Revisiting Prediction-Based Min-Entropy Estimation: Toward Interpretability, Reliability, and Applicability

Dongchi Han, Yuan Ma, Tianyu Chen, Shijie Jia, Na Lv, Fangyu Zheng, and Xianhui Lu

***Abstract*—Prediction-based min-entropy estimation methods, also known as predictors, are essential tools for assessing the security of entropy sources. As recommended in NIST SP 800-90B (90B), these methods estimate min-entropy by forecasting the outputs of entropy sources. Owing to their computational efficiency, considerable research has focused on enhancing the accuracy of predictors, including approaches based on deep neural networks (DNNs). However, concerns remain about their interpretability, reliability, and applicability, particularly for DNN-based predictors. In this paper, we first identify key deficiencies in existing prediction-based methods, including those in 90B and DNN-based predictors, which lead to unreliable estimates and poor adaptability across diverse entropy sources. To improve reliability, we model the predictor output distribution and revise the local predictability metric to produce more stable estimates with associated confidence levels. To enhance the interpretability of DNN-based predictors in entropy estimation, we provide the first theoretical analysis linking neural network optimization objectives to min-entropy, clarifying the suitability and learnability of different architectures. We further reveal the inapplicability of existing methods under time-varying sources and propose a new estimation framework that combines online learning, change detection, and Bayesian optimization for dynamic model updates. The experimental results demonstrate that our methods surpass existing approaches in terms of reliability and applicability, especially when dealing with time-varying sources.**

***Index Terms*—Min-entropy estimation, random number, deep neural network, interpretability.**

I. Introduction

ENTROPY sources are fundamental components in cryptography for generating random numbers. They rely on unpredictable physical occurrences, such as noise from electronic devices, or non-physical events, such as system interrupt characteristics, to produce raw random numbers. Random number generators (RNGs) further leverage these entropy sources to produce high-quality random numbers necessary for creating encryption keys, initialization vectors, and other security parameters. However, if the entropy source produces random numbers without sufficient randomness, the RNG's output becomes predictable. This vulnerability can lead to the cracking of cryptographic keys, leakage of sensitive information, and eavesdropping on communications [1]. Therefore, it is crucial to ensure the quality of the entropy source output.

The NIST SP 800-90B (short for 90B) standard, developed by the National Institute of Standards and Technology (NIST), is crucial for assessing the quality of randomness in data sequences generated by entropy sources [2]. This standard provides guidelines for designing and testing these sources. It uses min-entropy estimation to measure randomness, which has been widely used in the Cryptographic Module Verification Program (CMVP) of America and Canada. The 90B employs black-box testing with two entropy estimation approaches: (1) entropic statistical methods, which identify major flaws but often overestimate or underestimate randomness—introducing security risks or performance trade-offs [3], [4], [5]; and (2) prediction-based methods (predictors), which forecast future outputs from entropy sources and calculate the min-entropy based on these predictions [2].

A. Progress and Issues of the Predictors

Predictors have outperformed the statistical approaches [6], drawing increasing research interest. However, studies have highlighted persistent issues with stability and applicability, emphasizing the need for more robust and generalizable models [7], [8], [9]. Notably, predictors utilizing DNNs are quite attractive due to their ability to manage intricate outputs from various types of entropy sources. The design of DNN-based predictors has primarily progressed in two technical directions. The first concerns data preprocessing, where methods such as change detection are employed to segment data streams and extract statistical features. This approach enables even relatively simple network architectures to process time-varying entropy sources effectively [7]. The second focuses on network architecture design, aiming to improve predictive performance through more expressive models. For instance, more complex structures such as the temporal pattern attention-based long

Received 23 June 2024; revised 3 June 2025 and 29 July 2025; accepted 26 August 2025. Date of publication 8 September 2025; date of current version 18 September 2025. This work was supported in part by the CAS Project for Young Scientists in Basic Research under Grant YSBR-035, in part by IIE Project under Grant E4Z00111A4, in part by the National Natural Science Foundation of China under Grant 62272457, and in part by the National Cryptologic Science Fund of China under Grant 2025NCSF02005. The associate editor coordinating the review of this article and approving it for publication was Dr. Qiben Yan. *(Corresponding author: Yuan Ma.)*

Dongchi Han, Yuan Ma, Tianyu Chen, Shijie Jia, Na Lv, and Xianhui Lu are with the State Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100085, China, and also with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100864, China (e-mail: handongchi@iie.ac.cn; mayuan@iie.ac.cn; chentianyu@iie.ac.cn; jiashijie@iie.ac.cn; lvna@iie.ac.cn; luxianhui@iie.ac.cn).

Fangyu Zheng is with the School of Cryptology, University of Chinese Academy of Sciences, Beijing 100864, China (e-mail: zhengfy1028@hotmail.com).

Digital Object Identifier 10.1109/TIFS.2025.3607168

short-term memory (TPA-LSTM) network have been proposed [10]. Additionally, model compression techniques like pruning and quantization help eliminate redundancy, improve accuracy, and reduce output variability [9].

Although DNNs show unique advantages over traditional statistical methods in random number prediction, the underlying mechanism of the prediction and the performance for different entropy sources are lacking in the literature. The problems of existing predictors can be elaborated in the following three aspects.

- **Interpretability:** The relationship between the neural network optimization objective and min-entropy remains theoretically unexplained, raising doubts about the validity of using the network's accuracy to represent min-entropy. Furthermore, it is unclear what neural networks actually learn during entropy estimation. Key questions remain open: *which architectures are suitable for entropy estimation, and what representations do these networks capture in the process?*
- **Reliability**: Problems with the measure of local predictability in entropy sources could lead to large fluctuations in results. Up to 24% fluctuations were observed for some DNN-based predictors in our experiments.
- **Applicability:** All predictors struggled with entropy sources exhibiting sudden distribution shifts, and change detection techniques [7] did not resolve the issue. Additionally, when deterministic periodic signals were introduced into a random source, the 90B predictors and some DNN-based predictors tended to overestimate the results.

B. Contributions and Paper Organizations

This paper revisits prediction-based min-entropy estimation, addressing reliability and applicability concerns through a novel local predictability measure and a new DNN estimation framework. We analyze how different neural network architectures behave in entropy estimation tasks and propose optimized network structures for various entropy sources. Our key contributions include:

1) **Theoretical analysis and comprehensive experiments** are made to analyze and identify the underlying issues that lead to the unreliability and limited applicability of existing predictors. Thus, we have provided a clear understanding of the challenges that must be addressed.
2) **A theoretical analysis** of the relationship between the neural network optimization objective and min-entropy is presented. We also provide **a detailed interpretation** of what different architectures learn from various entropy sources through weight visualization and result analysis. Based on these insights, we propose neural network structures tailored to different entropy sources. As far as we know, we are the first to provide the interpretation for DNN-based predictors.
3) **A redesigned local predictability measure** is proposed, grounded in **a theoretical model** of predictor outputs, to mitigate result fluctuations. In addition, we develop **a new DNN training framework** that integrates online learning, change detection, and Bayesian optimization, improving the adaptability and reliability of entropy estimation for time-varying sources.
4) **A comprehensive evaluation** of our proposed methods and framework across multiple datasets is made. Experimental results show that our designed networks converge faster and achieve higher accuracy. Integrating our proposed framework, our predictors significantly outperform current ones on time-varying data. This offers new insights into entropy estimation design.

The paper is organized as follows. In Section II, we introduce the entropy estimation methods used by 90B and the existing DNN-based predictors. In Section III, we analyze the limitations of the current prediction-based methods. In Section IV, we interpret the underlying mechanics for DNN-based predictors in entropy estimation tasks. In Section V, we illustrate the redesigned measure and the new framework devised, followed by comprehensive experiments to demonstrate the improved performance of our methods. Section VI discusses limitations and adversarial threats, and proposes mitigation approaches. Section VII concludes the paper.

II. Preliminaries

A. Min-Entropy Definition in 90B

According to 90B, the entropy source produces outputs described by the random variable X. When X is an independent discrete random variable that can take on values from a finite set $S = \{x_1, x_2, x_3, \ldots, x_k\}$, if the probability of X being equal to x_i is $P\{X = x_i\} = p_i$ for $i = 1, 2, 3, \ldots, k$, then the min-entropy is:

$$H_\infty = \min_{1 \le i \le k}(-\log_2 p_i) = -\log_2 \max_{1 \le i \le k} p_i. \tag{1}$$

B. The Predictors Outlined in 90B

The predictors are initialized using a limited number of samples before evaluating the entire sequence. For each sample, the predictor generates a predicted value based on prior accumulated knowledge, then updates itself with the observed value to improve its accuracy. These predictors are composed of multiple sub-predictors, each with different parameters, and they dynamically select the best-performing sub-predictor's results. These predictors include the MultiMCW prediction estimate, the Lag prediction estimate, the MultiMMC prediction estimate, and the LZ78Y prediction estimate.

By evaluating both the *global* predictability and the *local* predictability of the entropy source outputs, the predictors determine the min-entropy of the entropy source [6]. The details are shown as follows.

Global Predictability. This metric counts the number of correct predictions in the entire sequence, known as accuracy in machine learning [2]. The accuracy of a predictor on a test sequence is indicated by $\hat{p}_{acc} = c/n$, with c as the number of correct predictions and n as the length of the test sequence. Assuming each prediction outcome is independent and identically distributed, a 99% upper confidence bound for

the estimate $\hat{p}$ can be computed [6]. This upper confidence bound is denoted as p_{global}:

$$p_{global} = \begin{cases} 1 - \left(\frac{\alpha}{2}\right)^{\frac{1}{n}}, & \text{if } \hat{p}_{acc} = 0, \\ 1, & \text{if } \hat{p}_{acc} = 1, \\ \hat{p}_{acc} + 2.576\sqrt{\frac{\hat{p}_{acc}(1-\hat{p}_{acc})}{n-1}}, & \text{otherwise.} \end{cases}$$

Local Predictability. This measure assesses min-entropy by examining consecutive accurate predictions made by the predictor. A series of precise forecasts indicates a swift shift of the entropy source outputs to a highly predictable state. Kelsy et al. [6] suggested that the statistical tools used for analyzing recurrent events can provide a local prediction accuracy p_{local} as a basis for calculating min-entropy.

Let ε denote a characteristic of finite sequences, which can be detected within any specific sequence $E_{j1}, \ldots, E_{jn}$. Suppose r is a positive integer, and ε indicates the presence of a consecutive success run of length r in a series of Bernoulli trials, which is a recurrent event [11]. Since the experiment can be repeated indefinitely, the sample space of the random variable ε is defined on the set of infinite sequences [11].

Let f_n denote the probability of ε occurring for the first time in the nth experiment:

$$f_n \sim \frac{(x-1)(1-px)}{(r+1-rx)q} \cdot \frac{1}{x^{n+1}}. \tag{2}$$

In n experiments, the probability of ε not occurring can be expressed as $q_n = f_{n+1} + f_{n+2} + \ldots$, which can be simplified to:

$$q_n \sim \frac{1-px}{(r+1-rx)q} \cdot \frac{1}{x^{n+1}}. \tag{3}$$

The probability of not encountering a recurrent event ε with a length of $r+1$ with 99% certainty can be expressed by the following formula:

$$q_n = 99\% = \frac{1-p_{local}x}{(r+2-(r+1)x)(1-p_{local})} \cdot \frac{1}{x^{n+1}}, \tag{4}$$

This equation is used in 90B to compute the local prediction accuracy p_{local}, where n represents the number of predictions, and x is the positive real root of the polynomial [11]:

$$1 - x + (1-p_{local})p_{local}^{r}x^{r+1} = 0, \tag{5}$$

where x can be easily determined by approximation methods. The final result of the predictors can be represented as:

$$\hat{H}_{\infty} = -\log_2(\max(p_{global}, p_{local})). \tag{6}$$

C. DNN-Based Entropy Estimation Methods

In 2018, Yang et al. [8] introduced DNN-based predictors using two basic architectures, FNN and RNN. Then, Lv et al. [12] discovered that 90B could not detect sequences with long-range dependencies, such as M-sequences generated by linear feedback shift registers (LFSRs). They also evaluated Yang et al.'s predictors on a broader dataset, proposed hyperparameter tuning strategies, and improved the FNN-based predictor. In 2019, Truong et al. [13] proposed the RCNN model to evaluate classical and quantum noise sources. Experiments with linear congruential generators (LCGs) demonstrated the model's ability to predict pseudo-random sequences from higher-order LCGs. In 2020, Zhu et al. [7] identified discrepancies in the 90B predictors when applied to time-varying sources. To address this, they proposed the CDNN model, which incorporates change detection (via the ADWIN2 algorithm) for data segmentation and feature extraction. CDNN outperformed the 90B predictors on low-bit-width, time-varying sources. Li et al. [10] later introduced the TPA-LSTM model with an attention mechanism for analyzing White Chaos-Based RNGs. This model further improved prediction accuracy on high-order LCGs and outperformed RCNN. In 2023, Li et al. [9] optimized TPA-LSTM through quantization and pruning, reducing model complexity and storage while preserving accuracy. Comparative experiments confirmed TPA-LSTM's superiority over the 90B predictors.

In prior work, DNN models were used for min-entropy estimation through supervised learning within a three-phase framework [9]. The first phase is data preprocessing, where the predictor constructs a dataset by grouping each sequence of L adjacent k-bit values as a feature vector, with the $(L+1)$th value serving as the label. Each feature-label pair forms a sample, and all samples together constitute the dataset. Unlike the predictors in 90B, this framework partitions the input data into separate training, validation, and test sets. The second phase involves model training using a train-validation strategy over multiple epochs. Training stops either after a preset number of epochs or via early stopping when validation performance plateaus. Model parameters are saved at each epoch, and the version achieving the best validation performance is selected for testing. The third phase estimates min-entropy by evaluating the selected model on the test set. Similar to 90B, two metrics, p_{global} and p_{local}, are used to quantify prediction performance.

Previous research has shown that DNN-based predictors offer superior entropy estimation accuracy compared to the 90B predictors. In the next section, we evaluate their limitations through experiments using a range of models, including RNN and FNN (with Tanh activation) from [8], FNN (with ReLU activation) from [12], CDNN from [7], and TPA-LSTM from [9]. Since both LSTM and CNN have capabilities for analyzing time series, we also used these two models to demonstrate our findings, and they use one-hot encoding. LSTM passes the final hidden state to a classification layer, while the CNN (ReLU-activated) flattens convolutional outputs into a 1D vector, which is processed by a fully connected ReLU layer before classification. Both CNN and LSTM have an input channel size of 2^k.

III. New Findings and Analysis on Reliability and Applicability of Existing Predictors

We examine the reliability and applicability issues of current predictors. First, we assess local predictability in entropy sources, identify causes of unreliability, and experimentally validate our findings. Next, we analyze how training methods lead to poor performance on time-varying entropy sources. Our experiments reveal a composite source where both 90B and some DNN-based predictors overestimate results. We pinpoint

TABLE I
HYPERPARAMETER SETTINGS

Hyperparameter	Value
Batch size	1280
Epochs	100
Learning rate	0.0005
History length	10
Ratio of Training, Validation, and Test Sets	6:2:2

their shortcomings and lay the groundwork for discussing the interpretability of neural networks.

A. Experiments Setup

The experiments are conducted on a Ubuntu 20.04 system with an NVIDIA GeForce RTX 2080 Ti GPU, and the DNN models are implemented using PyTorch. Since previous works use Keras to implement models, we adopt Keras's default weight initialization method in PyTorch. For model training, we employ the Adam optimizer. We set the same hyperparameter values for the selected models to exclude the influence of discrepancies in hyperparameters. Specifically, we follow the hyperparameter settings from the latest work on DNN-based predictors in [9], as shown in Table I. We monitor the validation loss and stop training if it does not decrease for five consecutive epochs, thereby preventing overfitting.

B. The Problem for the Reliability

Current research highlights a critical issue: existing predictors often show significant result fluctuations due to the local predictability measure (p_{local}), compromising their reliability. Zhu et al. [7] reported that their CDNN model underestimated entropy due to the use of p_{local} on gradually changing Markov data. Li et al. [9] found that the 90B predictors also face this issue. They demonstrated that p_{local}'s high sensitivity to the parameter r led to underestimation, causing significant result fluctuations across tests on different data from the same source.

Upon further analysis, we argue that the root cause is not the sensitivity to r, but rather the definition of local predictability itself. As defined in 90B, when a predictor correctly forecasts the output of an entropy source for r consecutive times, the source is considered locally predictable, resulting in lower entropy estimates. However, this phenomenon is a recurrent event inherent in the prediction results. According to Equation (3), the probability of such an event depends on r, the test set length n, and the predictor's accuracy p. Therefore, given a stable predictor and a sufficiently long sequence, these events will inevitably occur, resulting in a significant underestimation.

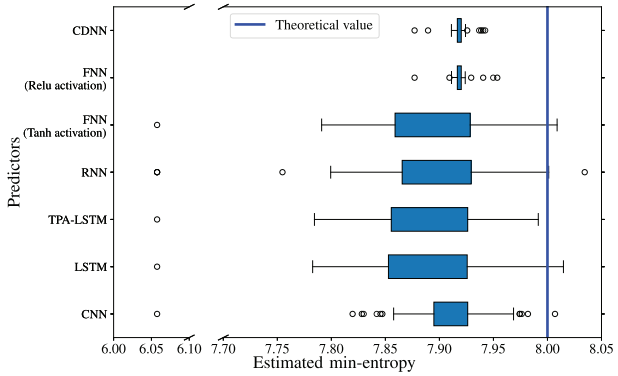
Furthermore, existing DNN-based predictors rely on random weight initialization, leading to varying estimation results when the same sequence is tested multiple times. This raises a significant question: do these DNN-based predictors exhibit underestimation due to p_{local} when repeatedly testing the same sequence? Unfortunately, the specific influencing factors and the impact degree of underestimation caused by p_{local} in DNN-based entropy estimators have not been thoroughly investigated. We believe that for the same sequence, each random initialization effectively results in a different predictor. With sufficient tests, one of these predictors will inevitably encounter recurring events, leading to underestimation.

To verify our analysis, we conduct comparative tests using two representative random sources: a discrete uniform distribution and a first-order Markov model. From these sources, we generate two random sequences, each containing one million 8-bit samples. Next, we test seven different DNN-based predictors, each utilizing a unique model (see Section II-C). These predictors are referred to by the names of their respective models. Each predictor performs 100 estimations on the two generated sequences to assess the stability of the predictor's output. The results, comparing the theoretical min-entropy values to the estimated min-entropy values from the different DNN-based predictors, are shown in Fig. 1. The theoretical min-entropy value for the discrete uniform distribution is calculated using Equation (1), while the first-order Markov model's min-entropy is derived following the method in [6].

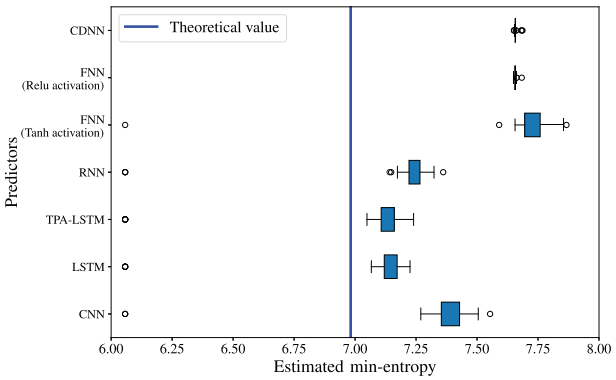
From Fig. 1, we observe that, except for CDNN [7] and FNN [12], all other predictors show significant fluctuations. In Fig. 1 (a), fluctuations reach up to 24%, and in Fig. 1 (b), they reach up to 13%. These fluctuations are due to significant underestimation, which causes the noticeable abnormal point near the entropy value of 6.0. We now analyze the reasons for these fluctuations. As shown in Equation (6), the result of each predictor is determined by the higher value between p_{local} and p_{global}. Random weight initialization in DNNs causes fluctuations in p_{global}. In Fig. 1 (a), the values of p_{global} range from 0.37% to 0.47%, and in Fig. 1 (b), the values of p_{global} for the best-performing predictors (TPA-LSTM and LSTM) range from 0.65% to 0.75%. With r values of 1 or 2, p_{local} is less than 0.36%, making p_{global} the determining factor for estimated min-entropy. In cases of significant underestimation, all r values for these predictors are 3, and the corresponding p_{local} is around 1.5%. Consequently, p_{local} determines the estimated min-entropy for each predictor. In Fig. 1 (a), the recurrent events occurred once for each predictor. In Fig. 1 (b), TPA-LSTM experienced 11 recurrent events, while LSTM experienced 7. According to Equation (3), these frequencies are consistent with the predictors' accuracy.

Additionally, CDNN [7] and Lv et al.'s FNN [12] yield relatively stable estimates. Analysis of their intermediate results reveals that the ReLU activation results in predominantly "zero" inputs to the classification layers, causing the predictors to output the same prediction value consistently. As a result, these predictors did not show underestimation from p_{local} in our experiments. However, their prediction patterns deviate from the first-order Markov model, leading to noticeable overestimation and raising applicability concerns, which we discuss in the next subsection.

Based on the above experimental results, we confirm that the successive correct predictions produced by the predictor are recurrent events, leading to underestimations caused by p_{local}. The probability of such events correlates with predictor accuracy, and this issue also exists in DNN-based predictors. Furthermore, random weight initialization in DNNs can cause significant underestimation even when testing the same

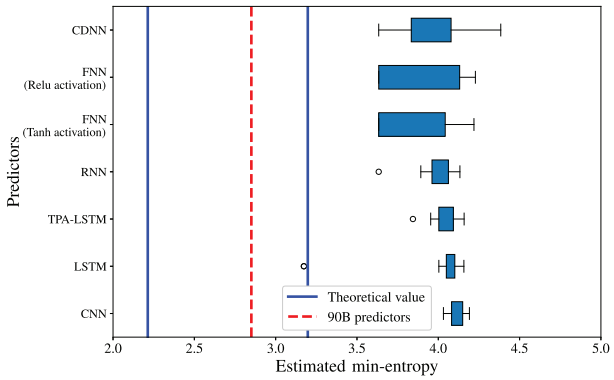


(a) Results on the discrete uniform distribution random sequence.

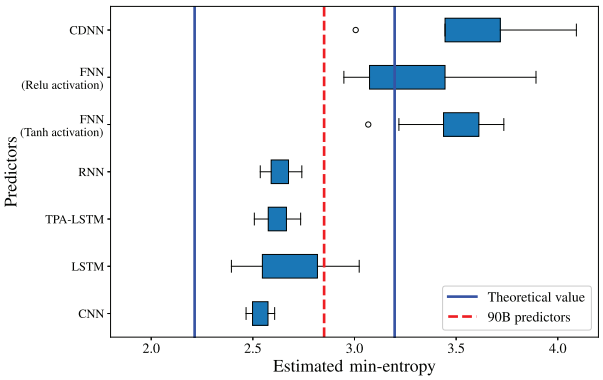


(b) Results on the first-order Markov model random sequence.

Fig. 1. The repeating estimation results of different DNN-based entropy predictors for stable data.



(a) Results on the first-order Markov model with sudden change using the ratio of 1:1.



(b) Results on the first-order Markov model with sudden change using the ratio of 4:1.

Fig. 2. The repeating estimation results of different DNN-based entropy predictors with different data size ratios of training to testing.

random sequence multiple times. This leads to substantial fluctuations in estimation results, thus reducing the reliability.

C. The Problem for the Applicability

Existing research on the applicability problem mainly focuses on time-varying sources. Zhu et al. [7] first addressed this issue with CDNN, which uses change detection and performs well on low-width, time-varying sources. However, due to limitations in its training methods, CDNN struggles when there is a significant discrepancy between the test and training datasets [7]. Furthermore, our earlier experiments show that the applicability problem persists for CDNN and FNN with stationary sources. Therefore, this subsection explores the factors affecting the applicability by analyzing the *training method*s and *predictive models*.

1) Training Methods: Existing DNN-based predictors use data partitioning to split data into training and test sets. This approach can lead to two potential scenarios: (1) If the training set contains data similar to the test set, the model will adjust its parameters accordingly, which helps maintain accuracy during testing if the similar data in the training set is sufficient. (2) Conversely, if the training set is entirely different from the test set, the model's performance during testing will be biased due to the absence of relevant data during training.

Although the 90B predictors use a learn-while-testing approach to adapt to recently arrived data and avoid the issues mentioned above, they can still temporarily overestimate when there are significant changes in the data. This occurs because the accumulated historical knowledge becomes less effective. However, as training progresses and the model adapts to the new data, the accuracy increases over time. Therefore, we figure out that when testing the same sequence with both 90B and DNNs, the following scenarios occur. If the DNNs' training set matches the first case, DNNs will outperform 90B due to 90B's partial overestimation. On the contrary, if the training set matches the second case, 90B will outperform DNNs, as DNNs are being optimized towards incorrect data.

To validate the conclusion, we test typical time-varying data sequences with sudden changes, where each stable segment follows a first-order Markov model. The sequence consists of 1,000,000 4-bit samples, with the sudden change occurring in the middle. The theoretical entropy of the first part is about 3.2 bits per sample, and that of the rest is about 2.2 bits per sample. We adopt two data partitioning methods: 1:1 and 4:1 (the data size ratio of training to testing), and the estimation results by different DNN-based predictors are shown in Fig. 2.

Fig. 2 (a) shows that all DNN-based predictors provide overestimated results at the ratio of 1:1. The 90B predictors outperform DNN-based predictors. Fig. 2 (b) shows the results

TABLE II
RESULTS OF THE COMPOSITE SOURCE (ENTROPY PER SAMPLE)

Predictors	CDNN [7]	FNN [12]	FNN [8]	**TPA-LSTM [9]**
Results	4.424	4.320	4.591	**4.065**
Predictors	RNN [8]	LSTM	CNN	90B Predictors
Results	4.109	4.066	4.108	4.789

of the ratio of 4:1. All DNN-based predictors outperform their 1:1 split counterparts and 90B.

The above experimental results demonstrate that training methods significantly impact the predictor's applicability. Data partitioning can cause the predictor to fail on unseen data. Conversely, using a learn-while-testing approach like 90B may lead to slight overestimation due to model update inefficiencies. Therefore, a new training method should be designed to enhance the predictor's applicability to time-varying sources.

2) Predictive Models: We have noted that structures like CDNN and FNN are inadequate for handling data correlations. Linear layers create a linear combination of inputs, which, even with activation functions, fail to capture dependencies between time steps. In contrast, recurrent structures like RNNs maintain and update a hidden state over time, enabling them to capture such sequential relationships.

The 90B predictors ignore temporal dependencies between time steps. Their approach is simple: they track (prefix, next value) frequencies and predict the most common next value. However, they perform no further analysis on these patterns.

Based on this understanding, we hypothesize that superimposing a deterministic signal on noise introduces correlations. Models with an FNN structure should struggle to handle these correlations effectively. Additionally, this increases the variety of key-value pair patterns and reduces the frequency of specific pairs. This further reduces the likelihood of 90B predictors selecting the correct pairs, thus the overestimation occurs.

To verify this, we generate a composite source by combining a high-frequency sine wave f_1 and a simulated Wiener process f_2, as described by the Equations (7). These signals are combined by addition before sampling and quantization to replicate common physical signal combinations. Such signals are common in the physical world. For example, in communication systems, the received signal is a combination of the transmitted signal and noise. This mixed signal serves as the composite source. We obtain sequences of one million samples, and each sample width is quantized to 8 bits. We evaluated these sequences using both the 90B predictors and DNN-based predictors. As the digitization process makes precise calculation of true entropy is infeasible, so lower estimation results are considered to be more accurate. The results are presented in Table II.

$$\begin{aligned} f_1(x) &= 0.5\sin(2\pi\cdot 10^5\cdot x)+0.5,\ \ x\in[0,1],\\ f_2(w(t)) &= w(t)-\lfloor w(t)\rfloor,\ \ t\in\{0,1,2,\ldots,10^6\},\\ w(t+1)-w(t) &\sim N(1,0.05),\ \ w(0)=0. \end{aligned} \tag{7}$$

It is observed that all DNN-based predictors surpass the 90B predictors in performance. This experiment shows that 90B predictors tend to overestimate entropy when using physical signals with periodic components as entropy sources. In comparison, recurrent DNN structures are more capable of handling such composite sources. However, this also presents a challenge. The black-box nature of neural networks prevents us from interpreting what information they extract from the relationships between time steps. Therefore, it is difficult to understand how they improve entropy estimation. We will investigate this question in the following section.

IV. INTERPRETING DEEP NEURAL NETWORKS IN MIN-ENTROPY ESTIMATION TASKS

Current research often uses DNN models for entropy estimation without clarifying what DNNs have actually learned. This section explores three key questions: *What is the relationship between the model optimization objective and min-entropy? What do neural networks learn, and which structures are best for entropy estimation?* We use the parameter settings from Table I and the existing training method of DNN predictors. Based on our analysis, we propose a variety of DNN architectures designed for different types of entropy sources.

A. Model Optimization Objective and Min-Entropy

According to Kelsey et al. [6], min-entropy can be estimated from the prediction accuracy of a predictor, denoted as p_{global}. While this approach is widely used in practice [7], [8], [9], [10], [12], [13], the theoretical connection between the model optimization objective and the resulting min-entropy estimate remains unexplored. In this section, we analyze how the optimization objective of DNNs shapes prediction accuracy, and how this relates to the underlying data's true min-entropy.

1) Model Optimization Objective: Existing DNN-based entropy estimation methods adopt a supervised learning framework and formulate entropy estimation as a classification task [7], [8], [9], [10], [12], [13]. Let x denote the history outputs, and y is the corresponding output symbol from the entropy source. The training pairs (x, y) are assumed to be independently and identically drawn from an unknown joint distribution $p_{\text{data}}(x, y)$ of the entropy source. A neural network with parameters θ models the conditional distribution $p_{\text{model}}(y \mid x;\theta)$, aiming to approximate the true conditional distribution $p_{\text{data}}(y \mid x)$, and is optimized via the cross-entropy loss [14]:

$$J(\theta) = -\mathbb{E}_{(x,y)\sim p_{\text{data}}}\left[\log p_{\text{model}}(y \mid x;\theta)\right].$$

This objective corresponds to the cross-entropy between the true and learned conditional distributions, which can be decomposed as $H(p_{\text{data}}) + D_{\text{KL}}(p_{\text{data}}\|p_{\text{model}})$. Since $H(p_{\text{data}})$ is constant with respect to θ, minimizing the loss is equivalent to minimizing the KL divergence $D_{\text{KL}}(p_{\text{data}}\|p_{\text{model}})$ between the two distributions [14].

2) Connection to Min-Entropy: Let N be the total number of samples used for prediction. For the i-th sample, let y_i denote the true output of the entropy source given input history x_i, following the distribution $p_{\text{data}}(y_i \mid x_i)$. The predicted label is $\hat{y}_i(x_i) = \arg\max_y p_{\text{model}}(y \mid x_i)$. The classification accuracy is defined as: Accuracy $= \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(y_i = \hat{y}_i(x_i))$, where $\mathbb{I}(\cdot)$ is

the indicator function equal to 1 if the prediction is correct and zero otherwise.

We assume an optimal model in which the learned distribution perfectly matches the true conditional distribution, i.e., $p_{\text{model}}(y_i \mid x_i) = p_{\text{data}}(y_i \mid x_i)$. Under this assumption, the best expected classification accuracy is:

$$\text{Accuracy}_{Best} = \frac{1}{N}\sum_{i=1}^{N} \max_{y} p_{\text{data}}(y \mid x_i). \tag{8}$$

When x_i and y are independent, which corresponds to an independently and identically distributed (i.i.d.) entropy source, the classification accuracy is related to the min-entropy H_∞ of the source as follows:

$$H_\infty = -\log_2(\text{Accuracy}_{Best}) \leq -\log_2(\text{Accuracy}). \tag{9}$$

Therefore, under the i.i.d. assumption of the entropy source, the model's prediction accuracy is an upper bound on the source's min-entropy. This upper bound becomes tight when the model ideally approximates the true distribution.

We now consider the non-independent case. For a stochastic process $\{Z_t\}_{t=1}^{\infty}$, where each Z_t takes values from a finite alphabet $\mathcal{Z}$, the min-entropy rate is defined as [15]:

$$H_\infty := \lim_{n\to\infty} \frac{1}{n} \min_{z^n} \left[-\log_2 P(z^n)\right].$$

Let $z^n = (z_1, \ldots, z_n)$, and let P denote the true distribution p_{data}. When the entropy source outputs a sequence of fixed length N, the finite-sample min-entropy rate, denoted as H_∞^N, relates to the asymptotic definition as follows:

$$H_\infty \leq -\frac{1}{N}\left(\min_{z^N} \log_2 P(z^N)\right) = H_\infty^N. \tag{10}$$

To further simplify the modeling, we introduce a k-th order Markov assumption, where $N+k \gg k$. Under this assumption, the joint probability can be factorized as $P(z^N) = \prod_{i=k+1}^{N+k} P(z_i \mid z_{i-k}^{i-1}) \cdot P(z^k)$. Then, given a fixed sequence and assuming the first k states are known, the empirical min-entropy rate from position $k+1$ onward can be estimated as:

$$H_\infty^N = -\frac{1}{N} \min_{z^N} \sum_{i=k+1}^{N+k} \log_2 P(z_i \mid z_{i-k}^{i-1}). \tag{11}$$

According to Jensen's inequality, the following relationship holds for the min-entropy:

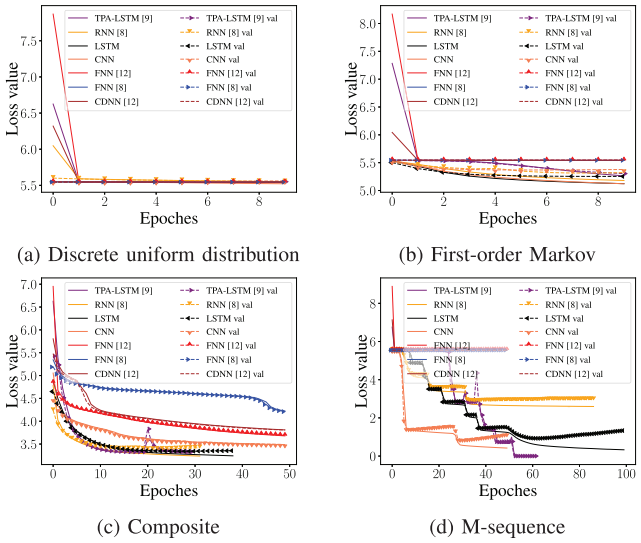
$$\log_2\left(\frac{1}{N}\sum_{i=k+1}^{N+k} P(z_i \mid z_{i-k}^{i-1})\right) \geq \frac{1}{N}\sum_{i=k+1}^{N+k} \log_2 P(z_i \mid z_{i-k}^{i-1}), \tag{12}$$

and since $P(z_i \mid z_{i-k}^{i-1}) \leq \max_{z_i} P(z_i \mid z_{i-k}^{i-1})$ holds for each term, we have:

$$\frac{1}{N}\sum_{i=k+1}^{N+k} P(z_i \mid z_{i-k}^{i-1}) \leq \frac{1}{N}\sum_{i=k+1}^{N+k} \max_{z_i} P(z_i \mid z_{i-k}^{i-1}). \tag{13}$$

Therefore, based on Equations (11)–(13), when the input sequence z^N provided to the neural network coincides with the one that minimizes the min-entropy, we have:

$$-\log_2(\text{Accuracy}_{Best}) \leq H_\infty^N. \tag{14}$$



(a) Discrete uniform distribution (b) First-order Markov

(c) Composite (d) M-sequence

Fig. 3. Loss curve for different sources.

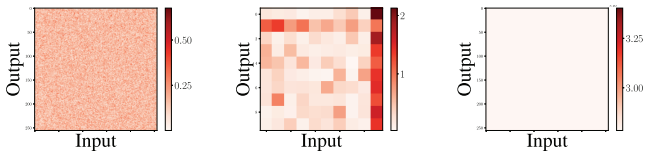
Under the optimal modeling assumption, a model's prediction accuracy provides a lower bound on the sequence's min-entropy. However, this bound no longer holds if the model is imperfect. In such cases, the estimated min-entropy can only be conservatively interpreted as an upper bound, as the true entropy H_∞ may be lower than model performance suggests.

Specifically, when a DNN-based predictor is applied to a time-varying entropy source, distributional shifts can cause its prediction accuracy to deviate from the best. To mitigate this, we propose a new framework in Section V-B that enhances model applicability to time-varying sources.

B. What DNNs Learn in Entropy Estimation

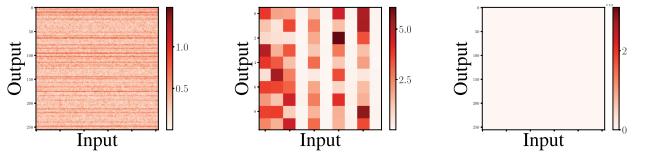
As shown in Fig. 3, we compare the training and validation loss curves of several DNN architectures on three stationary sources: a discrete uniform distribution, a first-order Markov source, and a composite entropy source. To further assess prediction capability, we also include results on a 24th-order M-sequence, a deterministic pseudo-random sequence with theoretical entropy equal to zero, following prior work [9], [13]. In Fig. 3 (a). All models converge quickly for the discrete uniform distribution source. In Fig. 3 (b), LSTM shows the fastest convergence speed and lowest loss for the first-order Markov source. In Fig. 3 (c), the TPA-LSTM achieves the fastest convergence speed for the composite entropy source but experiences a gradient explosion after converging. Finally, in Fig. 3 (d), for the M-sequence, only TPA-LSTM achieves perfect predictions, with the loss dropping to 0. While CNN also converges quickly but does not reach 0.

The findings suggest that TPA-LSTM, RNN, and LSTM excel in analyzing composite sources, first-order Markov sources, and long-correlated sources. Unlike RNN and LSTM, which process data sequentially, the CNN uses 2^k channels, with each channel extracting features from the entire history length. In the previous experiment, extracting features by time steps seemed to yield better results. So, we consider using the history length L as channels to simulate the behavior of RNN



(a) Heatmap of CNN (2^k channels) (b) Heatmap of CNN (L channels) (c) Heatmap of TPA-LSTM

Fig. 4. Heatmap for a first-order markov source.



(a) Heatmap of CNN (2^k channels) (b) Heatmap of CNN (L channels) (c) Heatmap of TPA-LSTM

Fig. 5. Heatmap for an M-sequence.

and LSTM. This method enables the convolutional kernels to extract features at each time step and combine them linearly. Next, we will understand the impact of this modification by explaining what the neural network has learned.

We conduct an analysis using weight visualization. Fig. 4 (a) and (b) show the heatmap of the convolution kernel matrix for CNN with 2^k channels, CNN with L channels, and the attention matrix of the TPA-LSTM when processing first-order Markov sources. The CNN heatmap is generated by taking the absolute value of each convolution kernel and summing them. Fig. 4 (a) indicates that the CNN with 2^k channels doesn't learn any features. While in Fig. 4 (b), the CNN with L channels primarily focuses on the last element of the input data. This demonstrates that it can accurately identify the key feature of the first-order Markov source, where the output mainly depends on the previous element. Additionally, the attention matrix of TPA-LSTM is all zeros, proving that the attention mechanism does not work. As shown in Fig. 5, CNNs can effectively learn complex patterns within the M-sequence, focusing on crucial parts of the input and using these for inference. Furthermore, as Fig. 5 (c) shows, the attention mechanisms in TPA-LSTM do not work, as the prediction function entirely relies on the LSTM layer.

While structures like LSTMs lack intuitive weight features for directly analyzing decision patterns, their learning behavior can still be studied via prediction outcomes. Fig. 5 compares the STL (Seasonal and Trend decomposition using Loess) decomposition of prediction results from TPA-LSTM, LSTM, FNN [12], and the 90B predictors for processing composite entropy sources. STL decomposition [16], a technique widely used in time series analysis, separates data into trend, periodic, and residual components. We present the prediction results after training for 10 epochs. As shown in Fig. 3 (c), these models converged within 10 epochs. As Fig. 5 shows, DNN models capture periodic components more effectively than 90B predictors, with recurrent networks showing stronger capabilities in trend detection. This demonstrates that recurrent architectures are better suited for learning trend and seasonal

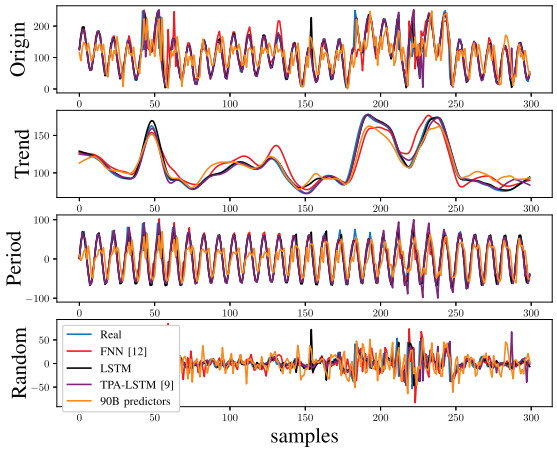
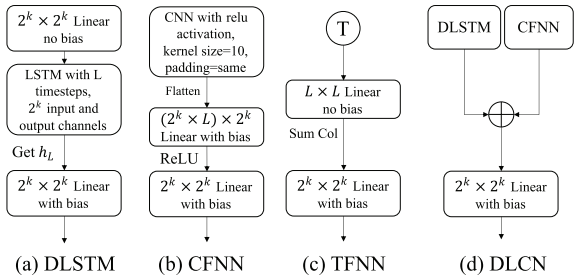


Fig. 6. Results on each component of the decomposed composite entropy source.



(a) DLSTM (b) CFNN (c) TFNN (d) DLCN

Fig. 7. The DNN models we design.

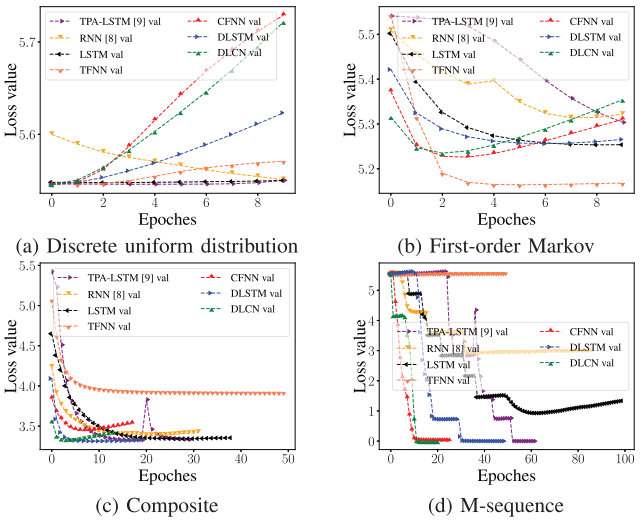
patterns in such data, improving entropy estimation accuracy and addressing the issue raised in the previous section.

C. DNN Models Tailored for Different Entropy Sources

From the previous experiments, we can conclude that the attention mechanism of TPA-LSTM is ineffective and relies on the LSTM component entirely. By explicitly extracting features at each time step like LSTM and RNN, CNNs can also handle these entropy sources, learn their statistical characteristics, and identify correlations in M-sequences.

Based on these findings, we design specialized predictors for different types of entropy sources. For composite sources, we propose **DLSTM**, an LSTM-based model preceded by a learnable $2^k \times 2^k$ weight layer. One-hot encoding maps each input symbol to a sparse 2^k-dimensional vector, forming an $L \times 2^k$ input matrix. A learnable weight layer re-encodes this into a denser representation, optimized during training.

For first-order Markov sources, where statistical features are highly evident and each time step holds no additional information, we suggest using a fully connected layer instead of a CNN to simplify the network. The input is transposed and fed into the network, and instead of flattening, we sum the columns to concentrate information at the final time step. This architecture is referred to as **TFNN**. And for M-sequence data, we retain CNNs with L channels, **CFNN**, to capture local patterns across time steps.



(a) Discrete uniform distribution (b) First-order Markov

(c) Composite (d) M-sequence

Fig. 8. Validation Loss for different sources.

Specifically, the four distinct structures we proposed are shown in Fig. 7. Since the type of entropy source is often unknown in practice, we explore combining models for broader generalization. Finally, by merging DLSTM and CFNN into a single network, **DLCN**, we observe faster convergence speed while maintaining comparable accuracy. It is noted that these networks all use one-hot encoding.

Fig. 8 illustrates the validation loss curves of our four models across different entropy sources. Our models achieve comparable initial performance for the uniform distribution but subsequently experience overfitting. Moreover, our designed networks converge faster on the first-order Markov source, with TFNN showing the lowest and most stable validation loss curve. The DLSTM model exhibits the lowest and most stable validation loss for the composite source, while DLCN quickly converges to a similar point but then shows signs of overfitting. Regarding the M-sequence, CFNN and DLCN converge the fastest, with similar speeds. These results demonstrate the effectiveness of our models.

V. The Proposed Methods and Evaluations

In response to the issues identified with existing work, we introduce a redesigned metric to improve the reliability of estimation results. We continue to leverage neural networks for the predictor design, necessitating a revamped training approach to enhance applicability. By integrating online learning, Bayesian parameter optimization, and change detection, we have developed a novel entropy estimation framework for DNN predictors. This framework enables DNN predictors to quickly adapt to data changes, effectively addressing applicability issues and ensuring robustness for time-varying data.

To evaluate the effectiveness of our proposed methods, we conduct verification experiments on various simulated datasets with known and unknown min-entropies as well as real-world data. We compare our method against all entropy estimators presented in 90B, including statistical estimators, and DNN-based predictors discussed in Section II-C.

A. Redesigning Local Predictability Measure P_{local}

Based on the analysis from Section III, the successive correct predictions used in the local predictability measure are recurrent events. The probability of these events occurring depends on the predictor's accuracy. Rare recurrent events with the highest r-value can cause significant fluctuations, impacting the reliability of the results.

Two primary factors can lead to such a state. First, the entropy source may not function as expected, producing results that are easily predictable by the predictor. This issue falls under the domain of health testing [2] and is beyond the scope of this paper. Second, the entropy source functions as expected. Stable prediction accuracy means that rare recurrent events with high r values do not imply local predictability, but may instead cause significant entropy underestimation. For time-varying sources, accuracy may fluctuate, and regions with high p_{global} are more prone to recurrent events and potential adversarial exploitation. To differentiate these cases, we examine the statistical structure of prediction outputs for signs of instability.

1) Modeling the Prediction Outputs: We model the predictor outputs over the interval $[0, T]$ as segment-wise stable, with a total of $N = f_s \cdot T$ samples, where f_s denotes the sampling frequency. The true segment boundaries t_i are governed by a time-varying switching frequency function $w(t)$, which is not directly observable. Within each true segment i, the predictor achieves a stable accuracy p_i. To assess prediction stability without access to t_i, we employ **blind segmentation**, dividing the binary prediction sequence into $m = N//L$ fixed-length, non-overlapping segments of size L.

Let the j-th blind segment of length L be denoted by $[s_j, s_j+L)$, where s_j is its starting index. Each blind segment may span multiple true segments. Let the length of overlap between blind segment j and true segment i be $l_{i,j} = |[s_j, s_j+L) \cap [f_s t_i, f_s t_{i+1})|$.

We define the weight $w_{i,j} = l_{i,j}/L$, representing the proportion of blind segment j that overlaps with true segment i, which satisfies $\sum_i w_{i,j} = 1$. The mean prediction accuracy of the j-th blind segment is then given by $\tilde{p}_j = \sum_i w_{i,j}\, p_i$.

Now, consider the prediction outcomes within the j-th blind segment: $X_{s_j}, X_{s_j+1}, \ldots, X_{s_j+L-1}$, where each $X_k \sim \text{Bernoulli}(p_i)$, depending on the true segment i to which index k belongs. The sample mean over the j-th blind segment is defined as $\hat{S}_j = \frac{1}{L}\sum_{k=s_j}^{s_j+L-1} X_k$, and the expected value is $\mathbb{E}[\hat{S}_j] = \frac{1}{L}\sum_{k=s_j}^{s_j+L-1} \mathbb{E}[X_k] = \sum_i w_{i,j} p_i = \tilde{p}_j$.

By Lyapunov's Central Limit Theorem (CLT) [11], when L is sufficiently large, the sample mean $\hat{S}_j$ approximately follows a normal distribution $\mathcal{N}(\tilde{p}_j, \sigma_j^2)$. Let $p_{\hat{S}}(x)$ denote the distribution of all blind segment means $\hat{S}_j$. Since each $\hat{S}_j$ is approximately Gaussian with segment-specific parameters $(\tilde{p}_j, \sigma_j^2)$, the overall distribution can be modeled as a Gaussian mixture:

$$p_{\hat{S}}(x) = \frac{1}{m}\sum_{j=1}^{m} \mathcal{N}(x; \tilde{p}_j, \sigma_j^2). \tag{15}$$

2) Detecting Instability via Statistical Tests: When the accuracy is stable, all $\hat{S}_j$ are identically distributed, and their empirical distribution converges to a single Gaussian. In

contrast, if the accuracy is unstable, the means $\hat{S}_j$ are no longer drawn from a single distribution, but instead form a mixture of Gaussians with different means and/or variances. When the differences between component means are substantial, such deviations from normality, particularly multimodality or skewness, can be effectively detected using the Shapiro–Wilk (SW) test [17], which is known for its sensitivity to these distributional characteristics [18].

In extreme cases, segments may share the same mean but differ in variance, forming a Gaussian mixture with identical means but different levels of variability. In finite samples, such mixtures may resemble a normal distribution, causing the SW test to fail to reject normality and potentially leading to a false inference of stability. To address such scenarios, we recommend complementing the SW test with Levene's test [19] to assess variance homogeneity across segments, thereby improving the detection of instability.

To analyze the variance of a mixture sample, we apply the law of total variance:

$$\sigma_j^2 = \frac{1}{L}\left(\underbrace{\sum_i w_{i,j} p_i(1-p_i)}_{\text{within-segment}} + \underbrace{\sum_i w_{i,j}(p_i - \tilde{p}_j)^2}_{\text{between-segment}}\right). \quad (16)$$

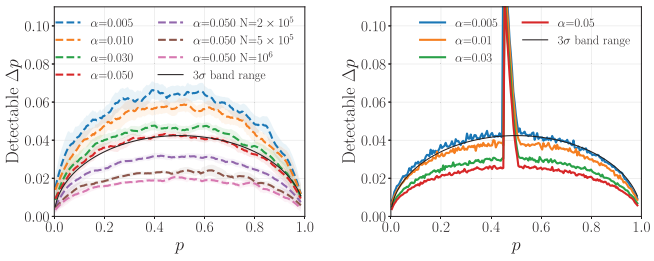
When a segment spans multiple true regions with differing p_i, the second term (between-segment drift) becomes non-zero. The greater the differences between regions, the more dominant this drift term becomes, leading to an inflated overall variance, allowing Levene's test to effectively detect the underlying *instability*. In edge cases where the accuracy switches from p to $1-p$, the overall variance may remain unchanged, yet the distribution may differ significantly in multimodality and skewness. The SW test remains sensitive to such changes.

3) Threshold Selection: The distributions of the SW and Levene's test statistics under alternative hypotheses are analytically intractable, especially when applied to data with complex or mixed distributions [17], [20]. Therefore, we use Monte Carlo simulations to evaluate its performance across various p-variation scenarios.

To ensure the validity of the SW and Levene's tests, we use Monte Carlo simulations to select a segment length L such that the sample mean $\hat{S}_j$ approximates normality via the CLT. This is essential because the SW test evaluates normality and may yield unreliable results when L is too small. Conversely, if L is too large, the number of segments $m = N//L$ may be insufficient for stable testing. Based on empirical results, we choose $L = 5000$ and $N \geq 100000$.

We consider a worst-case scenario where the prediction results undergo a single abrupt change, randomly generating two consecutive segments of different lengths with accuracies p and $p + \Delta p$. With only one distributional shift, statistical evidence for instability is minimal, making detection more difficult. In contrast, multiple shifts typically provide stronger cumulative signals and are easier to identify.

Prediction outcomes are sampled from a Bernoulli distribution with accuracy p. Starting from $p = 0.0039$ (ideal for 8-bit entropy), we increment p by 0.005 and run 1,000 simulations per setting. To ensure statistical validity, the minimum segment length is set equal to the blind segmentation length L; otherwise, both tests may fail to detect the change. A discussion on this constraint and its relevance to adversarial scenarios is provided in Section VI-B.



(a) SW test (b) Levene's test

Fig. 9. Minimum detectable fluctuation in p by two tests.

Fig. 9 (a) illustrates the minimal detectable difference Δp under various total sample sizes $N \in \{10^5, 2\times10^5, 5\times10^5, 10^6\}$ at a significance level of $\alpha = 0.05$, along with the effect of varying α for fixed $N = 10^5$. Smoothed curves with variability bands are presented. As N increases, variability decreases and detection sensitivity improves. Conversely, smaller α values require larger deviations for significance.

According to the CLT, segment-wise means from a stationary process are expected to lie within a normal fluctuation band, typically within $\pm 3\sigma$, similar to the confidence interval used in NIST SP 800-22 [21] for determining the proportion of sequences passing a test. To avoid false detections of instability, it is essential not to misinterpret these natural variations. As shown in the figure, for $N = 10^5$, setting $\alpha = 0.05$ aligns well with this theoretical range. We therefore recommend using $\alpha = 0.05$ for this setting, while for other values of N, the threshold can be empirically determined.

Fig. 9 (b) presents the performance of the Levene's test under the same simulation settings. A sharp discontinuity is observed near $p = 0.5$, which corresponds to the extreme case where the variance remains nearly constant, causing the Levene's test to lose sensitivity. In contrast, the SW test remains robust due to its sensitivity to distributional shape beyond variance alone. For $N = 10^5$, Levene's test performs best with $\alpha \in [0.005, 0.01]$; for conservative detection, we select $\alpha = 0.01$ as the default.

4) The Case of Stable Accuracy: If both statistical tests are passed, we conclude that the predictor's accuracy remains stable throughout the testing process. We can address the underestimation by examining the statistical characteristics of recurring events using hypothesis testing. Specifically, we utilize the statistic N_n, the number of occurrences of recurrent event ε [11], to evaluate the null hypothesis. In our approach, the null hypothesis H_0 asserts that the probability of ε occurs frequently, while the alternative hypothesis H_1 indicates that ε is rare.

We will now detail the use of N_n to construct a hypothesis test. In n experiments, the mean and variance of ε occurrences

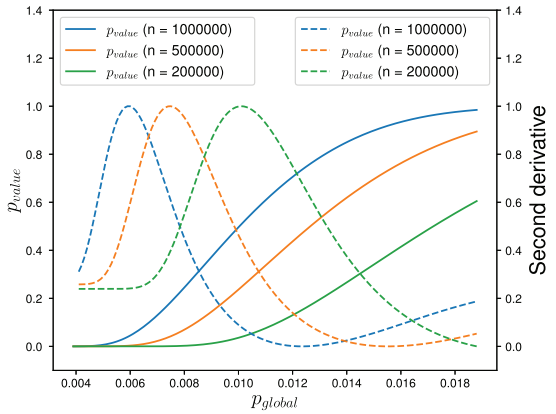


Fig. 10. The relationships of different parameters (dashed line: second derivative).

are given by [11]:

$$\mu = \frac{1-p^r}{qp^r}, \sigma^2 = \frac{1}{(qp^r)^2} - \frac{2r+1}{qp^r} - \frac{p}{q^2},$$

where $q = 1 - p$, and p denotes the success probability of the experiment. As the predictor's accuracy remains stable, we can use the predictor's accuracy p_{global} as the parameter p. For sufficiently large n, the number N_n of consecutive sequences of length r in n experiments follows an asymptotic normal distributions [11], that is $N_n \sim N(n/u, \sigma\sqrt{n/u^3})$.

We can calculate a p_{value} based on N_n. First, standardize N_n to obtain the statistic V:

$$V = \frac{N_n - \frac{n}{u}}{\sigma\sqrt{\frac{n}{u^3}}} \sim N(0,1),$$

then the p_{value} can be calculated using the complementary error function [21]:

$$p_{value} = \frac{1}{2} erfc\left(\frac{V}{\sqrt{2}}\right). \tag{17}$$

The p_{value} represents the strength of evidence for the null hypothesis, and α is the significance level. If the $p_{value} \geq \alpha$, we accept the null hypothesis and consider the recurrent event common. If the $p_{value} < \alpha$, we reject the null hypothesis and consider the event rare. Therefore, we can use α to determine whether the highest r-value event is rare.

To determine the threshold α, we investigate how parameters such as r, N_n, and the number of experiments n, affect the p_{value}. As depicted in Fig. 10, under the conditions $r = 3, N_n = 1$, we illustrate the variation in p_{value} with changes in p_{global} and n. Additionally, we present the second derivative of the p_{value}, scaled to the range of 0 to 1. There is a sudden increase in the p_{value} at a specific point, characterized by a steep rise in the curve's slope. This acceleration point marks the transition of an event from unlikely to probable and we use it as the threshold α. This acceleration point is determined by identifying the maximum value of the second derivative of the p_{value} curve. In Fig. 10, with $n = 1,000,000$, the threshold is 0.04 (round to two decimal places), and with $n = 500,000$, the threshold is 0.05. Therefore, in practical applications, the threshold must be determined based on the specific conditions of the experiment.

5) Influence on the Min-Entropy: Our method affects the estimated min-entropy differently from that of Kelsey et al. [6] (Equation (6)), as follows:

$$H_\infty(\epsilon, r, V) = -\log_2\left(\max\left\{p_{global}, \max_{\mathcal{P}} p_{local}\right\}\right),$$
$$\mathcal{P} = \begin{cases} \mathcal{P}(\epsilon, r, V), & \text{if all tests are passed,} \\ \mathcal{P}(\epsilon, r), & \text{otherwise.} \end{cases} \tag{18}$$

Algorithm 1 The Strategy of Calculating p_{local}

Input: The highest r of recurrent events, r_{max};
The number of experiments, n;
The number of consecutive events, N_n;
The global prediction accuracy, P_{global};
The list of prediction result, P_{list};
The number of groups for variance testing, gs_v;
The number of groups for normality testing, gs_n;
Output:
The local prediction accuracy, P_{local};
1: $r = r_{max}$;
2: By binary search method find the value of P_{local} by solving the Equation (4);
3: **if** not (Shapiro-Wilk($P_{list}//gs_n$) and Levene($P_{list}//gs_v$)) **then**
4: **return** P_{local}
5: **end if**
6: $p = P_{global}$, $q = 1 - p$;
7: **while** $r > 0$ **do**
8: $\mu = \frac{1-p^r}{qp^r}$, $\sigma^2 = \frac{1}{(qp^r)^2} - \frac{2r+1}{qp^r} - \frac{p}{q^2}$;
9: $p_{value} = erfc\left(\left(N_n - \frac{n}{u}\right) / \left(\sigma\sqrt{\frac{n}{u^3}}\right)\right)$;
10: **if** $p_{value} > \alpha$ **then**
11: Recalculate P_{local} based on the Equation (4);
12: **return** P_{local};
13: **end if**
14: $r = r - 1$;
15: **end while**
16: **return** P_{local};

If the prediction sequence passes both statistical tests, we consider the predictor stable and constrain the local probability estimates to the set $\mathcal{P}(\epsilon, r, V)$. This set includes all probability values p that could lead to the occurrence of a recurrent event ϵ of length at most r, subject to the constraint V, which mitigates underestimation. To compute the local probability p_{local}, we adopt the strategy described in Algorithm 1. Otherwise, we use the set $\mathcal{P}(\epsilon, r)$. In this case, the maximum probability p within the selected set is computed according to Equation (4). Section V-C.1 analyzes this method's practical effectiveness.

B. A General Estimation Framework for DNN Predictors

As we analyzed previously, the training method impacts the applicability of predictors on time-varying entropy sources. The training approach of 90B works well with unknown data but requires a tuning mechanism to converge with minimal data. In deep learning, a similar training method is known as online or continuous learning. However, there are two main

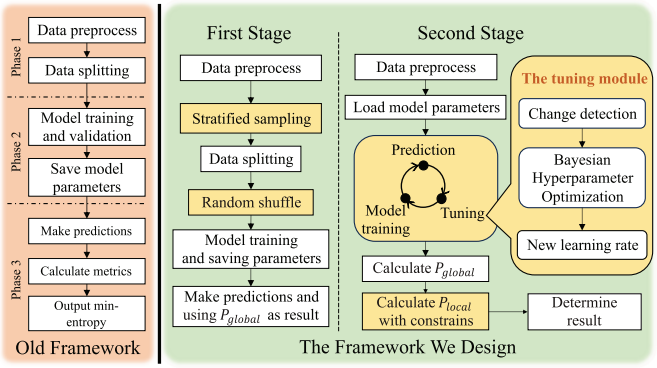


Fig. 11. Comparison between different frameworks, major improvements are highlighted in yellow.

issues with online learning: initial inaccuracy and the lack of an adjustment mechanism, which are similar to 90B.

To address these issues, we propose an online adaptive min-entropy estimation framework for DNN predictors. This framework employs online learning [22] and includes a tuning module with change detection techniques and Bayesian parameter optimization. It dynamically adjusts the model's learning rate based on input data, allowing for real-time adaptation to data variations and enhancing the performance in analyzing time-varying data. We improve our model's early predictions by integrating pre-trained model parameters. Fig. 11 shows a comparison between the three-phase framework and the new framework we designed. The new framework consists of the following two stages.

1) First Stage: This stage initializes the model parameters for the second phase, enhancing the model's initial predictive capabilities. During this stage, the DNN model is trained using one million sample outputs from the entropy source and continues to employ the existing DNN predictors' training method. However, the challenges of data partitioning remain, necessitating alternative methods to mitigate this issue. To address this problem, we employ stratified sampling before data splitting, followed by random shuffling. Specifically, we partition training and test sets by labels through stratified sampling. The resulting training and test sets are then randomly shuffled to mix the samples evenly, obtaining a training set and test set with identical distributions. Random shuffle disrupts the original data structure, so the p_{local} metric no longer applies. After this, the neural network can learn the entropy source data globally without issues from data partitioning. This stage will output a min-entropy as a reference, but not as the final result of entropy estimation.

2) Second Stage: The second stage estimates entropy. In this stage, we implement an online adaptive learning approach, complemented by online learning and a tuning module. The stage can facilitate a process where the model first predicts, tunes as necessary, and then trains. Online learning processes the data in its original sequence, allowing the use of the p_{local} metric for min-entropy calculation. The tuning module is designed to prevent overfitting and address sudden changes in data. Online learning enables real-time model adjustments for slowly varying data but can lead to overfitting on localized data, resulting in inaccurate estimations for new data. Abrupt data shifts can render prior knowledge obsolete, which degrades performance and causes overestimation errors. The tuning module mitigates these issues by continuously monitoring performance and adjusting the model when necessary.

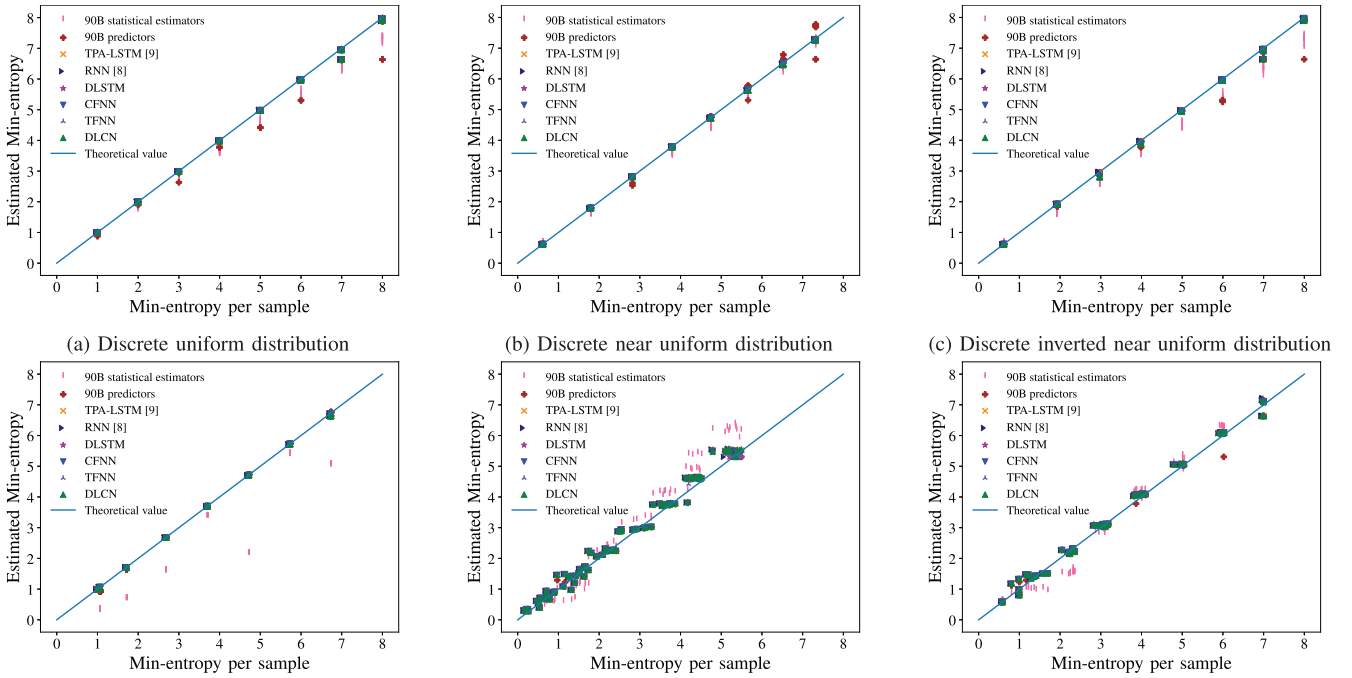
As detailed in Fig. 11, the workflow of the tuning module begins with a change detection mechanism that monitors model performance, identifying and responding to data variations. We have chosen the DDM algorithm [23] to monitor model performance, given its wide usage in the deep learning field [24]. When the model's performance declines, the module adjusts its parameters to accommodate new data. There are two ways to adjust model parameters: retrain the model or modify the learning rate to accommodate new data over time. Our experiments show that simple neural network architectures can quickly converge to near-actual entropy values through learning rate adjustments. Thus, we choose the latter adjustment strategy, employing a widely used Bayesian optimization algorithm [25] in deep learning to search for the optimal learning rate from $[10^{-5}, 10^{-1}]$. When performance declines, we use 60 batches of data for searching (the reasons are shown in Section V-C.2).

C. Experiments on Simulated Sources

1) Experiments on Stationary Sources: We first evaluate the performance of our methods on stationary sources used by previous studies [4] and [6]. We employ datasets adhering to five prevalent statistical distributions commonly referenced in the literature, namely: Discrete Uniform Distribution, Discrete Near Uniform Distribution, Discrete Inverted Near Uniform Distribution, Normal Distribution Rounded to Integers, and First-order Markov Model. We employ the Dirichlet and uniform distributions to randomly generate transition probability matrices for the First-order Markov Model. Specifically, 80 simulated sources are generated for each distribution, with 2 million data samples from each source. The theoretical entropy of each source can be derived from the respective known distributions. For the statistical estimators and predictors defined in 90B, we adopt the minimum estimated value across all methods as the final result.

Our proposed framework uses one million samples for the first stage and another million for the second. Accordingly, only the latter half is used when applying the 90B test suite. Fig. 12 shows the results on simulated sources. The statistical estimators in 90B tend to underestimate entropy on i.i.d. sources but overestimate it on Markov sources, especially with high symbol widths.

In contrast, both DNN-based predictors and the 90B predictors produce more accurate results. In Fig. 12 (a), (b) and (c), the 90B predictors provide several significant underestimated results for higher theoretical entropy due to the effect of p_{local}. Our local predictability measure mitigates this by preventing severe underestimation from low-probability ε and permitting high-probability ε in p_{local} calculations. Consequently, DNN-based predictors using our local predictability measure only exhibit a minor underestimation, for example in Fig. 12 (a), (c) and (f). Excluding these underestimations, the accuracy of DNN-based predictors aligns closely with the 90B predictors.



(a) Discrete uniform distribution

(b) Discrete near uniform distribution

(c) Discrete inverted near uniform distribution

(d) Normal distribution rounded to integers

(e) First-order Markov model with transition matrix generated using Dirichlet distribution

(f) First-order Markov model with transition matrix generated using uniform distribution

Fig. 12. Comparisons on different simulated stationary sources.

The error generated by our method is dependent on the number of experiments n and the predictor's accuracy p_{global}. In this experiment, we use 1,000,000 samples, resulting in the maximum observed error of 6.53%, as shown in Fig. 12 (f). When $n = 200,000$ and a recurrent event with $r = 3$ occurs, like the test sequence in Section III-B, which is based on a uniform distribution and employs the 90B method, this could result in an error as high as 24%. However, our approach will not incur any error, as the p_{value} is below the threshold. When $p_{\text{global}} = 0.0101$, the corresponding p_{value} equals the threshold. Our methods would lead to a potential maximum fluctuation of 8.64% in results. Above this threshold, the probability of recurrent events increases rapidly, making it easier for attackers to correctly predict the key consecutively. Thus, p_{local} is required to reflect this phenomenon.

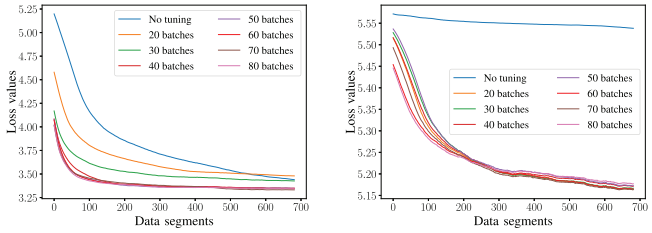
2) Experiments on Time-Varying Sources: To validate the effectiveness of our proposed framework on time-varying data, we conduct comparative experiments with the framework in [9]. We use the same type of time-varying sequences in Section III-C. In this experiment, two sequences of 1,000,000 8-bit samples are used, with a sudden change occurring at the midpoint. While the traditional framework adopts a 1:1 data partitioning method, our framework uses the first half of these sequences for the first stage and the latter half for the second stage. This ensures that both frameworks estimate entropy without prior exposure to the test data.

The relative error between the theoretical entropy and estimation results by different DNN-based predictors is shown in Table III. The "Down" sequence features a drop in entropy from 6.91 to 5.45 bits per sample at the midpoint, while "Up" represents a rise from 5.42 to 7.01. Relative error is computed with respect to the latter half. Table III illustrates that our proposed estimation framework enhances the applicability of predictors to new data. The TFNN, being better suited to this type of entropy source, yields the lowest relative error rate.

TABLE III

COMPARISON OF TRADITIONAL FRAMEWORK AND OUR PROPOSED FRAMEWORK

	Relative error			
	Old framework		Our framework	
	Up	Down	Up	Down
TPA-LSTM [9]	0.124	0.450	0.099	0.218
RNN [8]	0.126	0.457	0.062	0.116
DLSTM	0.128	0.456	0.064	0.083
CFNN	0.129	0.455	0.028	0.047
TFNN	0.133	0.459	**0.025**	**0.042**
DLCN	0.124	0.443	0.033	0.060



(a) Loss of DLCN on the composite entropy source

(b) Loss of TFNN on the first-order Markov source

Fig. 13. The loss curve for different batches used in the tuning module.

Next, we demonstrate the effectiveness of the tuning module. We randomly initialize a model to simulate a performance decline, and use our framework's second stage to process the data. Fig. 13 illustrates the loss decrease on the composite source and the Markov source in the second stage. This figure includes multiple loss curves corresponding to the case of

TABLE IV
COMPARISON OF PREDICTION PERFORMANCE AND ENTROPY ESTIMATION UNDER DIFFERENT SOURCES

Estimator	2-order Markov		Varying f_1		Multiple f_1	
	Up	Down	Up	Down	Up	Down
TPA-LSTM [9]	3.532	2.752	4.693	4.477	4.159	4.195
RNN [8]	3.645	3.117	4.996	4.745	4.205	4.243
DLSTM	3.554	2.773	4.694	**4.452**	**4.157**	**4.186**
CFNN	3.783	2.617	4.999	4.967	4.252	4.333
TFNN	3.717	3.354	6.638	5.307	4.344	4.488
DLCN	3.429	2.484	**4.673**	4.490	4.159	4.204
DLCN-Flatten	**3.358**	**2.361**	4.692	4.481	4.186	4.206
CFNN-Flatten	3.379	2.394	6.613	5.307	4.371	4.543
MultiMCW	3.925	3.748	7.059	6.638	6.182	6.471
Lag	3.966	3.727	**6.638**	**5.306**	**4.534**	**4.940**
MultiMMC	**3.651**	**2.380**	**6.638**	6.638	4.572	4.943
LZ78Y	3.919	3.475	**6.638**	6.638	4.562	4.959
MCV	3.916	3.804	6.983	6.994	6.930	6.957
Collision	3.478	4.000	8.000	8.000	8.000	8.000
Markov	3.945	3.938	7.489	7.489	7.014	7.180
Compression	3.717	3.319	**5.400**	**5.379**	**4.309**	**4.712**
t-Tuple	**3.217**	**2.322**	6.793	6.715	5.668	5.858
LRS Test	3.521	3.035	6.671	6.278	4.714	5.442

not using the tuning module and the case of using different batches of data to search for the learning rate. Using the tuning module allows the model to converge with less data compared to solely employing online learning. Notably, when the number of batches exceeds 60, there is no significant downward trend in the loss curves, and the results are very similar. Since an increase in the number of batches leads to an increase in the search time, we use 60 batches for tuning.

To evaluate model performance on complex entropy sources where theoretical entropy is difficult to compute, we consider three settings: second-order Markov processes, periodically varying composite sources, and multi-periodic sources, and report only empirical results.

For the second-order Markov source, state transitions were designed using Dirichlet and uniform distributions. The "Up" setting denotes a shift from Dirichlet to uniform, while "Down" indicates the reverse. For the composite source with periodically varying frequency, we extended the formulation in Equations (7) by allowing the period of the f_1 function to vary over time. The period of the f_1 was controlled by two sinusoidal functions: $10^5 \cdot (0.5 \cdot \sin(2\pi \cdot 10x) + 1.5)$ and $10^5 \cdot (0.5 \cdot \sin(2\pi \cdot 100x) + 1.5)$. "Up" and "Down" correspond to transitions between these regimes. For sources with multiple periodic components, we used two or four distinct f_1 functions with different periods and normalized amplitudes, where "Up" corresponds to a transition from two to four components, and "Down" the reverse. The results are shown in Table IV.

In each case, 2 million samples were generated with a distributional change at the midpoint. Models were trained on the first half and evaluated on the second. Although both DNN-based predictors and 90B predictors were trained and tested on the full dataset, we report only the average results over the second half for consistency, as DNNs were evaluated exclusively on this segment. Given that distributional changes could introduce bias in the estimation, the 90B statistical estimators were applied solely to the second half of the data.

To better handle second-order Markov processes, we simplified the model architectures. Specifically, we proposed CFNN-Flatten, a variant of CFNN that bypasses the intermediate linear and ReLU layers by feeding the unfolded features directly to the classification layer. A similar modification was made to DLCN, resulting in DLCN-Flatten. This flattened design yielded the best performance among all predictors. Among statistical estimators, the t-Tuple Estimate performed particularly well, and our models achieved comparable results.

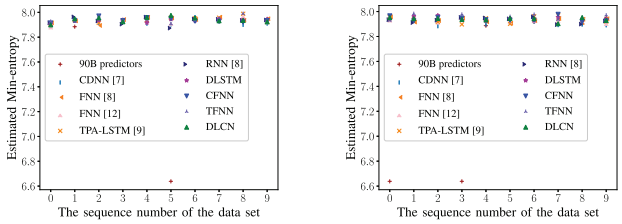
However, in experiments involving the two composite sources, all 90B estimators significantly overestimated entropy compared to the DNN-based predictors, highlighting their limitations in handling non-stationary or time-varying data.

3) Comparative Analysis with Statistical Estimators: 90B defines six statistical estimators: the Most Common Value (MCV) Estimate (applicable only to i.i.d. data), the Collision Estimate, the Compression Estimate, the t-tuple Estimate, the Markov Estimate, and the LRS Estimate. Among these, the LRS Estimate is known to consistently overestimate entropy, as previously reported in the literature [3]. In our evaluation, all observed underestimations on i.i.d. and low-width first-order Markov models can be attributed to the Compression Estimate [2]. This estimator assumes a worst-case distribution regardless of the actual data, resulting in a systematic bias relative to the true min-entropy. As analyzed by Zhu et al. [5] and Kim et al. [4], it provides only a lower bound under worst-case assumptions, rather than reflecting the true min-entropy. In contrast, predictors learn directly from the empirical distribution, enabling them to reflect the statistical characteristics of the data and approximate the true min-entropy more accurately in our experiments.

Besides, the Compression Estimate, Collision Estimate, and Markov Estimate are designed for stationary bitstring-type sources and continue to operate at the bit level even when applied to high-symbol-width data [2]. For example, the Markov Estimate constructs a transition matrix based on individual bits rather than full symbols, making it incompatible with such data and often resulting in overestimation. Furthermore, these estimators assume stationarity and lack mechanisms to adjust to temporal variation, which limits their reliability in real-world non-stationary environments.

The t-tuple Estimate often yields the lowest entropy for high-symbol-width first-order Markov models by counting empirical frequencies of t-length tuples. However, as symbol width increases, the number of possible tuples grows exponentially. Even with datasets containing one million samples, the data may be insufficient for reliable frequency estimation, resulting in inaccurate and often overestimated entropy values. This issue is exacerbated under non-stationary conditions, where distributional shifts render previously collected frequency counts unrepresentative of the current source behavior, further degrading reliability.

In contrast, DNN-based predictors trained using cross-entropy loss do not rely on frequency counting. They learn directly from observed patterns and can be continuously updated to reflect evolving distributions when integrated into our framework. This enables them to adapt to both data sparsity and temporal variation, resulting in greater robustness



(a) RDSeed (b) RDRand

Fig. 14. Comparison on RDSeed and RDRand.

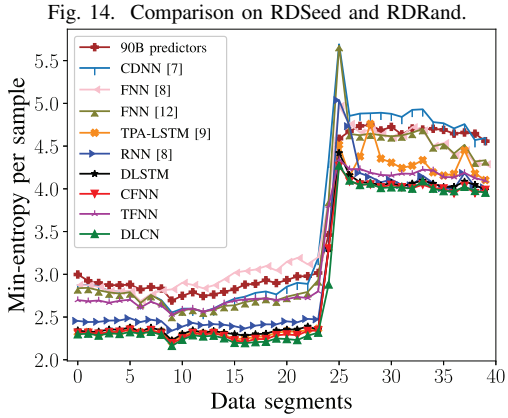


Fig. 15. Comparison on the raw data of CPU time-based jitter with sudden change.

under high-symbol-width and non-stationary conditions. In our experiments, models equipped with our framework consistently outperform 90B estimators under distributional shifts.

D. Experiments on Real-World Sources

We compare our results with all existing predictors as follows. We begin by testing two entropy sources known for their robust randomness. The first is the output from Intel's on-chip hardware RNG, accessed via the RDRand instruction and seeded by an on-chip entropy source [26]. The second is the direct output from the on-chip entropy source, obtained through the RDSeed instruction. We collect ten sequences from each entropy source, each sequence containing two million samples, and the data usage and testing methods are the same as testing stationary sources. As Fig. 14 shows, except for the underestimation caused by p_{local} in 90B predictors, the accuracy of DNN-based predictors is similar to 90B.

We test data known for limited randomness, sourced from CPU instruction execution time jitter, gathered by a CPU jitter-based RNG [27]. This is a typical composite and time-varying entropy source. The randomness originates from the interactions among various components within the CPU. The higher the CPU and IO load, the better the randomness generated from the jitter. We gathered 5,000,000 8-bit samples, initially under low CPU load. Later, we opened a browser to view 4K videos, increasing CPU load and IO operations, and leading to a sudden entropy increase. The first million samples are used for the first stage, and the remaining 4,000,000 samples for the second stage with results output for every 100,000 samples to illustrate the estimation process. Fig. 15 shows the output of DNN-based predictors using our framework and the lowest result of the 90B predictors. The 90B predictors overestimate the sequence and lack adaptability at the mutation point.

TABLE V
COMPARISON OF EXECUTION TIME ON RAW DATA OF CPU TIME-BASED JITTER WITH SUDDEN CHANGES

Model	First Stage	Per-Epoch	Tuning Time (×Count)	Second Stage
CDNN [7]	165.15s	5s	38s (×4)	478.31s
FNN [8]	340.66s	3s	37s (×3)	210.49s
FNN [12]	435.08s	4s	44s (×3)	238.97s
TPA-LSTM [9]	1290.05s	36s	304s (×2)	972.35s
RNN [8]	396.33s	8s	63s (×1)	196.30s
DLSTM	263.52s	8s	56s (×3)	379.58s
CFNN	182.80s	7s	56s (×1)	170.65s
TFNN	96.05s	6s	53s (×3)	276.24s
DLCN	144.05s	8s	80s (×1)	215.64s

Conversely, those DNN-based predictors using our framework can quickly adapt to the new data. As a consequence, DLCN gives the best result.

Table V summarizes the runtime performance of all neural network models across both stages. Although the second stage processes four times more data (4×10^6 samples) than the first stage (10^6 samples), the runtime increases only moderately. CDNN exhibits relatively lower efficiency due to the overhead introduced by its change detection module. This result highlights the efficiency and applicability of our estimation framework. Despite using online learning techniques, our method is not designed for real-time (on-the-fly) entropy estimation. Instead, it performs offline analysis after data collection, similar to 90B, prioritizing accuracy over latency.

VI. DISCUSSION

A. Limitations in Practical Deployment

Although a DNN-based predictor trained with cross-entropy can theoretically provide a lower bound on min-entropy with an optimal model, achieving this in practice depends on factors such as model capacity, training quality, and data availability. In the worst case, a mismatch between the model and the entropy source can degrade prediction accuracy and lead to significant overestimation of min-entropy, particularly under non-stationary conditions with distributional shifts between the training and testing data. As shown in Section III-C, models like CDNN and FNN are inherently unsuitable for first-order Markov models and consistently overestimate entropy. Under a distributional shift, all models trained using traditional methods fail to reliably estimate such sources.

While our proposed framework mitigates the effects of non-stationarity more effectively than the traditional method, optimal convergence still depends heavily on the model's capacity. As demonstrated in our experiments on both simulated and real-world non-stationary sources, even with our approach, persistent overestimation occurs when the model architecture is not well matched to the target entropy source. Therefore, the risk of overestimation remains an inherent challenge in prediction-based entropy estimation methods.

B. Adversarial Entropy Sources: Threats and Mitigations

In practical cryptographic applications, attackers may create entropy sources that appear statistically random while hiding structural flaws, causing estimators to overestimate true

entropy. As noted in NIST SP 800-90B, black-box statistical methods assume the source is honest and at least partially understood [2]. When randomness is deliberately fabricated, such methods fail to detect underlying weaknesses. Consequently, black-box entropy estimation is only effective when (1) the evaluator and adversary have comparable knowledge and control over the source, and (2) attacks cause statistically detectable deviations. Within this framework, we examine two adversarial strategies targeting our estimation method.

1) Structural Mismatch Attack: In this scenario, the adversary is assumed to know the neural network architecture used for entropy estimation. For instance, if the estimator is based on FNN [8], the attacker can craft a composite entropy source that exploits the model's blind spots. In contrast, our DLCN achieves consistent performance across diverse entropy sources and accommodating second-order Markov models through flattening, specifically countering such adaptive attacks. For enhanced security, we recommend using multiple diverse DNNs and selecting the lowest estimate. Moreover, further research is needed to extend the applicability of current models to unknown and more complex entropy sources.

2) Periodic Switching Attack: In this strategy, the adversary alternates between a high-entropy source and a low-entropy source in a fixed cycle, attempting to evade detection by window-based statistical tests. If the adversary aligns the switching period with the segment length used in SW and Levene's test, each window may contain a statistically consistent average, making the variation undetectable. To counter this, we recommend using multiple or adaptive window sizes during testing. When the window is shorter than half the switching period, the scenario matches our Monte Carlo setup, both tests remain effective. While adversaries could increase switching frequency, this reduces samples per sub-period, making patterns statistically indistinguishable and harder to exploit. Under our threat model, such high-frequency switching provides no practical advantage.

VII. Conclusion

In this paper, we first review existing predictors' unreliability and inapplicability issues. Moreover, through interpretability analysis of neural networks in entropy estimation, we develop lightweight predictors suited to different types of entropy sources. We improve reliability by refining the predictability metric and tackle applicability across time-varying sources with a novel DNN training framework. Experimental results confirm effectiveness, and our interpretability findings offer insights for building more robust predictors.

References

[1] J. Bouda, M. Pivoluska, M. Plesch, and C. Wilmott, "Weak randomness seriously limits the security of quantum key distribution," *Phys. Rev. A, Gen. Phys.*, vol. 86, no. 6, Dec. 2012, Art. no. 062308.

[2] M. S. Turan, E. B. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, "Recommendation for the entropy sources used for random bit generation," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. SP 800-90B, 2018, p. 102.

[3] J. Woo, C. Yoo, Y.-S. Kim, Y. Cassuto, and Y. Kim, "Generalized LRS estimator for min-entropy estimation," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 3305–3317, 2023.

[4] Y. Kim, C. Guyot, and Y.-S. Kim, "On the efficient estimation of min-entropy," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3013–3025, 2021.

[5] S. Zhu, Y. Ma, T. Chen, J. Lin, and J. Jing, "Analysis and improvement of entropy estimators in NIST SP 800–90B for non-IID entropy sources," *IACR Trans. Symmetric Cryptol.*, vol. 2017, no. 3, pp. 151–168, Sep. 2017.

[6] 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)*, Saint-Malo, France. Cham, Switzerland: Springer, Sep. 2015, pp. 373–392.

[7] S. Zhu, Y. Ma, X. Li, J. Yang, J. Lin, and J. Jing, "On the analysis and improvement of min-entropy estimation on time-varying data," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1696–1708, 2020.

[8] 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. Int. Conf. Secur. Privacy Commun. Syst.*, 2018, pp. 231–250.

[9] H. Li, J. Zhang, Z. Li, J. Liu, and Y. Wang, "Improvement of min-entropy evaluation based on pruning and quantized deep neural network," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1410–1420, 2023.

[10] 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.

[11] W. Feller, *An Introduction to Probability Theory and its Applications*, vol. 1. Hoboken, NJ, USA: Wiley, 1957, ch. 13.

[12] N. Lv et al., "High-efficiency min-entropy estimation based on neural network for random number generators," *Secur. Commun. Netw.*, vol. 2020, pp. 1–18, Feb. 2020.

[13] 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.

[14] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[15] S. Kamath and S. Verdú, "Estimation of entropy rate and Rényi entropy rate for Markov chains," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 685–689.

[16] R. B. Cleveland, W. S. Cleveland, J. E. McRae, I. Terpenning, "STL: A seasonal-trend decomposition," *J. Off. Stat*, vol. 6, no. 1, pp. 3–73, 1990.

[17] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, nos. 3–4, pp. 591–611, Dec. 1965, doi: 10.1093/biomet/52.3-4.591.

[18] N. M. Razali and Y. B. Wah, "Power comparisons of shapiro-wilk, Kolmogorov–Smirnov, Lilliefors and anderson-darling tests," *J. Stat. Model. Anal.*, vol. 2, no. 1, pp. 21–33, 2011.

[19] M. B. Brown and A. B. Forsythe, "Robust tests for the equality of variances," *J. Amer. Stat. Assoc.*, vol. 69, no. 346, pp. 364–367, Jun. 1974.

[20] J. L. Gastwirth, Y. R. Gel, and W. Miao, "The impact of Levene's test of equality of variances on statistical theory and practice," *Stat. Sci.*, vol. 24, no. 3, pp. 343–360, 2009. [Online]. Available: http://www.jstor.org/stable/25681315

[21] L. E. Bassham III et al., "SP 800–22 Rev. 1a. A statistical test suite for random and pseudorandom number generators for cryptographic applications," Nat. Inst. Standards Technol., Gaither, MD, USA, Tech. Rep. SP 800-22 Rev. 1a, Apr. 2010.

[22] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Netw.*, vol. 113, pp. 54–71, May 2019.

[23] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. Brazilian Symp. Artif. Intell. (SBIA)*, 2004, pp. 286–295.

[24] Q. Xiang, L. Zi, X. Cong, and Y. Wang, "Concept drift adaptation methods under the deep learning framework: A literature review," *Appl. Sci.*, vol. 13, no. 11, p. 6515, May 2023.

[25] S. Watanabe, "Tree-structured Parzen estimator: Understanding its algorithm components and their roles for better empirical performance," 2023, *arXiv:2304.11127*.

[26] J. P. Mechalas. (2018). *Intel Digital Random Number Generator (DRNG) Software Implementation Guide*. [Online]. Available: https://www.intel.com/content/dam/develop/external/us/en/documents/drng-software-implementation-guide-2-1-185467.pdf

[27] S. Müller, "CPU time jitter based non-physical true random number generator," in *Proc. Ottawa Linux Symp.*, 2014, pp. 23–48.