15]. In addition to the input slot and answer slot, the prompt tokens are understandable natural words. Manually creating intuitive templates based on domain knowledge is the simplest and most natural way. For example, in the task of severity classification, the hard prompt template can be defined as:

$$f_{prompt} = \text{“The severity is } [Z]. [X]\text{”} \quad (1)$$

TABLE II: Reconstructed vulnerability description by hard prompt template.

The severity is [Z]. Use after free in V8 in Google Chrome prior to 96.0.4664.110 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page.

Many new templates can be obtained by modifying the expression or position of natural language in the template. As shown in Table II, the input slot $[X]$ will be filled with vulnerability description text and reconstructed into a new input. PLM predicts the probability distribution over label words on the $[Z]$ position. The label word with the highest probability will be the intermediate generated answer of PLM.

Soft Prompt is also known as continuous prompt [15]. This prompt is not human interpretable natural language but continuous embeddings that can be tuned during the training of downstream tasks. Soft prompts overcome the difficulty of creating appropriate hard prompts for certain tasks, and prompt directly in the embedding space of the model. For example, soft prompt template can be defined as:

$$f_{prompt} = \text{"[SOFT] [SOFT] [SOFT] [Z] [SOFT] [X]"} \quad (2)$$

Soft prompt templates can be created by replacing natural words in hard templates with $[SOFT]$ tokens. The number of $[SOFT]$ token is variable and represents the length of continuous embeddings. These soft tokens can be initialized using embeddings of words in the hard prompt template, or through other methods [29].

Hybrid Prompt is a combination of hard prompt and soft prompt, consisting of hard tokens and soft tokens. Hard tokens are usually task-related important tokens that are not tuned during training, while soft tokens are tunable embeddings. Hybrid prompt templates can be obtained by retaining important tokens in hard template and replacing unimportant tokens with soft tokens. For example, hybrid prompt template can be defined as:

$$f_{prompt} = \text{"[SOFT] severity [SOFT] [Z] [SOFT] [X]"} \quad (3)$$

In this template, "severity" is a task-related token that we want not to be tuned, while other tokens such as "the" can be replaced and initialized as tunable embedding.

Verbalizer projects the target class labels to label words in the vocabulary. Label words are used to constrain the output space of PLM, that is, we only focus on the output probability of the model on these label words. Each target class label can correspond to one or more label words. With the verbalizer, the label word with the highest probability is mapped to the target class as the predict output. For the severity classification, one-one and one-many verbalizers can be defined as:

$$Verbalizer = \begin{cases} \text{"Low"} : & [\text{"low"}] \\ \text{"Medium"} : & [\text{"medium"}] \\ \text{"High"} : & [\text{"high"}] \\ \text{"Critical"} : & [\text{"critical"}] \end{cases} \quad (4)$$

$$Verbalizer = \begin{cases} \text{"Low"} : & [\text{"low"}, \text{"slight"}] \\ \text{"Medium"} : & [\text{"medium"}, \text{"moderate"}] \\ \text{"High"} : & [\text{"high"}, \text{"serious"}] \\ \text{"Critical"} : & [\text{"critical"}, \text{"urgent"}] \end{cases} \quad (5)$$

In this task, because the target class labels have specific meanings, they can be directly used as label words, or appropriate synonyms can be added.

C. Prompt Ensembling

Prompt Ensembling can combine the results of multiple prompt templates to take full advantage of different templates. Finding the optimal template is often challenging, and prompt ensembling can significantly reduce the cost of exploring the optimal performance template while achieving stable and better results on downstream tasks. In this study, we use the strategy of uniform averaging for prompt ensembling. Specifically, when predicting the class of an input in the inference stage, the output probabilities of multiple prompts and models on the same label word are uniformly averaged to achieve prompt ensembling. This process can be formally expressed as:

$$P(z|x) = \frac{1}{K} \sum_i^K P(z|f_{prompt,i}(x)) \quad (6)$$

where $f_{prompt,i}(\cdot)$ is the i -th prompt, and the type of each prompt template can be different. P represents the output probability of the model on label word z . We select K prompts with the best performance for ensembling. K can be set smaller, such as 3, to focus on a few of the best prompts.

IV. EXPERIMENTAL DESIGN

This section describes our research questions and related experimental designs. We collect and create vulnerability datasets, and set up baseline methods compared with the prompt learning. Besides, this section will introduce the evaluation metric and implementation details of the experiment.

A. Research Questions

This study aims to solve the tasks of predicting the severity and exploitability of vulnerabilities, and verify the effectiveness of the prompt learning method introduced to solve relevant challenges. Specifically, the research questions include:

RQ1: Compared with other methods, how effective is prompt learning in vulnerability severity classification?

RQ2: How effective is prompt learning in exploitability classification with few-sample scenarios?

RQ3: How do different prompt settings affect the performance of the prompt learning?

For severity classification, we design RQ1 to verify whether the prompt learning method has a better effect than other methods. For exploitability classification, we face the challenge of few training samples. Prompt learning is introduced to solve the above challenge. Meanwhile, considering the common data shortage problem in software engineering, we extend the

above scenario. In RQ2, we design experiments to compare the abilities of prompt learning and fine-tuning in various few-sample scenarios. We also verify the cross-tasks transferability of prompt learning. Since the design of prompt template and verbalizer is an important part, we design RQ3 to compare the impact of different settings on prompt learning performance, and verify the effectiveness of prompt ensembling.

B. Vulnerability Dataset

Experimental vulnerability data is obtained from NVD, which contains the CVSS severity score of vulnerabilities. For severity classification, we only collect vulnerability data with CVSS V3.X base score, and CVSS severity is used as the label. After processing, we ignore some vulnerabilities in which the CVSS score was 0. Before the publish date of January 1, 2022, a total of 93,512 vulnerability data are collected¹, the sample proportions of the four levels of Low, Medium, High, and Critical are 2%:40%:43%:15%. 90% of the samples are used as the training set to train models using multiple methods, 5% as the validation set to optimize models, and the remaining 5% as the test set to compare the effects of several methods.

For exploitability classification, we collect known exploited vulnerabilities from the CISA catalog [4], which is a subset of vulnerabilities in NVD. A total of 502 exploited vulnerabilities are collected as of March 16, 2022, with the label “exploited”. To construct a balanced dataset, combining random selection with manual review, we select the same number of other vulnerabilities from the complete NVD vulnerability dataset previously collected and set the label to “unexploited”. Therefore, we obtain an exploitability dataset with 1004 samples¹. The goal of our experiment is not only to use prompt learning to predict exploitability, but more importantly to explore the ability of prompt learning in few-sample or zero-sample scenarios. This is reasonable and meaningful, because in earlier time, we can get much less data or even no data. To thoroughly test the effect of the models trained in few-sample scenarios and reduce the contingency of test results, the size of the test set needs to be larger. In the experiment, the test set size is 500 and the validation set size is 200. The remaining data is used to construct the training set in various few-sample scenarios.

C. Baseline Building

In order to verify the effect of prompt learning in different tasks and scenarios, we set up several comparison methods or models, including Fine-tuning PLMs, TextCNN and Attention-based BiLSTM. The latter two models are used in the previous studies [6], [7], and we reproduce them according to the same network architecture. Fine-tuning PLMs method is the focus of comparison, because it has achieved better performance than traditional models in many downstream tasks. In addition, both it and prompt learning are PLMs-based paradigms.

Fine-tuning PLMs. Fine-tuning PLMs is a simple and widely-used method for text classification tasks. We select several PLMs as fine-tuning models, including BERT-base,

BERT-large, RoBERTa-base and RoBERTa-large, which are also used as the basic models for prompt learning. For these masked language models, the input sequence will be tokenized and a special [CLS] token will be added to the header of the sequence. The encoder [35] extracts the semantic information and outputs each token’s embedding in the last hidden layer. We take the embedding of the first token [CLS] as the representation of the whole sequence and feed it to a CLS head to predict the probability of the target label. CLS head is a feedforward neural network with softmax function, so additional parameters and optimization objective are introduced. These additional parameters and model’s parameters will be tuned by maximizing the probability of the correct label.

TextCNN. We reproduce a TextCNN [23] model, there are one convolutional layer followed by a 1-max-pooling layer, as well as an input layer and output layer. The input layer receives word embedding trained by the word2vec [36] as input. In the convolution and max-pooling layer, three different window sizes filters are used to extract features of phrases from input descriptions. After convolution, a 1-max-pooling operation is applied to each feature map obtained, thus the maximum value is selected as the most informative feature corresponding to a particular filter. These feature vectors will be connected and sent to the softmax classifier for classification.

Attention-Based BiLSTM. A Bi-directional LSTM [24] processes the input sequence in both forward and backward directions with two standard LSTMs, respectively [7]. For each word in the input sequence, the forward and backward hidden vectors are generated, and these two vectors are connected to form the hidden vector. Then, this hidden vector sequence will be fed to an attention-layer to encode word importance. The output representation vector of the whole sequence is finally fed into the softmax classifier for classification.

D. Evaluation Metrics

The severity classification has four labels: critical, high, medium and low. The exploitability classification has two labels: exploited and unexploited. For multi-class classification, we use weighted-F1 ($w\text{-}F1$) as the evaluation metric. Consider each class separately, taking this class as positive and other classes as negative. False negative (FN) means a positive sample is falsely predicted as negative and false positive (FP) means a negative sample is falsely predicted as positive. TP and TN represent true positive and true negative. We can get TP_i , FP_i , TN_i and FN_i values for each class i (0,1,2,3). Then, the Precision (P), Recall (R) and F1 score for each class can be calculated as follows:

$$P_i = \frac{TP_i}{TP_i + FP_i}, \quad R_i = \frac{TP_i}{TP_i + FN_i}, \quad F1_i = \frac{2 * P_i * R_i}{P_i + R_i} \quad (7)$$

For the overall evaluation, we use $w\text{-}F1$ to weight the F1 score of each class according to the sample proportion of each class in all samples (w_i), so as to avoid the class with very few samples having a large impact on the overall result.

$$w_i = \frac{\#sample_i}{\sum_{j=0}^3 \#sample_j}, \quad \text{weighted-F1} = \sum_{i=0}^3 w_i * F1_i \quad (8)$$

¹The datasets are available in: <https://sites.google.com/view/vulprompt>

Since the low-level has a small number of samples (2%), we don't use macro-F1, which treats each class equally and averages the F1 of each class uniformly. We don't want to pay more attention to the less important low-level vulnerabilities.

For binary classification, there are clear positive and negative classes. We use the common F1-score as the metric:

$$F1\text{-score} = \frac{2 \cdot P \cdot R}{P + R} \quad (9)$$

E. Experimental Details

Our experiment is based on Pytorch, Huggingface Transformers [37] and OpenPrompt [38] frameworks. Fine-tuning PLMs is implemented based on the Huggingface-Transformers framework. The implementation of prompt learning is based on OpenPrompt, through which prompt templates and verbalizers can be built quickly. TextCNN and Attention-Based BiLSTM are implemented based on Pytorch and tuned using hyperparameters from previous studies [6], [7].

We select BERT and RoBERTa as PLMs of the experiment. In the further pre-training stage, we further pre-training these PLMs on the vulnerability description corpus according to the settings in previous study [34], so as to make these models contain more domain knowledge. In the prompt learning experiment, we built various prompt templates and verbalizers based on OpenPrompt and set up different training strategies. For both PLMs-based prompt learning and fine-tuning, we train these models for 5 epochs and save the checkpoints with the best performance on the validation set. We set the same hyperparameters for fine-tuning and prompt learning for tuning, e.g., the learning rate is 1e-5, the optimizer is AdamW, the weight decay is set to 0.01, the lr_scheduler type is linear, the batch size is 16, etc.

In order to avoid the contingency of results, we set three different seeds for each experiment and repeat them three times respectively, and average the results. All experiments are run on a machine with NVIDIA GeForce RTX 3090 GPU.

V. EXPERIMENTAL EVALUATION AND RESULTS

In this section, we use the experimental results to answer the research questions previously raised and illustrate our findings.

A. RQ1: Compared with other methods, how effective is prompt learning in vulnerability severity classification?

Settings. For the vulnerability severity classification, we set up several baseline methods and models for comparison to test whether the prompt learning can achieve better performance. Fine-tuning PLMs is the main comparison object, because it and prompt learning are two different paradigms using PLMs. For the setting of prompt learning, we have tried a variety of combinations of prompt templates and verbalizers, including hard, soft and hybrid templates, one-one and one-many verbalizers. Some templates and verbalizers have been shown in equations (1) to (5), and more specific settings are shown in RQ3. Table III shows the performance from trying different prompt learning settings and prompt ensembling. The effect of different prompt settings on model performance and the effectiveness of prompt ensembling will be shown in RQ3.

TABLE III: Performance of several methods on vulnerability severity classification

Model	Method	$w\text{-}F1$ (%)	p-value
TextCNN	-	74.95	-
Attention-Based BiLSTM	-	74.71	-
BERT-base	fine-tuning	78.73	5.96E-07
	prompt learning	80.45	
BERT-large	fine-tuning	79.62	1.62E-06
	prompt learning	81.16	
RoBERTa-base	fine-tuning	79.37	4.21E-06
	prompt learning	80.76	
RoBERTa-large	fine-tuning	79.79	2.15E-06
	prompt learning	81.29	

Results. Table III shows the comparison results of different methods and models on the vulnerability severity test set. From this, we can see that fine-tuning PLMs has achieved greater performance improvement compared with traditional models. Taking RoBERTa-large, which has the best fine-tuning performance, as an example, its weighted-F1 is higher than TextCNN and Attention-Based BiLSTM respectively by 4.84% and 5.08%. This result fully shows that PLMs have learned rich knowledge in the pre-training stage and have a strong ability to extract the semantic information of the input text context.

From the table, we can observe that prompt learning paradigm performs better than fine-tuning paradigm on different PLMs. For BERT-base, BERT-large, RoBERTa-base and RoBERTa-large, prompt learning paradigm outperforms fine-tuning paradigm by 1.72%, 1.54%, 1.39% and 1.50%, respectively. For multiple experimental results of prompt learning and fine-tuning (three random seeds and three experiments under each seed), we perform independent samples T-test for statistical testing. The results show that there are statistical performance differences (p-value is much less than 0.05) between them. We present more detailed experimental results on the anonymous website¹. The above results indicate that prompt learning can extract the knowledge in PLMs more fully and effectively by transforming the classification task into the same cloze form as the PLMs pre-training stage. The transformation process simply reconstructs the input and adds task-related prompts. In addition, whether fine-tuning paradigm or prompt learning paradigm, the effect of a large model is better than that of base model. The rationale is larger PLMs contain more knowledge.

Answer to RQ1: For vulnerability severity classification, prompt learning achieves better results than fine-tuning PLMs and other models. In particular, the performance improvement of prompt learning paradigm compared with fine-tuning paradigm indicates that prompt learning can fully mine the domain knowledge in PLMs and realize better utilization of PLMs.

TABLE IV: Performance (F1-score(%)) of various methods on vulnerability exploitability classification in few-sample scenarios. 0, 32, 64, 128, 256 represent the number of training samples.

model	method	0	32	64	128	256
RoBERTa-base	fine-tuning	50	53.62	62.67	70.56	74.13
	prompt learning	52.70	63.74	67.13	73.19	75.41
RoBERTa-large	fine-tuning	50	58.53	65.87	72.87	74.82
	prompt learning	53.13	64.24	68.89	74.55	76.75
transferred RoBERTa-large1	fine-tuning	50	62.31	67.61	74.13	75.84
transferred RoBERTa-large2	prompt learning	55.41	69.27	71.80	75.92	77.92

B. RQ2: How effective is prompt learning in exploitability classification with few-sample scenarios?

Settings. We are not limited to the current scale of exploitability dataset, because we have been able to collect less or no data before this time. We simulate zero-sample and few-sample scenarios by controlling the size of the training set to explore the ability of prompt learning in these scenarios. We randomly select 0, 32, 64, 128, and 256 samples from the training set of exploitability dataset to form new training sets, and ensure the balance of the samples. Three random seeds are set for sampling to avoid the influence of the randomness of the training set on the model training. The focus of comparison with prompt learning is still fine-tuning paradigm. RoBERTa-base and RoBERTa-large are selected as the basic PLMs. We have tried many prompt settings in this experiment, including combinations of different templates and verbalizers. For example, the templates include “The exploitability is [Z]. [X]”, “The exploitability of the following vulnerability is [Z]. [X]”, etc. In this experiment, we only use hard prompt because it does not introduce additional tunable parameters, suitable for few-sample scenarios. The verbalizers include { “exploited”: “high”, “unexploited”: “low” }, { “exploited”: [“high”, “exploited”], “unexploited”: [“low”, “unexploited”] }, etc. In each sample training set and prompt setting, we conduct several experiments and average the overall results. Table IV shows the results of the prompt setting with the best performance. *Note that in order to more directly verify the effect of prompt learning in few-sample scenarios, we do not use prompt ensembling here.*

Results. Table IV shows the results obtained by prompt learning and fine-tuning in different few-sample scenarios. It can be observed that prompt learning paradigm achieves better performance than fine-tuning paradigm under five training set sizes. For RoBERTa-base and RoBERTa-large, on average, prompt learning performs 4.24% and 3.09% better than fine-tuning, respectively. Due to the introduction of additional CLS head parameters and random initialization, the fine-tuning paradigm needs to rely on a large amount of training data to tune the parameters, which is not effective in few-sample scenarios. Prompt learning paradigm does not introduce additional parameters when using hard templates. By simulating the form of pre-training stage, it can more efficiently stimulate PLMs and mine knowledge in them. This performance advantage is more obvious when there are fewer training samples. In the

scenario with only 32 training samples, for RoBERTa-base and RoBERTa-large, prompt learning outperforms fine-tuning by 10.12% and 5.71%, respectively. This result indicates that prompt learning can more effectively deal with scenarios with extremely scarce data, and only a small amount of training data is needed to simply tune PLMs to achieve promising results. For the fine-tuning paradigm, the few training data make the model learn little information, or even fail to converge.

Transferability of prompt learning. In the few-shot (sample) learning [39]–[41], the idea of transfer learning is very prevalent. Considering that vulnerability severity classification and exploitability classification belong to the same domain and are similar, we want to explore whether prompt learning has across-tasks transferability. The last row of Table IV shows the results obtained by the **transferred RoBERTa-large2** using the prompt learning method in different few-sample scenarios. Specifically, firstly, the prompt learning method is used to train RoBERTa-large in the severity classification task, so that the model can obtain a large amount of domain knowledge. Then modify the template and verbalizer to transfer to the exploitability classification task. For example, changing the hard template “The severity of the following vulnerability is [Z]. [X]” to “The exploitability of the following vulnerability is [Z]. [X]” requires changing only one word. Finally, use prompt learning method to train RoBERTa-large on the small-scale exploitability training set again. In contrast, **transferred RoBERTa-large1** is trained by fine-tuning before and after transfer. However, the CLS head needs to be replaced.

From the last four rows in the table, we can observe that for transferred RoBERTa-large2, the performance of prompt learning combined with transfer learning is on average 2.55% higher than that of prompt learning only, 5.65% higher than that of fine-tuning only, and 4.09% higher than that of fine-tuning combined with transfer learning. This performance advantage is more obvious in the scenario with only 32 training samples, indicating that transfer learning is effective and prompt learning has better transferability within the domain. In the case of hard prompt, the cost of transfer learning is simply to modify the prompt template and verbalizer. The knowledge learned before transfer is saved in PLM to the maximum extent. Moreover, the model structure has not changed. In the new task, only a tiny amount of training samples is needed to tune the model. However, for transferred RoBERTa-large1, the performance of fine-tuning combined with transfer learning is

only 1.56% higher than that of fine-tuning only. When the model is transferred to the exploitability classification, the original four-class CLS head will be discarded and a new two-class CLS head will be added. The new parameters still need relatively more training data to tune, resulting in poor transferability of fine-tuning in few-sample scenarios.

Answer to RQ2: Prompt learning has a strong ability to deal with few-sample scenarios and its performance surpasses fine-tuning. This performance advantage is more significant when training data is extremely scarce. In addition, when using hard prompt, prompt learning has a better across-tasks transferability within the domain because no additional parameters are introduced, which is suitable for transferring to tasks with few samples.

C. RQ3: How do different prompt settings affect the performance of the prompt learning?

We test the impact of different prompt settings on model performance by creating different types of templates and verbalizers. We conduct experiments on the severity classification task, and take RoBERTa-large as the basic PLM. The effectiveness of prompt ensembling is verified by comparing the effects of multiple prompts ensembling and a single prompt.

Impact of templates. To explore the impact of templates, here we fix the verbalizer as shown in equation (5). The types of prompt templates include: hard, soft and hybrid template. For hard templates, we manually add natural language prompts to build templates, such as T1, T4, and T7 in Table V. For soft templates, we create them by replacing all natural words in the above hard templates with soft tokens, such as T2, T5, and T8. For hybrid templates, we create them by retaining task-related words in the above hard templates and replacing non-important words with soft tokens, such as T3, T6, and T9.

The results are shown in Table V. We can observe that for RoBERTa, the performance of prefix templates (T1, T2, T3) is generally higher than that of postfix templates (T7, T8, T9), that is, putting the prompt in front of the input text can achieve better results. This may be because the position of [Z] in the prefix template is fixed, which can extract the context semantic more effectively. We can observe that the performance of the best-performing template (T6) is 0.85% higher than that of the worst-performing template (T7). Compared with hard templates, soft and hybrid templates achieve higher performance. Due to the large severity dataset, there are enough training samples to tune the model’s parameters and the tunable parameters introduced by soft and hybrid prompts. Soft tokens extend the possibilities of templates. Hybrid prompts keep the embeddings of task-related words not to be tuned and sometimes achieve better performance.

Impact of verbalizers. To test the effects of different verbalizers, the prompt template is fixed as T6 in Table V. We set up different one-one and one-many verbalizers, including equation (4), (5), etc. From the results in Table VI, we can observe that different verbalizers have an impact on the performance of prompt learning. By adding task-related label

TABLE V: Performance of different prompt templates on severity classification based on RoBERTa-large. The verbalizer is fixed as shown in equation (5).

	Template	Type	w-F1 (%)
T1	The severity is [Z]. [X]	Hard	80.22
T2	[SOFT]*3 [Z] [SOFT] [X]	Soft	80.34
T3	[SOFT] severity [SOFT] [Z] [SOFT] [X]	Hybrid	80.39
T4	The severity of the following vulnerability is [Z]. [X]	Hard	80.45
T5	[SOFT] * 7 [Z] [SOFT] [X]	Soft	80.57
T6	[SOFT] severity [SOFT]*3 vulnerability [SOFT] [Z] [SOFT] [X]	Hybrid	80.73
T7	[X] The severity is [Z].	Hard	79.88
T8	[X] [SOFT] * 3 [Z] [SOFT]	Soft	80.02
T9	[X] [SOFT] severity [SOFT] [Z] [SOFT]	Hybrid	79.95

TABLE VI: Performance of different verbalizers on severity classification based on RoBERTa-large. The template is T6.

	Verbalizer	w-F1 (%)
V1	low: ["low"], medium: ["medium"] high: ["high"], critical: ["critical"]	80.21
V2	low: ["low", "slight"] medium: ["medium", "moderate"] high: ["high", "serious"] critical: ["critical", "urgent"]	80.73
V3	low: ["low", "slight", "small"] medium: ["medium", "moderate", "middle"] high: ["high", "serious", "big"] critical: ["critical", "urgent", "huge"]	80.32

words with similar meaning to the target label to form a one-many verbalizer, the effect of prompt learning can be appropriately improved. However, this does not mean that more words in verbalizer are better. If the added words are not related to the task, they will interfere with the prediction.

Effectiveness of prompt ensembling. Based on the “Impact of template” experiment, we select K prompts with the best performance on the validation set for ensembling. Specifically, in the inference stage, the output probabilities of K prompt models on label words are uniformly averaged to calculate the probability after ensembling, so as to determine the final output class. We test the performance of the prompt ensembling with K from 1 to 5 to determine the appropriate K.

From the results in Table VII, we can observe that multiple prompts outperform a single prompt through ensembling. When K=3, RoBERTa-large achieves the best prompt ensembling performance. However, it is not that the more prompts involved in the ensembling, the higher the performance after ensembling. As more prompts participate in the ensembling, the uniform averaging strategy rather than weighted averaging can cause a less-performing prompt to affect the overall result. A single well-designed prompt can achieve performance that exceeds the fine-tuning paradigm. However, selecting an appropriate number of prompts for ensembling can further improve the performance of prompt learning, thereby reducing the cost of searching for an optimal performance template.

TABLE VII: Performance of top K prompts after ensembling. All prompts are based on Table V.

	Top 1	Top 2	Top 3	Top 4	Top 5
$w\text{-}F1$ (%)	80.73	81.17	81.29	81.22	81.23

Answer to RQ3: The design of templates and verbalizers will affect the performance of the prompt learning. In the case of sufficient training data, a hybrid or soft template can achieve better results. The effect of prompt learning can be appropriately improved by adding task-related words to the target labels to form a one-many verbalizer. In addition, selecting multiple prompts for ensembling usually improves prompt learning performance.

D. Threats to Validity

Experiments on prompt learning and fine-tuning paradigms are based on BERT and RoBERTa, but not on a wider range of PLMs. However, the experimental results on these two classical PLMs can also illustrate the superiority of the prompt learning. Although we have tried many template and verbalizer designs, the performance of prompt learning achieved thus is likely to be suboptimal. Finding the optimal prompt setting is difficult or inconclusive. We did not use more sophisticated prompt ensembling strategies, such as weighted averaging. In this suboptimal setting, the performance of prompt learning has exceeded the fine-tuning paradigm and traditional models.

In addition, factors such as the sampling construction of datasets and hyperparameters may also affect the stability of the experiment, although we set up multiple random seeds and repeat experiments to reduce the contingency of the results.

E. Discussion

We construct a small-scale balanced dataset for vulnerability exploitability classification, which facilitates the model training and the simulation of few-sample scenarios. However, this may not conform to real-world data distribution. Determining the unexploitable vulnerabilities and the true class distribution is difficult. But even with the limitations of our dataset, prompt learning has shown advantages in the same few-sample scenarios. We set large validation and test sets in this dataset in order to fully compare the performance of prompt learning with fine-tuning in few-sample scenarios and reduce the contingency. Once the advantage of prompt learning is determined, it is unnecessary to set large validation and test sets in a real data scarcity scenario. The model training only uses the prompt learning, for example, the better prompt learning setting can be selected through k-fold cross validation.

VI. RELATED WORK

A. Deep Learning in Vulnerability Detection and Analysis

Deep learning models have recently been adopted in vulnerability detection and analysis, due to their ability to solve complex problems even if the source code and software text are very diverse and massive [5], [42]. Zhou et al. [43] use graph neural networks to learn the semantic representation

of code to identify vulnerabilities. Feng et al. [44] use big programming language data to train CodeBERT, which can be used to solve many code intelligence tasks, such as defect detection [45]. Han, Gong et al. [6], [7] use TextCNN and BiLSTM models to predict the characteristics of vulnerabilities, such as severity, based on vulnerability descriptions. Different from the traditional models or the conventional fine-tuning PLMs method used in previous studies, we use the PLMs-based prompt learning to predict the severity and exploitability characteristics of vulnerabilities, achieving better performance. Das et al. [46] use BERT-based Siamese network to learn the similarity between vulnerability description and CWE description to map vulnerability to CWE. Through similarity learning and fine-tuning, the siamese network can handle the above few-sample task. However, this is different from our tasks. The prompt learning we use is oriented towards more general forms of classification tasks.

B. Prompt Learning

PLMs-based prompt learning has gradually become a new popular paradigm in NLP. Liu et al. [15] investigate the development and related studies of prompt learning. In this paper, we use the same formal expression to describe prompt learning. As PLMs use large-scale corpus in pre-training stage, Petroni et al. [47] believe that PLMs could be used as knowledge bases. They artificially construct natural language prompts to retrieve entity relationship knowledge in PLMs. Experiments show that prompts can mine the factual knowledge in PLMs. Schick et al. [16] use many prompt templates to transform the classification task into the cloze questions of PLMs, so as to complete the tasks of few-shot text classification and natural language inference. In order to pursue higher prompt learning performance, some studies have tried more types of prompt templates than manually crafted hard templates. In [26], [27], [48], Li, Lester, Zhong et al. use soft templates composed of continuous vectors, which can be tuned by training samples, so as to explore templates more suitable for downstream tasks. Liu, Han, et al. [29], [30] use hybrid templates to achieve promising performance on multiple tasks. In this study, we have tried various types of templates and combined prompt ensembling and transfer learning to improve the performance of prompt learning.

VII. CONCLUSION AND FUTURE WORK

In this paper, based on vulnerability description, we use PLMs-based prompt learning to predict severity and exploitability characteristics, and achieve performance beyond the traditional models and fine-tuning paradigm. Especially in the few-sample scenarios we set up, prompt learning has a great advantage. We also briefly evaluate the relevant settings that affect the performance of prompt learning. In the future, we will conduct large-scale experiments to explore different prompt settings. We will also try to compare prompt learning with other few-shot learning methods, such as data augmentation, meta-learning [49], etc. The trade-off between model training time and performance is also worth investigating.

ACKNOWLEDGMENT

This work is supported by the Basic Research Program of Jiangsu Province (Grant No. BK20201192). Dr. Xue's research is also supported by CAS Pioneer Hundred Talents Program.

REFERENCES

- [1] MITRE, "Common Vulnerabilities and Exposures," <https://cve.mitre.org/>, [Online; accessed 1-May-2022].
- [2] FIRST, "Common Vulnerability Scoring System," <https://www.first.org/cvss/>, [Online; accessed 1-May-2022].
- [3] CISA, "Binding Operational Directive 22-01," <https://www.cisa.gov/binding-operational-directive-22-01/>, [Online; accessed 1-May-2022].
- [4] CISA, "Known exploited vulnerabilities catalog," <https://www.cisa.gov/known-exploited-vulnerabilities-catalog/>, [Online; accessed 1-May-2022].
- [5] Y. Yang, X. Xia, D. Lo, and J. Grundy, "A survey on deep learning for software engineering," *ACM Computing Surveys (CSUR)*, 2020.
- [6] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," in *2017 IEEE International conference on software maintenance and evolution (ICSME)*. IEEE, 2017, pp. 125–136.
- [7] X. Gong, Z. Xing, X. Li, Z. Feng, and Z. Han, "Joint prediction of multiple vulnerability characteristics through multi-task learning," in *2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 2019, pp. 31–40.
- [8] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang *et al.*, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021.
- [9] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, vol. 63, no. 10, pp. 1872–1897, 2020.
- [10] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.
- [11] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [13] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
- [14] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.
- [15] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *arXiv preprint arXiv:2107.13586*, 2021.
- [16] T. Schick and H. Schütze, "Exploiting cloze questions for few shot text classification and natural language inference," *arXiv preprint arXiv:2001.07676*, 2020.
- [17] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 423–438, 2020.
- [18] G. Qin and J. Eisner, "Learning how to ask: Querying lms with mixtures of soft prompts," *arXiv preprint arXiv:2104.06599*, 2021.
- [19] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.
- [20] NIST, "National Vulnerability Database," <https://nvd.nist.gov/>, [Online; accessed 1-May-2022].
- [21] CISA, "Cybersecurity and Infrastructure Security Agency," <https://www.cisa.gov/>, [Online; accessed 1-May-2022].
- [22] REDSCAN, "NIST security vulnerability trends in 2020: an analysis," <https://www.redscan.com/>, [Online; accessed 1-May-2022].
- [23] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *arXiv preprint arXiv:1510.03820*, 2015.
- [24] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu, "Attention-based bidirectional long short-term memory networks for relation classification," in *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)*, 2016, pp. 207–212.
- [25] W. L. Taylor, "'cloze procedure': A new tool for measuring readability," *Journalism quarterly*, vol. 30, no. 4, pp. 415–433, 1953.
- [26] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.
- [27] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.
- [28] Y. Gu, X. Han, Z. Liu, and M. Huang, "Ppt: Pre-trained prompt tuning for few-shot learning," *arXiv preprint arXiv:2109.04332*, 2021.
- [29] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *arXiv preprint arXiv:2103.10385*, 2021.
- [30] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, "Ptr: Prompt tuning with rules for text classification," *arXiv preprint arXiv:2105.11259*, 2021.
- [31] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," *arXiv preprint arXiv:2002.06305*, 2020.
- [32] T. Zhang, F. Wu, A. Katiyar, K. Q. Weinberger, and Y. Artzi, "Revisiting few-sample bert fine-tuning," *arXiv preprint arXiv:2006.05987*, 2020.
- [33] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.
- [34] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?" in *China national conference on Chinese computational linguistics*. Springer, 2019, pp. 194–206.
- [35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
- [37] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.
- [38] N. Ding, S. Hu, W. Zhao, Y. Chen, Z. Liu, H.-T. Zheng, and M. Sun, "Openprompt: An open-source framework for prompt-learning," *arXiv preprint arXiv:2111.01998*, 2021.
- [39] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 403–412.
- [40] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.
- [41] E. Perez, D. Kiela, and K. Cho, "True few-shot learning with language models," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [42] C. Watson, N. Cooper, D. N. Palacio, K. Moran, and D. Poshyanyk, "A systematic literature review on the use of deep learning in software engineering research," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 2, pp. 1–58, 2022.
- [43] Y. Zhou, S. Liu, J. Siow, X. Du, and Y. Liu, "Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks," *Advances in neural information processing systems*, vol. 32, 2019.
- [44] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang *et al.*, "Codebert: A pre-trained model for programming and natural languages," *arXiv preprint arXiv:2002.08155*, 2020.
- [45] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang *et al.*, "Codexglue: A machine learning benchmark dataset for code understanding and generation," *arXiv preprint arXiv:2102.04664*, 2021.
- [46] S. S. Das, E. Serra, M. Halappanavar, A. Pothan, and E. Al-Shaer, "V2w-bert: A framework for effective hierarchical multiclass classification of software vulnerabilities," in *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2021, pp. 1–12.
- [47] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" *arXiv preprint arXiv:1909.01066*, 2019.

- [48] Z. Zhong, D. Friedman, and D. Chen, “Factual probing is [mask]: Learning vs. learning to recall,” *arXiv preprint arXiv:2104.05240*, 2021.
- [49] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, “Meta-learning in neural networks: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.