Tiny RNN Model with Certified Robustness for Text Classification

Yao Qiang, Supriya Tumkur Suresh Kumar, Marco Brocanelli and Dongxiao Zhu
Department of Computer Science
Wayne State University
Detroit, MI, USA
Email: {yao, supriyats, brok, dzhu}@wayne.edu

***Abstract*—Mobile artificial intelligence has recently gained more attention due to the increasing computing power of mobile devices and applications in computer vision, natural language processing, and internet of things. Although large pre-trained language models (e.g., BERT, GPT) have recently achieved the state-of-the-art results on text classification tasks, they are not well suited for latency critical applications on mobile devices. Therefore, it is essential to design tiny models to reduce their memory and computing requirements. Model compression has shown promising results for this goal. However, some significant challenges are yet to be addressed, such as information loss and adversarial robustness. This paper attempts to tackle these challenges through a new training scheme that minimizes the information loss by maximizing the mutual information between the feature representations learned from the large and tiny models. In addition, we propose a certifiably robust defense method named GradMASK that masks a certain proportion of words in an input text. It can defend against both character-level perturbations and word substitution-based attacks. We perform extensive experiments demonstrating the effectiveness of our approach by comparing our tiny RNN models with compact RNNs (e.g., FastGRNN) and compressed RNNs (e.g., PRADO) in clean and adversarial test settings.**

I. Introduction

Mobile artificial intelligence (AI) has recently found applications in a wide range of domains, including image classification, healthcare, recommender system, and sentiment analysis [1], [2]. Even with the rapid improvements of mobile hardware, the main obstacles in deploying deep neural networks (DNNs) on mobile devices are the high computational and memory requirements during training and inference processes. To alleviate this limitation, several model compression techniques, e.g., quantization [3], weight pruning [4], and knowledge distillation [5], have been applied to develop tiny DNNs for mobile deployment in computer vision tasks. Some tiny recurrent neural networks (RNNs) [6], [7] have also been developed and applied in lightweight (IoT) applications [8].

However, deploying tiny models for natural language processing (NLP) tasks is more challenging. NLP models include an embedding layer that maps discrete words and phrases to continuous vectors. The problem is that the embedding layer usually takes more parameters than the remaining networks due to the large vocabulary and high embedding dimension. In practice, the word embedding parameters account for 80% of the total parameters in a neural translation model of OpenNMT [9]. Therefore, it is desired to reduce the parameters of the embedding layer in tiny models. Many efforts have been made to compress word embeddings, e.g., quantizing each dimension of embeddings [10], filtering out uncommon words in vocabulary [11], and applying code-books to represent the original embeddings [12]. However, the existing works face the information loss problem during compression or need additional complex designs [13].

Unlike the above methods, we propose a simple but efficient method to reduce the embedding layer's parameters directly. Instead of using a high embedding dimension, we employ a much smaller one (i.e., decreasing from 200 to 5) for our tiny model. However, Patel et al. [14] have demonstrated that if the word embeddings do not get enough dimensions (a lower bound), it will fail to uphold the equality constraints, which ensures the correct embedding information. Therefore, our smaller embedding dimension may cause poor embedding performance and information loss. To mitigate these problems, we apply a feature mapping technique, which distills features from a large pre-trained model to a customized tiny model. Specifically, we add a regularization term in our training objective, which maximizes the mutual information between the features learned from the large and tiny models during the feature mapping. The mutual information is calculated by the Kraskov–Stogbauer–Grassberger (KSG) estimator, which has been demonstrated efficient to measure the similarity between dense word embeddings [15]. Similarly, we establish this design in the hidden state layer (e.g., LSTM) to further reduce our tiny model's parameters. As a result, our tiny model contains much fewer parameters than the large model, i.e., decreasing from 2M to 100K.

As DNNs are increasingly deployed in safety- and security-critical environments (e.g., healthcare and autonomous driving), a growing body of work is exploring the robustness properties of DNNs from the security perspective by proposing attacks (methods for crafting adversarial examples) and defenses (methods for making DNNs robust against such attacks). Thus, it is also desired for our tiny RNN model to be robust against adversarial attacks while employed on mobile devices. A few certified defense methods have been developed to provably guarantee the robustness of a text classifier against adversarial attacks [16]. However, most existing methods assume that the defenders are informed of how the attackers generate adversarial examples, which is unrealistic. Additionally, to our

best knowledge, no method has been proposed specially to improve the certified robustness of the tiny models for on-device NLP applications.

More recently, Zeng et al. [17] propose a certifiably robust defense method by randomly masking a certain proportion of the words in an input text. The authors claim their method can defend against word substitution based attacks and character-level perturbations. However, the random mask approach does not consider the efficiency and heavily relies on the total number of the masks. Differently, we design a novel certified defense method named GradMASK, which intentionally masks the most important words in the adversarial example instead of randomly masking. The importance scores are achieved by ranking the words' gradient values. Our method guarantees that the masked words make outstanding contributions to the incorrect predictions. Then, we take the average logits produced by the large model from the masked adversarial examples for soft label knowledge distillation in our training scheme. Thus, our tiny model gains certified robustness via knowledge distillation and does not need additional adversarial training to improve the robustness.

Specifically, our major contributions are summarized as: (1) We design a tiny RNN model for on-device text classification with a much fewer total number of parameters than the large RNN model and other compact/compressed RNN models. (2) We mitigate the information loss in model compression by maximizing the layer-wise feature mutual information and minimizing the soft label knowledge distillation loss in our new training scheme. (3) A novel certifiably robust defense method is employed to improve the robustness of tiny models against different types of adversarial attacks covering character-level and word-level perturbations. (4) The extensive experiments demonstrate the effectiveness of our approach by comparing our tiny models with baseline compact and compressed RNN models in multiple text classification tasks.

II. Related Work

A. Tiny RNN Models

RNN models have achieved significant success in learning complex patterns for temporal/sequential data (e.g., sensor signal, natural language). Beyond the classical LSTM and GRU architectures, more sophisticated RNNs with skip connections and residual blocks [18] and those combined with CNNs have been developed to allow the RNNs to go 'deeper' and achieve better performance. However, despite the state-of-the-art performance, these heavyweight RNN models are resource-hungry and unsuitable for on-device deployment.

Recently, tiny RNN models with small parameter sizes (e.g., 200K or less) have received increasing attention due to their high application potential to on-device deployment, including mobile and IoT environments. Ravi et al. [19] propose a new architecture that jointly trains a large neural network and a small projection network leveraging random projections to transform inputs or intermediate feature representations into bits. The projection network encodes lightweight and efficient computing operations in bit space to reduce memory footprint. Since then, a few more advanced projection networks have been proposed to achieve better performance [20], [21]. Some tiny models have been designed with novel compact RNN architectures [7]. Kusupat et al. [6] propose FastRNN/FastGRNN by adding residual connections and gating on the standard RNNs, which outperforms LSTM and GRU in prediction accuracy with fewer parameters. Other works consider compressing word embeddings directly to reduce the total number of parameters in RNN models [12], [22]. Unlike the above approaches, we design a tiny RNN model with fewer parameters in embedding and hidden states layers resulting in much smaller model size. We train the tiny model in a novel training scheme that minimizes the information loss via layer-wised feature mapping and soft label knowledge distillation with a novel certifiably robust defense method.

B. Mutual Information

Mutual information (MI) measures the mutual dependence between two variables. More specifically, it quantifies the "amount of information" obtained about one random variable by observing the other random variable. There is a vast literature on how to estimate MI. These approaches can be roughly categorized into four classes. The first class partitions the two variables into a finite number of bins of equal or unequal (adaptive) size and estimates MI based on discrete counts in each bin [23]. However, such methods suffer from the curse of dimensionality and are not practical to estimate MI of the high dimension embeddings and hidden state features in our DNN model. The second class first constructs kernel density estimates and then numerically integrates such approximate densities to estimate MI [24]. These approaches need a careful choice of the bandwidth parameters and the kernel functions limiting their applications. The third class applies neural-network-based estimation and trains a deep learning model to estimate the density ratio of the variables for MI estimation [25]. Such deep learning estimators are usually differentiable and scale well to high dimensions and large sample sizes. However, training a separate neural network is hardly justified and violates our desire to achieve tiny models. Therefore, we focus on the last class, which estimates MI from the k-nearest neighbor statistics [26]. In particular, the Kraskov–Stogbauer–Grassberger (KSG) estimator [26] has been demonstrated to measure the similarity of word embeddings efficiently in [15]. We apply (KSG) estimator to calculate and maximize MI between the features learned from the large and tiny models in our training scheme to minimize the information loss in compression.

C. Adversarial Robustness for Text Classification

Text adversarial attack is a severe problem in NLP applications. Attackers can mislead the text classifiers with a small amount of modification, e.g., replacing a certain number of characters or words. There are two types of adversarial attacks: character-level and word-level. Character-level attacks

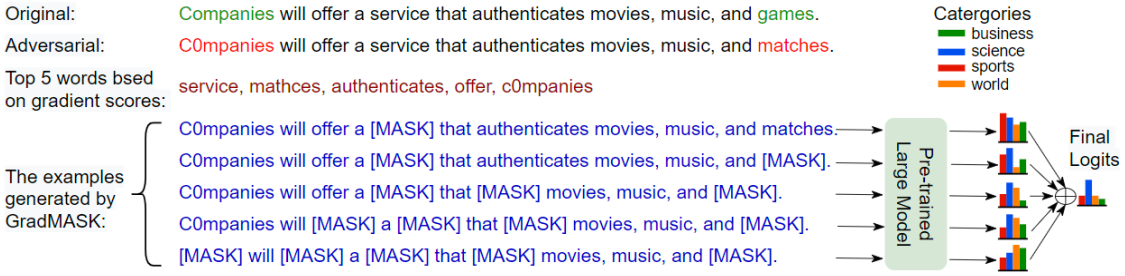

Fig. 1: Illustration of our GradMASK on a sample form AG's News dataset. Given an original text at the top, an adversarial example is generated by replacing the word "companies" with "c0mpanies" and "games" with "matches". The category of the original text is *science*, while the adversarial example misleads the model to predict it as *sports* due to the perturbations. We get the top-5 important words based on their gradient scores, such as "service", "matches", "authenticates", "offer", and "c0mpanies". Taking the adversarial example and the index of the top-5 words, GradMASK generates a set of masked adversarial samples by replacing these words with [MASK] iteratively. The pre-trained large model generates the logits for each sample. We take the average logits from the generated logits as the final logits (with the largest score on *science*) to be used in soft label knowledge distillation in our training scheme. $\oplus$ here denotes average operation.

generate adversarial examples that are very similar to the original examples by deleting, inserting, and modifying characters, such as DeepWordBug [27], Hotflip [28] and DeepFool [29]. Word-level attacks craft adversarial examples by modifying words while maintaining grammatical correctness and semantic consistency, e.g., PWWS [30] and SPGD [31].

Many methods have been proposed to defend the character-level attacks by recognizing or even correcting the perturbed characters [32], [33]. It is more difficult to defend word-level attacks since the perturbations have the correct spelling, grammar, and semantics. There are mainly three types of defense methods against word-level attacks in the literature: word re-encoding [34], adversarial training [35], [36] and adversarial data augmentation [17], [37]. Our work focuses on adversarial data augmentation, which augments the original training data with adversarial examples. The model trained with adversarial data augmentation obtains improved robustness to the perturbations.

Several works discuss that a model is certified robust when it is guaranteed to give the correct answer under any attacker, no matter the strength of the attacker and no matter how the attacker manipulates the input texts [16], [38]. Zeng et al. [17] propose a certifiably robust defense method named RanMASK by randomly masking a certain proportion of the words in an input text. However, RanMASK randomly masks several words in the adversarial example without considering the efficiency and heavily relies on the total number of the masks. We further propose an improved masking method leveraging the gradient to sort the words to be masked. Our proposed method is more efficient and does not require a large number of masked adversarial examples.

III. Method

A. KSG Mutual Information Estimator

We denote MI of two variables $\mathbf{X}$ and $\mathbf{Y}$ as:

$$\mathbf{I}(\mathbf{X};\mathbf{Y}) = \iint p_{\mathbf{XY}}(x,y)\log\frac{p_{\mathbf{XY}}(x,y)}{p_{\mathbf{X}}(x)p_{\mathbf{Y}}(y)}\mathrm{d}x\mathrm{d}y, \quad (1)$$

where $p_{\mathbf{XY}}(x,y)$ is the joint density of $\mathbf{X}$ and $\mathbf{Y}$. $p_{\mathbf{X}}(x) = \int_x p_{\mathbf{XY}}(x,y)\mathrm{d}x$ and $p_{\mathbf{Y}}(y) = \int_y p_{\mathbf{XY}}(x,y)\mathrm{d}y$ are the marginal densities. Given two features (i.e., embeddings and hidden states in RNN models) $\mathbf{X}$ and $\mathbf{Y}$ learned from the large and tiny models, our goal is to maximize $\mathbf{I}(\mathbf{X};\mathbf{Y})$ in our training scheme in order to minimize the information loss in model compression. However, it is difficulty to estimate MI for continuous variables since the joint and marginal densities are not known in practice.

Although there are many existing works on estimating MI, most of them are not suitable for our particular scenario, as discussed in Section II-B. The KSG estimator has been demonstrated to admit a particularly elegant expression for the similarity measurement of word embeddings [15]. Thus, we employ the KSG estimator to measure and maximize MI between two features (i.e., the embeddings and hidden states features) during the feature mapping process in our training scheme. We briefly illustrate how the KSG estimator can be applied to estimate MI here and refer the reader to [26] for its full derivation and justification.

MI can be equivalently expressed as $\mathbf{I}(\mathbf{X};\mathbf{Y}) = \mathbf{H}(\mathbf{X}) + \mathbf{H}(\mathbf{Y}) - \mathbf{H}(\mathbf{X},\mathbf{Y})$, i.e. the difference between the sum of marginal entropies and the joint entropy. Thus, it is sufficient to estimate various entropies in the above equation in order to estimate MI. Kozachenko et al. [39] first propose to estimate these differential entropies based on the nearest neighbour statistics. Recently, Kraskov et al. [26] modify this idea and construct the KSG estimator of MI as:

$$\begin{aligned}&\mathrm{KSG}(\mathbf{X};\mathbf{Y}) = \psi(D) + \psi(k) - \\ &\sum_{d=1}^{D}(\psi(n_x[d]+1) + \psi(n_y[d]+1)),\end{aligned} \quad (2)$$

where D is the embedding dimension, k denotes the number of nearest neighbours, and $\psi(x) = \Gamma'(x)/\Gamma(x)$ is the digamma function. $n_x[d], n_y[d]$ are certain nearest neighbour statistics, which are obtained by counting the number of neighbors that fall within less than $\epsilon[d]$ from x^d and y^d in the marginal spaces $\mathbf{X}$ and $\mathbf{Y}$. $\epsilon[d]$ here denotes the distance from $z^d = (x^d, y^d)$

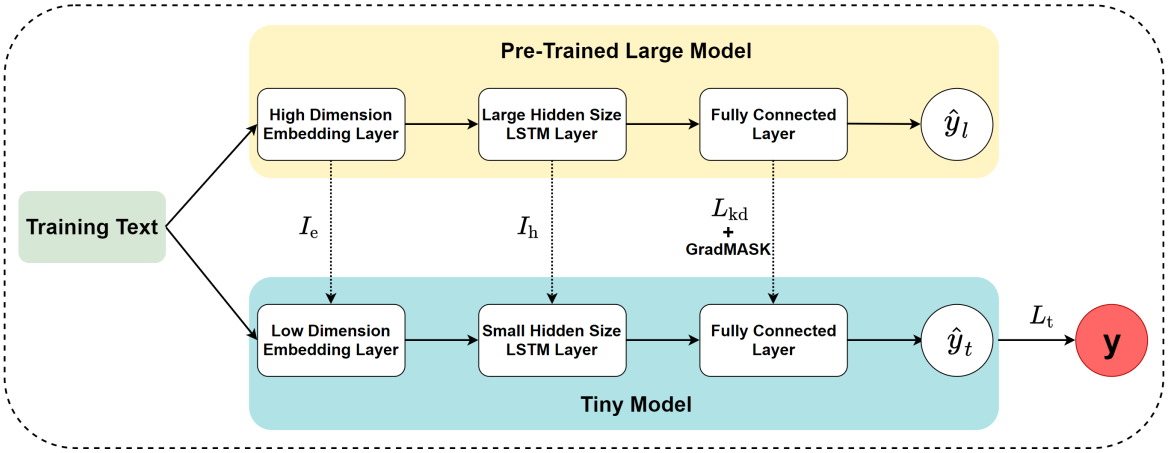

Fig. 2: Architecture of our tiny model training scheme. The first two feature mapping layers maximize the mutual information between the features from the large and tiny models. We apply our certified robust method GradMASK in the soft label knowledge distillation layer to improve the robustness of our tiny model. Finally, a task loss objective function is applied for the main classification tasks.

to its k-nearest neighbor in the joint space $(\mathbf{X}, \mathbf{Y})$. Finally, we set k as 3 and apply Equation 2 to approximately estimate MI in our experiments.

B. GradMASK: Certified Defense Method

Given an original input $\mathbf{x} = (x_1, \cdots, x_n)$, the adversarial attacks generate an adversarial example $\mathbf{x}' = (x'_1, \cdots, x'_n)$ by perturbing at most $m \leq n$ words in $\mathbf{x}$. We say $\mathbf{x}'$ is a good adversarial example of $\mathbf{x}$ if:

$$f(\mathbf{x}') \neq y, \quad ||\mathbf{x} - \mathbf{x}'|| \leq m \ . \tag{3}$$

y here denotes the group truth, and $||\mathbf{x} - \mathbf{x}'|| = \sum_{i=1}^{n} \mathbb{I}\{x_i \neq x'_i\}$ is the Hamming distance, where $\mathbb{I}\{\cdot\}$ is the indicator function. In this work, we consider both character-level perturbations where x'_i is a visually similar misspelling or typo of x_i (e.g., Replaceone) and word-level perturbations where x'_i is any of x_i's synonyms (e.g., PWWS). If a model f can give consistently correct predictions for all the possible adversarial examples $\mathbf{x}'$ regardless the type of perturbations under the constraint of $||\mathbf{x} - \mathbf{x}'|| \leq m$, we say it is certified robust against the adversarial attacks.

We propose a saliency guided mask operation named GradMASK by iteratively masking the words with high gradient values, i.e., more important words for model predictions. Saliency methods use gradient calculations to assign an importance score to individual features, reflecting their influences on the model prediction [40]–[42]. The gradient of the model output $f(\mathbf{x})$ with respect to the input $\mathbf{x}$ is given by $\nabla_{\mathbf{x}} f(\mathbf{x})$. Let $S(\cdot)$ be a sorting function such that $S_j(Z)$ is the j^{th} largest element in Z. Hence, $S(\nabla_{\mathbf{x}} f(\mathbf{x}))$ denotes the sorted gradients. We define GradMASK as $\mathcal{M}$ such that $M_k(S(\mathbf{x}); \mathbf{x})$ replaced all x_i where $S(x_i) \in S_j(x_i)_{j=0}^{k}$ with a special token [MASK]. The masked words can simply be encoded as the embedding of [MASK]. For example, if $\mathbf{x}$ = "A, B, C, D, E" and $S(\nabla_{\mathbf{x}} f(\mathbf{x}))$ = "B, C, A, D, E", $M_2(S(\mathbf{x}); \mathbf{x})$ generates the masked sample as $\mathbf{x}_{\text{mask}}$ = "A, [MASK], [MASK], D, E". Specifically, we generate a set of $\mathbf{x'}_{\text{mask}}$ for the adversarial example $\mathbf{x}'$ as adversarial data augmentation with different k values (i.e., 1-5). We show an illustrated example in Fig. 1. For each adversarial example $\mathbf{x}'$, we totally generate 5 masked adversarial examples in our experiments.

Our certified defense method trains the tiny model by distilling the soft labels (or logits) generated from the large model, which ensembles the outputs of a number of masked adversarial samples. In particular, let $f_{\text{large}} : \mathbf{x}_{\text{mask}} \rightarrow y$ be the larger classifier, which is trained to classify the masked adversarial samples. The tiny model f_{tiny} applies knowledge distillation with objective loss function as:

$$\mathcal{L}_{\text{kd}} = \text{KL}(\text{Softmax}(\tilde{\mathbf{y}}_{\text{large}}/T), \text{Softmax}(\mathbf{y}_{\text{tiny}}/T)), \tag{4}$$

where KL denotes the Kullback–Leibler (KL) divergence loss. This soft label knowledge distillation is attributed to not only the privileged information on similarities among classes but also the label smoothing regularization. It can also accelerates the training convergence [43]. T here is the temperature to soften the outputs, which make the tiny model's logits $\mathbf{y}_{\text{tiny}}$ more closely resembled the large model's average logits $\tilde{\mathbf{y}}_{\text{large}}$

$\tilde{\mathbf{y}}_{\text{large}}$ is the average of the logits produced by the large model over all the individual masked adversarial samples: $\tilde{\mathbf{y}}_{\text{large}} = \sum_{i=1}^{k} \mathbf{y}_i / k$. k here denotes the number of generated masked adversarial samples. $\tilde{\mathbf{y}}_{\text{large}}$ further prevents the attacks from effectively identifying the weakness of our models. As a result, the tiny model can classify the original input $\mathbf{x}$ robustly against any adversarial attack that is allowed to perturb a certain number of words at either character or word level.

C. Tiny Model Training Scheme

Fig. 2 illustrates our training scheme, which is composed of two feature mapping layers to maximize mutual information, a soft label knowledge distillation layer for certified robustness, and a final target classification.

We denote an input text as $\mathbf{x} = \{x_1, \ldots, x_i, \ldots, x_n\}$, where i is the index and n is the number of words. The first layer in most NLP models applies an embedding layer with trainable

parameters $\mathbf{W}_e \in \mathbb{R}^{d \times V}$ to map each word x_i to a fixed-length d-dimension vector $\mathbf{e}_i \in \mathbb{R}^d$, where V denotes the vocabulary size. The embedded word vectors $\mathbf{e}$ are processed by the remaining layers. To retain sufficient embedding information, most models use a large vocabulary size V (ranging from hundreds of thousands to millions) and a high embedding dimension d (e.g., 100 or higher), resulting in a huge number of parameters in $\mathbf{W}_e$. In practice, the embedding parameters account for 80% of the total parameters. Therefore, it is desired to decrease the embedding parameters for deploying NLP models on memory-constrained mobile devices.

As there is a minimum required vocabulary size to achieve a specific performance [44], we propose a low dimension (i.e., 5, 10, and 20) embedding layer in the tiny model to decrease the number of parameters in $\mathbf{W}_e$ (i.e., from 2M to 100K). In order to mitigate the embedding information loss caused by the low dimension, we apply a feature mapping method that allows the tiny model to learn embedding information from the pre-trained large model via maximizing the mutual information between the embedded features. In more detail, a regularization term is added into our training objective to maximize $\mathbf{I}(\mathbf{e}_\mathrm{l}, \mathbf{e}_\mathrm{t})$, which is measured by the KSG estimator. $\mathbf{e}_\mathrm{l}$ and $\mathbf{e}_\mathrm{t}$ here denote the embedded features from the large and tiny models, respectively.

As shown in Fig. 2, following the embedding layer is the LSTM layer. The number of parameters in LSTM layer can be calculated as: $4 \times h \times ((d+1)+h)$ [45], where h is the size of hidden state and d is the input size. So h greatly affects the number of parameters. As such, we set a small hidden state size in LSTM hidden layer without using attentions, shortcut connections, or other sophisticated additions to reduce the number of parameters in the tiny model. Similarly, we maximize $\mathbf{I}(\mathbf{h}_\mathrm{l}, \mathbf{h}_\mathrm{t})$ during the hidden state feature mapping.

The last layer in our training scheme exploits soft label knowledge distillation, which enforces the tiny model to mimic the prediction behavior of the large model by training the former with more informative soft labels generated by the latter [5]. We further apply our certified defense method in this layer to make the tiny model robust against different adversarial attacks as presented before.

Our general training objective function consists of several components, formally:

$$\mathcal{L} = \lambda_1 \mathcal{L}_\mathrm{t} + \lambda_2 \mathcal{L}_\mathrm{kd} - \lambda_3 (\mathcal{I}_\mathrm{e} + \mathcal{I}_\mathrm{h}) \tag{5}$$

where $\mathcal{L}_\mathrm{t} = \mathrm{CE}(y, \hat{y})$ denotes the CrossEntropy loss for text classification task. $\mathcal{L}_\mathrm{kd}$ is the soft label knowledge distillation loss as described in Equation (4). $\mathcal{I}_\mathrm{e}$ and $\mathcal{I}_\mathrm{h}$ are two regularization terms representing $\mathbf{I}(\mathbf{e}_\mathrm{l}, \mathbf{e}_\mathrm{t})$ and $\mathbf{I}(\mathbf{h}_\mathrm{l}, \mathbf{h}_\mathrm{t})$, respectively. The negative symbol means maximizing these two terms. In this way, these two regularization terms help increase the mutual information between the embedding and hidden state features of the tiny and pre-trained large models. λ_1, λ_2, λ_3 are tuning parameters to leverage the relative importance of different terms in our objective function.

IV. Experiment Setup

Datasets: Our experiments are conducted with multiple text classification tasks, such as sentiment analysis (Amazon and Yelp), news categorization (AG's News), and topic classification (Yahoo). We keep the same training and test set splitting as [46]. Statistics of the datasets are listed in TABLE I.

Dataset	#Training	#Testing	#Class
Amazon	3,600,000	400,000	5
Yelp	560,000	38,000	5
AG's News	120,000	7,600	4
Yahoo Answers	1,400,000	60,000	10

TABLE I: Datasets statistics details

Environment: Our models are implemented in PyTorch with two NVIDIA GeForce RTX 3090 GPU and a 24Gb memory. Adam optimizer is utilized, and the learning rate is consistently set as 0.001 in all the experiments. All models are trained in 20 epochs with batch size 64. We fine tune the hyper-parameters with several values and use the optimal selections, i.e., T is as 3, λ_1, λ_2, λ_3 are 0.5, 0.5, 0.8, respectively.

A. Adversarial Attacks

To evaluate the certified robustness of our tiny model, we consider both character-level (i.e. Replaceone [27] and Gradient [47]) and word-level (i.e. PWWS [30]) adversarial attacks. These attacks are one-off attacks, which are more efficient and with minimal alteration that is more suitable for real-world applications. We use the same settings as in [27], [30] for our experiments. In more detail:

Replaceone: Gao et al. [27] apply different black-box scoring functions and word transformation methods to generate adversarial samples with minimum word changes. We use Replaceone, an efficient yet effective scoring function, to find the most important words, then swap two adjacent letters to generate new words in the adversarial samples.

Gradient: Samanta et al. [47] propose a white-box attack that uses gradients to identify salient words in the original samples and modifies them to generate adversarial samples.

PWWS: Ren et al. [30] propose a white-box attack using a new word replacement order determined by both the word saliency and the classification probability. The generated adversarial samples are lexically/grammatically correct and semantically similar to the original samples.

B. The Compared Methods

Our goal is to develop an adversarially robust tiny model for NLP tasks, which achieves competitive performance in classifying both clean and adversarial samples and satisfies the on-device resource constraints. We compare with two existing on-device compressed neural network models for NLP applications (i.e., SGNN [20] and PRADO [21]) in terms of the test performance using clean samples only since (1) they are not designed for defending against adversarial attacks and (2) there are no source codes available. Moreover, our tiny model is compared with the compact model, FastGRNN [6],

Attacks	Amazon		Yelp		Yahoo Answer		AG's NEWS	
	CE	Ours	CE	Ours	CE	Ours	CE	Ours
Clean	57.4	**61.8**	59.4	**62.2**	68.4	**72.3**	86.5	**90.2**
PWWS	25.1	**36.4**	29.4	**43.2**	32.4	**46.2**	33.3	**57.6**
Gradient	35.0	**56.6**	32.7	**56.9**	38.1	**57.4**	38.1	**83.7**
Replaceone	33.2	**46.9**	36.2	**49.8**	33.6	**50.3**	34.6	**61.1**

TABLE II: Comparison of classification performance and adversarial robustness of the tiny models trained with the conventional CE loss (CE) and our training schemes (Ours) on four benchmark datasets. Clean here indicates natural test samples. The embedding dimension (d) and hidden state size (h) are set as 5 in the tiny models. PWWS, Gradient, and Replaceone are three different adversarial attacks covering character-level and word-level perturbations. Best performances are bold-faced.

Type	Method	#Params	Yelp			Amazon			Yahoo Answers		
			Clean Acc	Gradient AdvAcc	Replace AdvAcc	Clean Acc	Gradient AdvAcc	Replace AdvAcc	Clean Acc	Gradient AdvAcc	Replace AdvAcc
Tiny	**Ours(tiny)**	**100K**	62.2	**56.9**	**49.8**	**61.8**	**56.6**	**46.9**	**72.3**	**57.4**	**50.3**
	PRADO	175K	**64.7***	-	-	61.2*	-	-	72.3*	-	-
	SGNN	500K	35.4*	-	-	39.1*	-	-	36.6*	-	-
	FastGRNN	250K	26.7	20.7	20.6	30.2	20.4	19.6	28.3	19.7	21.6
Large	**Ours(large)**	2M	**63.9**	**50.6**	**43.9**	**62.3**	**50.9**	40.3	**73.2**	49.3	**51.2**
	CNN-char	11M	62.0	45.7	40.8	59.6	47.0	**42.1**	71.2	43.5	44.9
	CNN-word	8M	60.5	44.8	37.6	57.6	47.6	41.1	71.2	**51.9**	46.6
	LSTM	2M	58.2	43.2	42.6	59.4	45.7	41.3	70.8	40.0	45.1

[1] -: not applicable. *: results reported in [21].

TABLE III: Comparison of our tiny models with other compressed, compact and large models. Best performances are bold-faced according to model type.

which is initially designed for IoT applications and recently demonstrates promising performance in NLP applications. Although our tiny model is intended for on-device deployment, we compare it with some on-cloud large models that are designed to exploit the full extent of cloud resources, i.e., char-level CNN, word-level CNN, and LSTM [46] to demonstrate our competencies. We report the comparison results in TABLE II and TABLE III.

C. Evaluation Metrics

Our experiments consider the performance of the models trained with clean texts and their adversarial robustness towards various attacks. First, accuracy (Acc) is used to evaluate the models' performance for clean text classification tasks. Then, we report the adversarial accuracy (AdvAcc) to evaluate the robustness of the models on thousands of adversarial examples generated from different attacks. Finally, we report the models' number of parameters to compare their sizes.

V. Results and Discussion

A. Performance Comparison

TABLE II shows the performance and adversarial robustness of the tiny models, either trained conventionally with cross-entropy (CE) loss only or with the proposed training scheme (Ours). It is observed that our tiny models achieve an overall better performance on all the evaluation metrics. Our training scheme exploits feature mapping and soft label knowledge distillation enabling the tiny models to learn information from the pre-trained large models. As a result, it can reduce the information loss during model compression, leading to significant improvement in clean sample accuracy. In addition, the tiny models trained with soft label knowledge distillation in our training scheme improve their adversarial robustness compared to the ones trained with the CE loss in a vast majority of comparisons, highlighting the advantage of our certified defense method.

We further compare the performance of our tiny models with the other compressed, compact models and large models in both clean and adversarial example settings (wherever applicable). Note that the results of PRADO [21] and SGNN [20] are cited directly from the original papers since no source codes are made available. Therefore, we only compare the clean sample accuracy with these two compressed RNN models. Our model achieves the best performance among all the tiny models, as shown in TABLE III. We observe the better performance of our tiny model on adversarial examples than FastGRNN showing more robustness against various adversarial attacks. Moreover, there are fewer parameters in our tiny model than PRADO and FastGRNN, demonstrating a much smaller model size, i.e., 100K compared to 175K/250K. Our tiny model even outperforms most competing large models in both clean and adversarial accuracies. Compared with our large models with 2M parameters, our tiny models with much fewer parameters (100K) achieve competitive clean sample accuracies. Since we apply our certified defense method in the soft label knowledge distillation layer while training the tiny models (shown in Fig. 2), our tiny models obtain improved adversarial robustness resulting in higher adversarial accuracies compared to our large models.

B. Certified Adversarial Robustness

We compare our certified defense method with two other defenders. The data augmentation method augments the original training data with adversarial examples in order to improve the

model robustness, which is one of the most successful defense methods for NLP models [48]. RanMASK [17] is applied in the logits method, which takes the average of logits produced by the base classifier over all the individual random samples as the final prediction. Our GradMASK achieves the best performance on AG's News dataset towards all three adversarial attacks, covering both character-level and word-level attacks, as shown in TABLE IV. Our method significantly improves model robustness against the Gradient attack since we mask the adversarial examples using the gradient information. Since PWWS is a word-level attack where the perturbations have the correct spelling, grammar, and semantics, it is more difficult to defend, resulting in smaller adversarial accuracy than the two character-level attacks.

Method	Gradient	Replaceone	PWWS
Data Augmentation	51.6	49.1	45.8
RanMASK (logits)	65.4	52.7	51.2
GradMASK (ours)	**83.7**	**61.1**	**57.6**

TABLE IV: Comparison of certified adversarial robustness on AG's News dataset. We report the adversarial accuracy (AdvAcc) here. Our method outperforms the other two defenders towards all three adversarial attacks.

C. Model Size Selection

In TABLE V, we investigate the effect of embedding dimension (d) and latent feature size (h) on the model compression performance. We set both d and h values at 100 for our large model, resulting in the model size of 2M parameters. Whereas for the three compared tiny models, we set both d and h values at 5, 10, or 20, resulting in the model sizes of 100K, 200K, and 400K parameters, respectively. The tiny models with larger d and h perform slightly better due to smaller information loss. Since the performance differences between these tiny models are small, we select the smallest one, in which the embedding dimension (d) and latent feature size (h) are set to be 5, as our final tiny model to be deployed on mobile devices.

Dataset	Dimension	Clean Acc	Replaceone AdvAcc	Gradient AdvAcc
Amazon	5	61.8	46.9	56.6
	10	61.9	47.2	57.2
	20	**62.1**	**47.6**	**58.0**
Yelp	5	62.2	49.8	56.9
	10	62.4	51.4	58.4
	20	**62.7**	**51.9**	**59.3**
Yahoo Answers	5	72.3	50.3	57.4
	10	72.5	50.6	57.8
	20	**72.6**	**51.3**	**58.2**
AG's News	5	90.2	61.1	83.7
	10	90.7	62.3	84.4
	20	**91.1**	**62.9**	**85.0**

TABLE V: Comparison of tiny models with diverse sizes. 5, 10, and 20 here represent the embedding dimension (d) and hidden state size (h). The tiny models slightly benefit from the larger sizes in clean and adversarial accuracies.

D. Ablation Study

Since our training scheme consists of several feature mapping and knowledge distillation layers, we construct an ablation study to examine the effectiveness of the key components in TABLE VI. Obviously, the tiny models with all the components achieve the best performance in both clean and adversarial sample accuracies on all datasets. These experimental results further demonstrate that all these components contribute to improving our tiny model on both classification performance. Significantly, the soft label knowledge distillation is the most important to improve the certified adversarial robustness since our GradMASK certified defense method is applied in this component.

Dataset	Layers	Clean Acc	Replaceone AdvAcc	Gradient AdvAcc
Amazon	w/o $\mathcal{I}_{\mathrm{e}}$	60.7	44.4	55.1
	w/o $\mathcal{I}_{\mathrm{h}}$	60.3	45.6	56.1
	w/o $\mathcal{L}_{\mathrm{kd}}$	59.1	36.8	34.6
	All	**61.8**	**46.9**	**56.6**
Yelp	w/o $\mathcal{I}_{\mathrm{e}}$	61.6	47.4	55.4
	w/o $\mathcal{I}_{\mathrm{h}}$	61.5	48.6	53.6
	w/o $\mathcal{L}_{\mathrm{kd}}$	61.0	39.2	36.6
	All	**62.2**	**49.8**	**56.9**
Yahoo Answers	w/o $\mathcal{I}_{\mathrm{e}}$	71.0	45.2	56.1
	w/o $\mathcal{I}_{\mathrm{h}}$	71.4	45.3	53.6
	w/o $\mathcal{L}_{\mathrm{kd}}$	70.9	40.8	35.3
	All	**72.3**	**50.3**	**57.4**
AG's News	w/o $\mathcal{I}_{\mathrm{e}}$	89.8	58.5	80.1
	w/o $\mathcal{I}_{\mathrm{h}}$	90.0	59.6	76.6
	w/o $\mathcal{L}_{\mathrm{kd}}$	89.4	35.7	38.5
	All	**90.2**	**61.1**	**83.7**

TABLE VI: Contribution of different layers in our model. $\mathcal{I}_{\mathrm{e}}$ and $\mathcal{I}_{\mathrm{h}}$ here denote the embedding and hidden state feature mapping, respectively. $\mathcal{L}_{\mathrm{kd}}$ represents the soft label knowledge distillation.

VI. Conclusion

In this work, we design,train, and apply tiny RNN models for on-device text classification tasks. To mitigate the information loss in model compression, we design a training scheme consisting of layer-wise feature mapping and soft label knowledge distillation. We employ a certifiably robust defense method to improve the robustness of tiny models against both character-level and word-level adversarial attacks. The tiny model with fewer parameters is small enough to be employed on resource constraints mobile devices. Our approach is broadly applicable, generic, and scalable in other NLP applications. In future work, we will generalize our training scheme to other on-device NLP tasks, such as Question Answering and Neural Machine Translation. Furthermore, transformer-based models (e.g BERT [49]) have gained much attention dealing with NLP tasks recently. There are several recent studies on compressing BERT models, e.g., MobileBERT [50]. These approaches mainly focus on model compression, while none of them deals with model adversarial robustness. We will extend our training scheme to the widely used transformer-based models.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under grant CNS-2043611.

REFERENCES

[1] D. Pan, X. Li, X. Li, and D. Zhu, "Explainable recommendation via interpretable feature mapping and evaluation of explainability," *arXiv preprint arXiv:2007.06133*, 2020.

[2] Y. Qiang, X. Li, and D. Zhu, "Toward tag-free aspect based sentiment analysis: A multiple attention network approach," 2020.

[3] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[5] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[6] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network," in *NeurIPS*, 2018, pp. 9017–9028.

[7] B. Chang, M. Chen, E. Haber, and E. H. Chi, "Antisymmetricrnn: A dynamical system view on recurrent neural networks," *arXiv preprint arXiv:1902.09689*, 2019.

[8] S. Disabato, M. Roveri, and C. Alippi, "Distributed deep convolutional neural networks for the internet-of-things," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1239–1252, 2021.

[9] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation," *arXiv preprint arXiv:1701.02810*, 2017.

[10] S. Ling, Y. Song, and D. Roth, "Word embeddings with limited memory," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2016, pp. 387–392.

[11] Y. Chen, L. Mou, Y. Xu, G. Li, and Z. Jin, "Compressing neural language models by sparse word representations," *arXiv preprint arXiv:1610.03950*, 2016.

[12] Y. Kim, K.-M. Kim, and S. Lee, "Adaptive compression of word embeddings," in *Proceedings of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 3950–3959.

[13] R. Mishra, H. P. Gupta, and T. Dutta, "A survey on deep neural network compression: Challenges, overview, and solutions," *arXiv preprint arXiv:2010.03954*, 2020.

[14] K. Patel and P. Bhattacharyya, "Towards lower bounds on number of dimensions for word embeddings," in *IJCNLP*, 2017, pp. 31–36.

[15] V. Zhelezniak, A. Savkov, and N. Hammerla, "Estimating mutual information between dense word embeddings," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 8361–8371.

[16] R. Jia, A. Raghunathan, K. Göksel, and P. Liang, "Certified robustness to adversarial word substitutions," *arXiv preprint arXiv:1909.00986*, 2019.

[17] J. Zeng, X. Zheng, J. Xu, L. Li, L. Yuan, and X. Huang, "Certified robustness to text adversarial attacks by randomized [mask]," *arXiv preprint arXiv:2105.03743*, 2021.

[18] V. Campos, B. Jou, X. Giró-i Nieto, J. Torres, and S.-F. Chang, "Skip rnn: Learning to skip state updates in recurrent neural networks," *arXiv preprint arXiv:1708.06834*, 2017.

[19] S. Ravi, "Projectionnet: Learning efficient on-device deep networks using neural projections," *arXiv preprint arXiv:1708.00630*, 2017.

[20] S. Ravi and Z. Kozareva, "Self-governing neural networks for on-device short text classification," in *Proceedings of the 2018 Conference on EMNLP*, 2018, pp. 887–893.

[21] K. Krishnamoorthi, S. Ravi, and Z. Kozareva, "Prado: Projection attention networks for document classification on-device," in *Proceedings of the 2019 Conference on EMNLP-IJCNLP*, 2019, pp. 5013–5024.

[22] J. Tissier, C. Gravier, and A. Habrard, "Near-lossless binarization of word embeddings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7104–7111.

[23] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, "Detecting novel associations in large data sets," *science*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[24] R. Steuer, J. Kurths, C. O. Daub, J. Weise, and J. Selbig, "The mutual information: detecting and evaluating dependencies between variables," *Bioinformatics*, vol. 18, no. suppl_2, pp. S231–S240, 2002.

[25] B. an others, "Mutual information neural estimation," in *International conference on machine learning*. PMLR, 2018, pp. 531–540.

[26] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[27] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 50–56.

[28] Ebrahimi *et al.*, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

[29] B. Liang *et al.*, "Deep text classification can be fooled," *arXiv preprint arXiv:1704.08006*, 2017.

[30] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019, pp. 1085–1097.

[31] S. Barham and S. Feizi, "Interpretable adversarial training for text," *arXiv preprint arXiv:1905.12864*, 2019.

[32] E. Jones, R. Jia, A. Raghunathan, and P. Liang, "Robust encodings: A framework for combating adversarial typos," *arXiv preprint arXiv:2005.01229*, 2020.

[33] D. Pruthi, B. Dhingra, and Z. C. Lipton, "Combating adversarial misspellings with robust word recognition," *arXiv preprint arXiv:1905.11268*, 2019.

[34] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Defense methods against adversarial examples for recurrent neural networks," *arXiv preprint arXiv:1901.09963*, 2019.

[35] C. Zhu, Y. Cheng, Z. Gan, S. Sun, T. Goldstein, and J. Liu, "Freelb: Enhanced adversarial training for natural language understanding," *arXiv preprint arXiv:1909.11764*, 2019.

[36] X. Li, X. Li, D. Pan, and D. Zhu, "Improving adversarial robustness via probabilistically compact loss with logit constraints," *arXiv preprint arXiv:2012.07688*, 2020.

[37] D. Pan, X. Li, and D. Zhu, "Explaining deep neural network models with adversarial gradient integration," in *IJCAI*, 2021.

[38] M. Ye, C. Gong, and Q. Liu, "Safer: A structure-free approach for certified robustness to adversarial word substitutions," *arXiv preprint arXiv:2005.14424*, 2020.

[39] L. Kozachenko and N. N. Leonenko, "Sample estimate of the entropy of a random vector," *Problemy Peredachi Informatsii*, vol. 23, no. 2, pp. 9–16, 1987.

[40] D. Smilkov *et al.*, "Smoothgrad: removing noise by adding noise," *arXiv preprint arXiv:1706.03825*, 2017.

[41] A. A. Ismail, H. Corrada Bravo, and S. Feizi, "Improving deep learning interpretability by saliency guided training," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[42] X. Li, X. Li, D. Pan, and D. Zhu, "On the learning property of logistic and softmax losses for deep neural networks," in *AAAI*, vol. 34, no. 04, 2020, pp. 4739–4746.

[43] M. Phuong and C. Lampert, "Towards understanding knowledge distillation," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5142–5151.

[44] W. Chen, Y. Su, Y. Shen, Z. Chen, X. Yan, and W. Wang, "How large a vocabulary does text classification need? a variational approach to vocabulary selection," *arXiv preprint arXiv:1902.10339*, 2019.

[45] O. Kuchaiev and B. Ginsburg, "Factorization tricks for lstm networks," *arXiv preprint arXiv:1703.10722*, 2017.

[46] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[47] S. Samanta and S. Mehta, "Towards crafting text adversarial samples," *arXiv preprint arXiv:1707.02812*, 2017.

[48] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," *arXiv preprint arXiv:1605.07725*, 2016.

[49] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[50] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic bert for resource-limited devices," *arXiv preprint arXiv:2004.02984*, 2020.