Predicting pseudo-random number generator output with sequential analysis

Dmytro Proskurin[1,†], Maksim Iavich[2,†], Tetiana Okhrimenko[1,*,†], Okoro Chukwukaelonma[1,†] and Tetiana Hryniuk[1,†]

[1] *National Aviation University, 1 Liubomyra Huzara ave., 03058 Kyiv, Ukraine*
[2] *Caucasus University, 1 Paata Saakadze str., 0102 Tbilisi, Georgia*

Abstract

This study delves into the predictive capabilities of neural network models, specifically focusing on Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks as well as the combination of both in a hybrid architecture, for forecasting the outputs of various pseudo-random number generators (PRNGs). The investigation extends across a diverse set of PRNG algorithms, including Linear Congruential Generator (LCG), Mersenne Twister (MT), Xorshift, and Middle Square. Through meticulous analysis, the study evaluates the accuracy of these models in predicting single and continuous outputs generated from the mentioned PRNGs. The research findings illuminate the superior predictive performance of hybrid models, attributed to their adeptness at capturing long-term dependencies, a crucial factor in decoding the complexities of PRNG sequences. Additionally, the impact of model optimization techniques, including dropout and L2 regularization, on enhancing predictive accuracy is thoroughly explored. This comprehensive examination not only underscores the potential of neural networks in identifying deterministic patterns within PRNG outputs but also offers valuable insights into optimal model selection and configuration. The implications of this work are significant, paving new avenues in cryptography and securing random number generation by highlighting the predictability of PRNGs under advanced neural network models.

Keywords

random numbers, RNN, CNN, LSTM, GRU, hybrid model, PRNG

1. Introduction

In the ever-evolving landscape of machine learning, the ability to accurately predict future events based on sequential data stands as a cornerstone of numerous technological advancements and applications. From forecasting stock market trends to decoding human language, the significance of effective sequence prediction cannot be overstated. Central to this domain are Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, which have emerged as powerful tools in the machine learning arsenal for handling sequential data.

RNNs, known for their unique architecture that allows information to persist, have been instrumental in modeling time-dependent data [1]. However, their application is often marred by challenges such as the vanishing gradient problem, which hinders the learning of long-range dependencies [2]. Enter LSTMs, a special kind of RNN designed specifically to overcome these limitations. With their sophisticated internal mechanisms, LSTMs have set new benchmarks in sequence prediction tasks, demonstrating remarkable success where traditional RNNs falter [2, 3].

This paper embarks on a comprehensive exploration of RNNs and LSTMs in the context of sequence prediction. We delve into the architectural intricacies of these models, their strengths and weaknesses, and their performance across various sequence prediction scenarios.

Our study is particularly focused on datasets generated by different Pseudo-Random Number Generators (PRNGs), offering a unique lens through which the capabilities of these models can be examined and understood.

Through rigorous experimentation and analysis, we aim to shed light on the nuances of sequence prediction and provide insights that could guide future applications and research in this fascinating area of machine learning.

2. Background and related work

Recent advancements in sequence prediction have been significantly influenced by the development and refinement of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. These models have shown remarkable proficiency in handling sequential data, particularly in domains where understanding temporal dynamics is crucial.

CSDP-2024: Cyber Security and Data Protection, June 30, 2024, Lviv, Ukraine

*Corresponding author.
† These authors contributed equally.

proskurin.d@stud.nau.edu.ua (D. Proskurin); miavich@cu.edu.ge (M. Iavich); t.okhrimenko@npp.nau.edu.ua (T. Okhrimenko); kaelo@gmail.com (O. Chukwukaelonma); t.hryniuk@ukr.net (T. Hryniuk)

0000-0002-2835-4279 (D. Proskurin); 0000-0002-3109-7971 (M. Iavich); 0000-0001-9036-6556 (T. Okhrimenko); 0000-0002-1247-9854 (O. Chukwukaelonma); 0000-0003-0123-5241 (T. Hryniuk)

1. LSTM for Time Series Prediction: Studies have demonstrated the effectiveness of LSTM models in time series forecasting, a domain traditionally dominated by statistical methods like ARIMA. Unlike these methods, LSTMs can capture complex nonlinear relationships in time series data [2, 4]. Researchers have successfully applied LSTM models to forecast stock prices, energy demand, and weather patterns, achieving higher accuracy than traditional models, especially in scenarios with long-term dependencies and high volatility.

2. RNNs in Natural Language Processing (NLP): RNNs have been pivotal in advancing NLP. Their ability to process sequential text data has led to breakthroughs in machine translation, text generation, and sentiment analysis [1, 2, 5]. The sequential processing capability of RNNs allows them to maintain context in text, a critical factor in understanding human language [5]. However, vanilla RNNs often struggle with long-term dependencies [6], leading to the adoption of LSTMs and GRUs (Gated Recurrent Units) in more complex NLP tasks.

3. Sequence-to-Sequence Learning: The sequence-to-sequence learning framework, often implemented using LSTMs, has revolutionized tasks like machine translation. This approach involves training models on pairs of output and output sequences, enabling the model to learn mappings from one sequence to another. This framework has been crucial in developing models that can translate entire sentences with context, rather than translating on a word-by-word basis [2].

4. Challenges and Limitations: Despite their successes, RNNs and LSTMs are not without challenges. The vanishing gradient problem in RNNs, where the model loses its ability to learn long-range dependencies, has been partially addressed by LSTMs but still poses limitations [4]. Additionally, the training of these models can be computationally intensive, requiring significant resources for large datasets.

5. Future Directions: Ongoing research is exploring more efficient and effective variants of RNNs and LSTMs, such as attention mechanisms and Transformer models [4]. These developments aim to address existing limitations while enhancing the models' ability to process longer sequences and maintain context over extended periods.

3. Model architecture overview

Neural networks are artificial intelligence models that mimic human brain function [2]. A neural network connects processing units, similar to neurons, rather than manipulating zeros and ones like a digital model does [2]. The result depends on how the connections are organized and weighted. Neural networks are algorithms modeled after the human brain that recognize patterns. Sensory data is interpreted using machine perception, which labels or clusters raw information. They recognize numerical patterns in vectors, which must be converted into real-world data like as images, sounds, text, and time series [7]. Artificial Neural Networks (ANNs) are computing systems modeled after biological neural systems, including the human brain [8].

Convolutional Neural Networks (CNNs) are similar to standard artificial neural networks (ANNs) in that they use neurons to improve themselves through learning [8]. CNNs have made remarkable achievements. This neural network is now widely used in deep learning. Convolutional neural networks have revolutionized computer vision, enabling previously unthinkable feats like facial recognition, driverless automobiles, self-service supermarkets, and intelligent medical treatments, CNNs also differ from typical ANNs by focusing on picture pattern recognition. This allows us to encode image-specific properties into the architecture, making the network better suited for image-focused tasks while also lowering the number of parameters needed to set up the model [8, 9].

Hybrid Neural Networks (HNNs), which integrate the strengths of many neural networks, are becoming increasingly popular in computer vision applications including picture captioning and action identification. However, there has been limited research on the effective use of hybrid architectures for time series data, particularly for trend forecasting purposes [10]. HNNs use their internal structure to limit the interactions between process variables to align with physical models. Compared to regular neural networks, coupled models are more accurate, dependable, and generalizable [11].

Recurrent Neural Networks (RNNs) represent a paradigm shift in neural networks, specifically designed to recognize patterns in sequences of data [12]. Unlike traditional feedforward neural networks, RNNs possess a unique feature: the output from the previous step is fed back into the output of the current step. This looping mechanism allows RNNs to maintain an internal state that captures information about the sequence they have processed so far, making them ideal for tasks like speech recognition, language modeling, and time series forecasting [2, 12].

The core architecture of an RNN involves a hidden layer where the activation at a given time step is a function of the output at the same step and the activation of the hidden layer at the previous step [6]. This recurrent nature allows the network to maintain a form of memory [12]. However, RNNs are often challenged by long-term dependencies due to issues like vanishing and exploding gradients during backpropagation [2, 3], where the network becomes unable to learn and retain information from earlier time steps in the sequence [4].

Long Short-Term Memory Networks, a special kind of RNN, were developed to overcome the limitations of traditional RNNs. LSTMs are adept at learning long-term dependencies, thanks to their unique internal structure [3]. Unlike standard RNNs, LSTMs have a complex architecture with a series of gates: the forget gate, output gate, and output gate [3, 4]. These gates regulate the flow of information into and out of the cell, deciding what to keep in memory and what to discard, thereby addressing the vanishing gradient problem [4].

Forget Gate: Determines what information is discarded from the cell state [4, 13].

Output Gate: Updates the cell state with new information from the current output [13].

Output Gate: Determines the next hidden state and output based on the current output and the updated cell state [13].

This architecture allows LSTMs to make more precise decisions about what information to store, modify, and output. As a result, LSTMs have been successfully applied in various complex sequence modeling tasks, including machine translation, speech synthesis, and even generative models for music composition [3, 4, 13].

While both RNNs and LSTMs are designed for sequence processing, the key difference lies in their ability to handle long-term dependencies [4, 14]. Standard RNNs, while simpler and computationally less intensive, struggle with retaining information over longer sequences. LSTMs, with their intricate gating mechanism, excel in scenarios where understanding long-range contextual information is crucial [4].

The choice between RNNs and LSTMs often boils down to the specific requirements of the task at hand, the complexity of the sequences involved, and the computational resources available. LSTMs are generally preferred for more complex tasks with longer sequences [3], while RNNs might suffice for simpler tasks with shorter temporal dependencies [1].

4. Methodology

There are a large number of pseudorandom generators that differ in their characteristics, construction methods, and areas of possible application [15–19]. In our study, we employed datasets generated by four distinct PRNG algorithms, each offering unique challenges and characteristics for sequence prediction using RNN and LSTM models. These datasets serve as a testing ground to evaluate and compare the performance of different neural network architectures in sequence prediction tasks.

4.1. Linear congruential generator dataset

Description: The LCG is one of the oldest and simplest PRNG algorithms [20]. It generates random numbers using a linear equation [20]. The simplicity of its algorithm makes it a good baseline for evaluating the predictive capabilities of RNN and LSTM models.

Characteristics: The sequence generated by an LCG can exhibit patterns due to its linear nature. These patterns, while not immediately apparent, can be learned over time, making it an interesting case for sequence prediction models [20]. Despite their potential statistical issues, LCGs have the advantage of offering all the auxiliary qualities, such as seekability, numerous streams, and k-dimensional equidistribution [20].

4.2. Mersenne twister dataset

Description: The Mersenne Twister, specifically the MT19937 variant, is known for its long period and high-quality outputs. It's widely used in various applications due to its reliability and speed [21].

Characteristics: MT generates sequences that are far more complex and less predictable than LCG [20]. This complexity provides a challenging scenario for RNNs and LSTMs, testing their ability to model and predict more intricate and seemingly random sequences. In addition to its inability to produce the all-zero state, the Mersenne Twister also finds it difficult to act randomly in its nearly all-zero state [20].

4.3. Xorshift dataset

Description: Xorshift is a class of PRNGs that operates using XOR (exclusive or) and bit-shifting operations [20]. It's known for its simplicity and speed, often used in scenarios where the speed of random number generation is critical [20].

Characteristics: Despite its simplicity, Xorshift can produce high-quality random sequences [20]. The non-linear nature of its operations makes it an interesting case for studying how well neural network models can adapt to and predict outputs from non-linear algorithms [22]. A bitwise xor operation is a type of permutation that involves flipping certain bits in the target. It can be performed again to reverse the effects [20]. The conventional understanding of Xorshift would advise us to concentrate on lengthening the bits' period [20].

4.4. Middle square method dataset

Description: The Middle Square method is an older PRNG technique that generates random numbers by squaring the number and extracting the middle digits of the result [22]. It's less commonly used today due to certain limitations.

Characteristics: This method is prone to quickly converging to repetitive cycles or zeros, especially with certain seed values [23]. The predictability and potential repetition in the sequences makes it a unique dataset to test the models' ability to detect and adapt to less complex and potentially degenerative patterns [22]. The field of computer science began with the invention of the middle square [22]. It is possible to develop a viable version with a sufficiently long period (264 for each stream) thanks to modern 64-bit computing architecture. The fastest RNGs are comparable in processing speed [23]. This generator works well for parallel processing because of its simple stream capability. Because a square is nonlinear, it provides this generator with an edge over linearly-based generators in terms of data quality [22].

5. Dataset preparation

In our study on "Predicting PRNG Output with Sequential Analysis", we meticulously prepared a dataset to analyze the predictability of various Pseudorandom Number Generators (PRNGs), focusing on four widely recognized algorithms: Linear Congruential Generator (LCG) (Fig. 2), MiddleSquare (Fig. 4), Xorshift (Fig. 3), and Mersenne Twister (MT) (Fig. 1). Each of these PRNGs was chosen for its unique approach to generating sequences of pseudorandom numbers, providing a diverse test bed for our predictive models.

5.1. Data generation parameters

The dataset was generated using the following parameters to ensure consistency across all PRNGs:

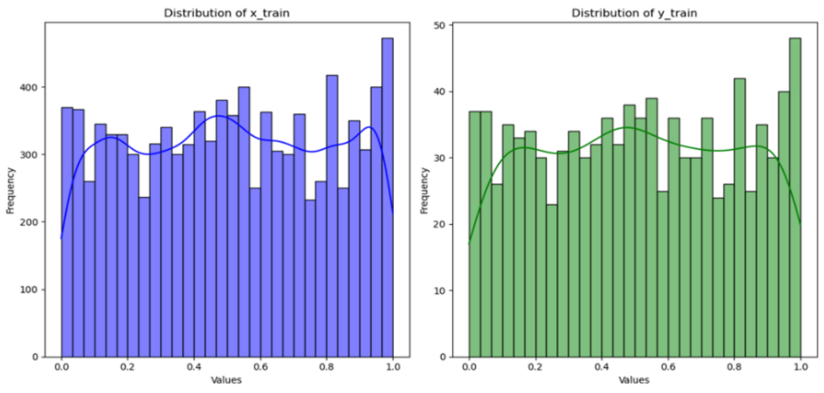

Figure 1: MT dataset distribution

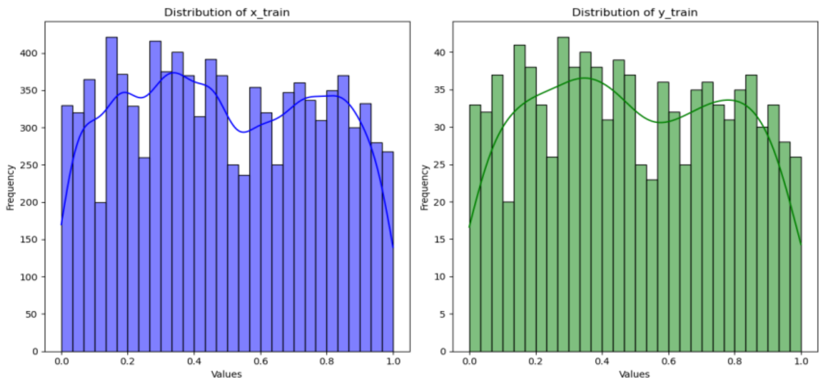

Figure 2: LCG dataset distribution

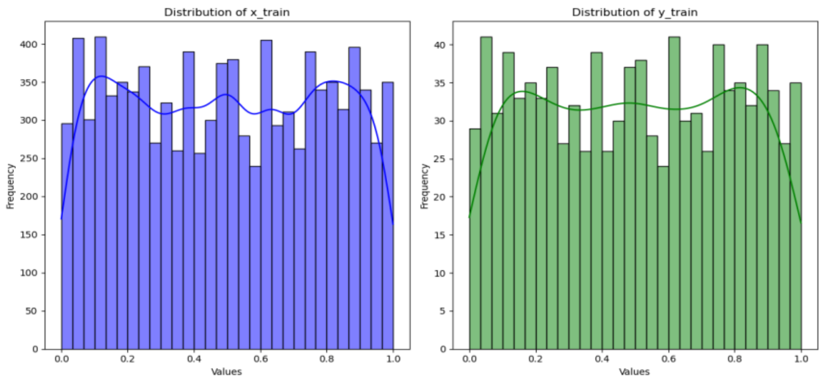

Figure 3: Xorshift dataset distribution

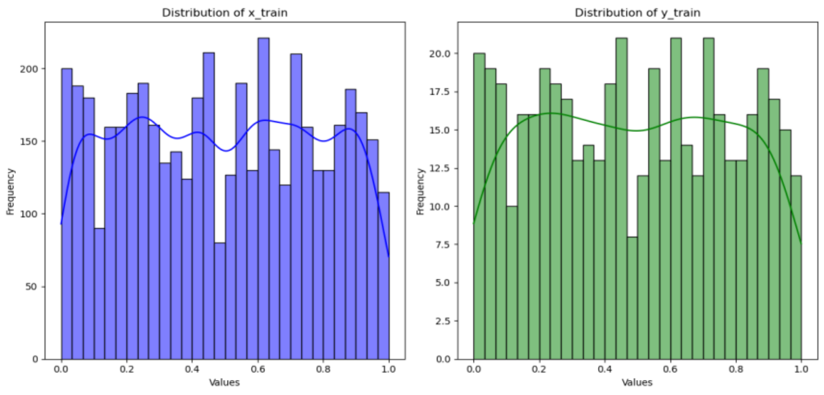

Figure 4: MiddleSquare dataset distribution

Sample Size: Each PRNG was used to generate a sequence of 10,000 numbers, with n = 10000, to create a substantial dataset for training and evaluation. Seed Value: A common seed value of 8956482 was applied to initialize each PRNG, ensuring that the starting point of the pseudorandom sequence was consistent across different generators. Word Size: For PRNGs where applicable, such as MiddleSquare, a word size of 8 bits was selected, balancing the need for computational efficiency with the desire for sequence complexity. Sequence Length: The output was segmented into sequences of length 10, which were then used as individual data points for the subsequent analysis. This sequence length was chosen to provide enough data for recognizing patterns without overwhelming the analytical models.

5.2. Dataset splitting

Once generated, the dataset was divided into three distinct sets to facilitate the training, testing, and validation of our predictive models:

Training Set: Used to train the models, allowing them to learn and adapt to the patterns inherent in the pseudorandom sequences generated by each PRNG.

Testing Set: Employed to assess the performance of the models on unseen data, providing an unbiased evaluation of their predictive capabilities.

Validation Set: Utilized during the model tuning phase to fine-tune parameters and prevent overfitting, ensuring that the models generalize well to new data.

This careful preparation and partitioning of the dataset were critical in establishing a robust foundation for our investigation into the predictability of PRNG outputs through sequential analysis. By standardizing the generation parameters and thoughtfully splitting the data, we aimed to create a fair and consistent testing environment for each of the predictive models applied in our study.

6. Model configuration

In our exploration of "Predicting PRNG Output with Sequential Analysis", we employed a comprehensive approach by leveraging various neural network architectures. Each model was selected based on its ability to process sequential data, a core characteristic of PRNG outputs. Our analysis incorporated Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and a custom Hybrid model, each designed to handle the intricacies of PRNG-generated data in distinct ways.

6.1. Convolutional neural networks

Application: Primarily used for single-value output prediction, CNNs are adept at identifying patterns within a fixed-size window of the sequence. This model excels at capturing local dependencies and spatial hierarchies in data, making it suitable for analyzing individual segments of the PRNG output.

6.2. Recurrent neural networks

Application: RNNs were employed for both single-value and continuous-value output predictions. Unlike CNNs, RNNs have a memory mechanism that allows them to process entire sequences of data, making them ideal for understanding the temporal dynamics and dependencies within PRNG outputs.

6.3. Long short-term memory networks

Application: Like RNNs, LSTMs were utilized for both single-value and continuous-value output predictions. LSTMs are a special kind of RNN capable of learning long-term dependencies. They are particularly effective in avoiding the vanishing gradient problem, enabling them to capture patterns over longer sequences of PRNG outputs.

6.4. Hybrid model

Configuration: The Hybrid model represents an innovative approach, integrating the strengths of CNNs and LSTMs into a singular architecture. It comprises:

- CNN Layer: For extracting local features within the subsequence of the PRNG output.
- LSTM Layer: To capture long-term dependencies and temporal patterns in the data, building upon the features extracted by the CNN layer.
- Dense Layer: Serving as the output layer, it synthesizes the information processed by the CNN and LSTM layers to make predictions.

Application: Designed for versatility, the Hybrid model is equipped to handle both single-value and continuous-value outputs, offering a robust solution for predicting PRNG outputs by leveraging the complementary strengths of convolutional and recurrent layers.

The strategic selection and configuration of these models underpin our analytical methodology. By employing a diverse array of architectures, each with its unique advantages, our study aims to comprehensively evaluate the predictability of PRNG outputs. The Hybrid model underscores our commitment to innovation, integrating multiple neural network paradigms to enhance predictive accuracy and insight into the sequential nature of PRNG-generated data.

7. Evaluation metrics

To rigorously assess the effectiveness of our models in predicting PRNG outputs, we employed a set of comprehensive evaluation metrics. These metrics are crucial for quantifying the accuracy of our predictions and facilitating a direct comparison between the different neural network architectures utilized in our study. Our evaluation framework is centered around the Mean Squared Error (MSE) and a specially devised Model Performance Score.

7.1. Mean squared error

MSE serves as the cornerstone of our evaluation strategy. It calculates the average squared difference between the actual and predicted values, offering a precise measure of the prediction error's magnitude. By squaring the errors, MSE gives more weight to larger errors, making it particularly sensitive to outliers and significant prediction inaccuracies.

In the context of predicting PRNG outputs, MSE provides a clear and direct measure of how closely the model's predictions align with the actual sequence of numbers generated by the PRNGs. A lower MSE indicates higher prediction accuracy, reflecting a model's ability to effectively capture and replicate the underlying patterns of the PRNG sequence.

7.2. Model performance score

Recognizing the need for a standardized metric that allows for an intuitive understanding of model performance, we introduced the Model Performance Score. This metric normalizes the MSE to a scale ranging from 0 to 1, where 0 represents the poorest performance (highest MSE) and 1 denotes perfect prediction accuracy (zero MSE).

The Model Performance Score is calculated by inversely scaling the MSE against a predetermined maximum error threshold. This approach ensures that the performance score is adjusted for the scale of the data and the expected variation in prediction accuracy, allowing for a fair comparison across different models and datasets.

This normalized score simplifies the interpretation of our results, providing a straightforward metric to gauge model effectiveness. It allows stakeholders to quickly assess the relative performance of each model in predicting PRNG outputs without delving into the complexities of raw MSE values.

Together, these evaluation metrics form the foundation of our analytical approach, enabling a nuanced analysis of model performance. MSE offers a detailed view of the prediction accuracy, while the Model Performance Score provides a high-level, comparative perspective. By incorporating both metrics, our study ensures a balanced and comprehensive evaluation of how well each neural network architecture can predict the seemingly unpredictable: output of pseudorandom number generators.

8. Experiment variables and observations

We conducted an extensive series of experiments to evaluate the predictive capabilities of various neural network configurations. These experiments were meticulously designed to explore the impact of different model parameters on the accuracy of PRNG output predictions. Below, we detail the variables involved in these experiments and highlight some critical observations related to model performance.

8.1. Experiment variables

To systematically assess the effects of various hyperparameters on model performance, we tested a wide array of combinations, encompassing:

- Activation Functions: We exper.
- imented with two popular activation functions, ReLU (Rectified Linear Unit) and tanh (Hyperbolic Tangent). These functions were chosen for their distinct characteristics in handling nonlinearities in the data.
- Number of Neurons: The neuron counts tested were 8, 16, and 32. This range allowed us to explore the models' capacity to learn and generalize from the data, balancing complexity with computational efficiency.
- Epochs: All models were trained for [1000] epochs, providing ample opportunity for learning and convergence.
- Model Layers: We varied the depth of the models by testing configurations with [1, 2, 4] layers. This variation aimed to understand how model depth influences learning and prediction accuracy.

Output Lengths: For continuous value prediction, output lengths of [1–4] were tested. This range was selected to assess the models' ability to forecast multiple steps in the PRNG sequence.

8.1.1. Impact of dropout and L2 regularization

One of the most notable findings from our experiments was the impact of dropout and L2 regularization techniques on model learning capabilities. Contrary to common practice in machine learning, where these techniques are employed to enhance model generalization and prevent overfitting, our experiments revealed that:

Models without dropout and L2 regularization demonstrated superior performance in learning and predicting PRNG outputs. The introduction of these regularization techniques led to models that were unable to adequately learn from the training data and, consequently, failed to predict accurately.

This observation suggests a unique aspect of predicting PRNG outputs: the data generated by PRNGs, while seemingly random, follows deterministic algorithms. The addition of regularization techniques, which are designed to introduce randomness and constraint to the learning process, may interfere with the model's ability to capture the underlying deterministic patterns of PRNG sequences.

The results of these experiments provide valuable insights into the design and optimization of neural network models for predicting PRNG outputs. Specifically, they underscore the importance of tailoring model configurations to the specific characteristics of the data and the task at hand. In the context of PRNG prediction, minimizing external sources of randomness and constraint (e.g., through dropout and L2 regularization) appears to be crucial for enabling models to learn and replicate the deterministic patterns that govern PRNG behavior.

9. Experiment results analysis

Our exhaustive investigation into predicting PRNG output through sequential analysis yielded compelling findings, elucidated through the analysis of the top-performing models for each PRNG. Here, we detail the significant outcomes for both single-output and continuous-output scenarios across different PRNGs: Xorshift, MT (Mersenne

Twister), LCG (Linear Congruential Generator), and MiddleSquare.

9.1. Single-output scenario analysis

For single-output predictions, our experiments have the following results.

Xorshift: The RNN model with 32 neurons, 5 layers, and the ReLU activation function emerged as the top performer, achieving a mean score of 0.9898 (Table 1). However, both Hybrid and CNN models came close to the same success rate suggesting that the Xorshift sequence characteristics are not particularly difficult to capture.

Table 1
Xorshift, single output results

Scenario	Model type	Neuron	Activation function	Epochs	Layers	Mean score
Xorshift	RNN	32	relu	1000	5	0.989848
Xorshift	CNN	32	relu	1000	3	0.983555
Xorshift	Hybrid	32	relu	1000	2	0.982930
Xorshift	CNN	16	relu	1000	2	0.981882
Xorshift	Hybrid	8	relu	1000	2	0.980837
Xorshift	CNN	32	relu	1000	5	0.979213
Xorshift	CNN	8	relu	1000	2	0.978671
Xorshift	CNN	32	relu	1000	2	0.977584
Xorshift	CNN	16	relu	1000	5	0.977577

50% of all models were able to reach 90% success thresholds (**Error! Reference source not found.**).

Nevertheless, more improvement can get this number higher.

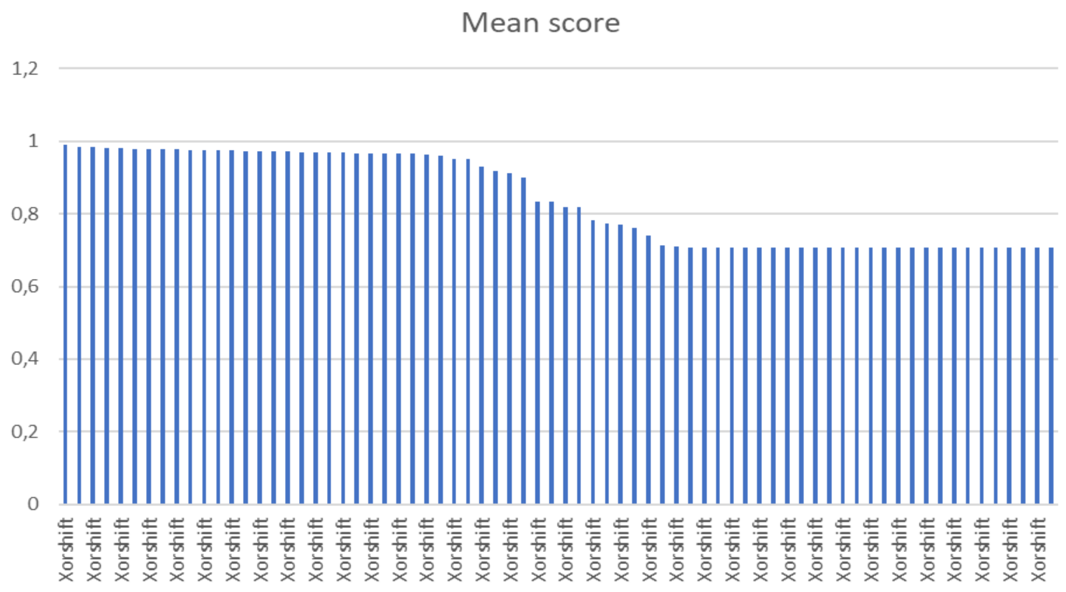

Figure 5: Xorshift, single output, all models results

MT: The CNN model with 8 neurons, 3 layers, and ReLU activation function led the pack with a mean score of 0.9832 (Table 2), indicating that CNN's feature extraction capabilities are effective at decoding the MT's output patterns.

28% of all models were able to reach 90% success thresholds (Fig. 6).

Table 2
MT, single output results

Scenario	Model type	Neuron	Activation function	Epochs	Layers	Mean score
MT	CNN	8	relu	1000	3	0.983227
MT	CNN	32	relu	1000	3	0.980932
MT	RNN	32	tanh	1000	5	0.978619
MT	CNN	32	relu	1000	2	0.978589
MT	CNN	16	relu	1000	2	0.977052
MT	LSTM	32	relu	1000	5	0.976916
MT	RNN	32	relu	1000	5	0.973991
MT	CNN	32	tanh	1000	2	0.973030
MT	CNN	32	relu	1000	5	0.972651
MT	CNN	16	relu	1000	5	0.972521

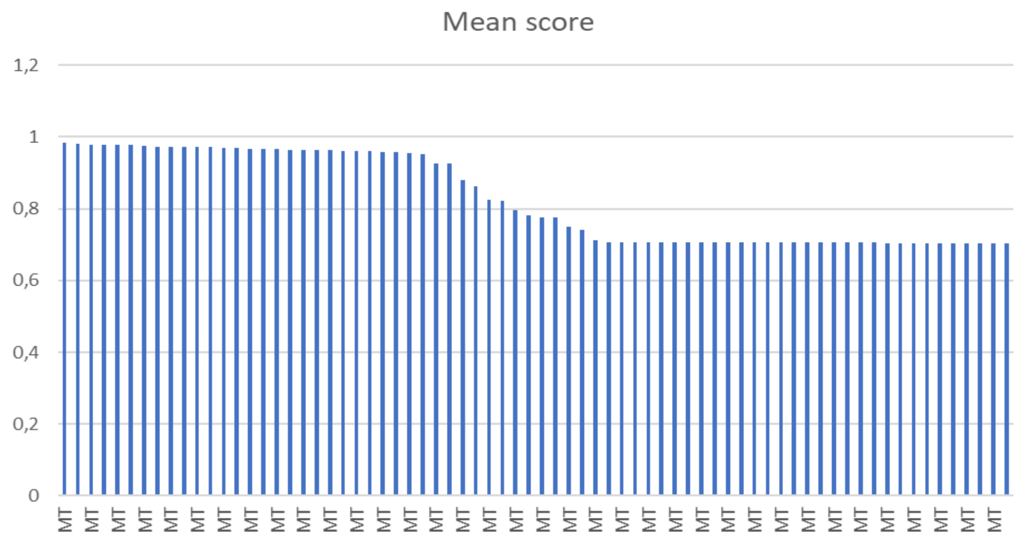

Figure 6: MT, single output, all models results

LCG: The Hybrid model, combining CNN and LSTM architectures with tanh activation, showed superior performance, especially with 32 neurons and 5 layers, reaching a mean score of 0.9831 (Table 3). This underscores the Hybrid model's robustness in capturing both local and long-range dependencies in LCG sequences.

Table 2
LCG, single output results

Scenario	Model type	Neuron	Activation function	Epochs	Layers	Mean score
LCG	Hybrid	32	tanh	1000	5	0.983155
LCG	Hybrid	8	tanh	1000	2	0.982143
LCG	Hybrid	8	tanh	1000	3	0.980809
LCG	Hybrid	32	tanh	1000	2	0.980579
LCG	Hybrid	16	tanh	1000	2	0.979036
LCG	CNN	8	relu	1000	2	0.978362
LCG	Hybrid	16	tanh	1000	3	0.978311
LCG	Hybrid	32	tanh	1000	3	0.977490
LCG	RNN	32	tanh	1000	3	0.976433
LCG	CNN	32	relu	1000	5	0.975615

60% of all models were able to reach 90% success thresholds (Fig. 7).

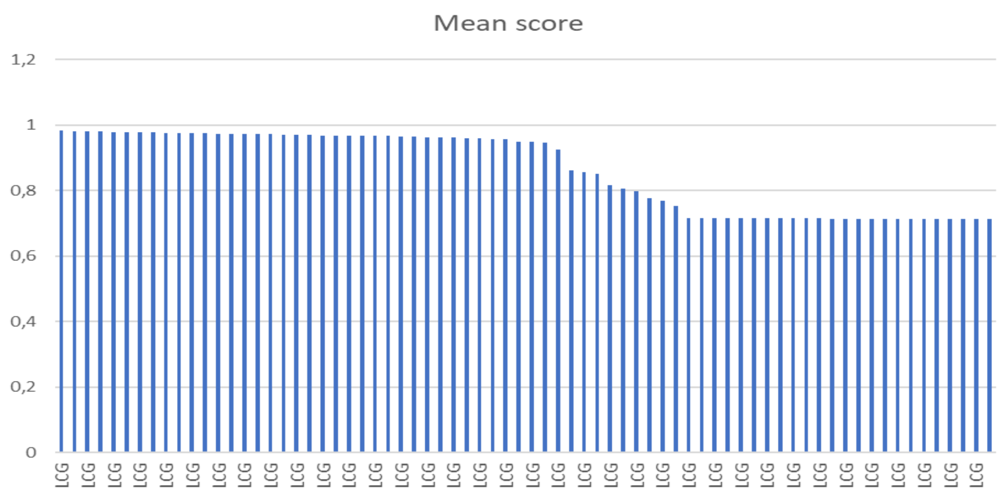

Figure 7: LCG, single output, all models results

MiddleSquare: The Hybrid model with tanh activation, 16 neurons, and 3 layers stood out with a mean score of 0.9883 (Table 4), highlighting the model's effectiveness in navigating the complex, squared calculations intrinsic to MiddleSquare algorithm.

Table 4
MiddleSquare, single output results

Scenario	Model type	Neuron	Activation function	Epochs	Layers	Mean score
MiddleSquare	Hybrid	16	tanh	1000	3	0.988377
MiddleSquare	CNN	32	relu	1000	3	0.987493
MiddleSquare	CNN	32	relu	1000	2	0.986276
MiddleSquare	Hybrid	32	tanh	1000	3	0.983554
MiddleSquare	Hybrid	16	tanh	1000	5	0.983326
MiddleSquare	Hybrid	8	tanh	1000	2	0.982954
MiddleSquare	CNN	32	relu	1000	5	0.980597
MiddleSquare	Hybrid	32	tanh	1000	5	0.980450
MiddleSquare	CNN	8	relu	1000	5	0.980440
MiddleSquare	Hybrid	16	tanh	1000	2	0.980428

64% of all models were able to reach 90% success thresholds (Fig. 8).

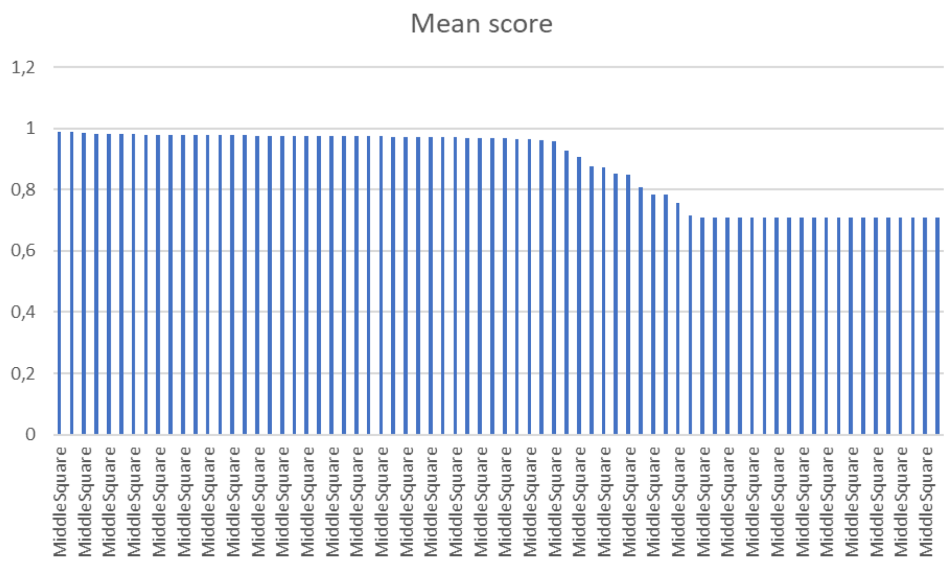

Figure 8: MiddleSquare, single output, all models results

These results underscore the nuanced relationship between PRNG algorithms and neural network architectures, suggesting that no single model architecture is universally superior. Instead, the optimal choice depends on the specific characteristics and mechanisms of the PRNG being predicted. While the models have been fine-tuned to achieve high predictive accuracy, the graphical analysis indicates that there is an inherent limitation to the exactness of these predictions.

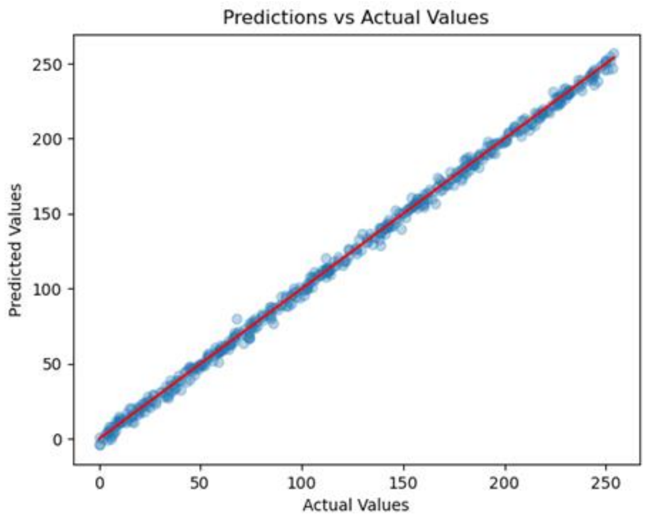

Figure 9: Prediction vs actual values, MiddleSquare with the best result (0.988377)

A further performance plot (**Error! Reference source not found.**) illustrates the correlation between the predicted and actual values of the PRNG sequence. The near-perfect linear alignment along the 45-degree line suggests that the model's predictions are highly correlated with the actual PRNG outputs. The tight clustering of the points around this line demonstrates the model's effectiveness in capturing the underlying pattern of the PRNG sequence. However, the slight deviation of points from the line implies that while the model can predict the general trend and distribution of the PRNG outputs, it cannot replicate the sequence with absolute precision.

The scatter plot showing predictions versus actual values (Fig. 10) for the Hybrid model using tanh activation, 16 neurons, and 3 layers reveals a close correspondence between predicted and actual values. However, the dispersion of points away from the line of perfect agreement (where predicted values equal actual values) suggests that while the model can approximate the PRNG's output with high fidelity, it cannot achieve complete accuracy. The variance from the line of perfect prediction could be attributed to the deterministic yet complex nature of PRNGs, which inherently limits the predictability even with sophisticated models.

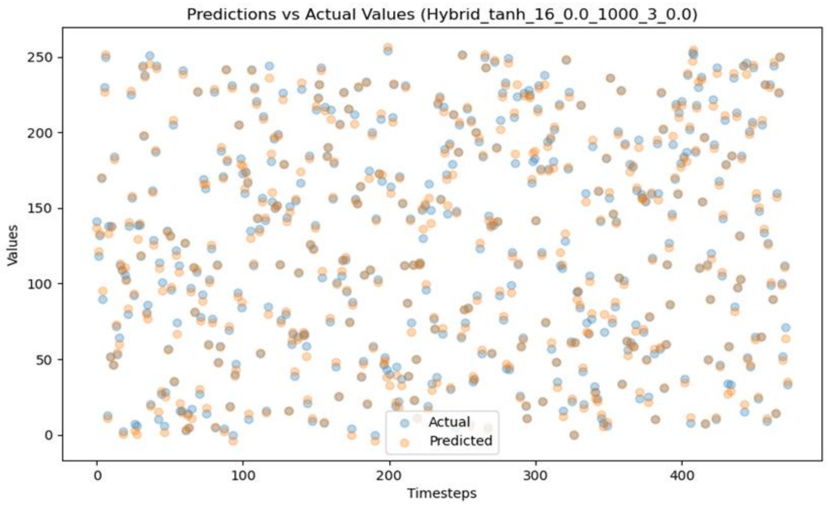

Figure 10: Prediction vs actual values, MiddleSquare with the best result (0.988377)

9.2. Continuous-Output Scenario Analysis

The continuous-output models demonstrated even higher predictive accuracy, with the Hybrid model configured for continuous predictions (Hybrid-C) achieving remarkable success.

For the MiddleSquare PRNG, the Hybrid-C model with tanh activation, 16 neurons, 3 layers, and an output length of 3 achieved a near-perfect mean score of 0.9955. Only 29% of all models were able to break the 90% success milestone (Fig. 11).

Table 5
MiddleSquare, continuous output results

Scenario	Model type	Neuron	Activation function	Epochs	Layers	Output length	Mean score
MiddleSquare	Hybrid-C	16	tanh	1000	3	3	0.995479
MiddleSquare	Hybrid-C	16	tanh	1000	2	2	0.992588
MiddleSquare	Hybrid-C	8	tanh	1000	5	2	0.990003
MiddleSquare	Hybrid-C	8	relu	1000	5	1	0.988989
MiddleSquare	Hybrid-C	32	relu	1000	5	3	0.988566
MiddleSquare	Hybrid-C	8	tanh	1000	3	1	0.988066
MiddleSquare	Hybrid-C	16	relu	1000	2	2	0.986359
MiddleSquare	Hybrid-C	16	tanh	1000	5	3	0.985923
MiddleSquare	Hybrid-C	16	tanh	1000	5	2	0.985797
MiddleSquare	Hybrid-C	8	tanh	1000	5	1	0.984813

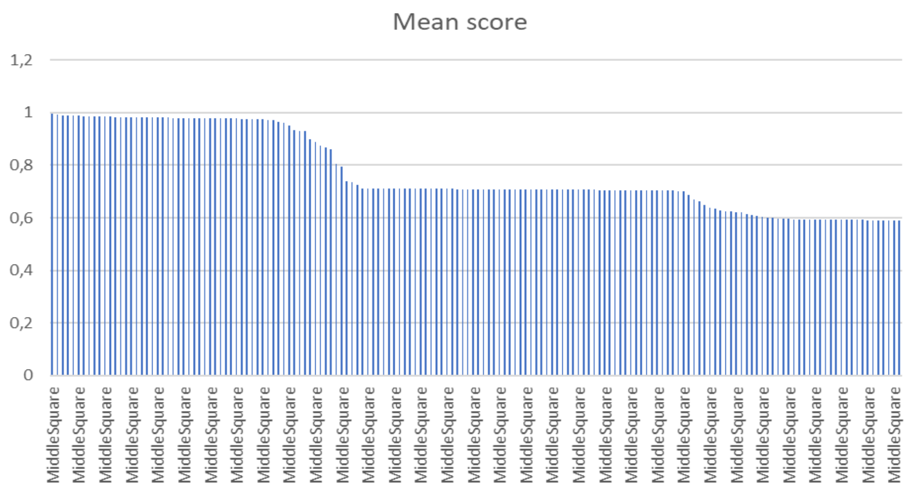

Figure 11: MiddleSquare, continuous output, all models results

For the LCG PRNG, the Hybrid-C model with tanh activation, 8 neurons, 5 layers, and an output length of 2 achieved a near-perfect mean score of 0.992055. 20% of all models were able to break the 90% success threshold (Fig. 12).

Table 3
LCG, continuous output results

Scenario	Model type	Neuron	Activation function	Epochs	Layers	Output length	Mean score
LCG	Hybrid-C	8	tanh	1000	5	2	0.992055
LCG	Hybrid-C	8	tanh	1000	5	3	0.989730
LCG	Hybrid-C	16	tanh	1000	5	5	0.987614
LCG	Hybrid-C	8	tanh	1000	3	5	0.986818
LCG	Hybrid-C	8	tanh	1000	2	5	0.985174
LCG	Hybrid-C	16	tanh	1000	3	2	0.984808
LCG	Hybrid-C	32	tanh	1000	3	2	0.984462
LCG	Hybrid-C	16	tanh	1000	2	1	0.984411
LCG	Hybrid-C	32	tanh	1000	3	1	0.983866
LCG	Hybrid-C	32	tanh	1000	2	1	0.983706

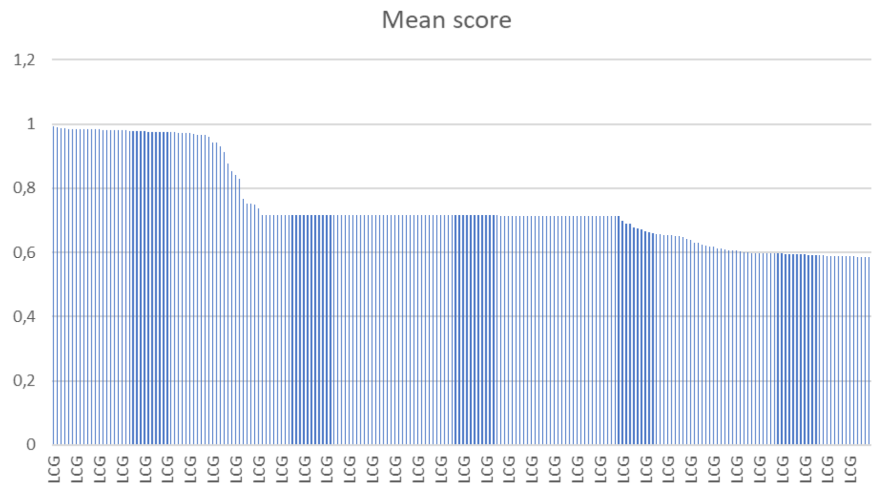

Figure 12: LCG, continuous output, all models results

For the Xorshift PRNG, the Hybrid-C model with relu activation, 16 neurons, 2 layers, and an output length of 2 achieved a near-perfect mean score of 0.987906 (Table 7). 15% of all models were able to break the 90% success threshold (Fig. 13).

Table 4
Xorshift, continuous output results

Scenario	Model type	Neuron	Activation function	Epochs	Layers	Output length	Mean score
Xorshift	Hybrid-C	16	relu	1000	2	2	0.987906
Xorshift	Hybrid-C	16	relu	1000	2	5	0.985753
Xorshift	Hybrid-C	8	relu	1000	5	5	0.985715
Xorshift	Hybrid-C	8	relu	1000	2	2	0.984238
Xorshift	Hybrid-C	32	relu	1000	2	2	0.983437
Xorshift	Hybrid-C	16	relu	1000	2	3	0.981247
Xorshift	Hybrid-C	8	relu	1000	2	1	0.980434
Xorshift	Hybrid-C	32	relu	1000	2	1	0.978930
Xorshift	RNN-C	32	tanh	1000	3	1	0.977685
Xorshift	Hybrid-C	32	relu	1000	2	3	0.976177

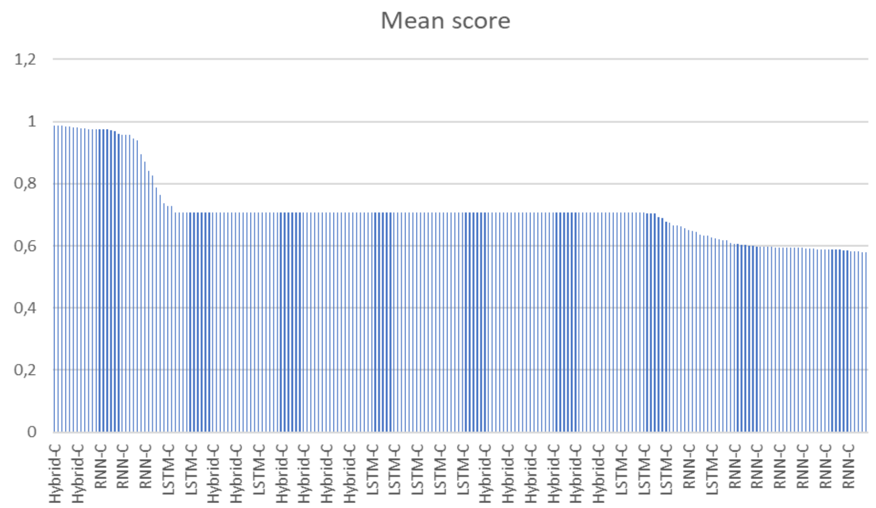

Figure 13: Xorshift, continuous output, all models results

For the MT PRNG, the Hybrid-C model with relu activation, 32 neurons, 2 layers, and an output length of 2 achieved a near-perfect mean score of 0.985006 (Table 8). 12% of all models were able to break the 90% success threshold (Fig. 14).

Table 5
MT, continuous output results

Scenario	Model type	Neuron	Activation function	Epochs	Layers	Output length	Mean score
MT	Hybrid-C	32	relu	1000	2	2	0.985006
MT	RNN-C	32	relu	1000	5	1	0.981523
MT	RNN-C	32	tanh	1000	5	1	0.980135
MT	Hybrid-C	16	tanh	1000	2	3	0.976324
MT	LSTM-C	32	relu	1000	5	1	0.976245
MT	Hybrid-C	8	tanh	1000	2	1	0.975258
MT	RNN-C	32	tanh	1000	3	1	0.974210
MT	RNN-C	32	relu	1000	3	1	0.972664
MT	RNN-C	32	relu	1000	2	1	0.965829
MT	RNN-C	32	tanh	1000	2	1	0.963914

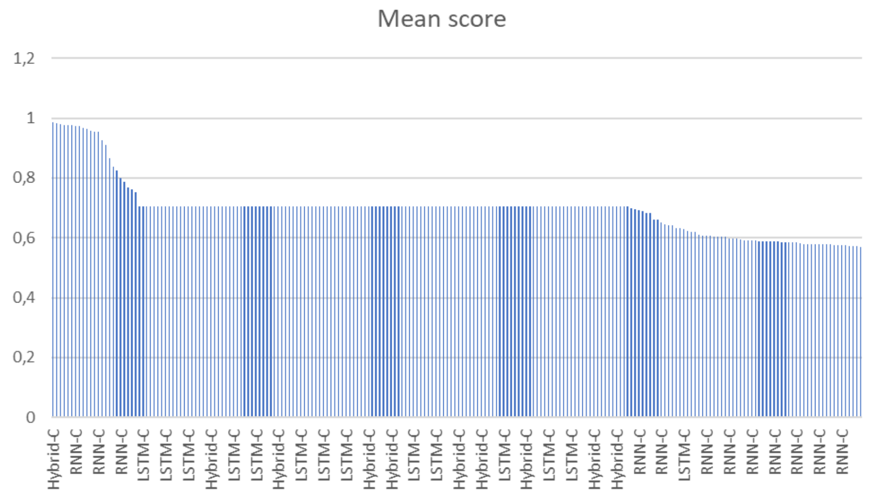

Figure 14: MT, continuous output, all models results

The examination of continuous-output models reveals a notable enhancement in predictive performance compared to single-output models. This is particularly evident in the context of predicting sequences generated by the MiddleSquare PRNG.

The performance plot illustrating the correlation between predicted and actual values (Fig. 15) for the continuous-output model shows an even tighter linear alignment than the single-output model. This near-perfect correlation, along with a high success score of 0.9955, reflects the model's exceptional predictive accuracy. The dense clustering of points along the diagonal suggests that the model can reliably predict the MiddleSquare PRNG's output with high confidence, and such precision is indicative of the model's ability to capture both the immediate and contextual dependencies within the PRNG's sequence.

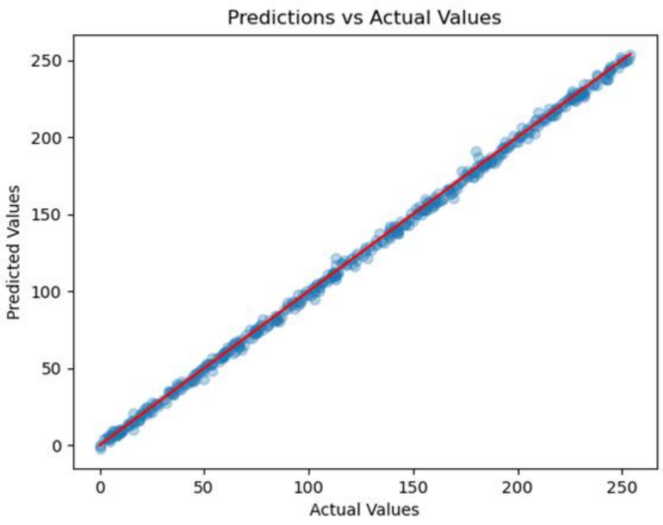

Figure 15: Prediction vs actual values, MiddleSquare with the best result (0.9955)

The scatter plot for the continuous-output Hybrid model, which integrates CNN and LSTM architectures (Hybrid-C), showcases a substantial concentration of points closely aligned with the line of perfect prediction (Fig. 16). The model, employing tanh activation with 16 neurons across 3 layers, exhibits a remarkable ability to track the actual values throughout the sequence. This tight clustering indicates a substantial reduction in prediction errors and a strong alignment with the true PRNG sequence, suggesting a deeper understanding of the underlying patterns by the model.

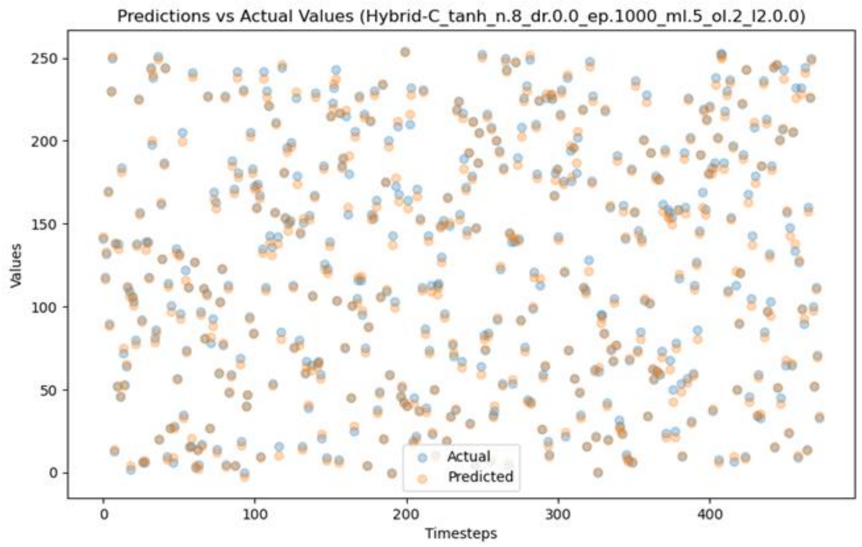

Figure 16: Prediction vs actual values, MiddleSquare with the best result (0.9955)

The continuous-output model's superior performance, as evidenced by the closer proximity of predicted to actual values and the higher success score, highlights the benefit of utilizing sequential context in PRNG output prediction. The ability to forecast the sequence with a success score reaching 0.9955 marks a significant milestone, suggesting that models incorporating sequence history can more effectively decode the deterministic yet complex structure of PRNG outputs.

This analysis implies that continuous-output models hold great promise for applications where forecasting accuracy over sequences is critical. The insights gleaned from this research can inform the development of more secure PRNGs, capable of withstanding sophisticated sequential analysis. Future work will likely explore the expansion of this approach to more complex and higher-dimensional sequences, potentially integrating additional layers of complexity and exploring the impact on model performance.

9.3. Model Performance Across PRNGs

Our study's findings highlight the nuanced nature of PRNG output prediction, with different models excelling for specific generators. This variation underscores the importance of model selection tailored to the characteristics of the PRNG being analyzed. For instance, the best-performing model for the Xorshift generator might leverage its unique XOR and shift operations, whereas the optimal model for the Mersenne Twister (MT) would need to account for its complex bit manipulation and tempering techniques.

Remarkably, the single-output models consistently achieved a 98% success rate across various PRNGs, demonstrating a high level of accuracy in predicting the next output value based solely on a single preceding value. This success rate is indicative of the models' ability to decipher the underlying deterministic patterns that govern PRNG outputs.

Even more impressive, the continuous-output model, which utilizes sequences of values to predict subsequent outputs, reached a 99% success rate. This improvement suggests that incorporating more context in the form of continuous output sequences enables the models to better capture the PRNGs' inherent algorithms, leading to more accurate predictions.

9.4. Implications for PRNG Analysis and Security

The success of our models in predicting PRNG outputs with such high accuracy has profound implications for the fields of cryptography and random number generation. While PRNGs are designed to produce sequences that are difficult to predict, our results suggest that advanced neural network models can uncover and exploit hidden patterns within these sequences. This finding calls for ongoing efforts to enhance the unpredictability and security of PRNGs, ensuring they remain robust against sophisticated analytical techniques.

10. Conclusions

This research delves into the predictability of PRNGs using advanced neural network models. Our study demonstrates that tested architectures possess a remarkable ability to predict the outputs of various PRNGs, with enhanced accuracy observed in continuous output prediction scenarios showcasing a superior performance in capturing long-term dependencies within PRNG sequences, affirming their suitability for complex sequence prediction tasks.

Our findings illuminate the nuanced dynamics of PRNG predictability and the potential vulnerabilities inherent within commonly used generators. By leveraging neural networks, we not only uncover the deterministic patterns masked as randomness but also push the boundaries of understanding in cryptographic security and random number generation.

Future research should explore the integration of more complex neural architectures and the application of these findings in real-world scenarios, such as secure communications and cryptographic key generation. The implications of our work suggest a pivotal shift towards more secure and unpredictable PRNG designs, bolstering the defenses against adversarial predictions and enhancing the integrity of cryptographic systems.

11. Future research directions

These findings have significantly advanced our understanding of the capabilities and limitations of current PRNG technologies when subjected to advanced neural network-based predictive models. The high success rates achieved by these models, particularly the 99% success rate with continuous-output models, not only demonstrate the feasibility of predicting PRNG outputs but also underscore the intricate patterns that deterministic algorithms generate—patterns that sophisticated models can uncover.

This study opens several avenues for future research, aimed at both improving PRNG designs and developing more advanced predictive models:

Advanced PRNG Algorithms: There is a clear need for the development of new PRNG algorithms that incorporate mechanisms specifically designed to counteract the capabilities of neural network-based predictive models. Future research should focus on exploring algorithmic complexities that can more effectively obscure deterministic patterns.

Neural Network Enhancements: Our research has shown that certain neural network architectures are more adept at predicting PRNG outputs than others. Investigating the development of novel neural network models or hybrid architectures that can more efficiently process and predict complex sequences is an exciting frontier. This includes exploring deeper networks, attention mechanisms, and other advanced features that could further improve prediction accuracy.

Cross-Disciplinary Approaches: Combining insights from cryptography, machine learning, and complexity theory could yield innovative approaches to both PRNG design and predictive modeling. Interdisciplinary research might uncover new principles for creating sequences that are inherently more difficult to predict, as well as models that are more adept at understanding complex patterns.

Real-World Application Scenarios: Applying our findings to real-world scenarios, where PRNGs are used under various constraints and for different purposes, will be essential. This includes testing PRNGs in environments with high-security requirements, such as in blockchain technologies, secure communications, and digital signatures.

Ethical Considerations and Security Implications: As research progresses in predicting PRNG outputs, it is imperative to consider the ethical implications and potential security risks associated with disseminating advanced predictive models. Developing guidelines and best practices for responsible research and application in this area is crucial.

Enhancing PRNG security: The ability of neural networks to predict PRNG outputs with such accuracy highlights an urgent need for the cryptographic community to re-evaluate and enhance the design and implementation of PRNGs. Ensuring that PRNGs can withstand analysis by advanced predictive models is crucial for maintaining the security and integrity of cryptographic systems, which rely heavily on the unpredictability of these generators.

Acknowledgment

This work was supported by the Shota Rustaveli National Foundation of Georgia (SRNSFG) [NFR-22-14060] as well as the Ministry of Education and Science of Ukraine (grant №0122U002361 "Intelligent system of secure packet data transmission based on reconnaissance UAV").

References

[1] K. Cho, et al., Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, arXiv:1406.1078 (2014).
[2] I. Sutskever, O. Vinyals, Q. Le, Sequence to Sequence Learning with Neural Networks, Advances in Neural Information Processing Systems 27 (2014).
[3] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation 9(8) (1997) 1735–1780.
[4] F. Gers, J. Schmidhuber, F. Cummins, Learning to Forget: Continual Prediction with LSTM. Neural Computation 12(10) (2000) 2451–2471.
[5] A. Graves, A.-R. Mohamed, G. Hinton, Speech Recognition with Deep Recurrent Neural Networks, IEEE International Conference on Acoustics, Speech and Signal Processing (2013).
[6] A. Karpathy, The Unreasonable Effectiveness of Recurrent Neural Networks (2015).
[7] M. Islam, G. Chen, S. Jin, An Overview of Neural Network, American J. Neural Netw. Appl. 5(1) (2019) 7–11. doi: 10.11648/j.ajnna.20190501.12.
[8] K. O'Shea, R. Nash, An Introduction to Convolutional Neural Networks (2015).
[9] Z. Li, et al., A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects (2021).
[10] T. Lin, T. Guo, K. Aberer, Hybrid Neural Networks for Learning the Trend in Time Series.
[11] D. Psichogios, L. Ungar, A Hybrid Neural Network-First Principles Approach to Process Modeling.
[12] A. Vaswani, et al., Attention Is All You Need. Advances in Neural Information Processing Systems 30 (2017).
[13] J. Brownlee, Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python, Machine Learning Mastery (2018).
[14] V. Desai, R. Patil, D. Rao, Using Layer Recurrent Neural Network to Generate Pseudo Random Number Sequences, Int. J. Comput. Sci. 9 (2012) 324–334.
[15] V. Maksymovych, et al., Hardware Modified Additive Fibonacci Generators Using Prime Numbers, Advances in Computer Science for Engineering and

Education VI, LNDECT 181 (2023). doi: 10.1007/978-3-031-36118-0_44.

[16] V. Maksymovych, O. Harasymchuk, M. Shabatura, Modified Generators of Poisson Pulse Sequences Based on Linear Feedback Shift Registers, Advances in Intelligent Systems and Computing, AISC 1247 (2021) 317–326.

[17] V. Maksymovych, O. Harasymchuk, I. Opirskyy, The Designing and Research of Generators of Poisson Pulse Sequences on Base of Fibonacci Modified Additive Generator, International Conference on Theory and Applications of Fuzzy Systems and Soft Computing, ICCSEEA 2018: Advances in Intelligent Systems and Computing 754 (2019) 43–53.

[18] R. Hamza, A Novel Pseudo Random Sequence Generator for Image-Cryptographic Applications, J. Info. Secur. Appl. 35 (2017) 119–127.

[19] O. Harasymchuk, Generator of Pseudorandom Bit Sequence with Increased Cryptographic Security, Metallurgical and Mining Industry: Sci. Tech. J. 6(5) (2014) 24–28.

[20] M. O'Neill, PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation (2014).

[21] M. Matsumoto, T. Nishimura, Dynamic Creation of Pseudorandom Number Generators (2015).

[22] B. Widynski, Middle-Square Weyl Sequence RNG (2017).

[23] K. Okada, et al., Learned Pseudo-Random Number Generator: WGAN-GP for Generating Statistically Robust Random Numbers, PLoS One 18(6) (2023). doi: 10.1371/journal.pone.0287025.