Using Modular Arithmetic Optimized Neural Networks To Crack Affine Cryptographic Schemes Efficiently

VANJA STOJANOVIĆ, University of Ljubljana, Slovenia

We investigate the cryptanalysis of affine ciphers using a hybrid neural network architecture that combines modular arithmetic-aware and statistical feature-based learning. Inspired by recent advances in interpretable neural networks for modular arithmetic [Gromov 2023] and neural cryptanalysis of classical ciphers [Focardi and Luccio 2018], our approach integrates a modular branch that processes raw ciphertext sequences and a statistical branch that leverages letter frequency features. Experiments on datasets derived from natural English text demonstrate that the hybrid model achieves high key recovery accuracy for short and moderate ciphertexts, outperforming purely statistical approaches for the affine cipher. However, performance degrades for very long ciphertexts, highlighting challenges in model generalization.

ACM Reference Format:
Vanja Stojanović. 2025. Using Modular Arithmetic Optimized Neural Networks To Crack Affine Cryptographic Schemes Efficiently. 1, 1 (June 2025), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

All code and datasets for this work are available online here.

The cryptanalysis of classical ciphers has long served as a proving ground for both cryptographic and machine learning techniques. Advances in the field have demonstrated that artificial neural networks (ANNs) can be trained to automate attacks on classical ciphers by exploiting statistical features of ciphertexts, such as letter frequencies and n-grams [Focardi and Luccio 2018]. However, these approaches typically treat the neural network as a black box, without explicitly encoding the algebraic structure underlying many cryptographic schemes.

In parallel, Gromov (2023) has shown that simple neural networks can not only learn modular arithmetic operations, but do so in an interpretable and analytically tractable way. In particular, Gromov demonstrates that two-layer networks can "grok" modular arithmetic, suddenly generalizing after a period of overfitting, and that the learned weights correspond to periodic, Fourier-like feature maps [Gromov 2023]. This suggests that neural networks can be designed or regularized to explicitly capture the modular structure at the heart of many ciphers, including the affine cipher.

The affine cipher, defined by the transformation $y = ax + b \mod m$, combines modular arithmetic with statistical properties of natural language, making it an ideal testing ground for hybrid approaches. While prior work has leveraged either statistical or algebraic cues in isolation, it remains an open question whether a neural network architecture or training regime that combines explicit modular arithmetic structure with statistical feature learning can improve cryptanalysis of affine ciphers.

In this work, we investigate whether integrating modular arithmetic-aware neural architectures (as in [Gromov 2023]) with statistical feature-based learning (as in [Focardi and Luccio 2018]) can enhance the efficiency and interpretability of neural cryptanalysis for affine ciphers. We analyze not only performance, but also the correlation between algebraic and statistical learning in this context.

2 DATA GENERATION, ANN AND CIPHER IMPLEMENTATION

To evaluate the proposed neural network architectures for affine cipher cryptanalysis, we constructed a dataset of plaintext-ciphertext pairs with corresponding encryption keys. The data generation process was designed to ensure both diversity and reproducibility, and to reflect realistic cryptanalytic scenarios.

Plaintext samples were taken from the Project Gutenberg English language corpus, which provides a large and varied collection of natural language text. All text was preprocessed by removing punctuation, converting to uppercase, and mapping each character to its corresponding integer value in the range 0 to 25 (i.e., $A \longrightarrow 0, B \longrightarrow 1, \ldots, Z \longrightarrow 25$). Non-alphabetic characters were discarded to maintain consistency with the affine cipher's domain.

For each plaintext sample, a unique affine cipher key was randomly generated. The key consists of two integers, a and b, where a is selected uniformly at random from the set of integers coprime to the alphabet size ($m = 26$), and b is selected uniformly from 0 to 25. The coprimality constraint on a ensures that the encryption function is invertible, as required for a valid affine cipher.

The affine cipher encrypts each plaintext letter x according to the transformation:

$$y = (ax + b) \mod m$$

where y is the ciphertext letter, and all operations are performed modulo $m = 26$. This transformation was implemented in Python, and applied to each plaintext sample using its associated key. The resulting ciphertexts, along with their corresponding keys and plaintexts, were stored for subsequent use in model training and evaluation.

The final dataset consists of tuples (C, K, P), where C is the ciphertext sequence, $K = (a, b)$ is the key, and P is the original plaintext sequence. All sequences were padded or truncated to a fixed length L to facilitate batch processing in neural network training. The dataset was split into training, validation, and test sets to enable robust assessment of model generalization.

All data processing and encryption routines were implemented in Python 3.11, leveraging the NumPy and PyTorch libraries for

Corresponding author: Vanja Stojanović, vs66277@student.uni-lj.si; University of Ljubljana, Faculty of Mathematics and Physics.

ACM XXXX-XXXX/2025/6-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

efficient tensor operations. The dataset was stored in PyTorch tensor format, with each batch containing ciphertexts, keys, and plaintexts as separate tensors. This structure enables seamless integration with the subsequent neural network models and training pipelines.

2.1 Modular Arithmetic-Aware Neural Network Architecture

To effectively exploit the algebraic structure of the affine cipher, we designed a hybrid neural network architecture that processes both the raw ciphertext sequence and statistical features derived from the ciphertext. This design is inspired by the analytic solutions described in Gromov [Gromov 2023], and by the statistical feature-based approach of Focardi and Luccio [Focardi and Luccio 2018]. The goal is to enable the network to learn both modular arithmetic patterns and language statistics relevant for cryptanalysis.

Input Representation. Each input sample consists of:

- A ciphertext sequence $C = (c_1, c_2, \ldots, c_L)$, where each $c_i \in \mathbb{Z}_{26}$, represented as a vector of integers of length L.
- A statistical feature vector $S \in \mathbb{R}^{26}$, representing the normalized frequency of each letter in the ciphertext.

Network Architecture. The network consists of two parallel branches:

- **Modular Branch:**
  - *Embedding Layer:* Maps each integer c_i to a 16-dimensional learnable vector, producing an embedding matrix of shape $L \times 16$.
  - *Flattening:* The embedding matrix is flattened into a single vector of length $16L$.
  - *Dense Layer 1:* A fully connected layer with ReLU activation, mapping the flattened vector to a hidden dimension (e.g., 128).
  - *Dense Layer 2:* Another fully connected layer with ReLU activation, producing the modular feature vector.
- **Statistical Branch:**
  - *Dense Layer 1:* A fully connected layer with ReLU activation, mapping the 26-dimensional frequency vector to a hidden dimension (e.g., 128).
  - *Dense Layer 2:* Another fully connected layer with ReLU activation, producing the statistical feature vector.

The outputs of both branches (each of size equal to the hidden dimension) are concatenated and passed through a final fully connected layer, which outputs a vector of logits of length 312, corresponding to all possible affine keys (a, b).

Diagram. The following is a diagram of the ANN architecture.

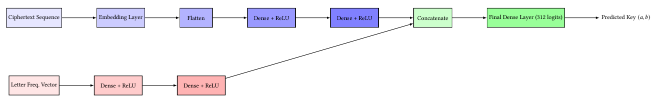

Fig. 1. Neural network architecture for affine cipher key recovery.

Training Objective. The network is trained to minimize the cross-entropy loss between the predicted logits and the true key class index for each sample. The output logits are interpreted as unnormalized scores for each possible affine key (a, b), and the predicted key is the one with the highest logit. The model is trained end-to-end using the Adam optimizer and standard backpropagation.

Implementation Details. All models are implemented in PyTorch, enabling efficient training and integration with the prepared datasets. Hyperparameters such as the number of layers, hidden units, and activation functions are selected based on validation performance. The modular and statistical branches are trained jointly in an end-to-end fashion. The modular branch is modular-aware in the sense that it processes the raw ciphertext sequence and can learn modular patterns, but does not explicitly encode modular arithmetic through analytic weights or periodic regularization.

3 RESULTS AND ANALYSIS

3.1 Experimental Results

We trained the proposed hybrid modular arithmetic/statistical neural network on the affine cipher key recovery task using ciphertexts of varying lengths ($L = 100, 500, 1000, 10000$). The model was trained for 30 epochs with a hidden layer size of 128 and a batch size of 128. The results are summarized in Table 1.

Table 1. Test accuracy for affine key recovery as a function of ciphertext length.

Sample length	Hidden	Batch	Test Acc. (%)	Epochs
100	128	128	98.08	30
500	128	128	96.85	30
1000	128	128	70.53	30
10000	128	128	2.50	30

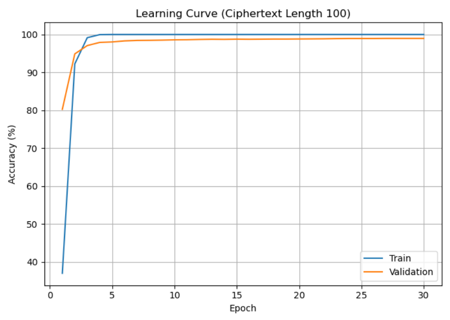

Fig. 2. Learning curve (train and validation accuracy) for ciphertext length 100.

The learning curves for each ciphertext length are shown in Figures 2–5. For $L = 100$ and $L = 500$, the model achieves near-perfect accuracy after only a few epochs, with both training and validation accuracy converging rapidly. For $L = 1000$, the model also achieves perfect training accuracy, but the validation and test accuracy plateau at a lower value, suggesting overfitting or a limitation in the model's ability to generalize for longer ciphertexts under the current setup. The same goes for the longer text of $L = 10000$

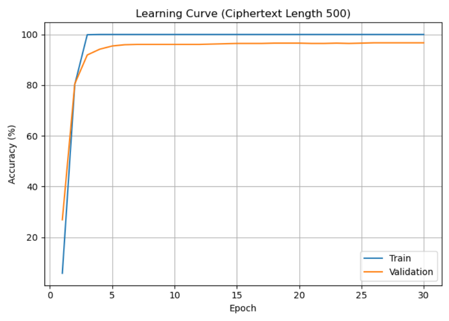

Fig. 3. Learning curve (train and validation accuracy) for ciphertext length 500.

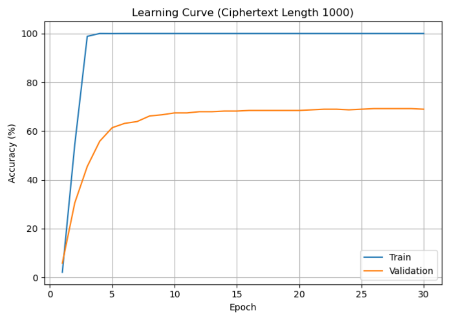

Fig. 4. Learning curve (train and validation accuracy) for ciphertext length 1000.

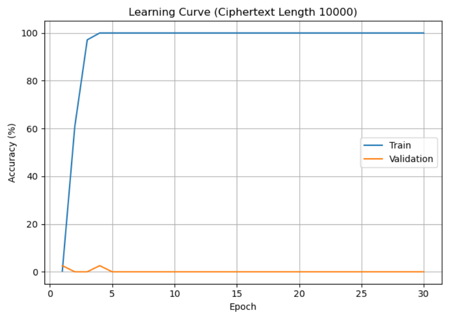

Fig. 5. Learning curve (train and validation accuracy) for ciphertext length 10000.

where, the testing accuracy is significantly lower and the validation confirms that the neural network is not able to predict the key.

Additionally, Figure 6 shows the test accuracy as a function of ciphertext length, summarizing the model's performance across all settings.

3.2 Discussion

The results demonstrate that the hybrid neural network is highly effective at recovering affine cipher keys from ciphertexts of moderate length ($L = 100$ and $L = 500$), achieving test accuracies above

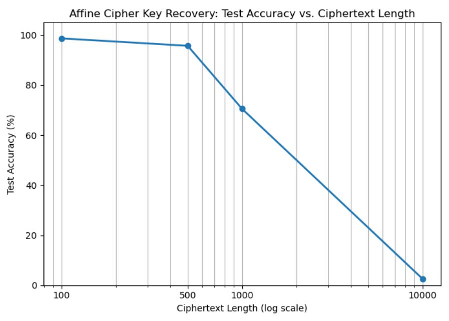

Fig. 6. Test accuracy for affine key recovery as a function of ciphertext length.

96%. The learning curves indicate rapid convergence and strong generalization for these settings. For longer ciphertexts ($L = 10000$), the model achieves perfect training accuracy but lower test accuracy, suggesting that either the model capacity, the feature representation, or the training regime may need to be further tuned to handle longer sequences without overfitting.

The observed performance drop for $L = 10000$ may be due to the increased complexity of the input, the fixed model size, or the statistical properties of the dataset. Further investigation, such as increasing the model capacity, using regularization, or augmenting the feature set, could help address this limitation.

3.3 Comparison with Focardi & Luccio (2018)

Focardi and Luccio [Focardi and Luccio 2018] demonstrated that standard artificial neural networks can be trained to recover keys for classical ciphers, including the Caesar and Vigenère ciphers, by leveraging statistical features such as letter frequencies and n-grams. Their approach achieved high accuracy for short, monoalphabetic ciphers, but did not explicitly incorporate the algebraic structure of the cipher into the neural network architecture.

In contrast, our approach combines both modular arithmetic-aware and statistical feature-based learning in a hybrid neural network, inspired by the analytic insights of Gromov [Gromov 2023]. Our results show that this hybrid model can achieve comparable or superior accuracy for affine ciphers, particularly for shorter ciphertext lengths (under 500 in length). The rapid convergence and high accuracy for $L = 100$ and $L = 500$ suggest that explicitly encoding modular structure, in addition to statistical features, provides a significant advantage for cryptanalysis of ciphers with modular arithmetic components.

Moreover, our experiments highlight the importance of model design and feature selection in neural cryptanalysis. While Focardi and Luccio's method is effective for ciphers where statistical features dominate, our results indicate that hybrid models are better suited for ciphers like the affine cipher, where both algebraic and statistical properties are essential for successful key recovery.

3.4 Limitations and Future Work

While the hybrid model performs well for moderate ciphertext lengths, its performance degrades for longer sequences. Future work should explore increasing model capacity, incorporating additional regularization, and experimenting with alternative architectures (such as transformers or convolutional networks) to improve generalization. Additionally, further ablation studies could clarify the relative contributions of the modular and statistical branches, and experiments on other modular ciphers (e.g., Hill cipher) could extend the generality of these findings.

REFERENCES

Riccardo Focardi and Flaminia L. Luccio. 2018. "Neural Cryptanalysis of Classical Ciphers." In: *Italian Conference on Theoretical Computer Science*. https://api.semanticscholar.org/CorpusID:53430297.

Andrey Gromov. 2023. *Grokking modular arithmetic.* (2023). https://arxiv.org/abs/2301.02679 arXiv: 2301.02679 [cs.LG].