CRYPTANALYSIS

Energy-Based and Neuro-Symbolic Methods for Advanced Cryptanalysis

Mentors:

1. GHRUANK KOTHARE
2. AFREEN KAZI

Mentees:

1. RISHIT MODI
2. ANUSHKA YADAV

Start

PROJECT OVERVIEW

- This project aims to explore the use of AI, neural networks, and neuro-symbolic learning to analyze and break encryption schemes.
- We focus on attacking modern encryption systems like AES using intelligent and adaptive methods.
- We are inspired by energy-based models, Hopfield networks, symbolic logic, and quantum-inspired techniques.
- The goal is to simulate advanced attacks and understand how encryption logic can be learned by machines.

PROJECT GOALS

1. **Cipher Cracking**

Learn to decrypt ciphertext without having the encryption key.

2. **Key Recovery**

Study how well modern encryption resists AI-based attacks.

3. **Symbolic Cipher Learning**

Train models to reverse engineer encryption logic based only on inputs and outputs.

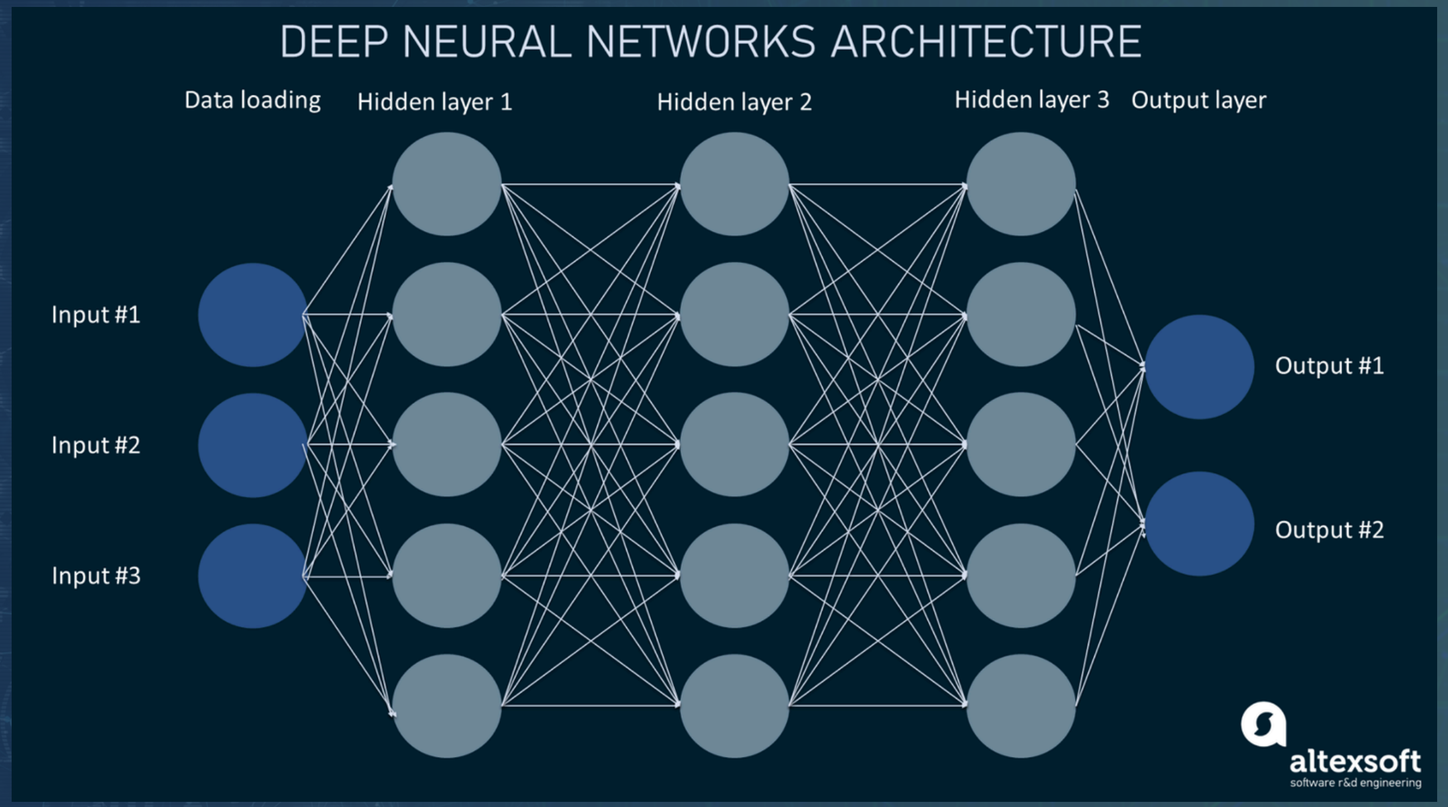
LEARNINGS

Neural Network:

- Weighted sum $Z = W \cdot X + b$
- Activation Functions: $\hat{y} = g(Z)$

Sigmoid: Good for binary outputs (0/1)
ReLU: Speeds up learning, avoids vanishing gradients

- Forward Propagation:
  Input → Hidden layers → Output
- Backpropagation: Error flows backward, adjusts weights to minimize loss
- Loss Function: Binary/Categorical Cross Entropy used to measure prediction error
  $L(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$
- Gradient Descent: Update weights to reduce loss ==> $w = w - \alpha * dw$



IMPROVING DEEP NEURAL NETWORKS

Weight Initialization

Prevents vanishing/exploding gradients

Optimization Techniques

- Gradient Descent,
- Adam Optimizer

Hyperparameter Tuning

Layer size, learning rate, iterations

Regularization (L2, Dropout)

Reduces overfitting

Gradient Checking

Debug tool to validate backpropagation

FROM THEORY TO APPLICATION

- DNNs to map ciphertext ↦ plaintext
- ReLU/Sigmoid to model encryption logic
- Vectorization for faster data handling
- Dropout & L2 to reduce overfitting
- Adam for quick optimization
- Gradient check for accuracy
- Mini-batch training for scalability

MINI PROJECT

Caesar Cipher Decrypter using Neural Networks

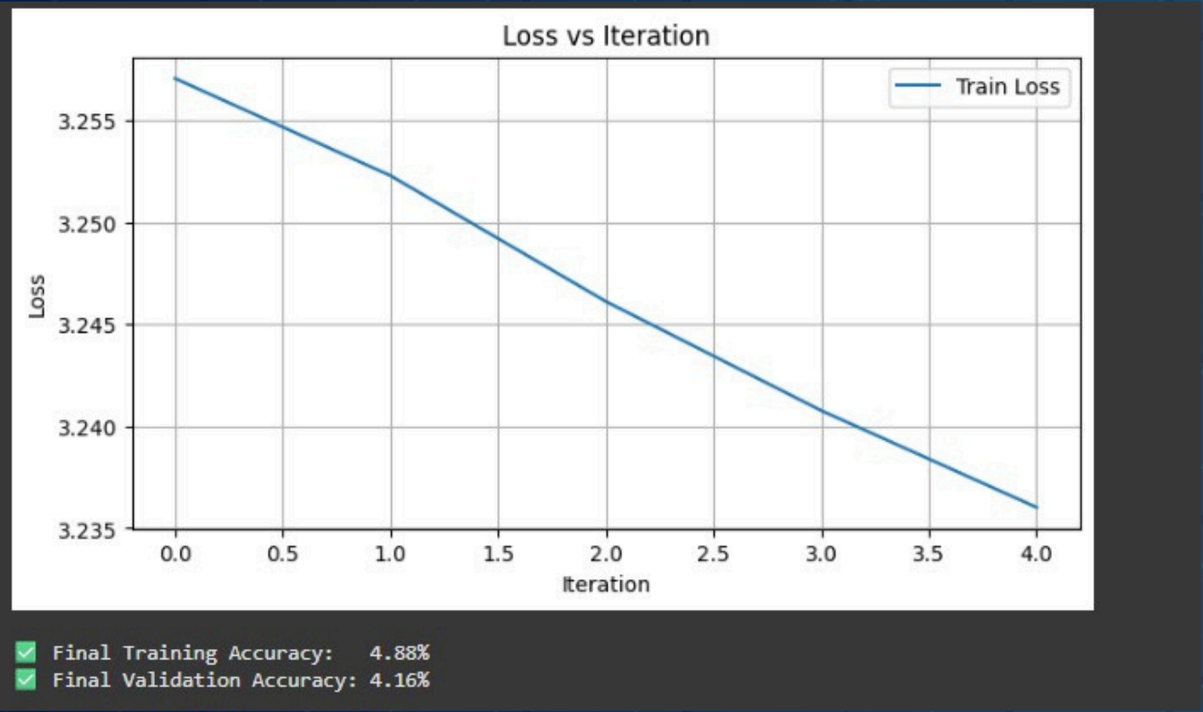
Objective:

- Encrypt plaintext using Caesar Cipher
- Decrypt ciphertext using a trained neural network
- No prior knowledge of shift required during decryption

Model Design:

- 2/4-layer Feedforward Neural Network
- Activation: ReLU (hidden), Sigmoid (output)
- Encoded input/output using one-hot vectors

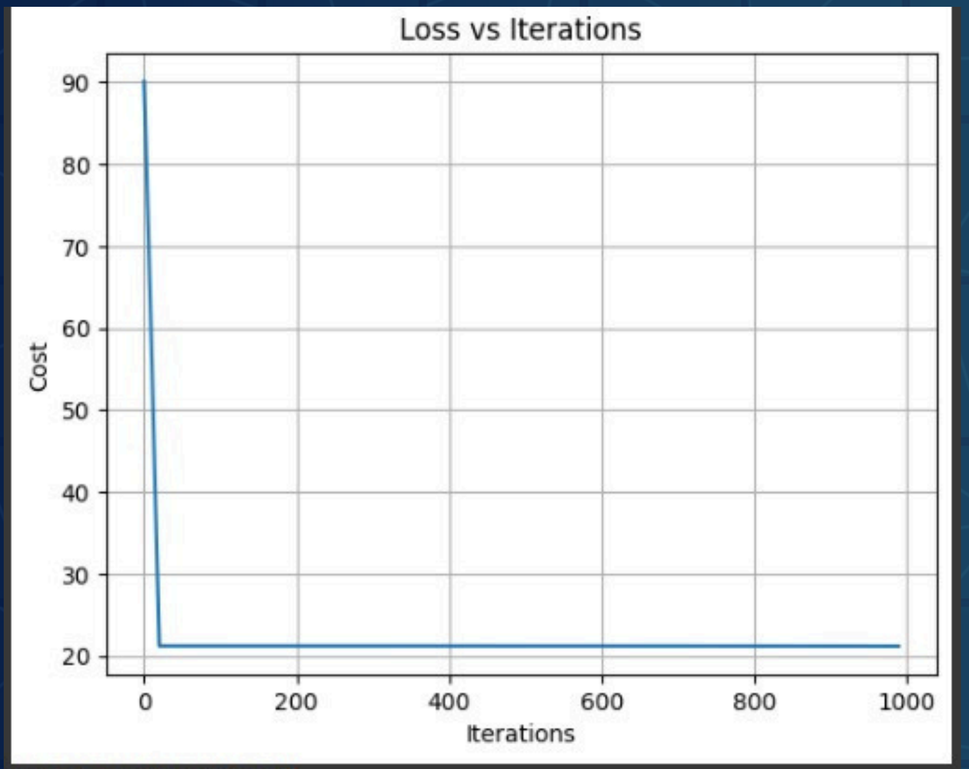
2 LAYER NEURAL NETWORK



Performance:

Train Accuracy: 4.88%
Test Accuracy: 4.16%

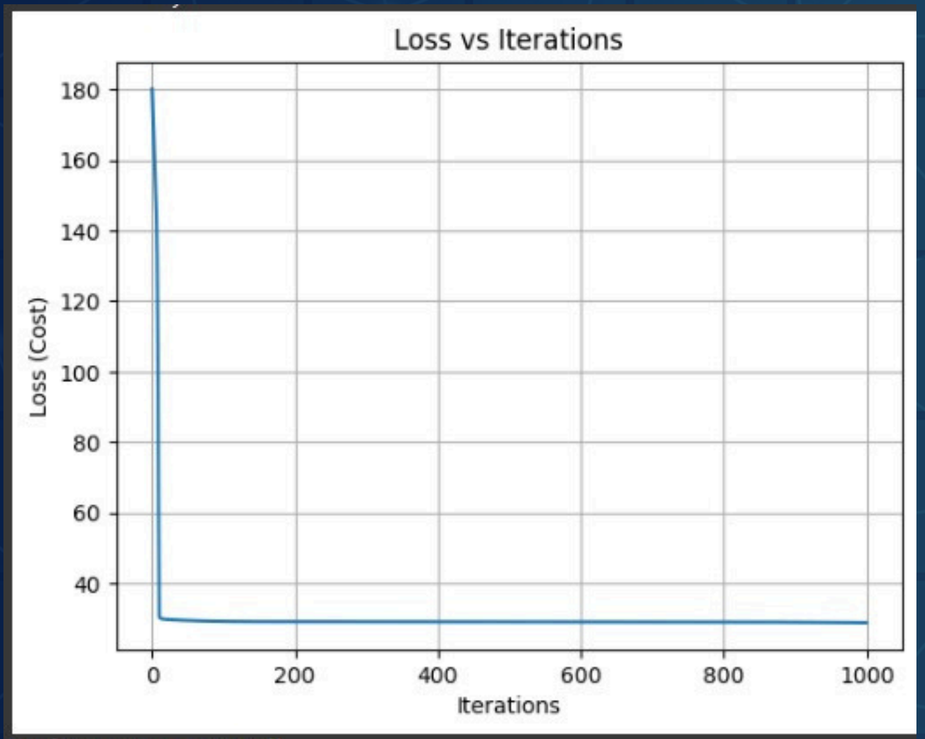
4 LAYER NEURAL NETWORK WITH FIXED KEY



Performance:

Train Accuracy: 96.15%
Test Accuracy: 96.15%

4 LAYER NEURAL NETWORK WITH VARIABLE KEY AND WORD LENGTH



Performance:

Train Accuracy: 97.49%
Test Accuracy: 97.50%

FUTURE WORK

Phase 1:

- Use basic neural networks to learn cipher patterns
- Train on Caesar encrypted text
- Measure character recovery accuracy

Phase 2:

- Apply sequence-to-sequence Transformer model
- Model learns token dependencies in ciphertext
- Output: Predicted plaintext sequences

Phase 3:

- Symbolic Logic Integration
- Fuse Transformer with symbolic rules (grammar, key patterns)
- Improves generalization and interpretability

FUTURE WORK - STEP 1

Goal:
Enhance Decryption Accuracy

- Integrate **Transformer** for better sequence modeling
- Train on diverse encrypted datasets (beyond Caesar)
- Focus on learning deep patterns in ciphertexts

FUTURE WORK - STEP 2

Goal:
Add Memory & Logic

- Embed **Hopfield Networks** for symbolic memory recall
- Fuse logical rules with neural inference
- Enable the model to generalize across cipher types

FUTURE WORK - STEP 3

Goal:
Optimize Key Recovery

- Use **Simulated Quantum Annealing** (**SQA**) for search
- Formulate decryption as energy minimization
- Test model on modern cryptosystems like AES

PROJECT APPLICATIONS

- Security Audit & Vulnerability Testing Tool
  - Use-case: Testing how weak a cipher or key management system is.
  - Why it matters: The system mimics real-world attackers using AI. Organizations can test custom or toy encryption algorithms for vulnerabilities before deployment.

- Logic Extraction from Black-Box Systems
  - Use-case: Reverse-engineering unknown or undocumented encryption systems.
  - Why it matters: Your neuro-symbolic module learns and reconstructs cipher logic from input-output pairs—useful in malware analysis or analyzing proprietary encryption.

- Proof-of-Concept for AI + Quantum-Inspired Cryptanalysis
  - Use-case: Show how simulated quantum annealing (SQA) and AI together shrink large keyspaces.
  - Why it matters: Useful for benchmarking hybrid AI-quantum cryptanalytic strategies—possibly relevant in post-quantum cryptography research.

THANK YOU!