Adversarial Neural Cryptography

Group : Infinity
Subhra Chakravorty (2021201078)
Mahesh Balaji Dudhanale (2021201010)
Vishal Pawar (2021201025)

Introduction

- A system may consist of neural networks named Alice and Bob.

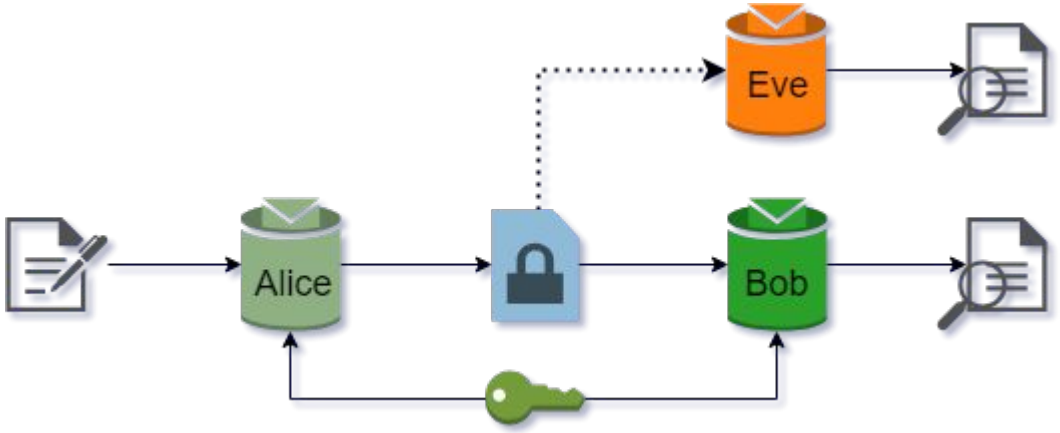
- We aim to limit what a third neural network named Eve learns from eavesdroping on the communication between Alice and Bob

Symmetric Encryption Setup

- Alice and Bob want to communicate securely over a public channel, and Eve is an attacker eavesdropping on the communication.

- For symmetric encryption, we may assume Alice and Bob both have a common secret key.

- Eve is able to capture all the encrypted messages Alice sends to Bob, but does not have access to the secret key.



Symmetric Encryption Setup

Objective

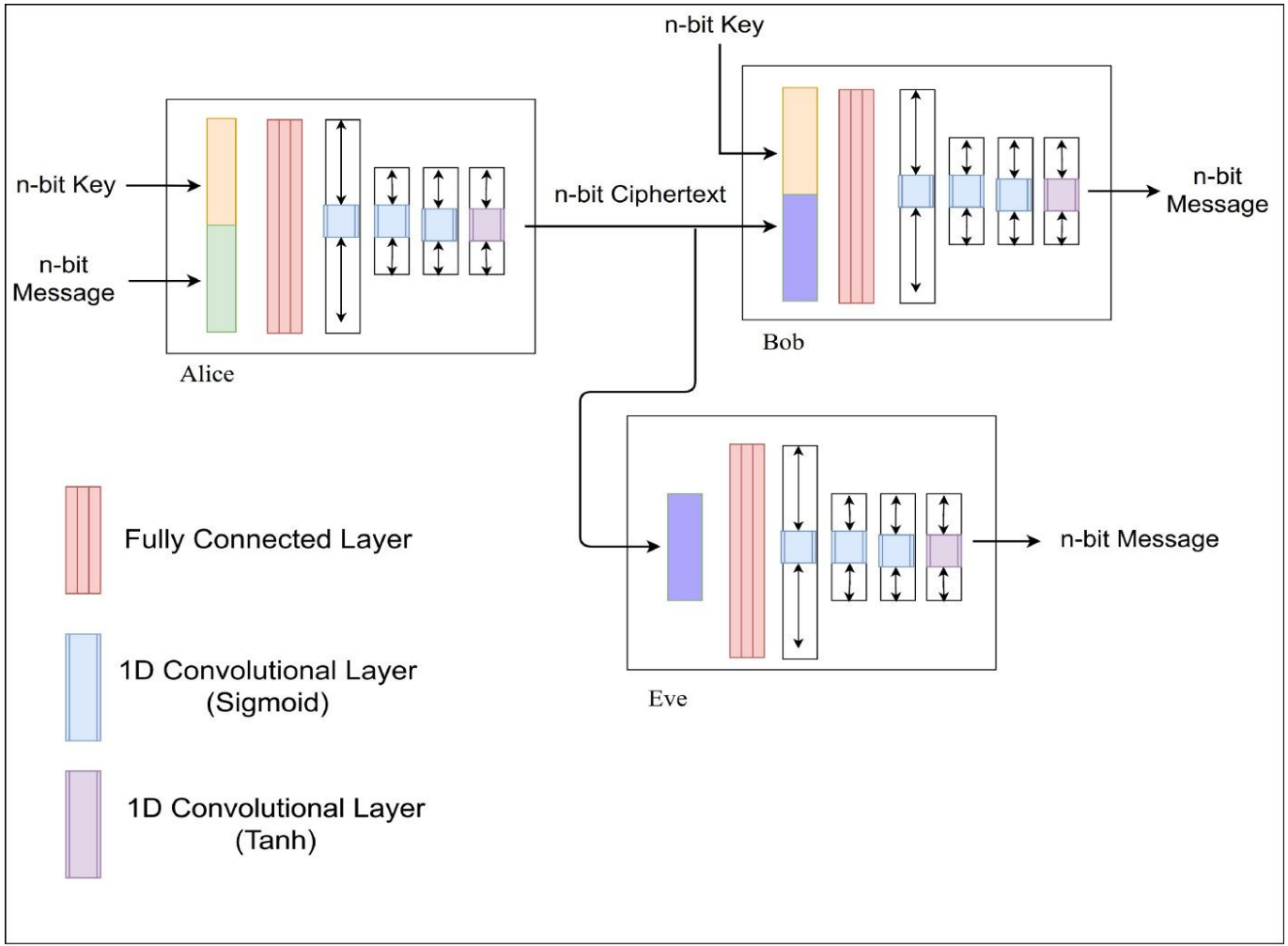
- Eve’s goal is simple: to reconstruct Original Message (P) accurately (in other words, to minimize the error between P and PEve).

- Alice and Bob want to communicate clearly (to minimize the error between P and PBob), but also to hide their communication from Eve.

- This is achieved by reducing error between Alice and Bob and simultaneously increasing loss of Eve’s Decryption.

Neural Network Architecture

- Mix and Transform Architecture is used.
- It has a first fully-connected (FC) layer, where the number of outputs is equal to the number of inputs. The plaintext and key bits are fed into this FC layer.
- Because each output bit can be a linear combination of all of the input bits, this layer enables but does not mandate mixing between the key and the plaintext bits.
- In particular, this layer can permute the bits. The FC layer is followed by a sequence of convolutional layers, the last of which produces an output of a size suitable for a plaintext or ciphertext.
- These convolutional layers learn to apply some function to groups of the bits mixed by the previous layer, without an a prior specification of what that function should be.


n-bit Key
n-bit Message
Alice
n-bit Ciphertext
n-bit Key
Bob
n-bit Message
Eve
n-bit Message
Fully Connected Layer
1D Convolutional Layer (Sigmoid)
1D Convolutional Layer (Tanh)

Loss Function

- Eve’s loss function : the L1 distance between Eve’s guess and the input plaintext.
- The loss function for Alice and Bob has two components, related to Bob’s reconstruction error and to the eavsdropper’s success.
- The first component is simply the L1 distance between Bob’s output and the input plaintext.
- The latter component, on the other hand, is (N/2 − Eve L1 error)2/(N/2)2.

Loss (Cont.)

For Eve :-

The loss for Eve is just the L1 distance between ainput0 and eoutput:

```
eveloss = K.mean(  K.sum(K.abs(ainput0 - eveout), axis=-1)  )
```

Instead of doing an average, sum is taken over all the bits in the message and its value represents the avg number of bits Eve guesses incorrectly.

We then take the average across the entire mini-batch with K.mean().

The minimum value of the loss is 0 (Eve guesses all the bits correctly), while the maximum is 16 (Eve is wrong about all the bits, in which case we flip the prediction and recover the message).

Loss (Cont.)

For Alice-Bob :-

The loss for Alice-Bob is slightly more complicated. First, we want Bob to successfully decrypt the ciphertext:

```
bobloss = K.mean(  K.sum(K.abs(ainput0 - bobout), axis=-1)  )
```

Would also like Alice to learn an encryption scheme which Eve can't break. In an ideal situation, Eve should do no better than random guessing, in which case she would correctly guess half the bits, or m_bits/2 correctly (corresponding to a loss value of 8).

```
K.square(m_bits/2 - eveloss)/( (m_bits//2)**2 )
```

The square term ensures the error grows large when Eve starts to guess a few of the bits correctly. This forces Alice to adapt more rapidly to attackers, and change its encryption scheme. Overall the Alice-Bob loss will be:

```
abeloss = bobloss + K.square(m_bits/2 - eveloss)/( (m_bits//2)**2 )
```

Training

Training alternates between Alice/Bob and Eve, with Alice/Bob training for one minibatch, and then Eve training for two minibatches.

We chose this ratio in order to give a slight computational edge to the adversary Eve without training it so much that it becomes excessively specific to the exact current parameters of Alice and Bob.

Additionally,

epoch_count = 20

batch_size = 512

n_batches = 128

Types of Cryptography Implemented

- Single Eve (CNN)
- Multi Eve (CNN)
- LSTM

Importance of Eve in Cryptography

Without Eve , Alice and Bob would quickly learn to ignore the key inputs and just pass the original message through.

But adversarial member of the cast is introduced : Eve.

Eve the eavesdropper is also a neural network, but Eve only gets the ciphertext as input, and not the key.

Two Eve Model

Need for Two Eve Model :-

In the CNN model built using 1 eve(adversary) the accuracy or strength of encryption depended mainly on eve's capability to reconstruct the original plaintext.

But since we were using only 1 eve, the efficiency of the model was only dependent on eve loss.

Hence alice could forget the involvement of the secret key entirely since the loss function did not take the key into account.

Advantage of two Eve

Hence we are introducing here another adversary:-

eve2 whose sole purpose is to recover the secret key.

In this new model, we have two attackers Eve1 and Eve2 with different attacking goals.

Eve1 focuses on recovering the plaintext and Eve2’s goal is to recover the secret key.

In this model, Alice cannot simply ignore the involvement of the secret key due to the existence of Eve.

Thank You