Defending Textual Neural Networks against Black-Box Adversarial Attacks with Stochastic Multi-Expert Patcher

Anonymous ACL submission

Abstract

Even though several methods have proposed to defend textual neural network (NN) models against black-box adversarial attacks, they often defend against a specific text perturbation strategy and/or require re-training the models from scratch. This leads to a lack of generalization in practice and redundant computation. In particular, the state-of-the-art transformer models (e.g., BERT, RoBERTa) require great time and computation resources. By borrowing an idea from *software engineering*, in order to address these limitations, we propose a novel algorithm, SHIELD, which modifies and re-trains only the last layer of a textual NN, and thus it "patches" and "transforms" the NN into a stochastic weighted ensemble of multi-expert prediction heads. Considering that most of current black-box attacks rely on iterative search mechanisms to optimize their adversarial perturbations, SHIELD confuses the attackers by automatically utilizing different weighted ensembles of predictors depending on the input. In other words, SHIELD breaks a fundamental assumption of the attack, which is a victim NN model remains constant during an attack. By conducting comprehensive experiments, we demonstrate that all of CNN, RNN, BERT, and RoBERTa-based textual NNs, once patched by SHIELD, exhibit a relative enhancement of 15%–70% in accuracy on average against 14 different black-box attacks, outperforming 6 defensive baselines across 3 public datasets. All codes are to be released.

1 Introduction

Adversarial Text Attack and Defense. After being trained to maximize prediction performance, textual NN models frequently become vulnerable to adversarial attacks (Papernot et al., 2016; Wang et al., 2019a). In the NLP domain, in general, adversaries utilize different strategies to perturb an input sentence such that its semantic meaning is preserved while successfully letting a target NN model output a desired prediction. Text perturbations are typically generated by replacing or inserting critical words (e.g., HotFlip (Ebrahimi et al., 2018), TextFooler (Jin et al., 2019)), characters (e.g., DeepWordBug (Gao et al.), TextBugger (Li et al., 2018)) in a sentence or by manipulating a whole sentence (e.g., SCPNA (Iyyer et al., 2018), GAN-based(Zhao et al., 2018)).

Since many recent NLP models are known to be vulnerable to adversarial black-box attacks (e.g., fake news detection (Le et al., 2020; Zhou et al., 2019b), dialog systems (Cheng et al., 2019), and so on), robust defenses for textual NN models are required. Even though several papers have proposed to defend NNs against such attacks, they were designed for either a specific type of attack (e.g., word or synonym substitution (Wang et al., 2021; Dong et al., 2021; Mozes et al., 2020; Zhou et al., 2021), misspellings (Pruthi et al., 2019), character-level (Pruthi et al., 2019), or word-based (Le et al., 2021)). Even though there exist some general defensive methods, most of them enrich NN models by re-training them with adversarial data augmented via known attack strategies (Miyato et al., 2016; Liu et al., 2020; Pang et al., 2020) or with external information such as knowledge graphs (Li and Sethy, 2019).

However, these augmentations often induce substantial overhead in training or are still limited to only a small set of predefined attacks (e.g., (Zhou et al., 2019a)). Hence, we are in search of defense algorithms that directly enhance NN models' structures (e.g., (Li and Sethy, 2019)) while achieving higher generalization capability without the need of acquiring additional data.

Motivation (Fig. 1). Different from white-box attacks, black-box attacks do not have access to a target model's parameters, which are crucial for achieving effective attacks. Hence, attackers often query the target model repeatedly to acquire the

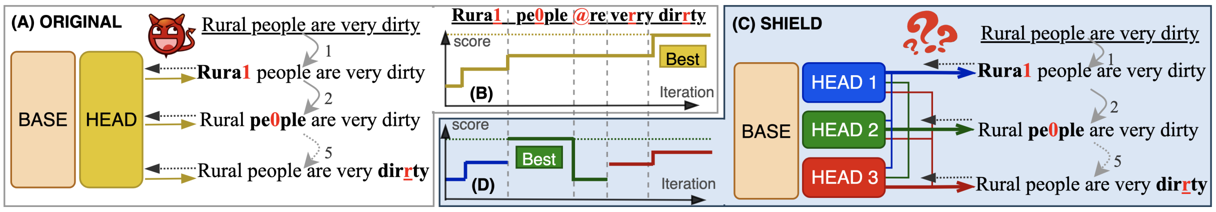
(A) ORIGINAL
Rural people are very dirty
Rura1 people are very dirty
Rural pe0ple are very dirty
Rural people are very dirrty
BASE
HEAD
Rura1 pe0ple @re verry dirrty
score
Best
(B)
Iteration
score
Best
(D)
Iteration
(C) SHIELD
Rural people are very dirty
HEAD 1
HEAD 2
HEAD 3
Rura1 people are very dirty
Rural pe0ple are very dirty
Rural people are very dirrty

Figure 1: Motivation of SHIELD: An attacker optimizes a step objective function (score) to search for the best perturbation by iteratively replacing each of the original 5 tokens with a perturbed one. **(A)** The attacker assumes the model *remains unchanged* and **(B)** gives coherent signal during the iteration search, resulting in the true *best* attack: "dirty"→"dirrty". **(C)** A model patched with SHIELD utilizes a weighted ensemble of 3 diverse heads depending on the input. Therefore, the ensemble weights keep changing over time during adversaries' perturbation search processes – the line width represents the ensemble weights. **(D)** SHIELD confuses the attacker with 3 *varying* distributions of the score, resulting in a sub-optimal attack "people"→"pe0ple".

necessary information for optimizing their strategy. From our analyses of 14 black-box attacks published during 2018–2020 (Table 1), all of them, except for *SCPNA* (Iyyer et al., 2018), rely on a searching algorithm (e.g., greedy, genetic) to iteratively replace each character/word in a sentence with a perturbation candidate to optimize the choice of characters/words and how they should be crafted to attack the target model (Fig. 1A). Even though this process is effective in terms of attack performance, they assume that the model's parameters remain "unchanged" and the model outputs "coherent" signals during the iterative search (Fig. 1A and 1B). Our key intuition is, however, *to obfuscate the attackers by breaking this assumption*. Specifically, we want to develop an algorithm that automatically utilizes a diverse set of models during inference. This can be done by training multiple sub-models instead of a single prediction model and randomly select one of them during inference to obfuscate the iterative search mechanism. However, this then introduces impractical computational overhead during both training and inference, especially when one wants to maximize prediction accuracy by utilizing complex SOTA sub-models such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b). Moreover, it also does not guarantee that trained models are sufficiently diverse to fool attackers. Furthermore, applying this strategy to existing NN models would also require re-training everything from the scratch, rendering the approach impractical.

Proposal. To address these challenges, we borrow ideas from *software engineering* where bugs can be readily removed by an external installation patch. Specifically, we develop a novel neural patching algorithm, named as SHIELD, which patches only the last layer of an already deployed textual NN

Attack Method	**Search Method**	**Atk Level**	**Sem. Presv.**	**Natr. Presv.**
SCPNA Iyyer et al.	TP	SN	✓	✓
TextBugger(TB) Li et al.	GD	CR	✓	
DeepWordBug(DW) Gao et al.	GD	CR	✓	
Kuleshov Kuleshov et al.	GD	WD	✓	✓
TextFooler(TF) Jin et al.	GD	WD	✓	
IGA Wang et al.	GN	WD		
Pruthi Pruthi et al.	GD	CR		
PWWS(PS) Ren et al.	GD	WD		
Alzantot Jia et al.	GN	WD		✓
BAE Garg and Ramakrishnan	GD	WD	✓	
BERT-Atk(BERTK) Li et al.	GD	WD	✓	
PSO Zang et al.	GN	WD		
Checklist Ribeiro et al.	GD	WD		
Clare Li et al.	GD	WD	✓	✓

TP: Template; GD: Greedy; GN: Genetics
CR: Character; WD: Word; SN: Sentence

Table 1: Different attack methods with i) how they search for adversarial perturbations, ii) their attack level, and iii) whether they maintain the original semantics (*Sem. Presv.*), pursue the naturalness of the perturbed sentence (*Natr. Presv.)*, or both of them.

model (e.g., CNN, RNN, transformers(Vaswani et al., 2017; Bahdanau et al.)) and transforms it into an ensemble of multi-experts or *prediction heads* (Fig. 1C). During inference, then SHIELD automatically utilizes a stochastic weighted ensemble of experts for prediction depending on inputs. This will obfuscate adversaries' perturbation search, making black-box attacks much more difficult regardless of attack types, e.g., character or word level attacks (Fig. 1C,D). By patching only the last layer of a model, SHIELD also introduces lightweight computational overhead and requires no additional training data. In summary, our contributions are as follows:

- We propose SHIELD, a novel neural patching algorithm that transforms a already-trained NN model to a *stochastic* ensemble of multi-experts with little computational overhead.

- We demonstrate the effectiveness of SHIELD. CNN, RNN, BERT, and RoBERTa-based textual models patched by SHIELD achieve an increase of 15%–70% on their robustness across 14 different black-box attacks, outperforming 6 defensive baselines on 3 public NLP datasets.
- To the best of our knowledge, this work by far includes the most comprehensive evaluation for the defense against black-box attacks.

2 The Proposed Method: SHIELD

We introduce *Stochastic Multi-Expert Neural Patcher* (SHIELD) which patches *only* the last layer of an already *trained* NN model $f(\mathbf{x}, \theta)$ and transforms it into an ensemble of multiple expert predictors with stochastic weights. These predictors are designed to be strategically selected with different weights during inference depending on the input. This is realized by two complementary modules, namely (i) a *Stochastic Ensemble (SE)* module that transforms $f(\cdot)$ into a randomized ensemble of different heads and (ii) a *Multi-Expert (ME)* module that uses Neural Architecture Search (NAS) to dynamically learn the optimal architecture of each head to promote their diversity.

2.1 A Stochastic Ensemble (SE) Module

This module extends the last layer of $f(\cdot)$, which is typically a fully-connected layer (followed by a softmax for classification), to an ensemble of K *prediction heads*, denoted $\mathcal{H}{=}\{h(\cdot)\}_j^K$. Each head $h_j(\cdot)$, parameterized by θ_{h_j}, is an expert predictor that is fed with a feature representation learned by *up to the second-last layer* of $f(\cdot)$ and outputs a prediction logit score:

$$h_j : f(\mathbf{x}, \theta^*_{L-1}) \in \mathbb{R}^Q \mapsto \tilde{y}_j \in \mathbb{R}^M, \qquad (1)$$

where θ^*_{L-1} are *fixed* parameters of f up to the second-last layer, Q is the size of the feature representation of $\mathbf{x}$ generated by the base model $f(\mathbf{x}, \theta^*_{L-1})$, and M is the number of labels. To aggregate all logit scores returned from all heads, then, a classical ensemble method would average them as the final prediction: $\hat{y}^*{=}\frac{1}{K}\sum_j^K \tilde{y}_j$. However, this simple aggregation assumes each $h_j(\cdot) \in \mathcal{H}$ learns from very similar training signals. Hence, when θ^*_{L-1} already learns some of the task-dependent information, $\mathcal{H}$ will eventually converge *not* to a set of experts but very similar predictors. To resolve this issue, we introduce stochasticity into the process by assigning *prediction heads with stochastic weights* during both training and inference. Specifically, we introduce a new aggregation mechanism:

$$\hat{y} = \frac{1}{K}\sum_j^K \alpha_j w_j \tilde{y}_j, \qquad (2)$$

where w_j weights $\tilde{y}_j$ according to head j's expertise on the current input $\mathbf{x}$, and $\alpha_j \in [0, 1]$ is a probabilistic scalar, representing how much of the weight w_j should be accounted for. Let us denote w, $\alpha \in \mathbb{R}^K$ as vectors containing all scalars w_j and α_j, respectively, and $\tilde{\mathbf{y}} \in \mathbb{R}^{(K\times M)}$ as the concatenation of all vectors $\tilde{y}_j$ returned from each of the heads. We calculate w and α as follows:

$$w = \mathbf{W}^T(\tilde{\mathbf{y}} \oplus f(\mathbf{x}, \theta^*_{L-1})) + \mathbf{b}, \qquad (3)$$

$$\alpha = \text{softmax}((w + \mathbf{g})/\tau), \qquad (4)$$

where $\mathbf{W} \in \mathbb{R}^{(K\times M+Q)\times K}$, $\mathbf{b} \in \mathbb{R}^K$ are trainable parameters, $\mathbf{g} \in \mathbb{R}^K$ is a noise vector sampled from the *Standard Gumbel Distribution* and therefore, probability vector α is sampled by a technique known as *Gumbel-Softmax* (Jang et al., 2016) controlled by the noise vector $\mathbf{g}$ and the inverse-temperature τ. Unlike the standard Softmax, the Gumbel-Softmax is able to learn a categorical distribution (over K heads) optimized for a downstream task (Jang et al., 2016). Annealing $\tau{\to}0$ encourages a pseudo one-hot vector (e.g., [0.94, 0.03, 0.01, 0.02] when $K{=}4$), which makes Eq. (2) a mixture of experts (Avnimelech and Intrator, 1999). Importantly, α is sampled in an inherently stochastic way depending on the gumbel noise $\mathbf{g}$.

While $\mathbf{W}, \mathbf{b}$ is learned to *deterministically* assigns more weights w to heads that are experts for each input $\mathbf{x}$ (Eq. (3)), α introduces *stochasticity* into the final logits. The multiplication of $\alpha_j w_j$ in Eq. (2) then enables us to use different sets of weighted ensemble models *while still maintaining the ranking of the most important head*. Thus, this further diversifies the learning of each expert and confuse attackers when they iteratively try different inputs to find good adversarial perturbations.

Finally, to train this module, we use Eq. (2) as the final prediction and train the whole module with *Negative Log Likelihood (NLL)* loss following the objective:

$$\min_{\mathbf{W},\mathbf{b},\{\theta_h\}_j^K} \mathcal{L}_{\text{SE}} = -\frac{1}{N}\sum_i^N y_i log(\text{softmax}(\hat{y}_i)). \qquad (5)$$

Algorithm 1 Training **SHIELD** Algorithm.

```
1: Input: pre-trained neural network $f(\cdot)$
2: Input: $\mathcal{O}, K, \tau, \gamma$
3: Initialize $\mathbf{W}, \mathbf{b}, \theta_{\mathcal{O}}, \{\beta\}_j^K$
4: repeat
5:     Freeze $\{\beta\}_j^K$ and optimize $\mathbf{W}, \mathbf{b}, \theta_{\mathcal{O}}$ via Eq. (5) in
       mini-batch from train set.
6:     Freeze $\mathbf{W}, \mathbf{b}, \theta_{\mathcal{O}}$ and optimize $\{\beta\}_j^K$ via Eq. (8) with
       $\gamma$ multiplier in mini-batch from validation set.
7: until convergence
```

2.2 A Multi-Expert (ME) Module

While the *SE* module facilitates stochastic weighted ensemble among heads, the *ME* module searches for the optimal architecture for each head that maximizes the diversity in how they make predictions. To do this, we utilize the DARTS algorithm (Liu et al., 2019a) as follows. Let us denote $\mathcal{O}_j = \{o_j(\cdot)\}_t^T$ where T is the number of possible architectures to be selected for $h_j \in \mathcal{H}$. We want to learn a one-hot encoded *selection* vector $\beta_j \in \mathbb{R}^T$ that assigns $h_j(\cdot) \leftarrow o_{j,\mathrm{argmax}(\beta_j)}(\cdot)$ during prediction. Since $\mathrm{argmax}(\cdot)$ operation is not differentiable, during training, we relax the categorical assignment of the architecture for $h_j(\cdot) \in \mathcal{H}$ to a softmax over all possible networks in $\mathcal{O}_j$:

$$h_j(\cdot) \longleftarrow \frac{1}{T}\sum_t^T \frac{\exp(\beta_j^t)}{\sum_t^T \exp(\beta_j^T)} o_{j,t}(\cdot). \quad (6)$$

However, the original DARTS algorithm *only* optimizes prediction performance. In our case, we also want to promote the diversity among heads. To do this, we force each $h_j(\cdot)$ to specialize in different features of an input, i.e., in how it makes predictions. This can be achieved by *maximizing* the difference among the gradients of the word-embedding $\mathbf{e}_i$ of input $\mathbf{x}_i$ w.r.t to the outputs of each $h_j(\cdot) \in \mathcal{H}$. Hence, given a fixed set of parameters $\theta_{\mathcal{O}}$ of all possible networks for every heads, we train all selection vectors $\{\beta\}_j^K$ by optimizing the objective:

$$\mathrm{minimize}_{\{\beta\}_j^K} \mathcal{L}_{\mathrm{experts}} = \sum_i^N \sum_{n<m}^K \Big(\mathrm{d}(\nabla_{\mathbf{e}_i}\mathcal{J}_n; \nabla_{\mathbf{e}}\mathcal{J}_m) - ||\nabla_{\mathbf{e}}\mathcal{J}_n - \nabla_{\mathbf{e}}\mathcal{J}_m||_2^2 \Big), \quad (7)$$

where $\mathrm{d}(\cdot)$ is the cosine-similarity function, and $\mathcal{J}_j$ is the NLL loss as if we only use a single prediction head h_j. In this module, however, not only do we want to maximize the differences among gradients vectors, but also we want to ensure the selected architectures eventually converge to good prediction performance. Therefore, we train the whole *ME* module with the following objective:

$$\mathrm{minimize}_{\{\beta\}_j^K} \mathcal{L}_{ME} = \mathcal{L}_{SE} + \gamma \mathcal{L}_{\mathrm{experts}}. \quad (8)$$

2.3 Overall Framework

To combine the *SE* and *ME* modules, we replace Eq. (6) into Eq. (1) and optimize the overall objective:

$$\begin{aligned} &\mathrm{minimize}_{\{\beta\}_j^K} \mathcal{L}_{\mathrm{ME}}^{\mathrm{val}} + \gamma \mathcal{L}_{\mathrm{experts}}^{\mathrm{val}} \quad \text{s.t.} \\ &\mathbf{W}, \mathbf{b}, \theta_{\mathcal{O}} = \mathrm{minimize}_{\mathbf{W},\mathbf{b},\theta_{\mathcal{O}}} \mathcal{L}_{\mathrm{SE}}^{\mathrm{train}}. \end{aligned} \quad (9)$$

We employ an *iterative* training strategy (Liu et al., 2019a) with the *Adam* optimization algorithm (Kingma and Ba, 2013) as in Alg. 1. By alternately freezing and training $\mathbf{W}, \mathbf{b}, \theta_{\mathcal{O}}$ and $\{\beta\}_j^K$ using a training set $\mathcal{D}_{\mathrm{train}}$ and a validation set $\mathcal{D}_{\mathrm{val}}$, we want to (i) achieve high quality prediction performance through Eq. (5) and (ii) select the optimal architecture for each expert to maximize their specialization through Eq. (7).

	#Class	#Vocab	#Example
MR (Pang and Lee, 2005)	2	19K	11K
CB (Anand et al., 2017)	2	25K	32K
HS (Davidson et al.)	3	35K	25K

Table 2: Statistics of experimental datasets.

3 Experimental Evaluation

3.1 Set-up

Datasets & Metric. Table 2 shows the statistics of all experimental datasets: Clickbait detection (CB) (Anand et al., 2017), Hate Speech detection (HS) (Davidson et al.) and Movie Reviews classification (MR) (Pang and Lee, 2005). We split each dataset into *train*, *validation* and *test* set with the ratio of 8:1:1 whenever standard public splits are not available. To report prediction performance on clean examples, we use the *weighted F1* score to take the distribution of prediction labels into consideration. To report the robustness, we report prediction *accuracy* under adversarial attacks (Morris et al., 2020), i.e., # of failed attacks over total # of examples. A failed attack is only counted when the attacker fails to perturb (i.e., fail to flip the label of a *correctly predicted* clean example).

Defense Baselines. We want to defend four textual NN models (base models) of different architectures, namely RNN with GRU cells (Chung et al.), *transformer*-based BERT (Devlin et al., 2019) and

RoBERTa (Liu et al., 2019b). We compare SHIELD with the following six defensive baselines:

- *Ensemble (Ens.)* is the classical ensemble of 5 different *base models*. We use the average of all NLL losses from the base models as the final training loss.
- *Diversity Training (DT)* (Kariyappa and Qureshi, 2019) is a variant of the *Ensemble* baseline where a regularization term is added to maximize the coherency of gradient vectors of the input text w.r.t each sub-model. DT diversifies the feature-level expertise among heads.
- *Adaptive Diversity Promoting (ADP)* (Pang et al., 2019) is a variant of *Ensemble* baseline where a regularization term is added to maximize the diversity among non-maximal predictions of individual sub-models. ADP diversifies the class-level expertise among heads.
- *Mixup Training (Mixup)* (Zhang et al., 2018; Si et al.) trains a *base model* with data constructed by linear interpolation of two random training samples. In this work, we use *Mixup* to regularize a NN to adapt linear transformation in-between the continuous embeddings of training samples.
- *Adversarial Training (AdvT)* (Miyato et al., 2016) is a semi-supervised algorithm that optimizes the NLL loss on the original training samples plus adversarial inputs.
- *Robust Word Recognizer (ScRNN)* (Pruthi et al., 2019) detects and corrects potential adversarial perturbations or misspellings in a text before feeding it to the base model for prediction.

Note that due to the insufficient memory of GPU Titian Xp to simultaneously train several BERT and RoBERTa sub-models, we exclude *Ensemble*, *DT*, and *ADP* baseline for them.

Attacks. We comprehensively evaluate SHIELD under 14 different black-box attacks (Table 1). These attacks differ in their attack levels (e.g., character, word, sentence-based), optimization algorithms for searching adversarial perturbations (e.g., through fixed templates, greedy, genetic-based search). Apart from lexical constraints such as limiting # or % of words to manipulate in a sentence, ignoring stop-words, etc., many of them also preserve the semantic meanings of a generated adversarial text via constraining the *l2 distance*

Model/Dataset	MR	HS	CB	AVG
RNN	0.73	0.88	$\underline{0.97}$	0.86
+Ensemble	**0.80**	**0.90**	$\underline{0.97}$	**0.89**
+DT	**0.80**	0.86	$\underline{0.97}$	0.88
+ADP	**0.80**	0.88	$\underline{0.97}$	$\underline{0.88}$
+Mixup	0.77	0.87	$\underline{0.97}$	0.87
+AdvT	0.76	$\underline{0.89}$	**0.98**	0.88
+ScRNN	0.79	0.85	0.96	0.87
+SHIELD	0.78	0.86	$\underline{0.97}$	0.87 (↑1.3%)
BERT	0.84	$\underline{0.90}$	**1.00**	$\underline{0.91}$
+Mixup	0.81	0.89	0.99	0.90
+AdvT	$\underline{0.85}$	**0.91**	0.99	**0.92**
+ScRNN	0.83	0.90	0.99	0.91
+SHIELD	**0.86**	0.90	$\underline{0.99}$	$\underline{0.91}$ (0%)
RoBERTa	$\underline{0.88}$	0.89	**1.00**	$\underline{0.92}$
+Mixup	**0.88**	**0.91**	0.99	**0.93**
+AdvT	0.87	0.89	0.99	0.92
+ScRNN	0.88	$\underline{0.90}$	0.99	0.92
+SHIELD	0.88	0.89	$\underline{0.99}$	0.92 (0%)

Table 3: Prediction F1 on clean examples. On average, SHIELD is still able to maintain the original fidelity.

between its representation vector and that of the original text produced by either Universal Sentence Encoder (USE) (Cer et al., 2018) or GloVe embeddings (Pennington et al., 2014). Moreover, to ensure that the perturbed texts still look natural, a few of the attack methods employ an external pre-trained language model (e.g., BERT(Devlin et al., 2019), L2W (Holtzman et al., 2018)) to optimize the log-likelihood of the adversarial texts. Due to computational limit, we only compare SHIELD with other baselines in 3 representative attacks, namely *TextFooler* (Jin et al., 2019), *DeepWordBug* (Gao et al.) and *PWWS* (Ren et al., 2019). They are among the most effective attacks. To ensure fairness and reproducibility, we use the external *TextAttack* (Morris et al., 2020) and *OpenAttack* framework for adversarial text generation and evaluation.

Implementation. We train SHIELD of 5 experts (K=5) with γ=0.5 in all experiments. For each expert, we set $\mathcal{O}_j$ to 3 (T=3) possible architectures: FCN with 1, 2 and 3 hidden layer(s). For each dataset, we use *grid-search* and try all $\tau \in \{1.0, 0.1, 0.01, 0.001\}$ to search for the best pairs of τ used during training and inference according to their prediction accuracy and defense performance under TextFooler (Jin et al., 2019) and DeepWordBug (Gao et al.) attacks on the *validation* set. We use 10% of the training set as a separate development set during training with early-stop to prevent overfitting. We then report the performance of **the best single model** *across all attacks* on the test set.

Dataset	**Movie Reviews**						**Hate Speech**						**Clickbait**					
	RNN		**BERT**		**RoBERTa**		**RNN**		**BERT**		**RoBERTa**		**RNN**		**BERT**		**RoBERTa**	
Attack	**Bef.**	**Aft.**	**Bef.**	**Aft.**	**Bef.**	**Aft.**	**Bef.**	**Aft.**	**Bef.**	**Aft.**	**Bef.**	**Aft.**	**Bef.**	**Aft.**	**Bef.**	**Aft.**	**Bef.**	**Aft.**
SCPNA	0.37	0.32	0.27	0.24	0.27	**0.28**	0.51	**0.72**	0.25	**0.29**	0.23	**0.3**	0.51	0.5	0.44	**0.49**	0.4	**0.4**
TB	0.2	**0.32**	0.28	**0.37**	0.28	**0.5**	0.35	**0.61**	0.48	**0.59**	0.51	**0.6**	0.79	**0.86**	0.87	**0.93**	0.89	**0.94**
DW	0.2	**0.44**	0.27	**0.42**	0.16	**0.55**	0.27	**0.47**	0.27	**0.55**	0.41	**0.55**	0.67	**0.9**	0.58	**0.95**	0.68	**0.96**
Kuleshov	0.01	**0.12**	0.07	**0.22**	0.05	**0.28**	0.04	**0.18**	0.09	**0.28**	0.03	**0.25**	0.37	**0.71**	0.52	**0.88**	0.63	**0.9**
TF	0.03	**0.18**	0.08	**0.26**	0.05	**0.39**	0.08	**0.24**	0.25	**0.42**	0.12	**0.37**	0.31	**0.78**	0.44	**0.92**	0.5	**0.93**
IGA	0.05	**0.29**	0.16	**0.32**	0.13	**0.5**	0.16	**0.34**	0.27	**0.35**	0.24	**0.33**	0.6	**0.8**	0.79	**0.95**	0.77	**0.96**
Pruthi	0.53	**0.56**	0.48	**0.49**	0.54	**0.54**	0.59	**0.71**	0.45	**0.59**	0.53	**0.59**	0.94	0.92	0.96	0.95	0.96	**0.96**
PS	0.09	**0.3**	0.14	**0.35**	0.15	**0.45**	0.3	**0.54**	0.32	**0.43**	0.32	**0.44**	0.46	**0.85**	0.64	**0.94**	0.66	**0.94**
Alzantot	0.21	**0.36**	0.42	**0.47**	0.46	**0.64**	0.27	**0.54**	0.51	**0.57**	0.56	0.55	0.73	**0.83**	0.92	**0.97**	0.9	**0.98**
BAE	0.44	**0.54**	0.38	**0.46**	0.43	**0.57**	0.6	**0.72**	0.38	**0.52**	0.43	**0.51**	0.83	**0.92**	0.4	**0.81**	0.39	**0.92**
BERTK	0.01	**0.18**	0.04	**0.17**	0.03	**0.23**	0.1	**0.21**	0.36	**0.48**	0.22	**0.36**	0.18	**0.65**	0.25	**0.86**	0.41	**0.86**
PSO	0.05	**0.07**	0.14	0.12	0.07	**0.15**	0.35	**0.54**	0.38	**0.4**	0.35	**0.4**	0.6	**0.64**	0.75	**0.87**	0.71	**0.87**
Checklist	0.7	**0.76**	0.84	**0.85**	0.88	**0.88**	0.86	0.81	0.89	**0.89**	0.88	**0.88**	0.98	**0.98**	0.99	**1.0**	1.0	**1.0**
Clare	0.16	**0.35**	0.23	**0.28**	0.27	**0.54**	0.76	0.72	0.79	0.78	0.72	**0.76**	0.7	**0.87**	0.48	**0.86**	0.68	**0.94**
Average	0.27	**0.36**	0.27	**0.46**	0.37	**0.52**	0.41	**0.51**	0.4	**0.49**	0.65	**0.75**	0.62	**0.8**	0.65	**0.88**	0.68	**0.9**
Relative ↑%	↑54.55%		↑33.33%		↑70.37%		↑40.54%		↑24.39%		↑22.5%		↑29.03%		↑35.38%		↑32.35%	

Bold, Red: **no worse** and decreased results from the base models

Table 4: Accuracy under adversarial attacks before (Bef.) and after (Aft.) patched with SHIELD.

The Appendix will include all details on all models' parameters, random seeds, and implementation. We will release the code of SHIELD.

3.2 Results

Due to space limitation, the results of CNN-based models are presented in the Appendix.

Fidelity We first evaluate SHIELD's prediction performance *without* adversarial attacks. Table 3 shows that all base models patched by SHIELD still maintain similar F1 scores on average across all datasets. Although SHIELD with RNN has a slightly decrease in fidelity on Hate Speech dataset, this is negligible compared to the adversarial robustness benefits that SHIELD will provide (More below).

Computational Complexity Regarding the space complexity, SHIELD can extend a NN into an ensemble model with a marginal increase of # of parameters. Specifically, with B denoting # of parameters of the base model, SHIELD has a space complexity of $\mathcal{O}(B+KU)$ while both *Ensemble*, *DT* and *ADP* have a complexity of $\mathcal{O}(KB)$ and $U \ll B$. In case of BERT with K=5, SHIELD only requires an additional 8.3%. While traditional ensemble methods require as many as 4 times additional parameters. During training, SHIELD *only* trains $\mathcal{O}(KU)$ parameters, while other defense methods, including ones using data augmentation, update all of them. Specifically, with K=5, SHIELD only trains 8% of the parameters of the base model and 1.6% of the parameters of other BERT-based ensemble baselines. During inference, SHIELD is also 3 times faster than ensemble-based *DT* and *ADP* on average.

Robustness Table 4 shows the performance of SHIELD compared to the base models. Overall, SHIELD consistently improves the robustness of base models in 154/168 (92%) cases across 14 adversarial attacks regardless of their attack strategies. Particularly, all CNN, RNN, BERT and RoBERTa-based textual models that are patched by SHIELD witness relative improvements in the average prediction accuracy from 15% to as much as 70%. Especially in the case of detecting clickbait, SHIELD can recover up to 5% margin within the performance on clean examples in many cases. This demonstrates that SHIELD provides a versatile neural patching mechanism that can quickly and effectively defends against black-box adversaries *without* making any assumptions on the attack strategies.

We then compare SHIELD with all defense baselines under TextFooler (TF), DeepWordBug (DW), and PWWS (PS) attacks These attacks are selected as (i) they are among the strongest attacks and (ii) they provide foundation mechanisms upon which other attacks are built. Table 5 shows that SHIELD achieves the best robustness across all attacks and datasets. On average, SHIELD observes an absolute improvement from +9% to +18% in accuracy over the second-best defense algorithms (DT in case

Dataset	**MR**			**HS**			**CB**			**AVG**
Attack	**TF**	**DW**	**PS**	**TF**	**DW**	**PS**	**TF**	**DW**	**PS**	
RNN	0.02	0.2	0.09	0.09	0.26	0.32	0.31	0.67	0.46	0.27
+Ens.	0.01	0.16	0.06	0.08	0.12	0.29	0.32	0.66	0.48	0.24
+DT	0.03	0.24	0.1	**0.32**	0.53	0.53	0.35	0.66	0.5	0.36
+ADP	0.02	0.18	0.09	0.18	0.27	0.35	0.33	0.66	0.47	0.28
+Mixup	0.01	0.14	0.04	0.07	0.42	0.29	0.27	0.64	0.44	0.26
+AdvT	0.01	0.3	0.09	0.17	0.18	0.35	0.33	0.69	0.51	0.29
+ScRNN	0.03	0.17	0.08	0.15	0.16	0.32	0.33	0.68	0.47	0.27
+SHIELD	**0.18**	**0.44**	**0.3**	0.26	**0.61**	**0.54**	**0.78**	**0.9**	**0.85**	**0.54**
BERT	0.09	0.2	0.19	0.26	0.16	0.38	0.49	0.5	0.49	0.31
+Mixup	0.11	0.3	0.22	0.15	0.19	0.22	0.39	0.48	0.57	0.29
+AdvT	0.11	0.25	0.19	0.37	0.47	**0.47**	0.69	0.73	0.81	0.45
+ScRNN	0.03	0.11	0.13	0.34	0.33	0.34	0.41	0.51	0.6	0.31
+SHIELD	**0.26**	**0.42**	**0.35**	**0.42**	**0.55**	0.43	**0.92**	**0.95**	**0.94**	**0.58**
RoBERTa	0.06	0.18	0.16	0.1	0.12	0.12	0.37	0.34	0.45	0.21
+Mixup	0.05	0.16	0.15	0.17	0.43	0.32	0.52	0.69	0.66	0.35
+AdvT	0.1	0.21	0.21	0.34	0.43	0.42	0.67	0.79	0.77	0.44
+ScRNN	0.04	0.18	0.15	0.19	0.38	0.32	0.57	0.74	0.7	0.36
+SHIELD	**0.39**	**0.55**	**0.45**	**0.37**	**0.55**	**0.44**	**0.93**	**0.96**	**0.94**	**0.62**

Underline: the second best result

Table 5: Accuracy of all defense baselines under TF, DW and PS attack.

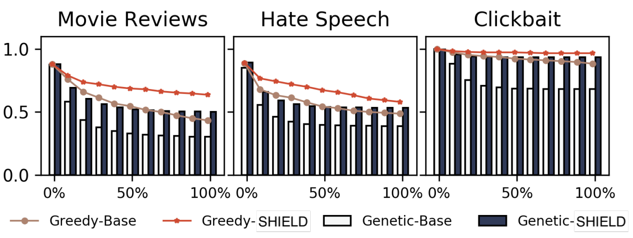

Figure 2: Average accuracy of RoBERTa before and after patched with SHIELD under greedy-based and genetic-based attacks with different percentages of # model queries up to 100% budget limit.

of RNN, and AdvT in case of BERT, RoBERTa). Moreover, SHIELD outperforms other ensemble-based baselines (DT, ADP), and can be applied on top of a pre-trained BERT or RoBERTa model with only around 8% additional parameters. However, that # would increase to 500% ($K \leftarrow 5$) in the case of DT and ADP, requiring over half a billion # of parameters.

4 Discussion

Performance under Budgeted Attacks. SHIELD not only improves the overall robustness of the patched NN model under a variety of black-box attacks, but also induces computational cost that can greatly discourage malicious actors to exercise adversarial attacks in practice. We define computational cost as # of queries on a target NN model that is required for a successful attack. Since adversaries usually have an attack budget on # of model queries (e.g. a monetary budget, limited API access to the black-box model), the higher # of queries required, the less vulnerable a target model is to adversarial threats. A larger budget is crucial for genetic-based attacks because they usually require larger # of queries than greedy-based strategies. We have demonstrated in Sec. 3.2 that SHIELD is robust even when the attack budget is *unlimited*. Fig. 2 shows that the performance of RoBERTa after patched by SHIELD also reduces at a slower rate compared to the base RoBERTa model when the attack budget increases, especially under greedy-based attacks.

Parameter Sensitivity Analyses. Training SHIELD requires hyper-parameter K, T, γ and τ. We observe that arbitrary value γ=0.5, K=5, T=3 works well across all experiments. Although we did not observe any patterns on the effects of K on the robustness, a $K \geq 3$ performs well across all attacks. On the contrary, different pairs of the inverse-temperature τ during training and inference witness varied performance w.r.t to different datasets. τ gives us the flexibility to control the sharpness of the probability vector α. When $\tau \rightarrow \infty$, α to get closer to one-hot encoded vector, i.e., use only one head at a time. By decreasing $\tau : 0.1 \rightarrow 0.001$, we involve more experts in final predictions. Table A.5 (Appendix) shows the best τ found using the validation set as explained in Sec. 3.1.

Ablation Tests. This section tests SHIELD with *only* either the *SE* or *ME* module. Table 6 shows that *SE* and *ME* performs differently across different datasets and models. Specifically, we observe that *ME* performs better than the *SE* module in case of Clickbait dataset, *SE* is better than the *ME* module in case of Movie Reviews dataset and we have mixed results in Hate Speech dataset. Nevertheless, the final SHIELD model which comprises both the *SE* and *ME* modules consistently performs the best across all cases. This shows that both the *ME* and *SE* modules are complementary to each other and are crucial for SHIELD's robustness.

5 Limitations and Future Work

In this paper, we limit the architecture of each expert to be an FCN with a maximum of 3 hidden layers (except the base model). If we include more options for this architecture (e.g., attention (Luong et al., 2015)), sub-models' diversity will significantly increase. The design of SHIELD is model-

Dataset	**Movie Reviews**			**Hate Speech**			**Clickbait**		
Attack	**TF**	**DW**	**PS**	**TF**	**DW**	**PS**	**TF**	**DW**	**PS**
RNN	0.02	0.2	0.09	0.09	0.26	0.32	0.31	0.67	0.46
+*SE* Only	0.02	0.17	0.08	0.09	0.2	0.32	0.52	0.72	0.61
+*ME* Only	0.02	0.14	0.07	0.13	0.03	0.01	0.57	0.79	0.61
+SHIELD	**0.18**	**0.44**	**0.3**	**0.26**	**0.61**	**0.54**	**0.78**	**0.9**	**0.85**
BERT	0.09	0.2	0.19	0.26	0.16	0.38	0.49	0.5	0.49
+*SE* Only	0.07	0.18	0.16	0.26	0.28	0.32	0.45	0.49	0.62
+*ME* Only	0.06	0.2	0.15	0.21	0.28	0.27	0.74	0.81	0.82
+SHIELD	**0.26**	**0.42**	**0.35**	**0.37**	**0.55**	**0.44**	**0.92**	**0.95**	**0.94**
RoBERTa	0.06	0.18	0.16	0.1	0.12	0.12	0.37	0.34	0.45
+*SE* Only	0.13	0.22	0.19	0.13	0.26	0.29	0.57	0.70	0.71
+*ME* Only	0.07	0.17	0.15	0.22	0.4	0.31	0.8	0.87	0.85
+SHIELD	**0.39**	**0.55**	**0.45**	**0.37**	**0.55**	**0.44**	**0.93**	**0.96**	**0.94**

Table 6: Complementary role of *SE* and *ME*.

agnostic and is also applicable to other complex and large-scale NNs such as *transformers-based* models. Especially with the recent adoption of *transformer* architecture in both NLP and computer vision (Carion et al., 2020; Chen et al., 2020), potential future work includes extending SHIELD to patch other complex NN models (e.g., T5 (Raffel et al., 2020)) or other tasks and domains such as Q&A and language generation.

6 Related Work

Defending against Black-Box Attacks. Most of previous works (e.g., (Le et al., 2021; Zhou et al., 2021; Keller et al., 2021; Pruthi et al., 2019; Dong et al., 2021; Mozes et al., 2020; Wang et al., 2021) in adversarial defense are designed either for a specific type (e.g., word, synonym substitution, misspellings) or level (e.g., character or word-based) of attack. Thus, they are usually evaluated against a small subset of ($\leq$4) attack methods. Even though there are works that propose general defense methods, they are often built upon *adversarial training* (Goodfellow et al., 2015) which requires training everything from scratch (e.g., (Si et al.; Miyato et al., 2016; Zhang et al., 2018) or limited to a set of predefined attacks (e.g., (Zhou et al., 2019a)). Even though adversarial training-based defense works well against different attacks on BERT and RoBERTa, its performance is far out-weighted by SHIELD (Table 5).

Contrast to previous approaches, SHIELD addresses not the characteristics of the resulted perturbations from the attackers but their fundamental attack mechanism, which is most of the time an iterative perturbation optimization process (Fig. 1). This allows SHIELD to effectively defend against 14 different black-box attacks (Table 1), showing its effectiveness in practice. To the best of our knowledge, by far, this works also evaluate with the most comprehensive set of attack methods in the adversarial text defense literature.

Ensemble-based Defenses. SHIELD is distinguishable from previous ensemble-based defenses on two aspects. First, previous approaches such as DT (Kariyappa and Qureshi, 2019), ADP (Pang et al., 2019) are mainly designed for computer vision. Applying these models to the NLP domain faces a practical challenge where training multiple memory-intensive SOTA sub-models such as BERT or RoBERTa can be very costly in terms of space and time complexities. In contrast, SHIELD enables to "hot-fix" a complex NN by replacing and training only the last layer, removing the necessity of re-training the entire model from scratch. Second, previous methods (e.g., DT and ADP) mainly aim to reduce the *dimensionality of adversarial subspace*, i.e., the subspace that contains all adversarial examples, by forcing the adversaries to attack *a single fixed* ensemble of diverse sub-models at the same time. However, our approach mainly aims to dilute the attack process with noisy signals by forcing the adversaries to attack stochastic, i.e., different, ensemble variations of sub-models at every inference passes. This helps SHIELD achieve a much better defense performance compared to DT and ADP across several attacks (Table 5).

7 Conclusion

This paper presents a novel algorithm, SHIELD, which consistently improves the robustness of textual NN models under black-box adversarial attacks by modifying and re-training only their last layers. By extending a textual NN model of varying architectures (e.g., CNN, RNN, BERT, RoBERTa) into a stochastic ensemble of multiple experts, SHIELD utilizes differently-weighted sets of prediction heads depending on the input. This helps SHIELD defend against black-box adversarial attacks by breaking their most fundamental assumption–i.e., target NN models remain unchanged during an attack. SHIELD achieves average relative improvements of 15%–70% in prediction accuracy under 14 attacks on 3 public NLP datasets, while still maintaining similar performance on clean examples. Thanks to its model- and domain-agnostic design, we expect SHIELD to work properly in other NLP domains.

References

Ankesh Anand, Tanmoy Chakraborty, and Noseong Park. 2017. We used neural networks to detect clickbaits: You won't believe what happened next! In *ECIR'17*, pages 541–547. Springer.

Ran Avnimelech and Nathan Intrator. 1999. Boosted mixture of experts: An ensemble learning scheme. *Neural computation*, 11(2).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR'15*.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. *ECCV'20*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Mark Chen, Alec Radford, Rewon Child, Jeff Wu, and Heewoo Jun. 2020. Generative pretraining from pixels. In *ICML'20*.

Minhao Cheng, Wei Wei, and Cho-Jui Hsieh. 2019. Evaluating and enhancing the robustness of dialogue systems: A case study on a negotiation agent. *ACL'19*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS'14 Workshop*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *ICWSM'17*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT'19*, pages 4171–4186.

Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. 2021. Towards robustness against natural language word substitutions. *arXiv preprint arXiv:2107.13541*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *ACL'18*. ACL.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *SPW'18*. IEEE.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. *EMNLP'20*.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICLR'15*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. *arXiv preprint arXiv:1805.06087*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *ACL'18*, pages 1875–1885.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *ICLR'17*.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. *EMNLP-IJCNLP'19*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? natural language attack on text classification and entailment. *arXiv preprint arXiv:1907.11932*.

Sanjay Kariyappa and Moinuddin K Qureshi. 2019. Improving adversarial robustness of ensembles with diversity training. *arXiv preprint arXiv:1901.09981*.

Yannik Keller, Jan Mackensen, and Steffen Eger. 2021. Bert-defense: A probabilistic model based on bert to combat cognitively inspired orthographic adversarial attacks. *arXiv preprint arXiv:2106.01452*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP'14*.

Diederik P Kingma and Jimmy Ba. 2013. Adam: A method for stochastic optimization. In *ICLR'13*.

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems, 2018. In *URL https://openreview.net/forum*.

Thai Le, Noseong Park, and Dongwon Lee. 2021. A sweet rabbit hole by darcy: Using honeypots to detect universal trigger's adversarial attacks. In *ACL'21*.

Thai Le, Suhang Wang, and Dongwon Lee. 2020. Malcom: Generating malicious comments to attack neural fake news detection models. In *ICDM'20*. IEEE.

Alexander Hanbo Li and Abhinav Sethy. 2019. Knowledge enhanced attention for robust natural language inference. *arXiv preprint arXiv:1909.00102*.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020a. Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502*.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. TextBugger: Generating Adversarial Text Against Real-world Applications. *NDSS'18*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. Bert-attack: Adversarial attack against bert using bert. *EMNLP'20*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019a. DARTS: Differentiable architecture search. In *ICLR'19*.

Kai Liu, Xin Liu, An Yang, Jing Liu, Jinsong Su, Sujian Li, and Qiaoqiao She. 2020. A robust adversarial training approach to machine reading comprehension. In *AAAI'20*, pages 8392–8400.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP'15*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Training methods for semi-supervised text classification. In *ICLR'16*.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *EMNLP'19*.

Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis D Griffin. 2020. Frequency-guided word substitutions for detecting textual adversarial examples. *arXiv preprint arXiv:2004.05887*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL'05*.

Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. 2019. Improving adversarial robustness via promoting ensemble diversity. In *ICML'19*.

Tianyu Pang, Kun Xu, and Jun Zhu. 2020. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *ICLR'20*.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *EuroS&P'16*, pages 372–387. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP'14*, pages 1532–1543.

Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *ACL'19*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR'20*, 21.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *ACL'19*, pages 1085–1097.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. *ACL'20*.

Chenglei Si, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. Better robustness by more coverage: Adversarial and mixup data augmentation for robust finetuning. *ACL'21 (Findings)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS'17*, pages 5998–6008.

Wenqi Wang, Lina Wang, Run Wang, Zhibo Wang, and Aoshuang Ye. 2019a. Towards a robust deep neural network in texts: A survey. *arXiv preprint arXiv:1902.07285*.

Xiaosen Wang, Hao Jin, and Kun He. 2019b. Natural language adversarial attacks and defenses in word level. *arXiv preprint arXiv:1909.06723*.

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

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *ACL'20*, pages 6066–6080.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *ICLR'18*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *ICLR'18*.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2021. Defense against synonym substitution-based adversarial attacks via dirichlet neighborhood ensemble. ACL'21.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019a. Learning to discriminate perturbations for blocking adversarial attacks in text classification. *arXiv preprint arXiv:1909.03084*.

Zhixuan Zhou, Huankang Guan, Meghana Moorthy Bhat, and Justin Hsu. 2019b. Fake news detection via nlp is vulnerable to adversarial attacks. *arXiv preprint arXiv:1901.09657*.

Model/Dataset	**MR**	**HS**	**CB**	**AVG**
CNN	0.719	**0.900**	0.966	0.862
+Ens.	0.770	0.881	0.975	0.875
+DT	0.767	0.890	0.972	0.876
+ADP	0.764	0.885	**0.977**	0.875
+Mixup	0.711	0.867	0.965	0.848
+AdvT	0.772	0.884	**0.977**	0.878
+ScRNN	0.758	0.854	0.972	0.861
+SHIELD	**0.787**	0.893	0.974	**0.885**

Table A.1: Prediction performance in F1 on clean examples of CNN-based NN models.

Dataset	**MR**		**HS**		**CB**	
Attack	**Bef.**	**Aft.**	**Bef.**	**Aft.**	**Bef.**	**Aft.**
SCPNA	0.35	**0.41**	0.27	**0.4**	0.58	0.53
TB	0.15	**0.35**	0.23	**0.48**	0.79	**0.82**
DW	0.13	**0.38**	0.1	**0.32**	0.71	**0.86**
Kuleshov	0.01	**0.13**	0.01	**0.11**	0.43	**0.63**
TF	0.01	**0.19**	0.03	**0.19**	0.44	**0.74**
IGA	0.05	**0.23**	0.1	**0.2**	0.6	**0.71**
Pruthi	0.49	**0.54**	0.47	**0.59**	0.94	0.9
PS	0.05	**0.28**	0.13	**0.34**	0.56	**0.81**
Alzantot	0.22	**0.3**	0.29	**0.36**	0.82	0.75
BAE	0.45	**0.5**	0.43	**0.55**	0.77	**0.85**
BERTK	0.0	**0.2**	0.01	**0.18**	0.32	**0.61**
PSO	0.03	**0.03**	0.23	**0.34**	0.58	0.56
Checklist	0.7	**0.77**	0.87	**0.88**	0.98	**0.98**
Clare	0.11	**0.3**	0.48	**0.67**	0.6	**0.81**
Average	0.2	**0.33**	0.26	**0.4**	0.65	**0.75**
Relative ↑%		↑65.0%		↑53.85%		↑15.38%

Table A.2: Accuracy of CNN-based NN models under adversarial attacks before (Bef.) and after (Aft.) being patched with SHIELD.

Dataset	**MR**			**HS**			**CB**			**AVG**
Attack	**TF**	**DW**	**PS**	**TF**	**DW**	**PS**	**TF**	**DW**	**PS**	
CNN	0.01	0.13	0.06	0.03	0.1	0.14	0.45	0.7	0.57	0.24
+Ens.	0.02	0.16	0.07	0.08	0.2	0.26	0.72	**0.87**	0.78	0.35
+DT	0.03	0.16	0.07	0.08	0.25	0.28	**0.75**	**0.87**	0.8	0.37
+ADP	0.0	0.11	0.04	0.08	0.19	0.21	0.19	0.67	0.44	0.21
+Mixup	0.03	0.18	0.1	0.07	**0.32**	0.24	0.13	0.6	0.37	0.23
+AdvT	0.02	0.17	0.07	0.1	0.18	0.27	0.33	0.73	0.55	0.27
+ScRNN	0.03	0.24	0.11	0.06	0.14	0.22	0.36	0.69	0.54	0.27
+SHIELD	**0.19**	**0.38**	**0.28**	**0.19**	**0.32**	**0.34**	0.74	0.86	**0.81**	**0.46**

Underline: the second best result

Table A.3: Accuracy of all defense baselines under TF, DW and PS attack on CNN-based NN models.

A ADDITIONAL RESULTS

- Table A.1 shows the performance on clean examples of all defense methods on CNN-based NN models.
- Table A.2 shows the performance of SHIELD against all 14 black-box attacks on CNN-based NN models.

Dataset	**MR**			**HS**			**CB**		
Attack	**TF**	**DW**	**PS**	**TF**	**DW**	**PS**	**TF**	**DW**	**PS**
CNN	0.01	0.13	0.06	0.03	0.1	0.14	0.45	0.7	0.57
+*SE* Only	0.02	0.15	0.07	**0.24**	**0.42**	**0.42**	0.46	0.64	0.61
+*ME* Only	0.18	0.19	0.07	0.1	0.25	0.29	0.60	0.80	0.69
+SHIELD	**0.19**	**0.38**	**0.28**	0.19	0.32	0.34	**0.74**	**0.86**	**0.81**

Table A.4: Ablation test of the *SE* and *ME* modules on CNN-based model.

Model	**Train**			**Test**		
	MR	**HS**	**CB**	**MR**	**HS**	**CB**
CNN+SHIELD	1e-2	1	1	1e-1	1e-1	1e-1
RNN+SHIELD	1	1e-2	1e-3	1e-3	1e-3	1
BERT+SHIELD	1e-2	1e-1	1	1e-1	1e-2	1e-3
RoBERTa+SHIELD	1	1	1e-3	1e-3	1e-3	1e-3

Table A.5: Inverse of the final hyper-parameter τ' values for the selected best SHIELD model for all datasets.

- Table A.3 compares the performance of SHIELD with all defense baselines on CNN-based NN models. SHIELD outperforms all baselines on average.
- Table A.4 shows the ablation test of SHIELD on CNN-based NN models.
- Table A.5 shows the final τ parameters found using brute-force search on the validation set as described in Sec. 3.1. We use this set of parameters to evaluate all the performance under adversarial attacks throughout the paper.

B REPRODUCIBILITY

B.1 Infrastructure and Source Code

- Software: All the implementations are written in Python (v3.7) with Pytorch (v1.5.1), Numpy (v1.19.1), Scikit-learn (v0.21.3). We rely on *Transformers* (v3.0.2) library for loading and training *transformers-based* models (e.g., BERT, RoBERTa).
- Hardware: We run all of the experiments on standard server machines installed with Ubuntu OS (v18.04), 20-Core Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz, 93GB of RAM, and a Titan Xp GPU.
- Dataset: We use the python library *datasets* (v.1.2.0) [1] by *Hugginface* to load all the

[1] https://huggingface.co/docs/datasets/#

benchmark datasets used in the paper. They are also available to download at the following links: Movie Reviews (http://www.eraserbenchmark.com/zipped/movies.tar.gz), Clickbait (https://github.com/saurabhmathur96/clickbait-detector), Hate Speech (https://github.com/t-davidson/hate-speech-and-offensive-language/raw/master/data/labeled_data.csv).

- Random Seed: To ensure reproducibility, we set a consistent random seed using *torch.manual_seed* and *np.random.seed* function for all experiments.

- Source Code: We will also release the source code of SHIELD upon acceptance of this paper.

B.2 Experimental Settings for Base Models

B.2.1 Architectures and Parameters

- CNN: We implement the CNN sentence classification model (Kim, 2014) with three 2D CNN layers, each of which is followed by a *Max-Pooling* layer. Concatenation of outputs of all Max-Pooling layers is fed into a Dropout layer with 0.5 probability, then an FCN + Softmax for prediction. We use an *Embedding* layer of size 300 with pre-trained *GloVe* embedding-matrix to transform each discrete text tokens into continuous input features before feeding them into the CNN network. Each of CNN layers uses 150 kernels with a size of 2, 3, 4, respectively.

- RNN: Because the original PyTorch implementation of RNN does not support double back-propagation on *CuDNN*, which is required by *DT* and SHIELD to run the model on GPU, we use a publicly available *Just-in-Time (JIT)* version of GRU of one hidden layer as RNN cell. We use an *Embedding* layer of size 300 with pre-trained *GloVe* embedding-matrix to transform each discrete text tokens into continuous input features before inputting them into the RNN layer. We flatten out all outputs of the RNN layer, followed by a Dropout layer with 0.5 probability, then an FCN + Softmax for prediction.

- BERT & RoBERTa: We use the *transformers* library from *HuggingFace* to fine-tune BERT and RoBERTa model. We use the *bert-base-uncased* version of BERT and the *RoBERTa-base* version of RoBERTa.

B.2.2 Vocabulary and Input Length

Due to limited GPU memory, we set the maximum length of inputs for transformer-based models, i.e., BERT and RoBERTa, to 128 during training. For CNN and RNN-based models, we use all the vocabulary tokens that can be extracted from the training set, and we use all of the vocabulary tokens provided by pre-trained models for BERT and RoBERTa-based models.

B.3 Experimental Settings for Defense Methods

1. SHIELD: For hyper-parameter γ, K and T, we arbitrarily set $\gamma{\leftarrow}0.5$, $K{\leftarrow}5$ and $T{\leftarrow}3$ and they work well across all datasets. For τ, we already described how to choose the best pair of τ during training and testing in Sec. 3.1.

2. *Ensemble*: We train an ensemble model of 5 sub-models, all of which have the same architecture as the base model. We use the average loss of all sub-models as the final loss to train the model.

3. *DT*: We follow the implementation described in Section 3 of the original paper (Kariyappa and Qureshi, 2019) and train an ensemble DT model with 5 sub-models, all of which have the same architecture as the base model. We set the hyper-parameter $\lambda \leftarrow 0.5$ as suggested by the original paper.

4. *ADP*: We follow the implementation described in Section 3 of the original paper (Pang et al., 2019) and train an ensemble ADP model with 5 sub-models, all of which have the same architecture as the base model. We set the hyper-parameters required by ADP to default values ($\alpha \leftarrow 1.0$ and $\beta \leftarrow 0.5$) as suggested by the original implementation.

5. *Mix-up Training (Mix)*: We sample $\lambda \in Beta(1.0, 1.0)$ as suggested by the implementation provided by the original paper (Zhang et al., 2018).

6. *Adversarial Training*: We use a *1:1* ratio between original training samples and adversarial training samples as suggested by (Miyato et al.,

2016). We specifically use the *AT* method as described in Sec. 3 of the original paper (Miyato et al., 2016).

7. *ScRNN*: We use the implementation and pretrained model provided by the original paper (Pruthi et al., 2019) that is available at `https://github.com/danishpruthi/Adversarial-Misspellings`.

B.4 Experimental Settings for Attack Methods

Since we use external open-source *TextAttack* (Morris et al., 2020) [2] and *OpenAttack* [3] framework for evaluating the performance of SHIELD and all defense baselines under adversarial attacks, implementation of all the attacks are publicly available. Specifically, we use the *TextAttack* framework for evaluating all the word- and character-level attacks, and use the *OpenAttack* for evaluating the sentence-level attack SCPNA.

B.5 Experimental Settings for Training and Evaluation

For every dataset, we train a single SHIELD model with the best τ parameters and evaluate this model with all of the adversarial attacks. In other words, since we have a total of 3 datasets (Movie Reviews, Hate Speech, Clickbait) and 4 base architectures (CNN, RNN, BERT, RoBERTa), we train a total of 12 SHIELD models for evaluation. This is done to ensure that we can evaluate the versatility of SHIELD's robustness against different types of attacks *without making any assumptions on their strategies*. During training, we use a *batch size* of 32, *learning rate* of 0.005, *gradient clipping* of 10.0.

For every attack evaluation, we generate a new set of adversarial examples for every pair of *attack method* and *target model*. In other words, since we have a total of 14 different attack methods, 3 datasets, and 4 possible architectures for the base models, this results in a total of 168 different sets of adversarial examples to evaluate in Table 4.

[2] `https://github.com/QData/TextAttack`

[3] `https://github.com/thunlp/OpenAttack`