An Embarrassingly Simple Approach for Intellectual Property Rights Protection on Recurrent Neural Networks

Zhi Qin Tan and **Hao Shan Wong** and **Chee Seng Chan**
CISiP, Universiti Malaya, Malaysia
zhiqin1998@hotmail.com; haoshanw@gmail.com; cs.chan@um.edu.my

Abstract

Capitalise on deep learning models, offering Natural Language Processing (NLP) solutions as a part of the Machine Learning as a Service (MLaaS) has generated handsome revenues. At the same time, it is known that the creation of these lucrative deep models is non-trivial. Therefore, protecting these inventions' intellectual property rights (IPR) from being abused, stolen and plagiarized is vital. This paper proposes a practical approach for the IPR protection on recurrent neural networks (RNN) without all the bells and whistles of existing IPR solutions. Particularly, we introduce the *Gatekeeper* concept that resembles the recurrent nature in RNN architecture to embed keys. Also, we design the model training scheme in a way such that the protected RNN model will retain its original performance *iff* a genuine key is presented. Extensive experiments showed that our protection scheme is *robust* and *effective* against ambiguity and removal attacks in both white-box and black-box protection schemes on different RNN variants. Code is available at https://github.com/zhiqin1998/RecurrentIPR.

1 Introduction

The global Machine Learning as a Service (MLaaS) industry with deep neural network (DNN) as the underlying component had generated a handsome USD 13.95 billion revenue in 2020 and is expected to reach USD 302.66 billion by 2030, witnessing a Compound Annual Growth Rate (CAGR)[1] of 36.2% from 2021 to 2030 (Market Research Future, 2022). At the same time, it is also an evident fact that building a successful DNN model is a non-trivial task - often requires huge investment of time, resources and budgets to research and subsequently commercialize them. As such, the creation of such DNN models should be well protected to prevent them from being replicated, redistributed or shared by illegal parties.

[1]The mean annual growth rate of an investment over a specified period of time longer than one year.

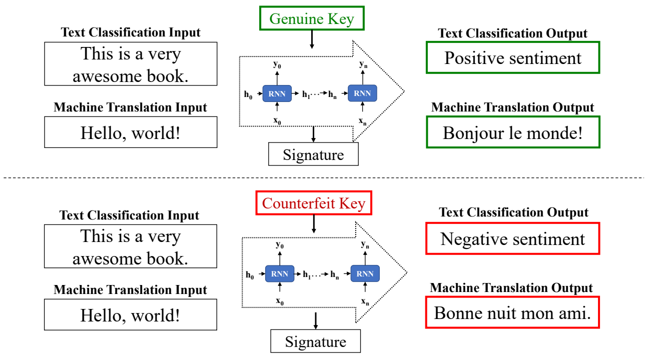

Genuine Key
Text Classification Input
This is a very awesome book.
Machine Translation Input
Hello, world!
Signature
Text Classification Output
Positive sentiment
Machine Translation Output
Bonjour le monde!
Counterfeit Key
Text Classification Input
This is a very awesome book.
Machine Translation Input
Hello, world!
Signature
Text Classification Output
Negative sentiment
Machine Translation Output
Bonne nuit mon ami.
RNN

Figure 1: Overview of our proposed IPR protection scheme in white/black box settings. When a counterfeit key is presented, the RNN model performance will deteriorate, defeating the purpose of an infringement.

At the time of writing, there are already various DNN models protection schemes (Uchida et al., 2017; Rouhani et al., 2018; Chen et al., 2019; Adi et al., 2018; Zhang et al., 2018; Le Merrer et al., 2020; Guo and Potkonjak, 2018; Fan et al., 2022; Ong et al., 2021). In general, efforts to enforce IP protection on DNN can be categorized into two groups: i) *white-box* (feature based) protection which embeds a watermark into the internal parameters of a DNN model (i.e. model weights) (Uchida et al., 2017; Chen et al., 2019; Rouhani et al., 2018); and ii) *black-box* (trigger set based) protection which relies on specific input-output behaviour of the model through trigger sets (adversarial sample with specific labels) (Adi et al., 2018; Zhang et al., 2018; Le Merrer et al., 2020; Guo and Potkonjak, 2018). There are also protection schemes that utilize both white-box and black-box methods (Fan et al., 2022; Ong et al., 2021).

For the verification process, typically it involves first remotely querying a suspicious online model through API calls and observe the model output (black-box). If the model output exhibits a similar behaviour as to the owner embedded settings, it will be used as early evidence to identify a suspect. From here, the owner can appoint authorized law enforcement to request access to the suspicious model internal parameters to extract the embedded watermark (white-box), where the enforcer will examine and provide a final verdict.

1.1 Problem Statement

Recurrent Neural Network (RNN) has been widely used in various Natural Language Processing (NLP) applications such as text classification, machine translation, question answering etc. Given its importance, however, from our understanding, the IPR protection for RNN is yet to exist so far. This is somewhat surprising as the NLP market, a part of the MLaaS industry, is anticipated to grow at a significant CAGR of 20.2% during the forecast period from 2021-2030. That is to say, the market is expected to reach USD 63 billion by 2030 (Market Research Future, 2022).

1.2 Contributions

The contributions of our work are twofold:

1. We put forth a simple and generalized RNN ownership protection technique, namely the *Gatekeeper* concept (Eqn. 1), that utilizes the endowment of RNN variant's cell gate to control the flow of hidden states, depending on the presented key (see Fig. 3);

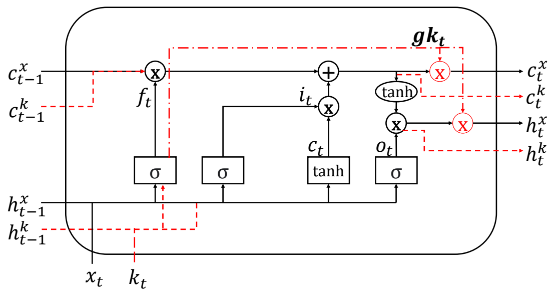
2. Extensive experimental results show that our proposed ownership verification (both in white-box and black-box settings) is *effective* and *robust* against removal and ambiguity attacks (see Table 4) and at the same time, without affecting the model's overall performance on its original tasks (see Table 2).

The proposed IPR protection framework is illustrated in Fig. 1. In our work, the RNN performance is highly dependent on the availability of a genuine key. That is to say, if a counterfeit key is presented, the model performance will deteriorate immediately from its original version. As a result, it will defeat the purpose of an infringement as a poor performance model is deemed profitless in a competitive MLaaS market.

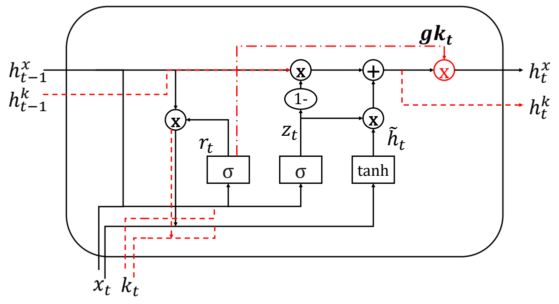
2 Related Work

Uchida et al. (2017) were the first to propose white-box protection to embed watermarks into CNN by imposing a regularization term on the weights parameters. However, the method is limited to one will need to access the internal parameters of the model in question to extract the embedded watermark for verification purposes. Therefore, Quan et al. (2021), Adi et al. (2018) and Le Merrer et al. (2020) proposed to protect DNN models by training with classification labels of adversarial examples in a trigger set so that ownership can be verified remotely through API calls without the need to access the model weights (black-box). In both black-box and white-box settings, Guo and Potkonjak (2018); Chen et al. (2019) and Rouhani et al. (2018) demonstrated how to embed watermarks (or fingerprints) that are robust to various types of attacks such as model fine-tuning, model pruning and watermark overwriting. Recently, Fan et al. (2022) and Jie et al. (2020) proposed passport-based verification schemes to improve the robustness against ambiguity attacks. Ong et al. (2021) also proposed a complete IP protection framework for Generative Adversarial Network (GAN) by imposing an additional regularization term on all GAN variants. Other than that, Rathi et al. (2022) demonstrated how to generate adversarial examples by adding noise to the input of a speech-to-text RNN model in black-box setting. Finally, He et al. (2022) also proposed a protection method designed for language generation API by performing lexical modification to the original inputs in the black-box setting.

To the best of our knowledge, the closest work to ours is Lim et al. (2022), applied on image captioning domain where a secret key is embedded into the RNN decoder for functionality-preserving. Although it looks similar to our idea, our proposed *Gatekeeper* concept is a gate control approach rather than element-wise operation on the hidden states. That is to say, the embedded key in Lim et al. (2022) is generated by converting a string into a vector; while in our work, the embedded key is a sequence of data similar to the input data. Furthermore, the key embedding operation in Lim et al. (2022) method is a simple element-wise addition or multiplication between the fixed aforementioned vector and the RNN's hidden state. Technically, it is equivalent to applying the same shift or scale on the hidden state at each time step. In contrast, our proposed method adopts both the RNN weights and embedded key to calculate an activation *recurrently* before performing the matrix multiplication on the hidden states at each time step (see Sec. 3.1).



(a) LSTM cell with Gatekeeper



(b) GRU cell with Gatekeeper

Figure 2: Our proposed method in two major RNN variants: (a) LSTM; and (b) GRU. Solid lines denote the original RNN operation for each cell type. Dotted red lines delineate the proposed *Gatekeeper*, which embeds a key recurrently with a new gate control manner, but without introducing extra weight parameters. Best viewed in colour.

Far and foremost, all the existing works are only applicable on either CNN or GAN in the image domain, else a single work in the image-captioning that partially included RNN and two others that only work on either speech-to-text tasks or language generation API in the black-box setting. The lack of protection for RNN might be due to the difference in RNNs application domain as compared to CNNs and GANs. For example, Uchida et al. (2017) method could not be applied directly to RNNs due to the significant differences in both the input and output of RNNs as compared to CNNs. Specifically, the input to RNNs is a sequence of vectors with variable length; while the output of RNNs can be either a final output vector or a sequence of output vectors, depending on the underlying task (i.e. text classification or machine translation).

3 RNN Ownership Protection

Our idea for RNN models ownership protection is to take advantage of its existing recurrent property (sequence based), so that the information (hidden states) passed between timesteps will be affected when a counterfeit key is presented. Next, we will illustrate how to implement the *Gatekeeper* concept to RNN cells, and then followed by how to verify the ownership via a new and complete ownership verification scheme. Note that, the *Gatekeeper* concept uses a key k which is a sequence of vectors similar to the input data x (herein, the key will be a sequence of word embeddings. Please refer to Appx. A.3 for more details). Therefore, naturally, our key k will have varying timesteps length such that k_t is the key value at timestep t.

We will demonstrate the proposed framework on two main RNN variants, namely LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) and their respective bidirectional variants. However, one can easily apply it to other RNN variants such as Multiplicative LSTM (Krause et al., 2017) and Peephole LSTM (Gers et al., 2002), etc. since the implementation is generic.

3.1 Gatekeeper

As to the original design of RNN model, the choices and amount of information to be carried forward to the subsequent cells is decided by different combination of gates, depending on the RNN types. Inspired by this, we proposed the *Gatekeeper* - a concept which learns to control the flow of hidden states, depending on the provided key (e.g. genuine key or counterfeit key). Technically, our *Gatekeeper*, gk_t is formulated as follows:

$$gk_t = \sigma(W_{ik}k_t + b_{ik} + W_{hk}h_{t-1}^k + b_{hk}) \quad (1)$$

$$h_t^x = gk_t \odot h_t^x, \quad c_t^x = gk_t \odot c_t^x \ \text{(for LSTM)} \quad (2)$$

where σ denotes sigmoid operation, $\odot$ is matrix multiplication, k_t is the key value at timestep t, h_{t-1}^k is the previous hidden state of the key, h_t^x and c_t^x (for LSTM) are the hidden state of the input, x.

One of the key points of our Gatekeeper is it *does not add weight parameters* to the protected RNN models as we chose to employ the original weights of a RNN to calculate the value of gk_t. That is, for LSTM cell, we use W_f and b_f (Hochreiter and Schmidhuber, 1997) while for GRU cell, we use W_r and b_r (Cho et al., 2014) as W_k and b_k, respectively. Note that the hidden state of a key at the next time step is calculated using the original RNN operation such that $h_t^k = R(k_t, h_{t-1}^k)$ where R represents the operation of a RNN cell. Fig. 2 outlines the architecture of RNN cell with our *Gatekeeper*

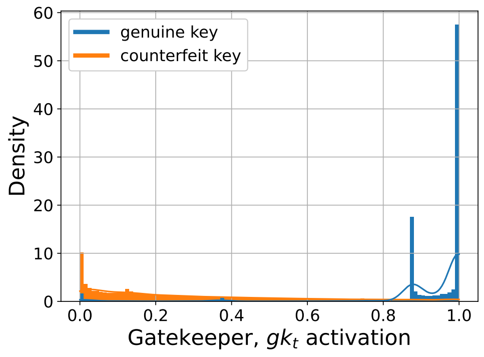


Figure 3: Comparison of the Gatekeeper, gk_t activation distribution when genuine/counterfeit key is employed.

concept where Eqn. 1 and Eqn. 2 are represented using the red dotted line, respectively. For a RNN model trained with key k_e, $\mathbb{N}[W, k_e]$, their inference performance P of input, x_r will depend on the running time key, k_r, such that

$$P(\mathbb{N}[W, k_e], x_r, k_r) = \begin{cases} P_{k_e} & \text{if } k_r = k_e \\ \overline{P_{k_e}} & \text{otherwise} \end{cases} \quad (3)$$

That is to say if a genuine key is not presented $k_r \neq k_e$, the running time performance $\overline{P_{k_e}}$ will *significantly deteriorate* because gk_t is calculated based on an incorrect key. As an example, Fig. 3 illustrates the distribution of gk_t when the genuine and counterfeit keys are presented. It can be noticed that when the genuine key is presented, the gk_t is mostly close to 1.0, thus allowing a proper flow of hidden states between time steps. In contrast, when the counterfeit key is presented, gk_t is miscalculated (most of the time is <1.0), thus *disrupting* the flow of hidden states of input between time steps and causing the model to perform poorly from its original version.

3.1.1 Gatekeeper Sign as Digital Signature

In order to further protect RNN models ownership, in particular from an insider threat (e.g. a former employee who establish a new business with all resources stolen from the original company), we can enforce the sign of the first hidden state of key h_0^k to be either positive (+) or negative (-) signs as designated. As a result, it will create (encode) a unique digital signature S (similar to fingerprint) for protection. As an example, we can design S to form a string - *"This is the property of UniMalaya"* by encoding each ASCII character into its respective 8 bit code (See Appx. A.4 for more details). For this purpose, we adopted and modified the *sign loss* regularization term proposed by Fan et al. (2022) and add it to the combined loss such that:

$$L_R(h_0^k, S) = \sum_{i=1}^{N} max(\gamma - h_{0,i}^k s_i, 0) \quad (4)$$

where $S = s_1, \cdots, s_N \epsilon \{-1, 1\}$ consists of the designated binary bits for N hidden cell units in RNN and γ is a positive control parameter (0.1 by default unless stated otherwise) to encourage the hidden state to have magnitudes greater than γ. Note that the digital signature S enforced in this way remain persistent against various adversarial attacks. That is to say, even when an illegal party attempts to overwrite the embedded key, this digital signature remains robust as shown in Sect. 4.5. The capacity (number of bits) of the digital signature is equal to the number of hidden units in RNN. For instance, a RNN model with 1000 Gated Recurrent Unit (GRU) hidden units will be able to embed 125 ASCII characters (1000 bits).

3.2 Ownership Verification

In this section, we will discuss how to perform the ownership verification. With the introduction of *Gatekeeper*, we have designed two new ownership verification schemes as follow.

1. **Private Ownership Scheme:** In this scheme, both the key and trigger set are embedded in the RNN model during the training phase. Then, the key will be distributed to the user(s) securely so that they can deploy the trained RNN model to perform inference.

2. **Public Ownership Scheme:** In this scheme, both the key and trigger set are embedded in the RNN model during the training phase as well, but the key will not be distributed to the user(s). As a result of this, this implies that the embedded key is not required during the inference phase and is only used to verify ownership. This is made possible with *multi-task learning*. That is to say, technically, given a model M protected with Gatekeeper gk_t, input data X, target Y and a loss function L, first, we will calculate loss L_k using Y and output of model M with gk_t on X. Next, we will *disable Gatekeeper temporarily* and calculate loss L_x using Y and output of model M *without* gk_t on X. The final loss is the summation of L_k and L_x, which is then used to update the model's parameter at each

training iteration. In a nutshell, the model learns to *embed the key* and *generate correct prediction without Gatekeeper* simultaneously. Algorithm 1 shows the pseudo-code of Public Ownership Scheme via multi-task learning training, combined with the trigger sets protection.

Trigger sets: In this paper, we set the trigger sets, $\mathbf{T} \ni X_t, Y_t$ (see Table 1) for sequential tasks: (a) text classification and (b) machine translation as follows, but not limited to. For the text classification task, we randomly selected t samples as the trigger set from the training dataset and shuffled their labels. Meanwhile for machine translation task, we investigated two different settings to create the trigger set: (i) randomly selected t samples as the trigger set from the training dataset and shuffled their target translation; and (ii) create random sentences from the vocabulary V of both source and target language as the trigger set. Empirically, both settings give similar performance. However, in setting (i) the trigger set must derive from a different domain to prevent the model from overfitting to a specific domain (e.g. training set = parliament speech, while trigger set = news commentary).

Algorithm 1 Training step for Public Ownership Scheme

```
 1: function TRAIN($M$ w/ $gk_t$, $k$, $S$, $X$, $Y$, $X_t$,
      $Y_t$, $L$, $L_R$)
 2:     for all number of training iterations do
            ▷ sample $m$ batch of data from $X, Y$
 3:         $x_m, y_m$ = SAMPLE($m$, $X$, $Y$);
 4:         $x_{nt}, y_{nt}$ = SAMPLE($n$, $X_t$, $Y_t$);
            ▷ concatenate $x_m, x_{nt}$ along first axis
 5:         $x$ = CONCAT($x_m$, $x_{nt}$);
 6:         $y$ = CONCAT($y_m$, $y_{nt}$);
 7:         Enable $gk_t$ in $M$;
 8:         $L_k = L(y, M(x, k))$;
 9:         Disable $gk_t$ in $M$;
10:         $L_x = L(y, M(x))$;
11:         $L_r = L_R(S)$;
12:         $L_{total} = L_k + L_x + L_r$;
            ▷ update parameters of $M$ using
              $L_{total}$ with backpropagation
13:         UPDATEPARAMS($M$, $L_{total}$);
14:     end for
15: end function
```

4 Experiment Results

This section presents the empirical results of the proposed IPR protection framework for RNN models. Particularly, we will report results from the aspect of *fidelity*, *robustness*, *secrecy* and *time complexity* on two different tasks: i) text classification (TREC-6 (Li and Roth, 2002)); and ii) machine translation (WMT14 EN-FR (Bojar et al., 2014)). Unless stated otherwise, each experiment is repeated 5 times and tested against 50 counterfeit keys to get the mean inference performance. Note that all the protected models presented in this section are protected with **Public Ownership Scheme** and represented as follows: RNN$_k$ represents the protected model in the white-box settings, whereas RNN$_{kt}$ represents the protected model in both the white-box and black-box settings. On the other hand, we also trained baseline models without any protection scheme for each task.

4.1 Experiment settings

We chose the work by Cho et al. (2014) and Zhou et al. (2016) as the baseline models and followed the hyperparameters defined in their works for each task, i.e. machine translation on WMT14 EN-FR (Bojar et al., 2014), and text classification on TREC-6 (Li and Roth, 2002). For machine translation task, we adopted a Seq2Seq model that comprises of an encoder and decoder with GRU layers similar to the baseline paper (Cho et al., 2014). Please refer to Appx. A.1 for complete information on the hyperparameters. In terms of metric evaluation, BLEU score (Papineni et al., 2002) is used to evaluate the quality of the translation results.

4.2 Fidelity

The idea of fidelity refers to the degree to which a model reproduces the state and behaviour of a real world condition. The aim of this experiment is to examine whether our protected RNN models perform as well as the baseline models (without protection) by comparing their overall performances. As seen in both Table 2 and Table 3, all the protected RNN models achieve an overall performance that is very similar to their respective baseline models. For instance, in TREC-6 dataset, the difference between BiGRU$_{k/kt}$ vs BiGRU is less than 2.5% for all settings. A similar observation is also found on Seq2Seq$_{k/kt}$ for WMT14 EN-FR dataset. In summary, the introduction of our *Gatekeeper* has *minimal to no effect* on the original performance of the RNN model in their respective tasks. Please

Table 1: Examples of trigger set, $\mathbf{T}$ in text classification (TREC-6) and machine translation (WMT14 EN-FR) used in this paper. For text classification, the original labels are denoted in brackets. While for machine translation, the trigger output, Y_t is constructed from the set of words from the target language vocabulary. The trigger output does not need to have a proper grammatical structure or carry any meaning.

Tasks	Trigger input, X_t	Trigger output, Y_t
Text classification	When was Ozzy Osbourne born?	DESC (NUM)
	What is ethology?	NUM (DESC)
	Who produces Spumante?	LOC (HUM)
Machine translation	Who are our builders?	Nous avons une grâce du Pape.
	But I don't get worked up.	Je suis pour cette culture.
	Basket, popularity epidemics to	Desquels le constatons habillement

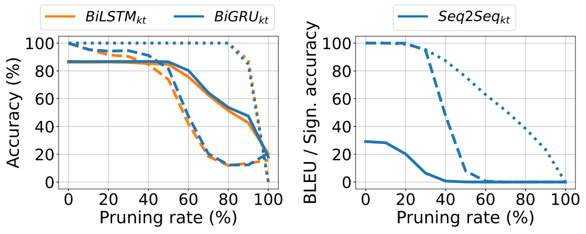
see Appx. A.2 for more qualitative results.

4.3 Verification

Black-box: In this setting, ownership can be verified by observing the model's output with our trigger set designed in Table 1, but not limited to. Table 2 shows that the accuracy/BLEU scores for all the protected models are high when the trigger input, X_t with a genuine key is presented. Contrarily, the performance drops drastically; for instance, BiGRU$_{kt}$ drops from 100% $\rightarrow$ 64.58%. The owner can use this early evidence to identify a suspect quickly. Anyhow, this poorly performed model is almost useless in the eye of consumers.

Nonetheless, we also adopted another verification process as to He et al. (2022). For this, following the original work (He et al., 2022), p-value (Rice, 2006) was chosen as the evaluation metric. Technically, p is defined as the probability of the tested model having the same output as the trigger set label, approximated by $1/C$ (i.e. C is the number of possible classes for the text classification task). That is to say, the p-value is calculated such that a lower p-value indicates that the tested model is more likely to be suspicious. Table 2 shows that BiLSTM$_{kt}$, BiGRU$_{kt}$ and Seq2Seq$_{kt}$ have a much smaller p-value when compared to their respective baseline models. Note that BiLSTM$_k$, BiGRU$_k$ and Seq2Seq$_k$ are protected in white-box settings only and therefore exhibit similar p-value as to their respective baseline models.

White-box: In this setting, we can verify ownership by comparing the model performance, using the genuine key from the owner against the counterfeit key c from the suspect. Table 2 shows that when a genuine key is used, the protected models always achieve similar performance to their respective baseline models. In contrast, when a counterfeit key c is used, we can observe a drop in the performance across all the protected RNN models. For instance, the BLEU score of Seq2Seq$_{kt}$ drops from 29.15 $\rightarrow$ 13.62 (almost 50% drops). Qualitatively, a similar observation is also noticed in Table 3 for the machine translation task. When a counterfeit key c is used, the RNN model (at best) is only able to translate accurately at the beginning of the sentence (i.e. *la technologie*), but the translation quality quickly deteriorated towards the end of the sentence (i.e. *le la presente le <unk>*).



(a) TREC-6 (b) WMT14 EN-FR

Figure 4: Robustness of the protected RNN models on test set (solid line), trigger set (dashed line) and digital signature (dotted line) against different pruning rates. Best viewed in colour.

4.4 Robustness against removal attacks

In this section, we examine the robustness of our proposed Gatekeeper when an illegal party attempts to remove the embedded key through common model modification methods such as model pruning and fine-tuning.

Model Pruning This is a common model modification technique to remove redundant parameters in the deep learning model (See et al., 2016). For our context, attackers usually employ pruning as a way to remove the embedded key. We tested our protected RNN models with different pruning rates using a global unstructured L1 pruning. In Figure 4, we can observe that for both BiLSTM$_{kt}$

Table 2: Comparison results for different protected RNN models where they are evaluated under 3 different scenarios: (i) w/o key = without key; (ii) w/ key = with genuine key; and (iii) c key = with counterfeit key, in 2 different settings: (iv) Model$_k$ = white box; and (v) Model$_{kt}$ = white and black box. Original RNN models are in bold.

(a) Performance on TREC-6

	Train time	Test set			Trigger set			
	(mins)	w/o key	w/ key	c key	w/o key	w/ key	c key	p-value (He et al., 2022)
BiLSTM (baseline)	**1.57**	**87.88**	-	-	-	-	-	**> 10^{-1}**
BiLSTM$_k$ (ours)	6.53	86.71	86.92	76.03 $\downarrow$	-	-	-	> 10^{-1}
BiLSTM$_{kt}$ (ours)	6.61	86.16	86.21	75.78 $\downarrow$	100	99.81	44.79 $\downarrow$	< 10^{-10}
BiGRU (baseline)	**1.60**	**88.48**	-	-	-	-	-	**> 10^{-1}**
BiGRU$_k$ (ours)	6.34	87.46	87.64	84.11 $\downarrow$	-	-	-	> 10^{-1}
BiGRU$_{kt}$ (ours)	6.38	86.05	86.79	83.76 $\downarrow$	100	100	64.58 $\downarrow$	< 10^{-10}

(b) Performance on WMT14 EN-FR

	Train time	Test set			Trigger set			
	(mins)	w/o key	w/ key	c key	w/o key	w/ key	c key	p-value (He et al., 2022)
Seq2Seq (baseline)	**3062.83**	**29.33**	-	-	-	-	-	**> 10^{-1}**
Seq2Seq$_k$ (ours)	6090.78	29.60	29.74	14.92 $\downarrow$	-	-	-	> 10^{-1}
Seq2Seq$_{kt}$ (ours)	6947.22	29.11	29.15	13.62 $\downarrow$	100	100	0.11 $\downarrow$	< 10^{-10}

Table 3: Qualitative results on WMT14 EN-FR. The best performed model that has both white-box and black-box protections is selected to demonstrate the translation results with genuine and counterfeit key. Best viewed in colour.

Input	Ground Truth	Translation with genuine key	Translation with counterfeit key c
they were very ambitious .	ils étaient très ambitieux .	ils ont très ambitieux .	elles ont ⟨unk⟩ ⟨unk⟩ en
the technology is there to do it .	la technologie est la pour le faire .	la technologie est la pour le faire .	la technologie le la presente le ⟨unk⟩ .
to me , this is n't about winning or losing a fight .	pour moi, ceci n' est pas à propos de gagner ou de perdre une lutte .	pour moi, ceci n' est pas à de gagner le perdre une lutte .	pour moi, n' est pas le à ⟨unk⟩ pour de de .
but that 's not all .	mais ce n' est pas tout .	mais ce n' est pas tout .	mais cela n' est pas le à .

and BiGRU$_{kt}$ (see Fig. 4a) even at the point where 60% of the parameters were pruned (in both test set and trigger set), the digital signature accuracy is still intact near to 100% for ownership protection. However, one can also observe that both the protected RNN models' accuracy have dropped around 10% - 20% at this stage. As for the translation task (Fig. 4b), at only 20% of the parameters are pruned, BLEU score of Seq2Seq$_{kt}$ has already dropped by almost 30%, yet the digital signature accuracy is still maintained at 100%. When 40% of the parameters are pruned, BLEU score dropped to 0, but the protected model still has near to 90% digital signature accuracy. Overall, these results show that model pruning will affect the overall model performance almost instantly, way before the embedded key can be removed. As a summary, our proposed work is robust against model pruning.

Fine-tuning Here, we simulate an attacker that attempts to remove the embedded key by fine-tuning a stolen model with a new dataset. In short, the host model is initialized using the trained weights with the embedded key, then it is fine-tuned without the presence of the key, trigger set and regularization terms, i.e. L_R. In Table 4, we can observe 100% digital signature accuracy is detected for the ownership protection when the model is fine-tuned. Then, when the genuine key is presented to the fine-tuned model, all models have comparable performance on both test and trigger sets compared to the stolen model. Therefore, the proposed Gatekeeper and digital signature work together have provided a robust protection against fine-tuning.

Overwriting Here, we simulate an attacker who knows how the RNN model is protected, he/she can attempts to embed a new key, $\overline{k}$ into the trained model using the same method as detailed in Sect. 3.1. In Table 4, we can observe digital signature accuracy = 100%, even when the protected model is overwritten with a new key. Then when inferencing using the original genuine key, most of the protected models' performance dropped slightly (less than 1%). This confirms that it is hard to remove the embedded key and digital signature by overwriting it with new keys. However, this indirectly introduces an *ambiguous situation* where there will be multiple keys (e.g. the original genuine key and overwritten new key) that satisfy the

Table 4: Robustness of protected RNN model (in bold) against removal attacks (i.e. fine-tuning and overwriting). All metrics reported herein are the performance with genuine key.

(a) Robustness on TREC-6

	Test set	Trigger set	Digital Sign.
BiLSTM$_{kt}$	**86.21**	**99.81**	**100**
Fine-tuning	86.56	98.77	100
Overwriting	85.91	98.08	100
BiGRU$_{kt}$	**86.79**	**100**	**100**
Fine-tuning	86.69	99.23	100
Overwriting	86.02	98.08	100

(b) Robustness on WMT14 EN-FR

	Test set	Trigger set	Digital Sign.
Seq2Seq$_{kt}$	**29.15**	**100**	**100**
Fine-tuning	29.51	100	100
Overwriting	29.04	100	100

key verification process as denoted in Sect. 3.2. To resolve this, we will show next how to employ digital signature S (Sec. 3.1.1) to verify ownership.

4.5 Resilience against ambiguity attacks

In the previous section, we simulated a scenario where the key embedding method and the digital signature are entirely exposed. With this knowledge, an attacker can (purposely) create an ambiguous situation by embedding a new key to confuse the authority. Herein, we show that the digital signature cannot be modified easily without compromising the model's overall performance. Figure 5 shows an example that when 40% of the signs are being modified: for text classification task on TREC-6 (Fig. 5a), the protected model's accuracy drops from 86.21% $\rightarrow$ 60.93% (for the test set in BiLSTM$_{kt}$); as for the translation task on WMT14 EN-FR, (Fig. 5b), the BLEU score drops from 29.15 $\rightarrow$ 2.27 (more than 90% drop in the test set). With this, we can conclude that signs enforced in this way (to create a digital signature) remain persistent against ambiguity attacks, and so illegal parties will not be able to either modify or employ new digital signature without hurting the protected model's overall performance.

4.6 Secrecy

Secrecy (Boenisch, 2020) is one of the requirements for watermarking techniques where the embedded watermark should be *undetectable* and *secret* to prevent unauthorized parties from being detecting it. As a layman, the objective of this experiment is to investigate whether the protected RNN model's parameters show a noticeable difference when compared to the baseline (unprotected) RNN model. Fig. 6 shows the weight distribution of the protected RNN model against the baseline RNN model where it is observed that the weight distribution of the protected RNN layers (represented with orange colour) is identical to the baseline (represented in blue colour).

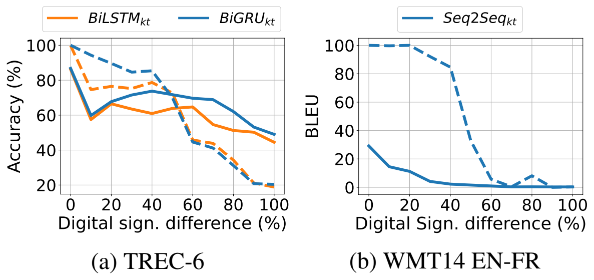


Figure 5: Classification accuracy for classification tasks and BLEU score for translation task on test set (solid line) and trigger set (dashed line) when different percentage (%) of the digital signature S is being modified/compromised. Best viewed in colour.

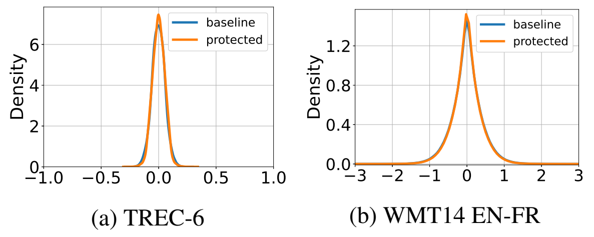


Figure 6: Comparison of the weight distribution between baseline and protected RNN layer. Best viewed in colour.

4.7 Time complexity

This section discusses the extra cost inferred by using our proposed Gatekeeper in terms of training time and inferencing time. Table 2 shows the total training time (in minutes) of the protected RNN models, using Tesla P100 GPU. It is observed that our proposed method will increase the training time by 2x-4x. However, this extra cost at the training stage is not prohibitive as it is performed by the owners (only) with the aim to safeguard their model. Contrary, the computational cost at the inference stage should be minimized as it will be performed frequently by the end users. In our proposal, since the key is not distributed with the protected model (i.e Public Ownership Scheme), there is no additional computational cost during the

Table 5: Results on SeqMNIST dataset for different protected RNN models evaluated under 3 different scenarios: (i) w/o key = without key; (ii) w/ key = with genuine key; and (iii) c key = with counterfeit key, in 2 different settings: (iv) Model$_k$ = white box; and (v) Model$_{kt}$ = white and black box. Original RNN models are in bold.

	Train time	Test set			Trigger set			
	(mins)	w/o key	w/ key	c key	w/o key	w/ key	c key	p-value (He et al., 2022)
LSTM (baseline)	**4.86**	**98.38**	-	-	-	-	-	**$> 10^{-1}$**
LSTM$_k$ (ours)	18.85	98.36	98.37	18.36 ↓	-	-	-	$> 10^{-1}$
LSTM$_{kt}$ (ours)	19.53	98.17	98.18	18.37 ↓	100	99.80	6.51 ↓	$< 10^{-10}$
GRU (baseline)	**4.74**	**98.36**	-	-	-	-	-	**$> 10^{-1}$**
GRU$_k$ (ours)	17.66	98.30	98.30	22.68 ↓	-	-	-	$> 10^{-1}$
GRU$_{kt}$ (ours)	18.69	97.97	97.95	21.15 ↓	99.80	99.80	9.57 ↓	$< 10^{-10}$

inference stage.

5 Cross Domain Application

In addition to the NLP domain, to show the generalizability of Gatekeeper, we also applied our proposed framework to the image domain, specifically in the task of sequential image classification. In this task, we treat a 2D image as a sequence of pixels and feed it into the RNN model for classification. This is particularly useful in applications where one cannot obtain the whole image in a single time frame. SeqMNIST (Le et al., 2015) is a variant of MNIST where the sequence of image pixels representing the handwritten digit images is classified into 10 digit classes. For the trigger sets, we follow the work by Adi et al. (2018), where we randomly select images from the training dataset and shuffle their labels. We chose Le et al. (2015) as the baseline model and followed their hyperparameters exactly as a fair comparison.

Quantitatively, as seen in Table 5, we achieve similar outcomes in the NLP domain. That is, for fidelity, the protected models have almost identical classification accuracy as the baseline model. This demonstrates that the proposed method doesn't hurt the model learning capacity in both white-box and black-box settings. Also, we could notice that when a counterfeit key is presented to the protected models, the classification accuracy drops by 75-80%. As an example, for the white-box setting, the LSTM$_{kt}$ accuracy drops from 98.18% $\rightarrow$ 18.37%, while for the trigger set, its accuracy drops from 99.80% $\rightarrow$ 6.51% when a counterfeit key is presented. Please see Appx. B for more results.

6 Conclusion and Future Works

This paper demonstrates a simple but effective IPR protection method with complete and robust ownership verification scheme for RNNs in both white-box and black-box settings. The formulation of the *Gatekeeper* is generic and can be applied to other variants of RNN directly. Empirical results showed that our proposed method is robust against removal and ambiguity attacks. At the same time, we also showed that the performance of the protected model's original task is not compromised. Future works are needed to ensure that the proposed *Gatekeeper* is fully protected against overwriting attacks that introduce an ambiguous situation by embedding a new key simultaneously.

Acknowledgement

This research is supported by the 2021 SATU Joint Research Grant under grant no. ST041-2021 from Universiti Malaya; and the GPUs used for this research was donated by the NVIDIA Corporation.

7 Broader Impact

Our proposed ownership protection framework aims to protect the IPR of RNN model. To compete with each other and gain business advantage, a large number of resources/budgets are continually being invested by giant and/or startup companies to develop new DNN models. Hence, we believe it is vital to protect these inventions from being abused, stolen or plagiarized. We believe that nobody with genuine intention will be harmed by this work. In the worst case scenario where if our proposed work fails to protect the RNN model; it just reflects the current status of RNN model as from our understanding, there is yet initiative of the IPR protection for RNN. In short, the ownership verification for RNNs will bring benefits to society by providing technical solutions to reduce plagiarism in deep learning and thus, lessen wasteful lawsuits and secure business advantages in an open market.

References

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX*, pages 1615–1631, Baltimore, MD.

Franziska Boenisch. 2020. A survey on model watermarking neural networks. *arXiv preprint arXiv:2009.12153*.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ales Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA.

Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *ICMR*, pages 105–113.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, Doha, Qatar.

Lixin Fan, Kam Woh Ng, Chee Seng Chan, and Qiang Yang. 2022. DeepIPR: Deep neural network ownership verification with passports. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6122–6139.

Felix Gers, Nicol Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, 3:115–143.

Jia Guo and Miodrag Potkonjak. 2018. Watermarking deep neural networks for embedded systems. In *ICCAD*, pages 1–8, New York, NY, USA.

Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. 2022. Protecting intellectual property of language generation apis with lexical watermark. In *AAAI*, pages 10758–10766.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Zhang Jie, Chen Dongdong, Liao Jing, Zhang Weiming, Hua Gang, and Yu Nenghai. 2020. Passport-aware normalization for deep model protection. In *NeurIPS*, page 22619–22628.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Benjamin Krause, Iain Murray, Steve Renals, and LU Liang. 2017. Multiplicative lstm for sequence modelling. In *ICLR*, pages 2872–2880.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Erwan Le Merrer, Patrick Perez, and Gilles Trédan. 2020. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING*, page 1–7, USA.

Jian Han Lim, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. 2022. Protect, show, attend and tell: Empowering image captioning models with ownership protection. *Pattern Recognition*, 122:108285.

Market Research Future. 2022. Machine learning as a service market estimated to cross usd 302.66 billion at a cagr of 36.2% by 2030. *GlobeNewswire*.

Ding Sheng Ong, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. 2021. Protecting intellectual property of generative adversarial networks from ambiguity attacks. In *CVPR*, pages 3630–3639.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *ACL*, page 311–318, USA.

Yuhui Quan, Huan Teng, Yixin Chen, and Hui Ji. 2021. Watermarking deep neural networks in image processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):1852–1865.

Pulkit Rathi, Saumya Bhadauria, and Sugandha Rathi. 2022. Watermarking of deep recurrent neural network using adversarial examples to protect intellectual property. *Applied Artificial Intelligence*, 36(1):2008613.

John A Rice. 2006. *Mathematical statistics and data analysis*. Cengage Learning.

Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2018. Deepsigns: A generic watermarking framework for ip protection of deep learning models. *arXiv preprint arXiv:1804.00750*.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301, Berlin, Germany.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 269–277.

Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting intellectual property of deep neural networks with watermarking. In *ASIACCS*, page 159–172, New York, NY, USA.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *COLING*, pages 3485–3495.

A Appendix

A.1 Hyperparameters

Table 6 summarizes all the hyperparmeters used in the experiments.

Table 6: Hyperparameters used in each tasks.

Hyperparameter	TREC-6	WMT14 EN-FR
Vocabulary size	-	15000
Max sentence length	30	15 (EN) / 20 (FR)
RNN hidden units	300	1000
Embedding dimension	300	300
Batch size	10	256
Bidirectional	Yes	No
Optimizer	Adam(Kingma and Ba, 2014)	Adam

A.2 Qualitative Results

Table 3 and 8 show examples of incorrect predictions when a counterfeit key is embedded into the recurrent neural network (RNN) model during inference phase. For classification tasks (i.e. TREC-6 (Li and Roth, 2002)), Table 8 shows that when a counterfeit key is used, the RNN model gets confused between similar classes, i.e. DESC and ENTY for TREC-6. Meanwhile, for machine translation task (i.e. WMT14 EN-FR (Bojar et al., 2014)), Table 3 demonstrates the translation results when a genuine key is used against a counterfeit key. It is observed that when a counterfeit key is used, the RNN model can still somehow translate accurately at the beginning of the sentence, but the translation quality quickly deteriorates toward the end of the sentence. This is in line with our idea and design of Gatekeeper where the information (hidden state) passed between timesteps would be disrupted with a counterfeit key and the output of RNN would deviate further from the ground truth the longer the timesteps are.

A.3 Methods to generate key

Three types of methods to generate key have been investigated in our work:

- *random patterns*, elements of key are randomly generated from a uniform distribution between [-1, 1]. For natural language processing (NLP) task, a sequence of random word embedding can be used.
- *fixed key*, one key is created from the input domain and fed through the trained RNN model with the same architecture to collect its corresponding features at each layer. The corresponding features are used in the Gatekeeper. For NLP task, a sentence from the input language domain is used as key.
- *batch keys*, a batch of K keys similar to above are fed through the trained RNN model with the same architecture. Each K features is used in the Gatekeeper, and their mean value is used to generate the final Gatekeeper activation.

In the *batch keys* method, the number of possible key combination is $(K \times l)^V$ where K is the number of keys used, l is the length/time step of key and V is the vocabulary size. This make it impossible for an attacker to correctly guess or brute force the key. Since batch keys provides the strongest protection (with the highest possible key combination), we adopt this key generation method for all the experiments reported in this paper.

Table 7: Example of hidden state output h_0^k and their respective sign (+/-) from LSTM$_{kt}$ when we embed digital signature S={*private signature goes here*}

Hidden state h_0^k	Sign (+/-)	ASCII code	Character
-0.1939	-1	112	p
0.1820	1		
0.2064	1		
0.1648	1		
-0.1795	-1		
-0.1670	-1		
-0.1778	-1		
-0.1711	-1		
-0.2059	-1	114	r
0.1685	1		
0.1767	1		
0.1876	1		
-0.1996	-1		
-0.1997	-1		
0.1882	1		
-0.1655	-1		
-0.1657	-1	105	i
0.1838	1		
0.2144	1		
-0.1840	-1		
0.1652	1		
-0.1818	-1		
-0.2118	-1		
0.1673	1		
-0.2330	-1	118	v
0.1882	1		
0.1740	1		
0.1909	1		
-0.1963	-1		
0.1868	1		
0.1882	1		
-0.1951	-1		

A.4 Gatekeeper Sign as Digital Signature

Sign (+/-) of the first hidden state of key h_0^k can be used to encode a digital signature S such as ASCII code (8 bits as one ASCII character). Note that the maximum capacity of an embedded digital signature depends on the number of hidden units in the protected RNN layer. For instance, in this paper, the model Seq2Seq$_{kt}$ has Gated Recurrent Unit (GRU) layer with 1000 units, so the maximum signature capacity that can be embedded is 1000 bits or 125 ASCII characters. For ownership verification, the embedded digital signature S can be revealed by decoding the learned sign of h_0^k. Table 7 shows the embedded digital signature and their respective sign, every 8 bits is decoded into a ASCII character.

B Cross Domain Application

In addition to NLP domain, we also applied our proposed frameworks on image domain, specifically in the task of sequential image classification. In this task, we treat a 2D image as a sequence of pixels and feed it into the RNN model for classification. This is particularly useful in cases where one cannot obtain the whole image in a single time frame. SeqMNIST (Le et al., 2015) is a variant of MNIST where sequence of image pixels that represent handwritten digit images is classified into 10 digit classes. For trigger sets in image domain, we follow the work by Adi et al. (2018) where we select ran-

Table 8: Qualitative results on TREC-6. The best-performed model that has both white-box and black-box protections is selected to demonstrate the classification results with genuine and counterfeit keys.

Input	Ground Truth	Prediction with genuine key	Prediction with counterfeit key
What is Mardi Gras ?	DESC	DESC	ENTY
What date did Neil Armstrong land on the moon ?	NUM	NUM	DESC
What is New York 's state bird ?	ENTY	ENTY	DESC
How far away is the moon ?	NUM	NUM	LOC
What strait separates North America from Asia ?	LOC	LOC	ENTY

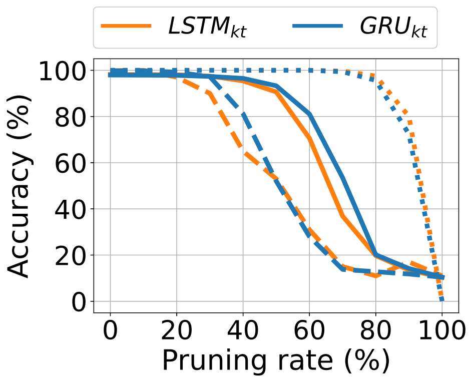


Figure 7: Classification accuracy on test set (solid line) and trigger set (dashed line), and digital signature accuracy (dotted line) against different pruning rates for SeqMNIST. Best viewed in colour.

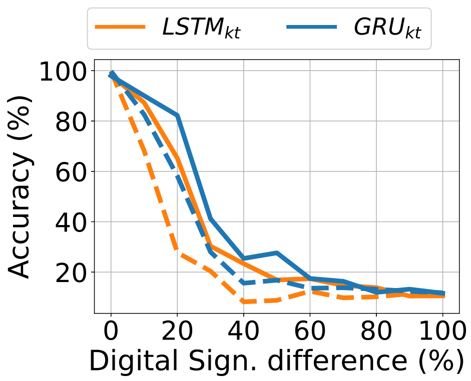


Figure 8: Classification accuracy on test set (solid line) and trigger set (dashed line) for SeqMNIST when different percentage (%) of the digital signature S is being modified/compromised. Best viewed in colour.

dom images from training dataset and shuffle their labels. We chose Le et al. (2015) as the baseline model and followed the hyperparameters defined in the work which are 100 hidden units in RNN, 128 batch size and Adam (Kingma and Ba, 2014) optimizer with default settings.

B.1 Quantitative and Qualitative Results

Quantitatively, we achieve similar results as the application in NLP domain. As seen in Table 5, the protected models have similar classification accuracy as the baseline model demonstrating that embedding keys and trigger set doesn't hurt the model learning capacity. Also, we can notice that when a counterfeit key is presented to the protected models, the classification accuracy dropped by 75-80%.

Furthermore, we also investigate the qualitative results in sequential image classification task. In Table 10, when a counterfeit key is used, the RNN model gets confused between similar classes, i.e. 5 and 6 for SeqMNIST.

B.2 Robustness against Removal Attacks

Pruning: We follow the same model pruning strategy in our main paper. Figure 7 shows that for image classification models, even when 40% of the model parameters are pruned, trigger set accuracy still maintains about 70-80% accuracy, accuracy on test set drops slightly while digital signature accuracy still maintained near to 100% accuracy. This proves that model pruning will hurt the model performance before the embedded watermarks can be removed and therefore our proposed work is robust against it.

Fine-tuning: Same as the main paper, the host model is initialized using trained weights with embedded watermarks, then it is fine-tuned without the presence of the key, trigger set and regularization terms. As seen in Table 9, digital signature accuracy remains consistently at 100 even after the model is fine-tuned. When the original genuine key is presented to the fine-tuned model, we are able to obtain comparable accuracy to the stolen model.

Overwriting: We also simulate an overwriting scenario where the attacker has knowledge of how the model is protected and attempts to embed a new key, $\overline{k}$ into the trained model using the same proposed method. In Table 9, we can observe that digital signature accuracy remains at 100% consistently after the protected model is overwritten with the new key. When inferencing using the original genuine key, the performance only dropped slightly. Empirically, this confirms that the embedded key and signature cannot be removed by overwriting it with new keys.

B.3 Resilience against ambiguity attacks

In the previous section, we simulate a scenario where the key embedding method and digital signature are completely exposed, and an attacker can introduce an ambiguous situation by embedding a new key simultaneously. However, we show that the digital signature cannot be changed easily. As shown

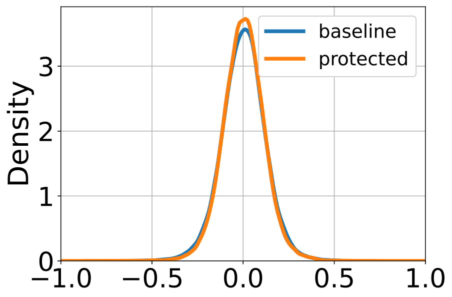


Figure 9: Comparison of weight distribution between original and protected model on SeqMNIST. Best viewed in colour.

Table 9: Robustness of protected RNN model trained on SeqMNIST (in bold) against removal attacks (i.e. fine-tuning and overwriting). All metrics reported herein are the performance with genuine key where acc. = accuracy.

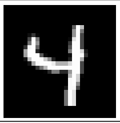
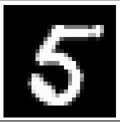
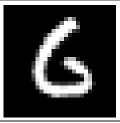
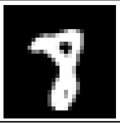
	Acc.	T acc.	Sign acc.
LSTM$_{kt}$	**98.18**	**99.8**	**100**
Fine-tuning	98.28	99.6	100
Overwriting	97.52	52	100
GRU$_{kt}$	**97.95**	**99.8**	**100**
Fine-tuning	98.09	99.4	100
Overwriting	97.53	78	100

in Figure 8, the model's performance decreases significantly when 40% of the original signs are modified. In sequential image classification task on SeqMNIST, the model's accuracy dropped from 98.18 → 23.37 (for the test set in LSTM$_{kt}$), which is merely better than a random guessing model. We can conclude that the signs enforced in this way are persistent against ambiguity attacks and illegal parties will not be able to employ new digital signatures without hurting the protected model's performance.

B.4 Secrecy

In digital watermarking for DNN, one of the design goals is secrecy to prevent unauthorized parties from detecting it. In other words, this means that the protected model's weights should not change when compared to a baseline (unprotected) model. Figure 9 shows the weight distribution of the protected models and baseline model, the weight distribution of the protected RNN layers is identical to the baseline RNN layers.

Table 10: Qualitative results on SeqMNIST. The best-performed model that has both white-box and black-box protections is selected to demonstrate the classification results with genuine and counterfeit keys.

Input	Ground Truth	Prediction with genuine key	Prediction with counterfeit key
	2	2	7
	4	4	7
	5	5	6
	6	6	0
	8	8	0