Enhancing Neural Models with Vulnerability via Adversarial Attack

Rong Zhang[1,*], **Qifei Zhou**[1,*], **Bo An**[2], **Weiping Li**[1,†], **Tong Mo**[1], **Bo Wu**[3]
[1]Peking University, Beijing, China
[2]Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]MIT-IBM Watson AI Lab, Massachusetts, United States
{rzhangpku, qifeizhou}@pku.edu.cn, anbo@iscas.ac.cn
{wpli, motong}@ss.pku.edu.cn, bo.wu@ibm.com

Abstract

Natural Language Sentence Matching (NLSM) serves as the core of many natural language processing tasks. 1) Most previous work develops a single specific neural model for NLSM tasks. 2) There is no previous work considering adversarial attack to improve the performance of NLSM tasks. 3) Adversarial attack is usually used to generate adversarial samples that can fool neural models. In this paper, we first find a phenomenon that different categories of samples have different vulnerabilities. Vulnerability is the difficulty degree in changing the label of a sample. Considering the phenomenon, we propose a general two-stage training framework to enhance neural models with **V**ulnerability via **A**dversarial **A**ttack (VAA). We design criteria to measure the vulnerability which is obtained by adversarial attack. VAA framework can be adapted to various neural models by incorporating the vulnerability. In addition, we prove a theorem and four corollaries to explain the factors influencing vulnerability effectiveness. Experimental results show that VAA significantly improves the performance of neural models on NLSM datasets. The results are also consistent with the theorem and corollaries. The code is released on https://github.com/rzhangpku/VAA.

1 Introduction

Natural Language Sentence Matching (NLSM) aims to compare two sentences and identify the relationship between them (Wang et al., 2017). NLSM serves as the core of many natural language processing tasks, such as question answering and information retrieval (Wang et al., 2016). 1) Most previous work develops a single specific neural model for NLSM tasks. We propose a general framework that can enhance various neural models proposed by previous work. 2) There is no previous work considering adversarial attack to improve the performance of NLSM tasks. This paper is the first to take advantage of adversarial attack to improve the performance of NLSM tasks. 3) Adversarial attack is usually used to generate adversarial samples that can fool neural models. In contrast, we take advantage of adversarial attack to enhance neural models.

We introduce a concept *vulnerability*, the difficulty degree in changing the label of a sample. Inspired by the finding that adversarial samples are more vulnerable than normal ones (Zhou et al., 2020), we first find a phenomenon that different categories of samples have different vulnerabilities. That is, it is easy to change the labels of samples belonging to some categories, whereas it is difficult for others. Here we take Quora Question Pairs (QQP) dataset as an example. If we want a `paraphrase` pair to be `non-paraphrase`, changing one word is sufficient. For example, simply adding "not" or "hardly" can completely change the meaning of a sentence. However, it is difficult to make two irrelevant sentences have the same meaning. Different categories of samples have different vulnerabilities. So the vulnerability can be considered as a feature, which can be used to predict the label of the sample. To the best of

[*]Rong Zhang and Qifei Zhou contribute equally to this paper.
[†]Weiping Li is the corresponding author.

Proceedings of the 28th International Conference on Computational Linguistics, pages 1133–1146
Barcelona, Spain (Online), December 8-13, 2020

Table 1: Two examples from QQP dataset. + means that the two sentences are `paraphrase`; − means that the two sentences are `non-paraphrase`.

Label	Sentence
+	S1: Why are people supporting Donald Trump? S2: Why do you think people are supporting Trump?
−	S3: What is the difference between the Moto2 and the Moto3 race? S4: Who is ahead in the race to sell self-driving cars?

our knowledge, no study has considered the vulnerability and found the phenomenon thus far. Our study is the first to identify the phenomenon and take advantage of the vulnerability to predict the label.

Four sentences are listed in Table 1 to explain the vulnerability. All four sentences are selected from QQP dataset. Initially, sentence pair $(\mathrm{S1}, \mathrm{S2})$ is `paraphrase`, i.e., S1 and S2 have the same meaning. Then, we modify S2 by adding "not", and S2 will be S2′: "Why do you think people are not supporting Trump?". $(\mathrm{S1}, \mathrm{S2}')$ is `non-paraphrase`, which means that the label of $(\mathrm{S1}, \mathrm{S2})$ has been changed. As for $(\mathrm{S3}, \mathrm{S4})$, it is `non-paraphrase`, i.e., S3 and S4 have different meanings. S3 and S4 have many different words, and many words need to be changed if we want $(\mathrm{S3}, \mathrm{S4})$ to be `paraphrase`. It is difficult to make $(\mathrm{S3}, \mathrm{S4})$ `paraphrase`. Thus far, $(\mathrm{S1}, \mathrm{S2})$ has larger vulnerability than $(\mathrm{S3}, \mathrm{S4})$.

This paper is the first to take advantage of adversarial attack to improve the performance of NLSM tasks. We adopt adversarial attack to implement the action of changing the label of a sample and measure the vulnerability of the sample. Recent studies have shown that neural models will be fooled by adversarial samples (Goodfellow et al., 2014; Kurakin et al., 2016; Hu and Tan, 2018). In the case of images, human-imperceptible modifications to the original inputs can mislead a neural model to provide incorrect predictions. In the case of texts, we can simply replace, delete, or add some words to change the output of a model. Such modification methods are referred to as adversarial attacks. Adversarial attack is usually used to generate adversarial samples to fool neural models. In contrast, we take advantage of adversarial attack to enhance neural models.

Considering the analysis above, we propose a general two-stage training framework to enhance neural models with **V**ulnerability via **A**dversarial **A**ttack (VAA). In stage one, we pre-train a model A, which can choose any neural model proposed by previous work. In stage two, we obtain the vulnerability of input sample via adversarial attack using model A, and we train a new model B by combining the original sample and its vulnerability as inputs. Experiments demonstrate that VAA is an effective framework that can be adapted to neural models, improving their performance on NLSM datasets.

Our main contributions are threefold:

1. We first find that different categories of samples have different vulnerabilities. Moreover, we define criteria to measure the vulnerability of a sample. In addition, we first take advantage of adversarial attack to enhance neural models for NLSM tasks. Furthermore, we prove a theorem and four corollaries to explain the factors influencing vulnerability effectiveness.

2. We propose a novel two-stage training framework called VAA by incorporating the vulnerability into the neural model. VAA significantly enhances the performance of the model on NLSM datasets via adversarial attack.

3. The proposed VAA framework is of general purpose and works well for many neural models. VAA does not require much additional computing resources because the model structures and parameters can be shared between model A and model B in the two stages.

2 Related Work

2.1 Natural Language Sentence Matching

There are two types of deep learning frameworks for Natural Language Sentence Matching (NLSM) tasks. The first framework (Shen et al., 2018; Choi et al., 2018; Conneau et al., 2017) is based on

Siamese architecture (Bromley et al., 1993). Two input sentences are encoded individually into sentence vectors with the same neural network encoder. Then, a matching decision is made solely based on the two independent sentence vectors. The major disadvantage of this framework is that there is no explicit lower-level semantic interaction between sentences, which may cause the loss of some important information.

Therefore, the matching-aggregation framework is proposed to deal with this problem. This framework compensates for the lack of interaction and uses the cross-sentence feature or inter-sentence attention. Wang et al. (2017) proposed a bilateral multi-perspective matching model for NLSM tasks. Tan et al. (2018) proposed multiway attention networks with multiple attention functions. Chen et al. (2017) proposed ESIM model and demonstrated the effectiveness of carefully designed sequential inference models based on chain LSTMs. Hong et al. (2020) proposed legal feature enhanced semantic matching network for similar case matching.

2.2 Adversarial Attack

Adversarial attack refers to the modification of a few pixels with small perturbations of images or changing a few words of text sequences. Adversarial samples will be obtained with such minor changes to normal samples. The target model will misclassify adversarial samples, which remain correctly classified by humans. Numerous methods have been proposed to mislead models in image classification tasks, including box-constrained L-BFGS (Szegedy et al., 2013), the fast-gradient sign method (FGSM) (Goodfellow et al., 2014), the basic iterative method (BIM) (Kurakin et al., 2016), the project gradient descent (PGD) (Madry et al., 2017), the jacobian-based saliency map attack (JSMA) (Papernot et al., 2016a), the Carlini & Wagner (C&W) attack (Carlini and Wagner, 2016), and the adversarial transformation networks (ATN) (Baluja and Fischer, 2017). Such adversarial attacks are gradient-based, score-based, transfer-based, or decision-based.

3 Vulnerability and VAA Framework

In our notation, we have two input sentences a and b in an NLSM task. A sample $X = (a, b)$ is a pair of two sentences a and b. There are n labels, and $y_{\text{true}} \in \{1, \ldots, n\}$ is the true label of sample X. $\mathbf{a} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{\ell_a}\}$ and $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{\ell_b}\}$ are the embeddings of sentences a and b, respectively. $\mathbf{a}_i$ or $\mathbf{b}_j \in \mathbb{R}^l$ is a word embedding of l-dimensional vector, which can be initialized with some pre-trained word embeddings. The goal of the NLSM task is to predict label $y_{\text{pred}} \in \{1, \ldots, n\}$ that indicates the relationship between a and b.

3.1 Vulnerability Measurement and Adversarial Embedding

3.1.1 How to Measure the Vulnerability of a Sample X?

Two *vulnerability-measure criterion*, F_1 and F_2, are defined to measure the vulnerability of a sample X. First, we pre-train a model that can successfully classify most samples. When a new sample X arrives, we attack the model with a specific attack strength ϵ to generate adversarial sample $\tilde{X}$. Normal sample $X = (a, b)$ and adversarial sample $\tilde{X} = (\tilde{a}, \tilde{b})$ are the original sentence pair and adversarial sentence pair, respectively. Normal embedding E and adversarial embedding $\tilde{E}$ are the word embeddings of X and $\tilde{X}$, respectively. We feed E and $\tilde{E}$ to the model to get vectors O and $\tilde{O}$, the outputs of the last dense layer, respectively.

$$\varphi_i = \frac{e^{O_i}}{\sum_j e^{O_j}}, \quad \psi_i = \frac{e^{\tilde{O}_i}}{\sum_j e^{\tilde{O}_j}} \tag{1}$$

$$i^* = \arg\max_i \varphi_i = \{i^* \mid \varphi_{i^*} = \max_i \varphi_i\} \tag{2}$$

$$F_1 = -\sum_i \varphi_i \times \log \psi_i, \quad F_2 = -\log \psi_{i^*} \tag{3}$$

where O_i and $\tilde{O}_i$ are the i-th items of vectors O and $\tilde{O}$, respectively. F_1 measures the vulnerability by considering the attack perturbations of all classes. F_2 measures the perturbation of the main class. Both

F_1 and F_2 measure the difference between O and $\tilde{O}$. It should be noted that if we have E and $\tilde{E}$ but do not have X and $\tilde{X}$, we can still obtain F_1 and F_2.

3.1.2 How to Obtain the Adversarial Embedding $\tilde{E}$?

VAA framework does not need adversarial sample $\tilde{X}$; the adversarial embedding $\tilde{E}$ is sufficient. We only need to modify the word embedding to measure the vulnerability, and do not have to find the real word corresponding to adversarial embedding $\tilde{E}$. Hence, a large number of adversarial attack methods can be employed directly by VAA. Most adversarial attacks proposed for either images (Brendel et al., 2017; Carlini and Wagner, 2016) or texts (Rosenberg et al., 2018; Papernot et al., 2016b) run extremely slowly. Owing to training time and memory constraints, we adopt the fast-gradient sign method (FGSM) (Goodfellow et al., 2014) to generate adversarial embedding $\tilde{E}$ rapidly. FGSM is the fastest gradient-based attack method and requires minimal memory.

FGSM can only generate adversarial embedding $\tilde{E}$ but cannot generate adversarial sample $\tilde{X}$, because there may be no real word corresponding to $\tilde{E}$. $\tilde{E}$ can be obtained with FGSM in the following two ways.

$$\tilde{E} = E + \epsilon \text{sign}(\bigtriangledown_E J(\theta, E, y_{\text{pred}})) \quad [L_\infty] \tag{4}$$

$$\tilde{E} = E + \epsilon \frac{\bigtriangledown_E J(\theta, E, y_{\text{pred}})}{\|\bigtriangledown_E J(\theta, E, y_{\text{pred}})\|_2} \quad [L_2] \tag{5}$$

where Equation (4) and Equation (5) refer to the L_∞- and L_2-norm variants, respectively. FGSM works by linearizing the loss function in the L_∞ or L_2 neighborhood of normal sample X. ϵ is the parameter that determines the perturbation size in the direction of the gradient. J, θ, and y_{pred} are the cross-entropy loss, the parameters of the model, and the predicted label of X, respectively. Note that the FGSM that we use is slightly different from the original one used in (Goodfellow et al., 2014). We replace y_{true} with y_{pred} here. Because we cannot obtain y_{true} in the test set, and we need to measure the vulnerability rather than to obtain adversarial sample $\tilde{X}$. Thus, y_{pred} is sufficient for our need.

3.2 VAA Framework

Suppose that there is a neural model A proposed by previous work for an NLSM dataset. We propose a general two-stage training framework VAA, which can enhance the performance of model A with vulnerability. Putting model A into the VAA framework will achieve better performance than only model A on the NLSM dataset. VAA framework is shown in Figure 1. In stage one, we pre-train model A. In stage two, we fix model A and fine-tune model B.

3.2.1 Stage One

Stage one is the same as a regular training process. In this stage, we train the model A on the NLSM dataset regularly. After stage one, we will obtain a well-trained model A, which is called pre-trained model A relative to stage two. Usually, neural models will adopt a multilayer perceptron (MLP) as the final layer. For the convenience in the following description, model A is divided into model A^- and the final MLP classifier.

3.2.2 Stage Two

First, we will get normal embedding E of normal sample X with some pre-trained word embeddings. Then, adversarial embedding $\tilde{E}$ will be obtained with E, as shown in Equations (4) and (5). $\tilde{E}$ is obtained via adversarial attack with the help of model A. Normal embedding E and adversarial embedding $\tilde{E}$ are fed into model A^-; then normal logit $V_1 \in \mathbb{R}^h$ and adversarial logit $V_2 \in \mathbb{R}^h$ are obtained, respectively.

$$\Delta V = V_1 - V_2 \tag{6}$$

We use vectors V_1 and V_2 instead of F_1 and F_2 as shown in Equation (3). Because F_1 or F_2 is only a numeric value, and using it here may result in the loss of some important information. Vector ΔV contains the vulnerability information of the sample X. Different categories of samples have different

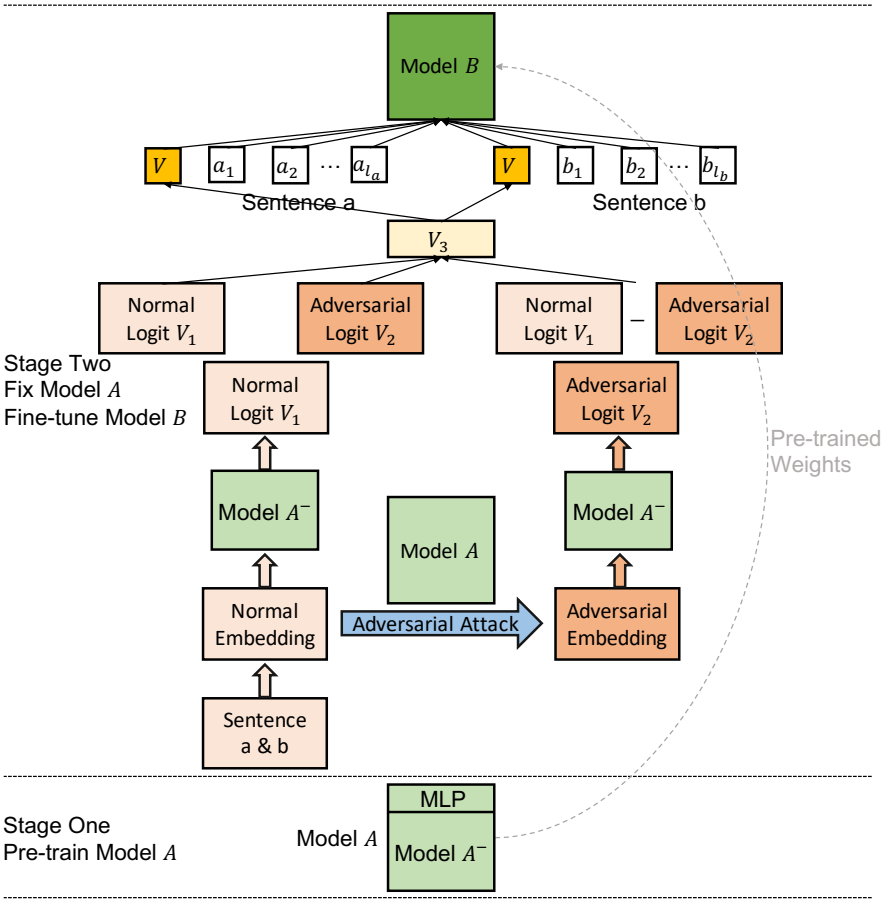


Figure 1: Overview of VAA framework for NLSM datasets. In stage one, we pre-train model A. In stage two, we fix model A and fine-tune model B.

vulnerabilities. That is, some categories of samples will have large ΔV, whereas others will have small ΔV. ΔV can measure the vulnerability just like F_1 or F_2.

V_1, V_2, and ΔV will be concatenated together to get $V_3 \in \mathbb{R}^{3h}$. Then, V_3 will be projected into a vector $V \in \mathbb{R}^l$, which has the same dimension as word embeddings $\mathbf{a}_i$ and $\mathbf{b}_j$.

$$V_3 = [V_1; V_2; \Delta V] \tag{7}$$

$$V = WV_3 + b \tag{8}$$

where $b \in \mathbb{R}^l$ is the bias and the projection is the weight parameter matrix $W \in \mathbb{R}^{l \times 3h}$.

Then V will be fed into model B along with the word embeddings $\mathbf{a}_i$ and $\mathbf{b}_j$. Model B and model A share the same model structure. The parameters of model B are initialized with the parameters of the pre-trained model A. Therefore, model B can be trained rapidly, and VAA does not require much additional computing resources. In stage two, we fix model A and fine-tune model B. That is, model A will not be trained; only model B will be trained.

VAA is a two-stage training framework; we first train stage one, then train stage two. VAA is a general framework that can be adapted to many neural models for NLSM tasks, because model A can choose any neural model proposed by previous work.

3.3 Factors Influencing Vulnerability Effectiveness

Theorem 1. *Suppose that there is a model predicting the label only with the vulnerability. Let n be the number of labels and P be the model accuracy representing the model performance. Suppose that*

the n categories all obey normal distribution, and $X_1 \sim \mathcal{N}(\mu, \sigma^2)$, $X_2 \sim \mathcal{N}(\mu + d, \sigma^2)$, ..., $X_n \sim \mathcal{N}(\mu + (n-1)d, \sigma^2)$. Then,

$$P = \frac{1-2S}{n} + 2S,$$
$$\textbf{\textit{where}}\ S = \int_{\mu}^{\mu+\frac{d}{2}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \tag{9}$$

Different categories of samples have different vulnerabilities. So the vulnerability can be considered as a feature, which can be used to predict the label of the sample. We suppose that the model predicts the label only with the vulnerability feature. The model performance is brought only by the vulnerability, so model accuracy P can represent *vulnerability effectiveness*. d represents the vulnerability difference between the n categories. Theorem 1 establishes the clear functional relationship between P and n, d, and explains the factors influencing vulnerability effectiveness. Then, we will have the following four corollaries for Theorem 1. The appendix shows the proofs of Theorem 1 and the four corollaries.

Corollary 2. *P is a monotonically decreasing function of n.*

Corollary 3.

$$\min_n P = \lim_{n\to+\infty} P = 2S, \quad \max_n P = P|_{n=2} = S + \frac{1}{2} \tag{10}$$

Corollary 4. *P is a monotonically increasing function of d.*

Corollary 5.

$$\min_d P = \lim_{d\to 0} P = \frac{1}{n}, \quad \max_d P = \lim_{d\to+\infty} P = 1 \tag{11}$$

Table 2: Two examples' vulnerability measured by F_2 as defined in Equation (3).

Label	Sentence	Vulnerability
+	S1: Why are people supporting Donald Trump? S2: Why do you think people are supporting Trump?	1.9
−	S3: What is the difference between the Moto2 and the Moto3 race? S4: Who is ahead in the race to sell self-driving cars?	0.1

4 Experiments

4.1 Model A and Datasets

We adapt VAA framework to different neural models on various datasets to verify the generalization ability of VAA. ESIM (Chen et al., 2017) and BERT (Devlin et al., 2019) are chosen as model A in Figure 1. We test VAA on some publicly available NLSM datasets, including Quora Question Pairs (QQP)[1], SNLI (Bowman et al., 2015), and MultiNLI (Williams et al., 2018). We use the same QQP dataset partition as (Wang et al., 2017)[2].

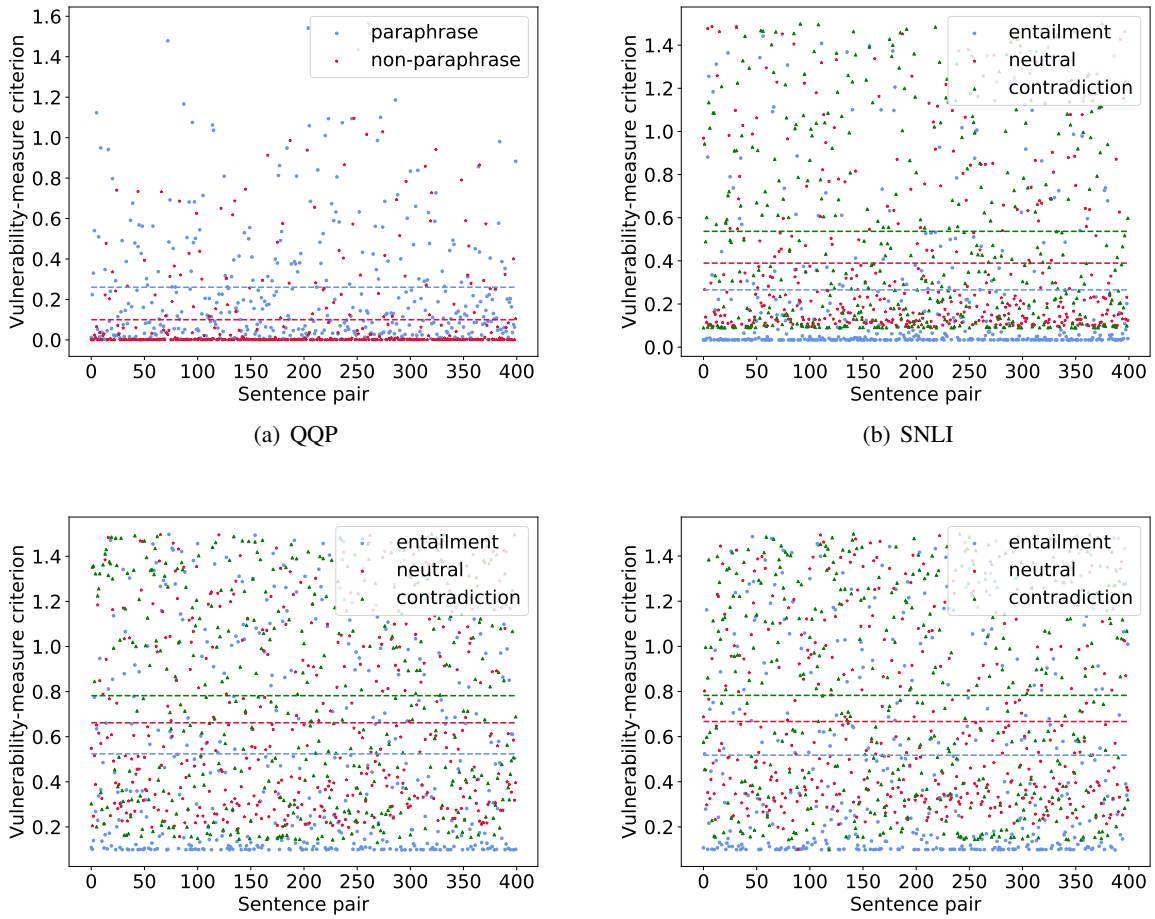
4.2 Vulnerability Analysis

4.2.1 Case Study and Vulnerability Visualization

The two examples in Table 1 are used to do a case study. Table 2 shows that vulnerability of the `paraphrase` pair is larger than that of the `non-paraphrase` pair. In Figure 2, we illustrate the vulnerability of samples in QQP, SNLI, and MultiNLI. Figure 2 shows that different categories of samples have different vulnerabilities in all these datasets. For the QQP dataset, `paraphrase` pairs have larger vulnerability than `non-paraphrase` pairs. As for SNLI and MultiNLI, both `contradiction` and `neutral` pairs have larger vulnerability than `entailment` pairs. Figure 2 also shows that the vulnerability difference between categories in QQP is more obvious than that in SNLI and MultiNLI.

[1]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[2]This QQP dataset partition is available at https://zhiguowang.github.io.



(a) QQP (b) SNLI

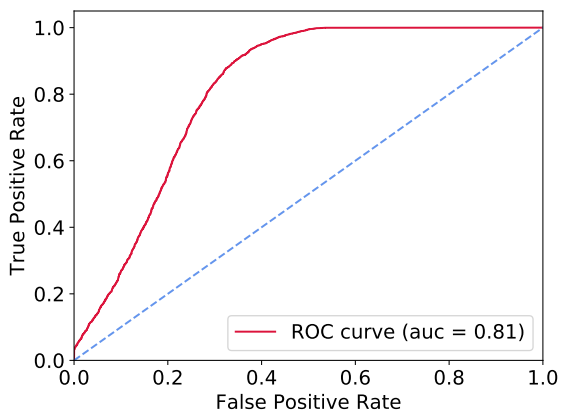
(c) MultiNLI (Matched) (d) MultiNLI (Mismatched)

Figure 2: Vulnerability visualization of samples in QQP, SNLI, and MultiNLI. The colored horizontal line represents the average vulnerability of the corresponding category. The vulnerability is measured by F_2 with the L_2-norm ($\epsilon = 0.02$) FGSM attack.

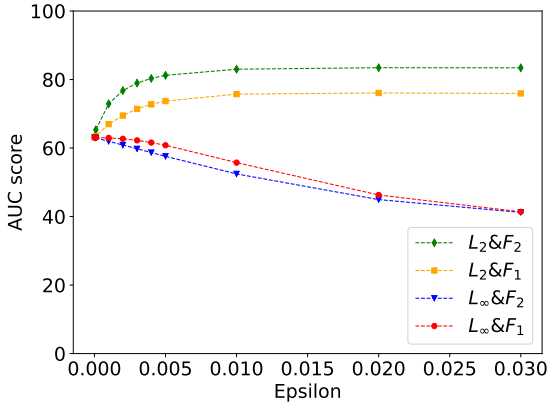
4.2.2 Vulnerability Effectiveness and Hyper-parameters Analysis

Different categories of samples have different vulnerabilities. So the vulnerability can be considered as a feature, which can be used to predict the label of the sample. We predict the label only with the vulnerability feature to demonstrate the effectiveness of vulnerability. With the prediction results, we can plot the receiver operating characteristic (ROC) curve and calculate the AUC score. We conduct the experiments on the QQP dataset which is taken as an example. Figure 3 (a) shows the ROC curve with F_2 of the L_2-norm FGSM attack, where ϵ is 0.02. The vulnerability is measured with F_2, and we predict the label only with the numeric value F_2. $\mathrm{AUC} = 0.81$ is really high, demonstrating the effectiveness of vulnerability.

To achieve better performance with the vulnerability, we should find more suitable hyper-parameters. Vulnerability-measure criterion F, perturbed norm L, and attack strength ϵ are all hyper-parameters. With different hyper-parameters setting, we will get different AUC scores. Figure 3 (b) shows the relationship between the AUC score and these hyper-parameters. From Figure 3 (b), we can know the optimal hyper-parameters setting. Figure 3 (a) and VAA framework adopt the optimal hyper-parameters setting.



(a) ROC curve with F_2 of the L_2-norm FGSM attack, where ϵ is 0.02. AUC $= 0.81$ is really high, demonstrating the effectiveness of vulnerability.



(b) The relationship between the AUC score and vulnerability-measure criterion F, perturbed norm L, attack strength ϵ. F can choose F_1 and F_2, as defined in Equation (3). L can choose L_∞ and L_2, as shown in Equations (4) and (5), respectively.

Figure 3: Vulnerability effectiveness and hyper-parameters analysis on the QQP dataset. We classify a sentence pair only with its vulnerability feature.

Table 3: Classification accuracy on the dev/test set of QQP, SNLI, and MultiNLI. We re-implement the ESIM and $\text{BERT}_{\text{BASE}}$, then report the results. The best result in each scenario is indicated in **bold**.

Model	QQP		SNLI		MultiNLI			
					Matched		Mismatched	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test
ESIM	86.36	86.00	87.50	86.81	73.03	72.81	73.14	72.56
ESIM+VAA	**87.30**	**87.16**	87.49	**87.05**	72.83	**73.09**	**73.43**	**72.62**
$\text{BERT}_{\text{BASE}}$	88.22	88.24	87.90	87.93	78.08	78.35	78.47	78.40
$\text{BERT}_{\text{BASE}}$+VAA	**89.62**	**89.64**	87.81	**88.10**	**78.71**	**78.79**	**78.93**	**79.14**

4.3 VAA Framework Analysis

4.3.1 VAA Framework Settings and Results

For convenience, both ESIM and $\text{BERT}_{\text{BASE}}$ are in their default settings. For the comparability of results, the method for generating adversarial embedding $\tilde{E}$ is the L_2-norm FGSM attack with $\epsilon = 0.02$, as shown in Equation (5), for all neural models and datasets.

Zhang et al. (2020) summarized the performances of previous works on these datasets. Compared with the small performance gain of each previous work, VAA significantly improves the performances of neural models on most datasets, as shown in Table 3. In particular, for the QQP dataset, VAA increases the accuracy on the dev/test sets by 1.23% on average. Table 3 is consistent with Corollary 2; VAA achieves better performance gain on QQP ($n = 2$) than on SNLI ($n = 3$) and MultiNLI ($n = 3$). n is the number of labels, as shown in Section 3.3. From Section 4.2.1, we know that the vulnerability difference between categories in QQP is more obvious than that in SNLI and MultiNLI. d represents the vulnerability difference between the n categories, so the d of QQP is larger than that of SNLI and MultiNLI. Thus, Table 3 is also consistent with Corollary 4, and VAA achieves better performance gain on QQP than on SNLI and MultiNLI.

4.3.2 Two-stage Training

Joint training models A and B is not adopted. Instead, we adopt two-stage training and fine-tune only model B in stage two for the following three reasons.

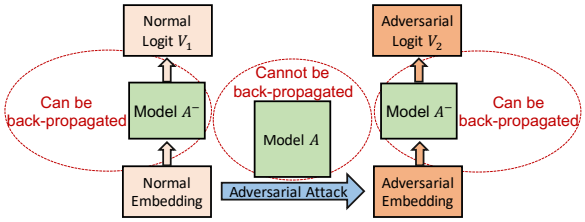


Figure 4: Back-propagation ability of model A in stage two.

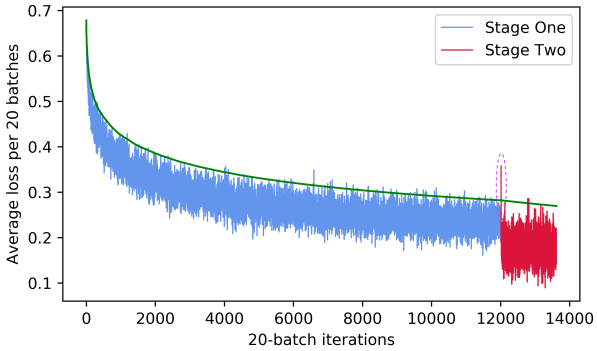


Figure 5: Loss in the two-stage training of VAA framework. To make the curve smoother, we average the loss as a value for every 20 batches.

1. It will reduce the accuracy to fine-tune many pre-trained word embeddings models, such as ELMo, GPT, and BERT, on a specific dataset. Similarly, fine-tuning model A in stage two will lead to overfitting the training data and reducing accuracy.

2. As shown in Figure 4, for each batch, we need to run model A three times in stage two to get the normal logit V_1, adversarial embedding $\tilde{E}$, and adversarial logit V_2, respectively. Further, the process of getting $\tilde{E}$ is an independent adversarial attack that cannot be back-propagated. Fine-tuning model A only optimizes the process of getting V_1 and V_2. The adversarial embedding $\tilde{E}$ will introduce more uncertainty owing to the partial optimization. As a result, adversarial logit V_2, vectors V_3 and V will also introduce more noise.

3. Fixing model A in stage two also speeds up the training process and thus saves computing resources.

To observe the change after incorporating the vulnerability, we record the average training loss per 20 batches, as shown in Figure 5. In stage one, the optimizer converges to a local minimum. After incorporating the vulnerability, the loss increases immediately because adding the vulnerability to the sample destroys the original feature space of the sample. However, after several iterations, the loss rapidly declines to a more optimized area than in stage one. It indicates that the vulnerability significantly improves the performance of neural models.

4.4 Difference Between VAA and Adversarial Training

Both VAA and adversarial training are methods that incorporate the vulnerability into the training process. Adversarial training (Goodfellow et al., 2014; Kurakin et al., 2016) is one of the most popular defense mechanisms in the current studies. The idea is to inject adversarial samples into the training process and train the model on a mix of normal samples and adversarial samples. The goal of adversarial training is to improve the accuracy on adversarial samples. However, adversarial training tends to overfit the specific attack used at training time, which causes the accuracy to decline on normal samples.

However, the goal of VAA is different from that of adversarial training. VAA uses the vulnerability to improve the accuracy on normal samples rather than adversarial samples. The methods of using the vulnerability are different as well. VAA encodes the vulnerability into the inputs, while adversarial training works by injecting adversarial samples into the training process.

5 Conclusion

In this paper, we first found a phenomenon that different categories of samples have different vulnerabilities. Besides, we defined criteria to measure the vulnerability of a sample. We proposed a general two-stage training framework called VAA to incorporate the vulnerability. VAA can significantly enhance the performance of neural models on NLSM datasets. Furthermore, we closely examined the factors that influence vulnerability effectiveness theoretically and experimentally.

In future work, there remains scope to improve VAA further. First, the vulnerability is currently obtained via adversarial attack. There may be some other methods that can better capture the vulnerability, e.g., integrating external knowledge vulnerability between phrase pairs. Second, VAA works by encoding the vulnerability into the inputs. We should explore some other methods to incorporate the vulnerability.

Acknowledgements

First of all, we would like to thank Zhiyuan Liu, Wei Tan, and Bizheng Wang for their instructive advice and useful suggestions on this paper. This work is supported by grants from the National Key Research and Development Program of China (2017YFB1400401, 2017YFC0803609).

References

Shumeet Baluja and Ian Fischer. 2017. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2017. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv: Machine Learning*.

Jane Bromley, Isabelle Guyon, Yann Lecun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition Artificial Intelligence*, 7(04):669–688.

Nicholas Carlini and David Wagner. 2016. Towards evaluating the robustness of neural networks.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *Computer Science*.

Zhilong Hong, Qifei Zhou, Rong Zhang, Weiping Li, and Tong Mo. 2020. Legal feature enhanced semantic matching network for similar case matching. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Weiwei Hu and Ying Tan. 2018. Black-box attacks against rnn based malware detection algorithms. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Nicolas Papernot, Patrick D Mcdaniel, Somesh Jha, Matthew Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016a. The limitations of deep learning in adversarial settings. *ieee european symposium on security and privacy*, pages 372–387.

Nicolas Papernot, Patrick D Mcdaniel, Ananthram Swami, and Richard E Harang. 2016b. Crafting adversarial input sequences for recurrent neural networks. *military communications conference*, pages 49–54.

Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. 2018. Query-efficient gan based black-box attack against sequence based machine and deep learning classifiers. *arXiv: Cryptography and Security*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4345–4352. AAAI Press.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *Computer Science*.

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway attention networks for modeling sentence pairs. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4411–4417. AAAI Press.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1340–1349.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144–4150. AAAI Press.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Rong Zhang, Qifei Zhou, Bo Wu, Weiping Li, and Tong Mo. 2020. What do questions exactly ask? mfae: Duplicate question identification with multi-fusion asking emphasis. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 226–234. SIAM.

Qifei Zhou, Rong Zhang, Bo Wu, Weiping Li, and Tong Mo. 2020. Detection by attack: Detecting adversarial samples by undercover attack. In *European Symposium on Research in Computer Security*, pages 146–164. Springer.

A Proof of Theorem 1

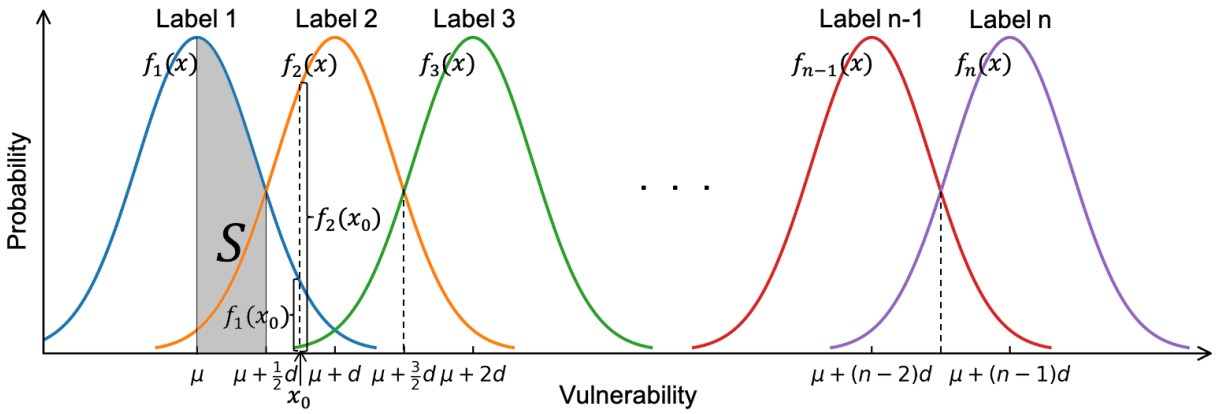


Figure 6: Hypothetical normal distribution of vulnerability. The horizontal axis represents the vulnerability, which can be measured by F_1 or F_2 as defined in Equation (3). The vertical axis represents the probability that a sample has the vulnerability.

Proof. Many random variables obey normal distribution in the real world; hence, the normal distribution assumption in Theorem 1 is reasonable. $X_1 \sim \mathcal{N}(\mu, \sigma^2)$; hence, the density function $f_1(x)$ is:

$$f_1(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{12}$$

In Figure 6, we consider a sample that has vulnerability x_0. We want to predict the label of the sample only with the vulnerability x_0. The probability that the sample has label 1 is $f_1(x_0)$. The probability that the sample has label 2 is $f_2(x_0)$. $f_1(\cdot)$ and $f_2(\cdot)$ are the density functions of X_1 and X_2, respectively. From Figure 6, we can find that $f_1(x_0) < f_2(x_0)$. We suppose that the model predicts the label only with the vulnerability. Hence the model should predict that the sample has label 2. Then, the probability of making the right prediction is $\frac{f_2(x_0)}{f_1(x_0)+f_2(x_0)}$. Through generalization to all samples, we can compute the model accuracy P.

$$P = \frac{S_{\text{upper}}}{S_{\text{all}}} \tag{13}$$

where S_{upper} is the area under the upper curve in Figure 6, and S_{all} is the total area under all the distribution curves. Now, we have to compute S_{upper} and S_{all}.

$X_1 \sim \mathcal{N}(\mu, \sigma^2)$; hence, we have Equation (14) and Equation (15).

$$\int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = 1 \tag{14}$$

$$\int_{-\infty}^{0} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = 0.5 \tag{15}$$

All X_i obey normal distribution, so we can calculate S_{all} with Equation (14).

$$S_{\text{all}} = n \tag{16}$$

Let

$$S \coloneqq \int_{\mu}^{\mu+\frac{d}{2}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \tag{17}$$

All X_i have the same variance σ^2 and the means form a tolerance of d arithmetic progression, so we have:

$$\begin{aligned} &\int_{\mu-\frac{d}{2}+kd}^{\mu+kd} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \\ =&\int_{\mu+kd}^{\mu+kd+\frac{d}{2}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \\ =&\int_{\mu}^{\mu+\frac{d}{2}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = S, \forall k \in [0, \ldots, n-1] \end{aligned} \tag{18}$$

From Figure 6, we can calculate S_{upper} with Equations (15), (17), and (18).

$$S_{\text{upper}} = (0.5 + S) \times 2 + 2S \times (n-2) \tag{19}$$

With Equations (13), (16), and (19), we can have the model accuracy P.

$$P = \frac{1-2S}{n} + 2S \tag{20}$$

□

B Proof of Corollary 2

Proof. With Figure 6, Equations (15) and (17), we know that $S < 0.5$, then $1 - 2S > 0$. With Equation (9), we know that P is a monotonically decreasing function of n. □

C Proof of Corollary 3

Proof. With Corollary 2 and Equation (9), we can have Equations (21) and (22).

$$\min_n P = \lim_{n\to+\infty} P = \lim_{n\to+\infty} \frac{1-2S}{n} + 2S = 2S \tag{21}$$

n is the number of labels; hence, $n \geq 2$. Then

$$\max_n P = P|_{n=2} = \frac{1-2S}{2} + 2S = S + \frac{1}{2} \tag{22}$$

□

D Proof of Corollary 4

Proof. With Equation (17), we know that S is a monotonically increasing function of d.

$$P = \frac{1-2S}{n} + 2S = (2 - \frac{2}{n})S + \frac{1}{n} \tag{23}$$

n is the number of labels; hence, $n > 1$. Then, $2 - \frac{2}{n} > 0$; hence, P is a monotonically increasing function of S. Thus, P is a monotonically increasing function of d. □

E Proof of Corollary 5

Proof.

$$\lim_{d\to 0} S = \lim_{d\to 0} \int_{\mu}^{\mu+\frac{d}{2}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = 0 \tag{24}$$

$$\begin{aligned} \lim_{d\to+\infty} S &= \lim_{d\to+\infty} \int_{\mu}^{\mu+\frac{d}{2}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \\ &= \int_{\mu}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = 0.5 \end{aligned} \tag{25}$$

With Corollary 4, Equations (24) and (25), we have:

$$\min_d P = \lim_{d\to 0} P = \lim_{d\to 0} \frac{1-2S}{n} + 2S = \frac{1}{n} \tag{26}$$

$$\max_d P = \lim_{d\to+\infty} P = \lim_{d\to+\infty} \frac{1-2S}{n} + 2S = 1 \tag{27}$$

□

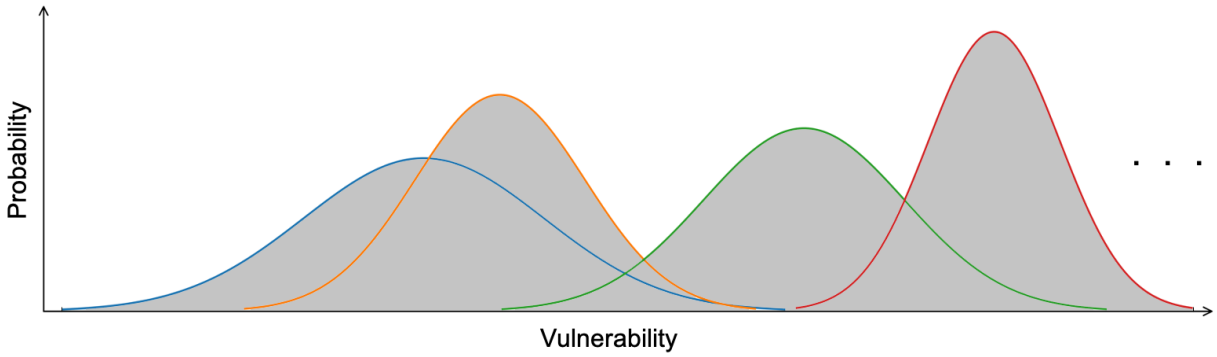


Figure 7: General distribution of vulnerability.

F General Distribution

If we discard the assumption of the normal distribution in Theorem 1, we can also have Equation (13), which can be established for any distribution. Figure 7 shows an example of a general distribution. $S_{\mathrm{all}} = n$ is always established. For the convenience of calculating S_{upper}, we assume the normal distribution, and we have Theorem 1.