Improvement of Min-Entropy Evaluation Based on Pruning and Quantized Deep Neural Network

Haohao Li, Jianguo Zhang, Zhihu Li, Juan Liu, and Yuncai Wang

***Abstract*— In the field of information security, the unpredictability of random numbers plays determinant role according to the security of cryptographic systems. However, limited by the capability of pattern recognition and data mining, statistical-based methods for random number security assessment can only detect whether there are obvious statistical flaws in random sequences. In recent years, some machine learning-based techniques such as deep neural networks and prediction-based methods applied to random number security have exhibited superior performance. Concurrently, the proposed deep learning models bring out issues of large number of parameters, high storage space occupation and complex computation. In this paper, for the challenge of random number security analysis: building high-performance predictive models, we propose an effective analysis method based on pruning and quantized deep neural network. Firstly, we train a temporal pattern attention-based long short-term memory (TPA-LSTM) model with complex structure and good prediction performance. Secondly, through pruning and quantization operations, the complexity and storage space occupation of the TPA-LSTM model were reduced. Finally, we retrain the network to find the best model and evaluate the effectiveness of this method using various simulated data sets with known min-entropy values. By comparing with related work, the TPA-LSTM model provides more accurate estimates: the relative error is less than 0.43%. In addition, the model weight parameters are reduced by more than 98% and quantized to 2 bits (compression over 175x) without accuracy loss.**

***Index Terms*— Min-entropy, TPA-LSTM, random number, pruning, quantization.**

Manuscript received 27 July 2022; revised 3 January 2023 and 22 January 2023; accepted 23 January 2023. Date of publication 30 January 2023; date of current version 6 February 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1803500, in part by the National Natural Science Foundation of China under Grant 62045009 and Grant 61731014, in part by the Natural Science Foundation of Shanxi Province under Grant 202103021224038 and Grant 20210302123185, and in part by the International Cooperation of Key Research and Development Program of Shanxi Province under Grant 201903D421012. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Stefano Tomasin. *(Corresponding author: Jianguo Zhang.)*

Haohao Li and Jianguo Zhang are with the Key Laboratory of Advanced Transducers and Intelligent Control System, Ministry of Education of China, College of Physics and Optoelectronics, Taiyuan University of Technology, Taiyuan 030024, China (e-mail: lihaohao1115@link.tyut.edu.cn; zhangjianguo@tyut.edu.cn).

Zhihu Li is with the China Electric Power Research Institute, Beijing 100192, China (e-mail: 13381009490@189.cn).

Juan Liu is with Nations Technology Company Ltd., Shenzhen 518057, China (e-mail: liu.juan@nationstech.com).

Yuncai Wang is with the Advanced Institute of Photonic Technology, Guangdong University of Technology, Guangzhou 510006, China (e-mail: wangyc@gdut.edu.cn).

Digital Object Identifier 10.1109/TIFS.2023.3240859

I. Introduction

RANDOM numbers are widely used in cryptography, authentication, spread spectrum communication and other information security fields, which are the security guarantee of encryption systems [1], [2], [3]. Random numbers are generated by random number generators (RNGs), and secure random numbers need to satisfy true randomness and be theoretically unpredictable. However, environmental noise and non-ideal characteristics in physical devices may compromise the security of RNGs [4]. When the output of RNGs does not provide sufficient unpredictability, it is susceptible to malicious attacks therefore leads to cryptosystem corruption [5], [6], [7]. To provide security assurance, RNGs are extensively tested during the design and verification process. The importance of random number security analysis cannot be overstated.

Currently, many research institutions or individuals have provided amounts of methods for testing and evaluating RNGs to guide designers, users, and evaluators in analyzing the security of RNGs. These methods can be roughly divided into two categories: white-box testing and black-box testing. The white-box testing approach requires understanding the internal structure and generation principles of RNGs. A mathematical model is established by appropriate assumptions [8], [9] so that the entropy of the output sequence can be calculated theoretically. This method is usually used to prove the theoretical security of hardware RNGs. While most RNGs still cannot be well modeled by stochastic models due to their complexity and diversity [10], [11], [12]. Relatively, the black box test method only relies on input samples, so it has good versatility for different RNGs.

Black-box testing methods can be mainly classified into statistic-based methods and prediction-based methods. The statistic-based methods evaluate the randomness of random numbers according to standard test suites, such as NIST SP 800-22 [13], AIS 31 [14], Diehard [15], and TestU01 [16]. These standard test suites detect whether statistical properties of random sequences, such as uniformity or independence, are significantly defective. However, limited by the ability of pattern recognition and data mining, they show deficiencies in random number security assessment. For example, some complex pseudo-random sequences can successfully pass the statistical test criteria due to their good statistical properties [17]. Therefore, it is difficult to meet the security assessment requirements for random numbers only by relying on statistical test criteria.

Prediction-based methods using machine learning techniques have been increasingly applied to random number security analysis with excellent performance [18], [19], [20]. In 2015, Kelsey et al. [21] developed a set of tools for estimating entropy, called predictors, which provide better performance than statistical-based entropy estimation methods. The predictors refer to machine learning algorithms that attempt to predict each sample in a sequence and update its internal state based on all observed samples. In 2018, Yang et al. [22] used fully-connected neural networks (FNN) and recurrent neural networks (RNN) as predictors to evaluate the min-entropy of entropy sources. For data with long-term dependence or large sample space, the neural network-based estimator can outperform the predictor provided by Kelsey et al. [21]. In 2019, Truong et al. [23] used the recurrent convolutional neural network (RCNN) to study the effect of deterministic classical noise on the security of quantum random number generator (QRNG) and applied it to uniformly distributed random numbers from QRNG and a congruential RNG. The experimental results showed the robustness of the neural network approach and its potential for quality benchmarking of RNG devices. In 2020, Li et al. [24] analyzed the security of white chaos-based RNG by introducing a deep learning model with a temporal pattern attention mechanism. The results of evaluating the random number of congruential RNGs showed that the deep learning model outperformed other existing models. However, deep learning models concurrently bring out issues of large number of parameters, high storage space occupation, and complex computations. Building high-performance predictive models becomes a major challenge for random number security analysis.

In this paper, we review existing methods for random number security analysis and also reconsider prediction-based evaluation methods, where the advantages of evaluation methods based on machine learning techniques (e.g., deep neural networks) can be evidence. For the challenge of random number security analysis: building high-performance predictive models, we propose an effective analysis method based on pruning and quantized deep neural network. We suggest a three-stage random number evaluation scheme that uses the min-entropy to represent the unpredictability of random numbers. Extensive experiments show that our proposed method is effective and can provide more accurate estimation results compared to other relevant methods.

The major contributions of this paper can be summarized as follows.

- We suggest a three-stage random number evaluation scheme. According to this scheme, we provide a temporal pattern attention-based long short-term memory (TPA-LSTM) prediction model to analyze the security of random numbers and use the min-entropy value to quantify the unpredictability of random numbers.
- Experiments using various simulated data sets with known min-entropy values were performed where, the results showed the advantages of our proposed model over the existing evaluation models. In addition, we further validated the effectiveness of pruning and quantized deep learning models in the field of random number security analysis.
- For the model quantization problem, we validated three existing clustering quantization strategies and the new K-means++ quantization strategy, which compresses the model faster without losing the accuracy of the model.

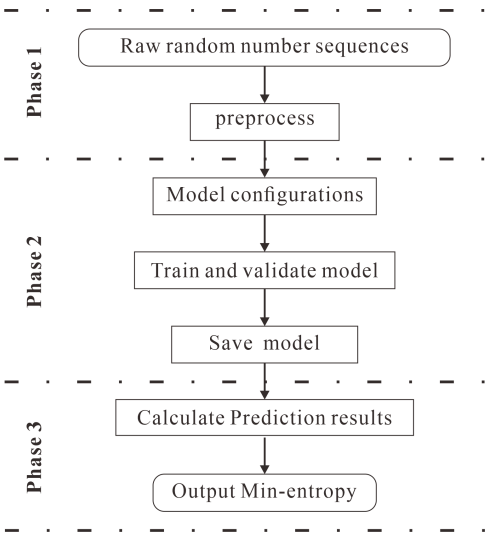


Fig. 1. The general workflow of random number evaluation scheme.

The remainder of this work is organized as follows. In Section II, we present the definition of min-entropy and our proposed scheme for min-entropy evaluation based on the TPA-LSTM prediction model. In Section III, we introduce the experimental setup and the method of neural network compression, including pruning and quantization. Section IV presents the evaluation results of the TPA-LSTM model and looks at how the model performs when pruned and quantized. Finally, Section V of the paper concludes the full paper and proposes future work.

II. Experiment Scheme

In this section, the general workflow of the evaluation scheme for random number security is illustrated in Fig. 1, which consists of three phases: data preprocessing, model training, and min-entropy evaluation. In the data preprocessing, the random number sequences collected are labeled after slicing into different proportions. In the model training, we provide and describe the TPA-LSTM model and the operations to train and validate the model. In min-entropy evaluation, we test the untrained random sequences with the saved model and calculate the min-entropy based on the prediction results.

A. Data Processing

In order to properly train a model that can successfully predict sequential data, the data needs to be pre-processed. As presented in Fig. 2, the data samples are split into different ratios in order to train and test the model. For all samples, the ratio of training data and test data is set to 4:1, in which 20% of

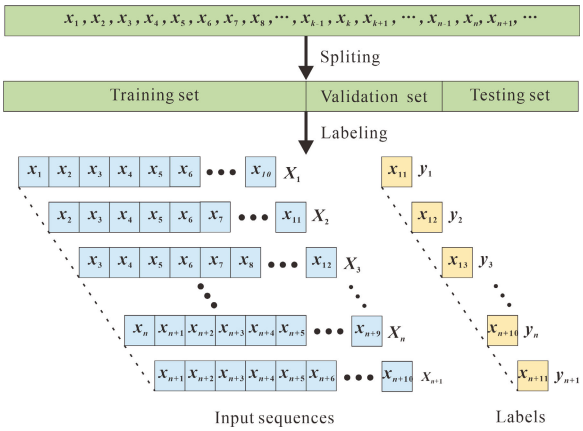


Fig. 2. Data processing.

the training data is used as the verification set. Afterwards, all the samples are labeled. 10 consecutive samples in the dataset are used as inputs and the corresponding latter samples are used as labels. Then, the data are sequentially shifted backward by 1 sample, and finally all input sequences and labels are obtained.

B. Model Training

Random numbers are a typical time series, and thus are best suited for solving such problems with recurrent neural networks (RNNs) [25]. Long Short-Term Memory Network (LSTM) [26], as a variant of RNNs, can improve the gradient disappearance or gradient explosion problem that occurs in RNN training to some extent. In addition, the attention mechanism has made progress in the fields of natural language processing and speech recognition. Compared with typical attention mechanisms, the introduction of temporal pattern attention [27] can better learn hidden associations in complex time-series data. Inspired by [24], we build the TPA-LSTM model as our prediction model.

The structure of the TPA-LSTM model used in this paper is shown in Fig. 3. First, the preprocessed 10 neighboring 8-bit integers are encoded into one-hot vectors, and each element of the integer index is transformed into a one-bit validly encoded binary vector. The encoded one-hot vectors are sequentially fed into the LSTM layer. The LSTM layer with 256 units learns the dependency and patterns in the sequence and outputs the hidden states corresponding to each time step in the input sequence. Subsequently, all the hidden states output by the LSTM layer are divided into hidden states $H = (h_1, h_2, \ldots, h_{t-1})$ and current state h_t. Convolution operations are performed on the row vectors of hidden states H using the Conv1D layer to improve the model's ability to learn temporal features. The Conv1D layer has 256 filters, each of length 10. The Conv1D layer with Relu activation function outputs H^C. For the h_t to be predicted, it interacts with each row of the H^C matrix and is normalized with a sigmoid function to produce an attention weight a_i:

$$a_i = sigmoid\left(\left(H_i^C\right)^T W_c h_t\right) \tag{1}$$

which represents the intensity of each row of the H^C matrix on the h_t to be predicted, i.e., the intensity of the influence of each time series on the h_t. The attention weights are used to do a weighted summation for each row of H^C to obtain the context vector v_t:

$$v_t = \sum_{i=1}^{m} a_i H_i^C \tag{2}$$

which represents the combined effect of all rows on h_t, i.e., the temporal aspect. The effects of all-time series on h_t are added to obtain the final TPA mechanism output h'_t:

$$h'_t = W_h h_t + W_v v_t \tag{3}$$

Finally, the results are optimized by a fully connected layer combined with a softmax classifier to obtain the final prediction results. There are 256 units in the model for each fully connected layer.

C. Min-Entropy Calculation

Researchers quantify the unpredictability through the concept of entropy, such as Shannon entropy, Rényi entropy, and min-entropy. Min-entropy proposed in NIST SP 800-90B [28] (90B for short) is the most relevant metric for cryptographic security and corresponds to the difficulty of guessing or predicting the most likely output of the entropy source. For an independent discrete random variable X taken from set $A = (x_1, x_2, \ldots, x_k)$ with probability $\Pr(X = x_i) = p_i, i = 1, 2, \ldots, k$, 90B draft [29] in 2012 gave the definition of min-entropy:

$$H_{\min} = \min_{1 \le i \le k} \left(-\log_2 p_i\right) = -\log_2 \max_{1 \le i \le k} \left(p_i\right) \tag{4}$$

In this paper, we calculate the min-entropy based on the prediction results and compared it with several estimators in final version of 90B officially published in 2018 [28]. These estimators include the Multi Most Common in Window (MultiMCW) predictor, Lag predictor, Multiple Markov Model with Counting (MultiMMC) predictor, and LZ78Y predictor. A brief introduction of the 4 predictors is as follows.

1) **MultiMCW predictor**: This predictor contains 4 sub-predictors with different window sizes to guess the future output data of the sequence based on the most common values in the window. The window sizes of the 4 sub-predictors are 63, 255, 1023, 4095.
2) **Lag predictor**: This predictor predicts the future output of the sequence using the lagged prediction method, which also contains several sub-predictors. It is intended to detect correlation and periodicity.
3) **MultiMMC predictor**: This predictor is a Markov predictor with counting, which also contains several sub-predictors. Each sub-predictor records the frequency at which a value in the sequence is transferred to another value and predicts the subsequent output, based on the highest frequency.
4) **LZ78Y predictor**: This predictor is based on the LZ78 encoding, which does not contain sub-predictors but uses a dictionary approach to record historical sequences

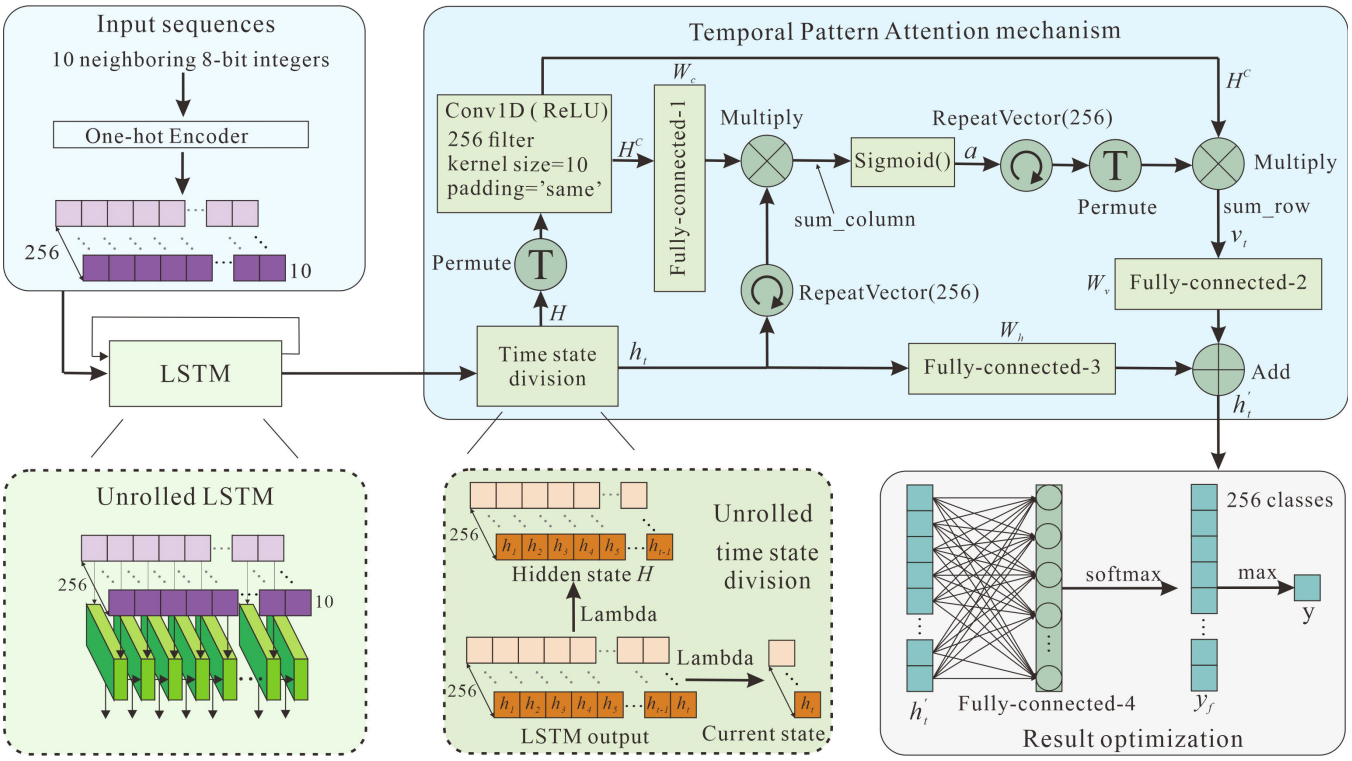


Fig. 3. Schematic diagram of TPA-LSTM predictive model.

and make predictions about the future. Each time the sequence is processed, every substring in the most recent sequences updates the dictionary or is added to the dictionary.

We use the trained model to test the untrained test set data and calculate the min-entropy based on the prediction results. Assuming that the sample size of the test set is N, the number of correct predictions is N_T, and the number of consecutive correct predictions is r, we can thus calculate the global prediction performance P_{global} and the local prediction performance P_{local}. P_r is the prediction accuracy, which is calculated as follows:

$$P_r = \frac{N_T}{N} \times 100\% \tag{5}$$

The confidence upper bound of P_r at a confidence level of 99% is taken as the global prediction probability P_{global} and the calculation formula is shown below:

$$P_{global} = \begin{cases} 1 - 0.01^{\frac{1}{N}}, if \ \ P_r = 0, \\ \min\left(1, P_r + 2.576\sqrt{\frac{P_r(1-P_r)}{N-1}}\right) \end{cases} \tag{6}$$

The local prediction probability P_{local} is calculated iteratively by (7) and (8):

$$0.99 = \frac{1 - P_{local} x_{10}}{(r+1-r x_{10})\, q} \times \frac{1}{(x_{10})^{N+1}} \tag{7}$$

$$x_j = 1 + (1 - P_{local})\, P_{local} x_{j-1},\ j = 1, 2, \ldots, 10 \tag{8}$$

where $x_0 = 1$. Ultimately, the min-entropy can be calculated from these two metrics by definition as shown in the following equation:

$$Min - entropy = -\log_2\left(\max\left(P_{global}, P_{local}\right)\right) \tag{9}$$

III. Compression Methods

In this section, we introduce pruning and cluster quantization methods, respectively. We first provide the principles of pruning and cluster quantization, and then describe the overview and details of the methods used. Pruning and cluster quantization methods can be performed separately. They can also be performed together to achieve higher compression rates.

A. Pruning

Deep learning models are often considered to be over-parameterized and have a lot of redundancy [30]. The main idea of network pruning is to remove a certain percentage of relatively unimportant parameters without affecting the network performance [31]. The importance of the parameters can be assessed by the second-order derivatives of the target loss function with respect to the weights, the absolute values of the parameters, and other metrics [32], [33], [34], [35]. Depending on the object of pruning, it can be divided into weight pruning, channel pruning, kernel pruning and neuron pruning.

Since the TPA-LSTM model has less parameters in the hidden layer, most of the parameters are located in the LSTM input layer. Therefore, we prefer to choose to compress the model using a weight pruning scheme rather than other schemes, which may lead to the loss of model accuracy. The most popular magnitude-based method [36] is used to evaluate the importance of each parameter. The pseudo-code of the pruning algorithm we used is shown in algorithm 1. The details of the algorithm are given later.

Algorithm 1 Weight Pruning Algorithm

Input: The model M with L layers;
The weight matrix to be pruned, W_i;
The expected sparsity, p;
Output: The pruned weight matrix, W_{ip}
Calculate pruning steps k;
Generate initial mask;
for i = 1 to k **do**
  for each layer i = 1 to L **do**
    Calculate threshold with sparsity p_i for W_i
    Pruning and Updata mask;
  end for
  Normal training;
end for
Return the pruned weight matrix, W_{ip}

The final sparse model is obtained by iteratively pruning the network. A portion of the unimportant weights are pruned layer by layer each time. The network is repaired once after each pruning in order to ensure the network accuracy during the pruning process.

Because the existing framework has limited support for sparse operation, we use binary mask to implement sparse structure. The network is initialized by:

$$\theta_s = \theta * M \tag{10}$$

where θ is the density parameter of the network and M is the binary mask. Using the gradual pruning technique proposed in [37], the sparsity is increased from an initial sparsity 0 to a final sparsity p. The mask will be updated every $\Delta t = 100$ steps to recover the loss of network accuracy:

$$k = \frac{sample\ size}{100 \cdot n \cdot (batch\ size)} \tag{11}$$

$$p_t = p + (0 - p)\left(1 - \frac{i\Delta t}{k\Delta t}\right)^3, i \in \{0, 1, \ldots, k\} \tag{12}$$

where n is pruning epochs. For each individual weight θ_i^s in every layer, its importance is defined as its absolute values:

$$S\left(\theta_s^i\right) = |\theta_s^i| \tag{13}$$

Given a certain sparsity p_t, the new mask is given by:

$$M = S\left(\theta_s\right) > d \tag{14}$$

where d is the p_t-th percentile of $S\left(\theta_s\right)$ with an ascending order. We use the compressed sparse row (CSC) format to store the pruned network. The number of parameters required for storage after pruning is $2a + z + 1$, where a is the number of non-zero elements and z is the number of rows.

B. Quantization

The main purpose of quantization is to reduce the storage capacity of the model and consists of two main approaches: weight sharing and using lower bits to represent the model parameters. In this paper we currently focus on the weight-sharing approach. Weight sharing allows multiple parameters of the network to share a weight size and reduces the parameter storage size. Chen et al. [38] determine the weight sharing by a hash function. Han et al. [36] use k-means clustering to identify shared weights for each layer of a trained network and examine three different initialization methods: random, density-based, and linear initialization.

The specific steps of clustering quantization are: (1) Load the pre-trained network model; (2) Use the k-means method to cluster the trained parameter matrix; (3) Retrain the network; After clustering, each item of the parameter matrix only needs to be represented by the corresponding cluster center number. The weights $W_i = \left\{\theta_1^i, \theta_2^i, \ldots \theta_q^i\right\}$ of each layer are partitioned into m classes $C = \{c_1, c_2, \ldots c_m\}, q \gg m$, so as to minimize the within-cluster sum of squares.

$$arg\min_{C} \sum_{i=1}^{m} \sum_{\theta^i \in c_i} \left\|\theta^i - c_i\right\|^2 \tag{15}$$

A brief description of the three methods is as follows.

1) **Random initialization**: Randomly choose m observations from the original weights as the initial centroids.
2) **Density-based initialization**: Linearly choose m cumulative distribution function (CDF) observations in the y-axis, then find the vertical intersection with the CDF observations on the x-axis as the initial centroids.
3) **Linear initialization**: Linearly choose m observations between the [min, max] of the original weights as the initial centroids.

Because centroid initialization impacts the quality of clustering and thus affects the prediction accuracy of the network, we also examine another improved initialization method, k-means++ [39]. The k-means++ initialization methods are introduced as follows:

Step 1: Randomly choose 1 observation from the original weights $W_i = \left\{\theta_1^i, \theta_2^i, \ldots \theta_q^i\right\}$ as the first initial centroid c_1.

Step 2: Calculate the shortest distance between each sample and all existing centroids. Then calculate the probability of each sample being selected for the next centroid.

Step 3: Repeat Step2 until m centroids are selected.

C. Compression Rate

In addition to comparing the performance changes of the models, we use the model compression ratio to compare the size of the deep learning models before and after compression. All parameters involved in the computation in the model are first counted, and then converted to bytes based on their data format, using the bytes as the criteria for calculating the compression rate. The compression rate is calculated as follows:

$$Compression\ rate = \frac{o \times b}{o_c \times b_c} \tag{16}$$

where o denotes the number of all the parameters involved in the calculation, including the weight parameters and bias parameters. b denotes the number of bytes required to store a single parameter. We consider $b = 32/8$ for all bias parameters

TABLE I
KNOW DISTRIBUTION PARAMETERS AND THEORETICAL MIN-ENTROPY

Data sets	P_m	H_{min}
uniform	$1/2^8$	8
near-uniform	0.00550	7.50635
triangle	$1/2^7$	7
normal	0.01534	6.02654

and pruning model parameters and $b = \log_2^m /8$ for quantization model parameters. The subscript $c = 1, 2, 3$ denotes the pruning model, quantization model and pruning quantization model, respectively.

IV. EXPERIMENT RESULTS AND ANALYSIS

To evaluate the effectiveness of our proposed model as well as the pruning and quantization approach, we implement the model and test it on different types of simulated data with known min-entropy. We compare the baseline model with the 90B estimate. Furthermore, we investigate the impact of pruning and quantizing models on different datasets.

A. Data Set

We train the prediction model on a number of representative simulated data sources, in which the probability distributions of outputs are known. Simulation datasets are generated using the following distribution families adopted in [21] and [22], and a new discrete triangular distribution is added.

- *Discrete Uniform Distribution*: The samples which come from an independent and identically distributed source are equally-likely

- *Discrete Near-Uniform Distribution*: All samples which come from an independent and identically distributed source are equally-likely except one. It has a higher probability than the rest.

- *Normal Distribution Rounded to Integers*: All samples which come from an independent and identically distributed source are drawn from a normal distribution and rounded to integer values.

- *Triangular Distribution Rounded to Integers*: All samples which come from an independent and identically distributed source are drawn from a triangular distribution and rounded to integer values.

- *M-sequence*: The longest linear feedback shift register sequence is a typical pseudo-random sequence widely used.

Ten simulated sources are created in each of the classes listed above except the last one. A sequence of 1M samples is generated from each simulated source. For M-sequence, a sequence of 1M samples is generated using simulated sources of different stages. All simulated data set samples are 8-bit integer values.

For each source, the theoretical min-entropy is derived from the known probability distribution to verify the prediction model. Table I lists the simulated entropy source parameters of several known probability distributions and the corresponding theoretical min-entropy. The theoretical min-entropy of periodic sequence is 0.

B. Experiment Setup

The model is trained with 1280 batch size and an optimizer adam with a manually set 0.0005 learning rate is introduced. The cross-entropy is used as an objective function, which can calculate the bias between the labels and the predicted values. The maximum number of training epochs is set to 200. Training is stopped early once the validation accuracy stops declining after 5 consecutive epochs to avoid overfitting the model. We calculate the validation loss and accuracy and save the corresponding parameters every epoch. The parameters with the highest validation accuracy are used as the final parameters of the trained model. Our experiments are implemented based on Keras and the backend of TensorFlow with python language and run them on a Windows 10 system configured with an Intel i9 10900x CPU and two NVIDIA RTX 2080ti GPUs.

C. Experiment Results and Analyses

We perform entropy estimation using our TPA-LSTM prediction model and 90B predictors. The evaluation results for different distribution families are shown in Fig. 4. We calculate the error between the min-entropy estimation results given by different predictors and the theoretical min-entropy values. For the uniform distribution data set, which is considered as full-entropy data, the 90B predictors give severely underestimated results and the results have large deviations. The deviations mainly come from the MultiMCW and Lag predictors. For the near-uniform distribution data set, all 90B predictors provide significantly overestimated results. It seems that the 90B predictors fail to predict the near-uniformly distributed data. For the triangular distribution data set, the estimates of the Lag predictor in 90B tend to slightly overestimate, while the MultiMCW predictor causes a serious underestimate in some experiments. For the normally distributed data set, the 90B predictors yield a substantial underestimate. Except Lag predictor, all the 90B predictors display heavy underestimation with big bias in some experiments. In contrast, the estimation results of TPA-LSTM prediction model for different distribution families are closer to the theoretical min-entropy values. The estimation results of TPA-LSTM prediction model have little error and bias, demonstrating the accuracy and robustness of TPA-LSTM prediction model. Due to the powerful learning ability of the TPA-LSTM model for data dependence, it can predict time series more accurately by approximating the probability distribution function.

Further, we analyze the reason why the 90B predictors show a significant underestimation of the min-entropy. We consider that the reason lies in the sensitivity of the local predictability probability P_{local} to the parameter r. The SP 800-90B suite contains multiple predictors and chooses the minimum estimate as the final result. As long as one of the predictors provides a (nearly) correct estimate, the final result rarely appears to be overestimated. However, if one of the predictors provides a significant underestimation, then the final result will take this low estimate regardless of whether the other predictors are correct. Thus, when a predictor computes a slightly larger value r, its corresponding probability P_{local} will

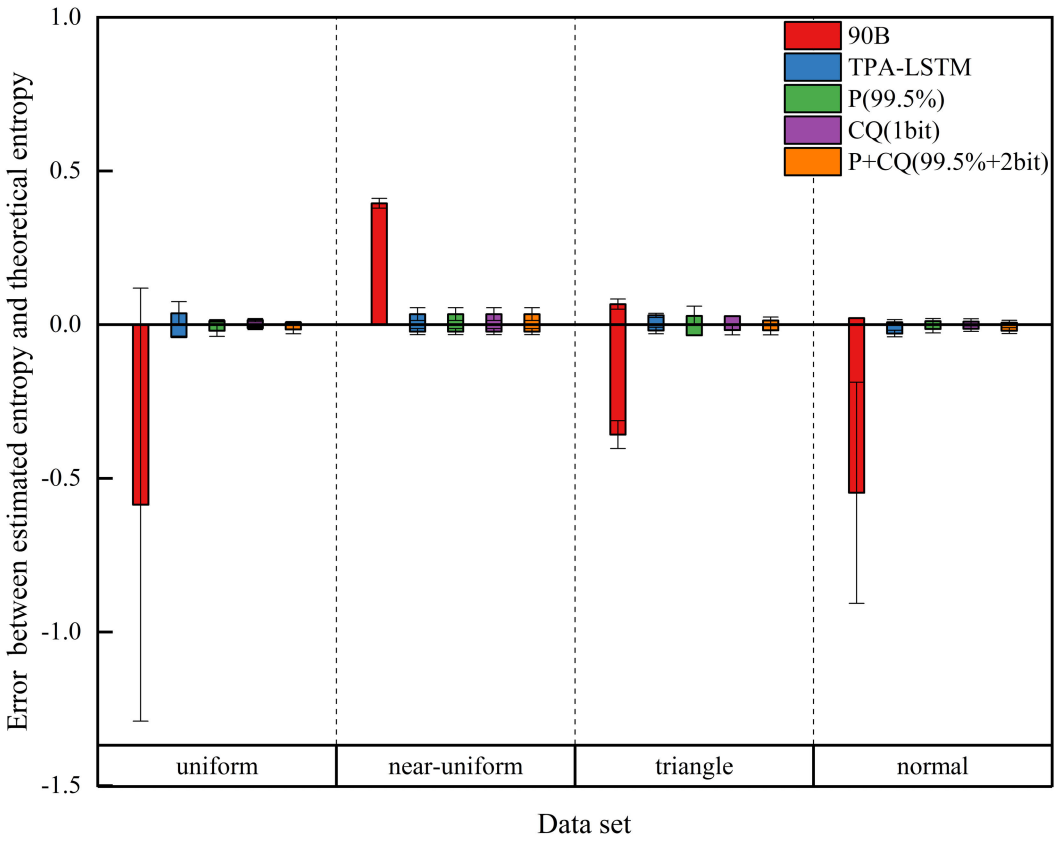


Fig. 4. Error comparison between the estimated min-entropy with different predictors and the theoretical min-entropy on different families of distributions.

TABLE II
RELATIVE ERROR OF DIFFERENT PREDICTORS ESTIMATIONS RESULTS

Data sets	90B	FNN	RNN	TPA-LSTM	P(99.5%)	CQ(1bit)	P+CQ(99.5%+2bit)
uniform	7.32%	1.37%	1.23%	0.48%	0.21%	**0.10**%	0.13%
near-uniform	5.26%	1.44%	1.41%	0.43%	0.43%	0.43%	**0.43**%
triangle	1.79%	0.86%	0.72%	0.37%	0.44%	0.31%	**0.23**%
normal	5.04%	1.12%	0.96%	0.84%	0.36%	0.30%	**0.28**%

far exceed P_{global}, resulting in a significant underestimation of the min-entropy derived from 90B.

After that, we use pruning and clustering quantization methods to reduce the complexity of the model and compress it. We investigate the effect of pruning and cluster quantization methods used separately and together. The sparsity p is set to 99.5% by sensitivity analysis. Using either the pruning method or the cluster quantization method alone, neither the pruned model nor the quantized model lost performance or even exceeded the performance of the original model. The TPA-LSTM model with pruned 99.5% parameters (compressed 83.2x) or quantized to 1 bit (compressed 29.4x) still accurately estimate the min-entropy for various distribution families. When the pruning and quantization methods are used together, Both the pruned and quantized models have no performance loss and even outperform the original model. The TPA-LSTM model with pruned 99.5% parameters and quantized to 2 bits (compressed 260.4x) has no performance loss and even outperforms the original model.

We compare the average relative errors of the estimation results of the FNN, RNN predictor [22], 90B's predictors, and our TPA-LSTM model. The lowest min-entropy value among the 90B's four predictors is used as 90B's estimation result. The average relative errors of the estimation results for various predictors are shown in Table II, where the best results are shown in bold. The experimental results show that the TPA-LSTM model outperforms the FNN, RNN, and 90B predictors in evaluating various distribution families with an average relative error as low as 0.37%. However, for the 90B's predictors, the average relative error is up to 7.32%. It further demonstrates the accuracy of TPA-LSTM. In addition, the average relative error of the pruned and quantized model is comparable to or even lower than that of the original model. It is indicated that the pruning and quantization methods can effectively reduce the complexity and storage of the model without any loss in model performance.

To further demonstrate the performance of the TPA-LSTM model and the effectiveness of the pruning and quantization methods, a more detailed study on the M-sequences data was carried out. We applied the FNN predictor, RNN predictor, 90B's predictors, and TPA-LSTM model to evaluate different stages of M-sequences whose theoretical min-entropy is

TABLE III
ESTIMATED RESULTS FOR M-SEQUENCES($H_{min} = 0.000$)

stage	8	10	12	14	16	18	20	22	24
MultiMCW	8.000	8.000	7.775	7.491	7.922	7.935	**7.967**	**7.995**	**7.916**
Lag	8.000	8.000	8.000	8.000	8.000	8.000	8.000	8.000	7.944
MultiMMC	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	8.000	8.000	7.998
LZ78Y	0.000	0.440	1.860	1.438	0.719	8.000	8.000	8.000	8.000
FNN	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.012	0.057
RNN	0.004	0.915	0.962	0.968	0.972	0.963	0.964	1.285	4.625
TPA-LSTM	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**

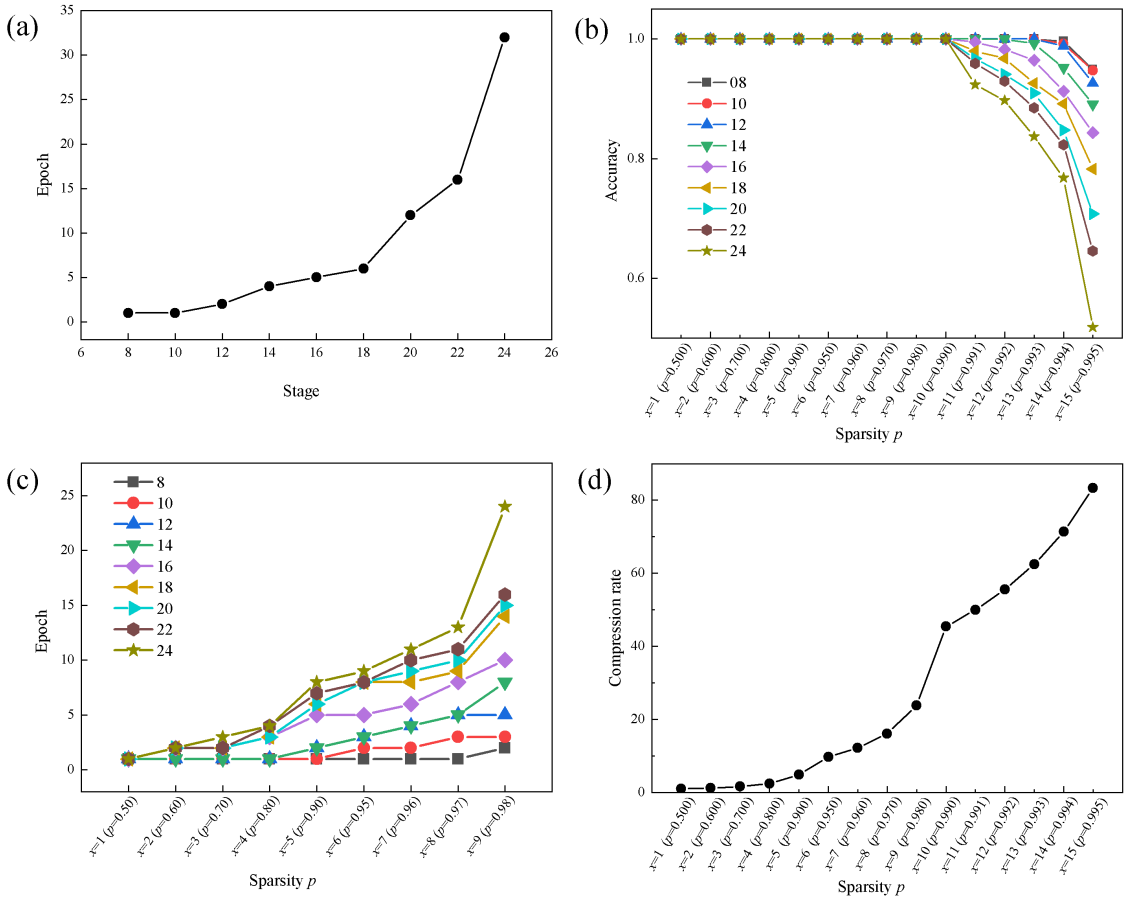


Fig. 5. Model training and pruning effect of M-sequences dataset. (a) The minimum training epochs required to converge to the highest accuracy; (b) The maximum accuracy of the model at different sparsity; (c) The minimum training epoch required for lossless compression models at different sparsity; (d) The compression rate of the model at different sparsity.

a known fixed value of 0. The min-entropy estimates for different stages of M-sequences are displayed in Table III, and the lowest entropy estimates from the 90B's predictors and our TPA-LSTM model for each stage are shown in bold.

For M-sequences with stage≤18, the MultiMMC predictor suggested in the final 90B presents the most accurate entropy estimation results. However, when the stage of M-sequences is higher than 18, the MultiMMC predictor gives completely wrong entropy estimation results, which is caused by the limitation of the machine learning algorithm itself. At this point, the period of the M-sequences ($2^{20} - 1$) is already larger than the input sample size. Without knowing the full period, the 90B's predictors fail to predict the sequence. The test results of the RNN predictor differ significantly from the theoretical value, probably because the gradient disappearance or the gradient explosion problem prevents it from learning a fixed pattern in the M-sequence. In contrast, the FNN predictor and our TPA-LSTM model are able to consistently give exactly correct results even when the M-sequences period is larger than $2^{24} - 1$ and the input sample is only 1M. This result suggests that the TPA-LSTM model has better performance in detecting long-term dependencies or patterns in long-term complex time series.

We record the minimum training epochs required to reach the highest accuracy for M-sequences of different stages, as shown in Fig. 5 (a). For the M-sequences with stage≤18, the minimum training epochs required increases slowly as the M-sequence stage increases. Nevertheless, when the stage of M-sequence is greater than 18, the training epochs required grows exponentially with the increase of M-sequences stages. By increasing the input samples and adjusting the

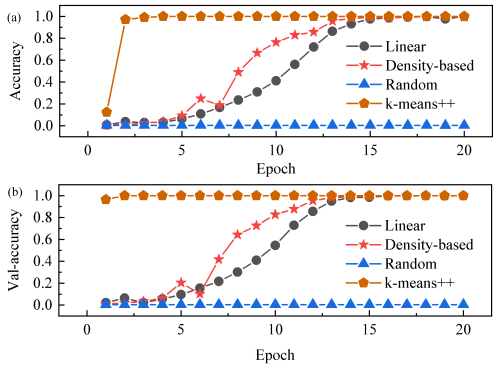


Fig. 6. Comparison of different initialization methods of clustering quantization on unpruned networks. (a)training set accuracy; (b) Verification set accuracy.

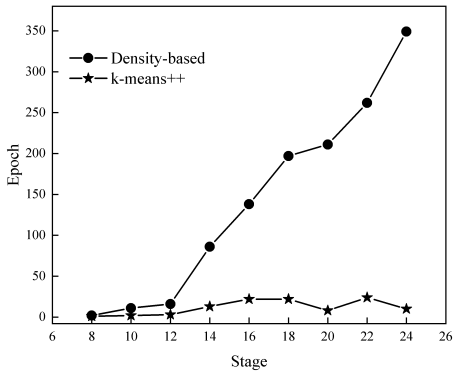


Fig. 7. Convergence speed comparison of different initialization methods.

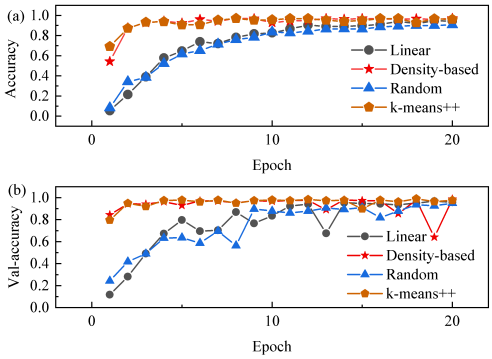


Fig. 8. Comparison of different initialization methods of clustering quantization on pruned network. (a)training set accuracy;(b) Verification set accuracy.

corresponding learning rate and batch size, the number of training epochs of the model can be effectively reduced.

To better validate the effectiveness of the pruning method and explore the parameter redundancy of the model, we setup a group of different sparsities p. Fig. 5 (b) shows the maximum accuracy that can be achieved for M-sequences at different sparsities. At this time, the model accuracy no longer increases even if the model is retrained for tens of epochs. As the sparsity p increases, the model accuracy starts to drop obviously after reaching a certain threshold value. Due to the different complexity of M-sequences, the models have unequal parameter redundancy. When beyond the certain sparsity threshold, M-sequences with high stages suffer from lower accuracy at the same sparsity. Even so, for different stages of M-sequences, the model accuracy with a sparsity less than or equal to 98% is still comparable to that of the original model. This experimental result further demonstrates the effectiveness of our pruning method.

When the sparsity is less than or equal to 98%, the minimum iterative pruning epochs required for the model to achieve lossless compression is shown in Fig. 5 (c). The parameter redundancy of the model is different when evaluating data of different complexity. Complex sequences tend to have less redundant parameters. Therefore, more epochs are needed to recover the accuracy loss of complex sequences due to pruning at the same sparsity. Finally, the model compression rates for different sparsities are shown in Fig. 5 (d).

We then examine the effect of the cluster quantization methods on the model performance, and conducted experiments on different stages of M-sequences. The experiments are first performed on the unpruned network to compare four different initialization centroid methods. Fig. 6 (a) and (b) show the variation of the training and validation set accuracy of the model with epochs when the model is quantified by different initialization centroid methods, respectively. The results indicate that the model quantified by the random initialization method loses its predictive power completely, and its accuracy drops to 0.39% after initialization and does not improve within the set clustering epochs. This may be due to that all the random initialized centroids are located near 0 value. According to the results of pruning experiments, it can be confirmed that the weights with small absolute values are of low importance for the model. The model accuracy can gradually recover to the highest after more epochs of training. In comparison, the model quantified by density-based and linear initialization approaches gradually returns to the best performance within the given clustering epochs. Ultimately model quantified by K-means++ initialization method achieves the best results and the model can converge to the peak precision rapidly.

We make further experiments using M-sequence datasets of different stages to compare the speed of the two best performing initialization methods: k-means++ and the density-based initialization. The training epochs required for the quantized model to converge to the top accuracy are shown in Fig. 7. The epochs required for the density-based initialization model scales up exponentially with the sequence complexity. Oppositely, the epochs required for the k-means++ initialization model increase only slightly. It is revealed again that the k-means++ initialization method converges significantly faster than the other initialization methods when quantizing the unpruned network.

Fig. 8 exhibits comparison of different initialization methods of clustering quantization on pruned network, where (a) and (b) denote the trend of accuracy with increasing epochs in the training and validation sets, respectively. The sparsity of the pruned model is $p = 97\%$. The experimental results suggest that the model quantified by all four initialization methods can gradually recover to the greatest accuracy as the training epochs increase. This is due to the fact that the pruned model has removed most of the relatively unimportant weight parameters. The Density-based and K-means++ initialization methods exhibit similar performance, and both surpass the other initialization methods. This is because the

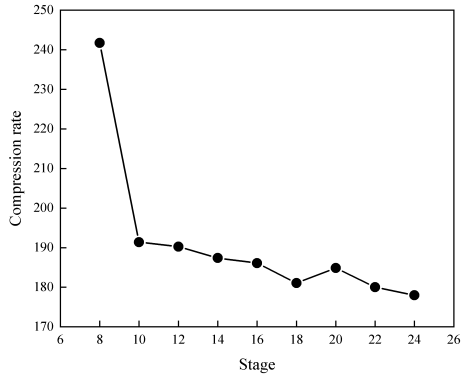


Fig. 9. Compression multiples of different stages of M-sequences using pruning and clustering quantification methods.

pruned network is quantized to 2 bits, i.e., $2^2 = 4$ clustering centers. At this point, the linear initialization and random initialization centroids deviate significantly from the majority weights, while the density-based initialization method and the K-means++ initialization method ensure that the majority weights are close to the clustering centroids and therefore require fewer training epochs.

Finally, we apply the pruning and cluster quantization together to compress the model, and the model size can be compressed by more than 175 times, and the compression multipliers for different stages of the M-sequences are plotted in Fig. 9.

V. Conclusion

In this paper, we proposed an approach based on pruning and quantized deep neural networks to estimate entropy more accurately for various random sequences. A three-phase random number evaluation scheme has been given to assess the unpredictability of random numbers. It should be noted that our scheme is RNG-agnostic and thus can evaluate the unpredictability of any RNG. Because random sequences belong to time series, this feature is just conducive to the TPA-LSTM model. Similarly, the introduction of temporal pattern attention allows better learning of hidden associations in complex time series data than typical attention mechanisms. Therefore, our proposed TPA-LSTM prediction model can evaluate various random sequences more accurately compared to the 90B predictors. By pruning and quantization operations, the complexity and storage size of the model can be reduced, and the model accuracy is not degraded. Extensive experiments have been conducted to evaluate the performance of our proposed method, and allow us to draw the following conclusions.

1) The proposed TPA-LSTM model has better prediction accuracy compared to 90B. For min-entropy evaluation, the relative error of our proposed method is only 0.43% at the highest, while the 90B is as high as 7.32%.
2) We firstly validate the application of pruned and quantized deep neural networks in the field of entropy evaluation. The model is compressed by more than 175x and the accuracy is comparable to or even better than the original model.
3) The complexity of the data affects the efficiency and results of pruning and quantization, and we can ensure the accuracy of the model by increasing the corresponding training epochs.
4) The k-means++ initialization method can quantize the unpruned network faster and without loss. Its effect is comparable to the density-based initialization method in quantizing the pruned network, and both outperform other initialization methods.

We believe that the method based on pruning and quantized deep learning prediction models is expected to provide an effective complement to evaluate the safety and quality of RNGs. In the future, we will further explore and enhance the predictive power and generalization ability of the model. Secondly, knowledge distillation and binarization methods can be considered to further simplify the model. In addition, we consider building an online fast evaluation system for random sequences. The hardwareization of the prediction model is realized to evaluate the security of random sequences in real time while ensuring the evaluation accuracy. This will provide new ideas for the online security analysis of RNGs.

References

[1] W. Wei, G. Xie, A. Dang, and H. Guo, "High-speed and bias-free optical random number generator," *IEEE Photon. Technol. Lett.*, vol. 24, no. 6, pp. 437–439, Mar. 15, 2012.
[2] P. Li et al., "Parallel optical random bit generator," *Opt. Lett.*, vol. 44, no. 10, pp. 2446–2449, 2019.
[3] M. Sciamanna and K. A. Shore, "Physics and applications of laser diode chaos," *Nature Photon.*, vol. 9, pp. 151–162, Feb. 2015.
[4] K. Keigo, Y. Terashima, K. Kanno, and A. Uchida, "Entropy evaluation of white chaos generated by optical heterodyne for certifying physical random number generators," *Opt. Exp.*, vol. 28, no. 3, pp. 3686–3698, Jan. 2020.
[5] A. Martínez, A. Solis, R. D. H. Rojas, A. U'Ren, J. Hirsch, and I. P. Castillo, "Advanced statistical testing of quantum random number generators," *Entropy*, vol. 20, no. 11, p. 886, Nov. 2018.
[6] D. Lambić, "Security analysis and improvement of the pseudo-random number generator based on piecewise logistic map," *J. Electron. Test.*, vol. 35, no. 4, pp. 519–527, Aug. 2019.
[7] Z. Wang, H. Yu, Z. Zhang, J. Piao, and J. Liu, "ECDSA weak randomness in bitcoin," *Future Gener. Comput. Syst.*, vol. 102, pp. 507–513, Jan. 2020.
[8] W. Killmann and W. Schindler, "A design for a physical RNG with robust entropy estimators," in *Proc. 10th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Aug. 2008, pp. 146–163.
[9] Y. Ma, J. Lin, T. Chen, C. Xu, Z. Liu, and J. Jing, "Entropy evaluation for oscillator-based true random number generators," in *Proc. 16th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, vol. 8731, Sep. 2014, pp. 544–561.
[10] B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE Trans. Comput.*, vol. 56, no. 1, pp. 109–119, Jan. 2007.
[11] J. D. J. Golic, "New methods for digital generation and postprocessing of random data," *IEEE Trans. Comput.*, vol. 55, no. 10, pp. 1217–1229, Oct. 2006.
[12] P. Z. Wieczorek and K. Golofit, "Dual-metastability time-competitive true random number generator," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 1, pp. 134–145, Jan. 2014.
[13] A. L. Rukhin, J. Soto, and J. R. Nechvatal, *Sp 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. Gaithersburg, MD, USA: NIST Special Publication, 2010. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf
[14] W. Killmann and D. W. Schindler, *BSI AIS 31: Functionality Classes and Evaluation Methodology for True (Physical) Random Number Generators, Version 3.1*. Bonn, Germany: T-Systems GEI GmbH and Bundesamt für Sicherheit in der Informationstechnik (BSI), 2001.

[15] G. Marsaglia. (1996). *The Marsaglia Random Number CDROM Including the Diehard Battery of Tests of Randomness*. [Online]. Available: http://www.stat.fsu.edu/pub/diehard/

[16] P. L'Ecuyer and R. Simard, "TestU01: A C library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 1–40, Aug. 2007.

[17] D. Hurley-Smith and J. Hernandez-Castro, "Certifiably biased: An in-depth analysis of a common criteria EAL4+ certified TRNG," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 4, pp. 1031–1041, Apr. 2018.

[18] F. Fan and G. Wang, "Learning from pseudo-randomness with an artificial neural network–does god play pseudo-dice?" *IEEE Access*, vol. 6, pp. 22987–22992, 2018.

[19] T. Fischer, "Testing cryptographically secure pseudo random number generators with artificial neural networks," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2018, pp. 1214–1223.

[20] Y. Feng and L. Hao, "Testing randomness using artificial neural network," *IEEE Access*, vol. 8, pp. 163685–163693, 2020.

[21] J. Kelsey, K. A. McKay, and M. S. Turan, "Predictive models for min-entropy estimation," in *Proc. 17th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, vol. 9293, Sep. 2015, pp. 373–392.

[22] J. Yang, S. Zhu, T. Chen, Y. Ma, N. Lv, and J. Lin, "Neural network based min-entropy estimation for random number generators," in *Proc. 14th European-Alliance Innov. (EAI) Int. Conf. Secur. Privacy Commun. Netw. (SecureComm)*, vol. 255, Aug. 2018, pp. 231–250.

[23] N. D. Truong, J. Y. Haw, S. M. Assad, P. K. Lam, and O. Kavehei, "Machine learning cryptanalysis of a quantum random number generator," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 403–414, Feb. 2019.

[24] C. Li et al., "Deep learning-based security verification for a random number generator using white chaos," *Entropy*, vol. 22, no. 10, p. 1134, Oct. 2020.

[25] Y. Yu, X. Si, C. Hu, and Z. Jianxun, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, 2019.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] S.-Y. Shih, F.-K. Sun, and H.-Y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Mach. Learn.*, vol. 108, nos. 8–9, pp. 1421–1441, Sep. 2019.

[28] M. S. Turan et al. (2018). *Recommendation for the Entropy Sources Used for Random Bit Generation*. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-90b/final

[29] E. Barker and J. Kelsey. (2012). *NIST Draft Special Publication 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation*. [Online]. Available: http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf

[30] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. D. Freitas, "Predicting parameters in deep learning," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, vol. 2, Dec. 2013, pp. 2148–2156.

[31] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[32] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 4860–4874.

[33] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, vol. 1, Dec. 2015, pp. 1135–1143.

[34] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," in *Proc. Brit. Mach. Vis. Conf.*, 2015, p. 31.

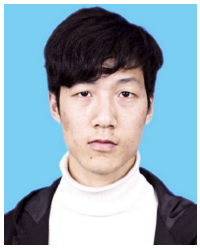
[35] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6071–6079.

[36] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *Fiber*, vol. 56, no. 4, pp. 3–7, 2015.

[37] M. H. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," in *Proc. 6th Int. Conf. Learn. Represent., (ICLR)*, 2018, pp. 1–11.

[38] W. Chen, J. T. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. ICML*, 2015, pp. 2285–2294.

[39] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2007, pp. 1027–1035.

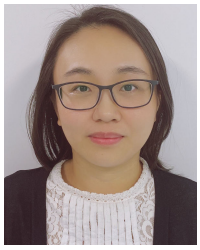


Haohao Li received the B.S. degree in measurement and control technology and instrumentation from Henan University, Henan, China, in 2020. He is currently pursuing the M.S. degree in instrument science and technology with the Taiyuan University of Technology, Taiyuan, China. His research interests include deep learning and random number security analysis.

Jianguo Zhang received the B.S. degree in automation, the M.S. degree in detection technology and automatic equipment, and the Ph.D. degree in circuit and system from the Taiyuan University of Technology, Taiyuan, China, in 2002, 2005, and 2013, respectively.

He is currently an Associate Professor with the Key Laboratory of Advanced Transducers and Intelligent Control System, Ministry of Education of China, College of Physics and Optoelectronics, Taiyuan University of Technology. His research interests include information security technology, deep learning, and hardware implementation of neural networks.

Zhihu Li received the M.S. degree in applied mathematics from the Beijing University of Posts and Telecommunications, Beijing, China, in 2004. He is currently a Senior Expert of the China Electric Power Research Institute and the Executive Director of IEEE PES Smart Grid Block Chain Sub-Technical Committee (China). His research interests include cryptography application technology, data security, and artificial intelligence technology.



Juan Liu received the B.S. degree in electronic information and engineering and computer science and technology and the M.Eng. degree in electronic information and engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005 and 2007, respectively. She is currently the Executive Director working at Nations Technology Company Ltd., Shenzhen, China. Her current research interests include hardware security, deep learning, and IC implementation.

Yuncai Wang received the B.S. degree in semiconductor physics from Nankai University, Tianjin, China, in 1986, and the M.S. and Ph.D. degrees in physics and optics from the Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Taiyuan, Shanxi, China, in 1994 and 1997, respectively.

In 1986, he joined the Taiyuan University of Technology (TYUT), Taiyuan, as a Teaching Assistant. From 1994 to 1998 and from 1998 to 2003, he was a Lecturer and then an Assistant Professor with the Department of Physics, TYUT. From 2003 to 2018, he was a Professor with the College of Physics and Optoelectronics and the Director of the Key Laboratory of Advanced Transducers and Intelligent Control System, Ministry of Education of China, TYUT. In 2019, he joined the Guangdong University of Technology (GDUT), Guangzhou, China, where he is currently the Deputy Director of the Advanced Institute of Photonic Technology. His research interests include nonlinear dynamics of semiconductor lasers, all-optical analog-to-digital conversion, and chaotic secure communication.

Dr. Wang is a fellow of the Chinese Instrument and Control Society and a Senior Member of the Chinese Optical Society and the Chinese Physical Society.