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

Vanja Stojanović[1], **Žiga Lesar**[2], **Ciril Bohak**[2]

[1]*Faculty of Mathematics and Physics, University of Ljubljana, Slovenia*
E-mail: vs66277@student.uni-lj.si
[2]*Faculty of Computer and Information Science, University of Ljubljana, Slovenia*
E-mail: {ziga.lesar, ciril.bohak}@fri.uni-lj.si

Abstract

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 and neural cryptanalysis of classical ciphers, 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 attains 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.

1 Introduction

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 [2]. 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 [3] 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. 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 [3]) with statistical feature-based learning (as in Focardi and Luccio [2]) 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. All code and datasets for this work are available in a GitHub repository.[1]

2 Related Work

Gromov [3] demonstrates that simple two-layer ANNs can learn modular arithmetic tasks through a phenomenon known as "grokking," where generalization emerges suddenly after extensive training. Notably, the study shows that this learning process corresponds to the discovery of interpretable periodic features, akin to Fourier components. The paper even derives analytic solutions for network weights when learning additive modular functions, as exemplified in Equation (1).

$$f(n, m) = f_1(n) + f_2(m) \mod p \tag{1}$$

This provides strong support for our work by confirming that neural networks can learn the fundamental operations involved in affine ciphers and by offering insights into the mechanism underpinning this learning—namely, the encoding of modular features.

Another relevant contribution explores the use of standard ANNs for automating the cryptanalysis of classical ciphers, including Caesar (shift), Vigenère, and substitution ciphers [2]. This study adopts a ciphertext-only setting and leverages known statistical weaknesses of the ciphers. For the shift cipher, a neural network is trained to map ciphertext frequency distributions to the corresponding shift key. This network is then repurposed to attack Vigenère ciphers by segmenting ciphertext into monoalphabetic subtexts based on hypothesized key lengths and applying the trained shift model. This approach validates the broader applicability of neural networks in classical cryptanalysis, reinforcing the feasibility of our method for the affine cipher.

[1] https://github.com/Vanja-S/Using-Modular-Arithmetic-Optimized-Neural-Networks-to-Crack-Affine-Cryptographic-Schemes-Efficiently

Further work by Jeong et al. [4] expands on the potential of deep learning for cryptanalysis. Their findings highlight how neural models can effectively reveal vulnerabilities in encryption schemes, illustrating the capacity of neural cryptanalysis to discover weaknesses in cryptographic algorithms. This aligns closely with Gromov's results, affirming neural networks as powerful tools for addressing cryptographic challenges.

Finally, Dubey et al. [1] focus on incorporating modular arithmetic into neural network design. Their research emphasizes the development of architectures that naturally accommodate the structural requirements of cryptographic tasks. These insights directly inform our approach, which integrates modular branches within the network to enhance its cryptanalytic capabilities.

3 Data Generation, ANN and Cipher Implementation

To evaluate our neural network's architecture (proposed in Section 3.1) for the 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 with Equation (2).

$$h : \{A, B, C, \ldots, Z\} \to \mathbb{Z}_{26} \tag{2}$$

i.e., $A \mapsto 0, B \mapsto 1, \ldots, Z \mapsto 25$. Non-alphabetic characters were discarded to maintain consistency with the affine cipher's domain.

For each plaintext sample, a distinct affine cipher key was generated at random. The key comprises two integers, a and b, where a is drawn uniformly at random from the set of non-zero elements of $\mathbb{Z}_{26}$ that are coprime to the alphabet size (that is $m = 26$), and b is drawn uniformly at random from $\{0, 1, \ldots, 25\}$. There are exactly 12 possible values for a, corresponding to the integers in this range for which $\gcd(a, 26) = 1$. The requirement that a be coprime to 26 ensures that the encryption function is invertible, which is necessary for the affine cipher to be valid.

The affine cipher encrypts each plaintext letter x according to the transformation Equation (3).

$$y = (ax + b) \mod m \tag{3}$$

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 randomly shuffled and split into training, validation, and test sets in proportions of 80%, 10%, and 10%, respectively.

All data processing and encryption routines were implemented in Python 3.11, leveraging the NumPy and PyTorch libraries for efficient tensor operations. All models were also implemented with PyTorch. The dataset was stored in PyTorch tensor format, with each batch containing ciphertexts, keys, and plaintexts as separate tensors.

3.1 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 by Gromov [3] and by the statistical feature-based approach by Focardi and Luccio [2]. The goal is to enable the network to learn both modular arithmetic patterns and language statistics relevant for cryptanalysis.

InputRepresentation: 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, where L is the size of the dataset $L \in \{100, 500, 1000, 10\,000\}$.
- 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 (128 in our experiments).
- *Dense Layer 2*: Another fully connected layer with ReLU activation, producing the modular feature vector (dimension 128).

Statistical Branch:

- *Dense Layer 1*: A fully connected layer with ReLU activation, mapping the 26-dimensional frequency vector to a hidden dimension (128 in our experiments).
- *Dense Layer 2*: Another fully connected layer with ReLU activation, producing the statistical feature vector (dimension 128).

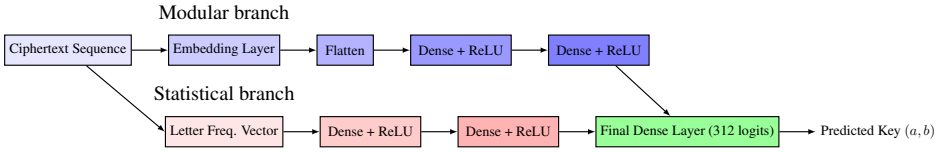


Figure 1: Neural network architecture for affine cipher key recovery.

The outputs of both branches (each of size equal to the hidden dimension) are concatenated and passed through a final fully connected layer *without activation*, which outputs a vector of logits of length 312, corresponding to all possible affine keys (a, b). These logits are used as input to the cross-entropy loss function during training.

System diagram: A diagram of the ANN architecture is presented in Figure 1.

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: Hyperparameters, including the number of layers, hidden units (128), and activation functions (ReLU), were selected based on initial validation and kept fixed for all experiments. The modular and statistical branches are trained jointly end-to-end. While the modular branch does not explicitly encode modular arithmetic, its design — processing the raw ciphertext sequence via embeddings — enables the network to learn modular patterns from data.

4 Results and Analysis

4.1 Experimental results

We trained the proposed ANN on the affine cipher key recovery task using ciphertexts of varying lengths (L = 100, 500, 1000, 10 000). 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.

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

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

The learning curves for each ciphertext length are shown in Figures 2 and 3. 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 $L = 1\,000$, 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 $L = 10\,000$ where the testing accuracy is significantly lower and the validation confirms that the neural network is not able to predict the key.

Additionally, Figure 3 shows the test accuracy as a function of ciphertext length, summarizing the model's performance across all settings.

4.2 Discussion

The results demonstrate that the hybrid neural network is highly effective at recovering the affine cipher keys from ciphertexts of moderate length ($L = 100$ and $L = 500$), achieving test accuracies above 96 %. The learning curves indicate rapid convergence and strong generalization for these settings. For longer ciphertexts ($L = 10\,000$), 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.

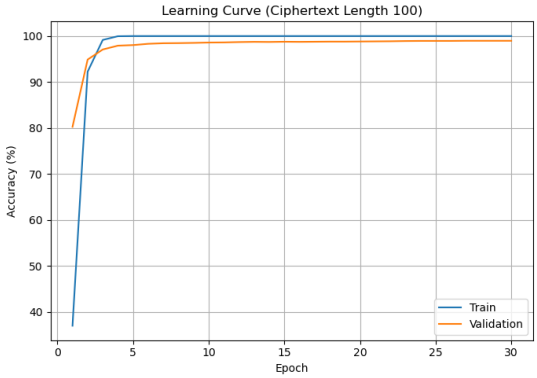
The observed performance drop for $L = 10\,000$ may be due to the increased complexity of the input, the fixed model size, underfitting, the embedding layer properties of the modular branch, or the statistical properties of the dataset. Further investigation, such as increasing the model capacity, using regularization, or augmenting the feature set, could help address this limitation.

4.3 Comparison with Forcardi and Luccio

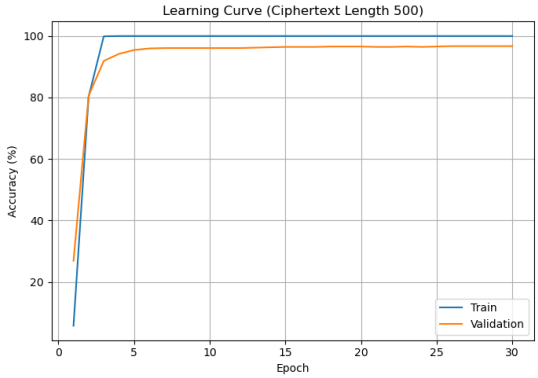
Focardi and Luccio [2] demonstrated that standard artificial neural networks can be trained to recover keys for classical ciphers, including the Caesar and Vigenère ciphers, by leveraging statistical features such as letter frequencies and n-grams. Their approach achieved high accuracy for short, mono-alphabetic 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 [3]. 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.

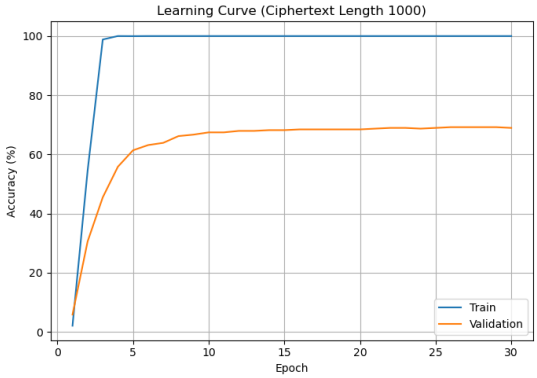
Moreover, our experiments highlight the importance of model design and feature selection in neural cryptanalysis. While Focardi and Luccio's method is effective for ciphers where statistical features dominate, our results indicate that hybrid models are better suited for ciphers



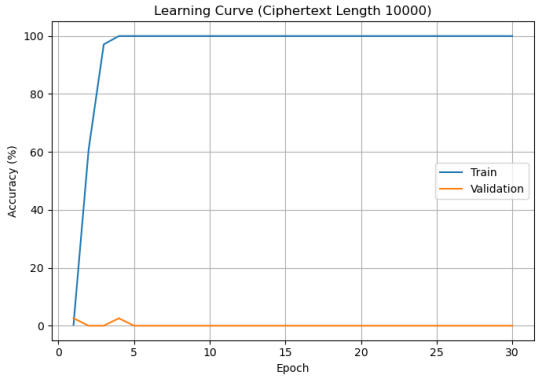
(a) Ciphertext length is 100.



(b) Ciphertext length is 500.



(c) Ciphertext length is 1 000.



(d) Ciphertext length is 10 000.

Figure 2: Learning curve (train and validation accuracy) for ciphertexts of different lengths.

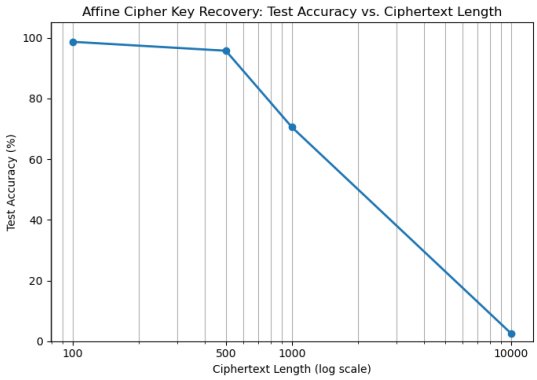


Figure 3: Test accuracy for affine key recovery as a function of ciphertext length.

like the affine cipher, where both algebraic and statistical properties are essential for successful key recovery.

4.4 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.

5 Conclusion

We have shown that the proposed hybrid neural network combining modular arithmetic-aware and statistical feature-based learning can accurately recover affine cipher keys from moderate-length ciphertexts. Explicitly modeling modular structure enables superior performance over purely statistical approaches for shorter texts. However, the model's effectiveness diminishes with longer ciphertexts, indicating challenges in generalization. These results emphasize the value of incorporating algebraic priors into neural cryptanalysis, especially for ciphers with modular components. Future work should focus on improving scalability and robustness, as well as extending this approach to other cryptographic schemes. Our findings highlight the promise of interpretable, structure-aware neural models for advancing automated cryptanalysis.

References

[1] A. Dubey, A. Ahmad, M. Pasha, R. Cammarota, and A. Aysu. Modulonet: neural networks meet modular arithmetic for efficient hardware masking. *Iacr Transactions on Cryptographic Hardware and Embedded Systems*, pages 506–556, 2021.

[2] Riccardo Focardi and Flaminia L. Luccio. Neural cryptanalysis of classical ciphers. In *Italian Conference on Theoretical Computer Science*, 2018.

[3] Andrey Gromov. Grokking modular arithmetic, 2023.

[4] O. Jeong, E. Ahmadzadeh, and I. Moon. Comprehensive neural cryptanalysis on block ciphers using different encryption methods. *Mathematics*, 12:1936, 2024.