# ADVERSARIAL NEURAL NETWORK

MSCS 630L 711 20S

Dr. Pablo Rivas

Prabuddha Banerjee
Prabuddha.banerjee1@marist.edu

## 1. Abstract

Neural networks can learn to use secret keys to protect information from other neural networks. In a multiagent system we ensure confidentiality properties of multiagent system, and those are specified in terms of properties of an adversary.
Thus, a system may consist of neural networks named Alice and Bob, and we aim to limit what a third neural network named Eve learns from eavesdrop- ping on the communication between Alice and Bob. We demonstrate that the neural networks can learn how to perform different forms of encryption and decryption of an image in order to meet confidentiality goals.

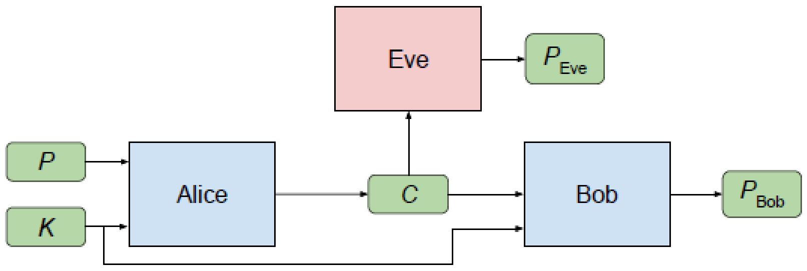
## 2. Introduction.

Specifically, we focus on developing a binary-string block cipher in ECB mode, and we specify the model in terms of adversarial networks. Thus, a system may consist of neural networks named Alice and Bob, and we aim to limit what a third neural network named Eve learns from eavesdropping on the communication between Alice and Bob. We do not prescribe specific cryptographic algorithms to these neural networks; instead, we train end-to-end, adversarially.
Artificial neural networks are well known for their ability to selectively explore the solution space of a given problem. This feature finds a natural niche of application in the field of cryptanalysis. At the same time, neural networks offer a new approach to attack ciphering algorithms based on the principle that any function could be reproduced by a neural network, which is a powerful proven computational tool that can be used to find the inverse-function of any cryptographic algorithm.
I got encouraged to work in such an amazing area of research on which even Google is working. He even motivated us to work in areas where the ideas of mutual learning, self learning, and stochastic behavior of neural networks and similar algorithms can be used for different aspects of cryptography, like public-key cryptography, solving the key distribution problem using neural network mutual synchronization, hashing or generation of pseudo-random numbers.
In this spirit, further work could be done, considering other tasks, for example steganography, pseudorandom-number generation, or integrity checks. Finally, neural networks may be useful not only for cryptographic protections but also for attacks. While it seems improbable that neural networks would become great at cryptanalysis, they may bequite effective in making sense of metadata and in traffic analysis.

## 3. Background and Related Work.

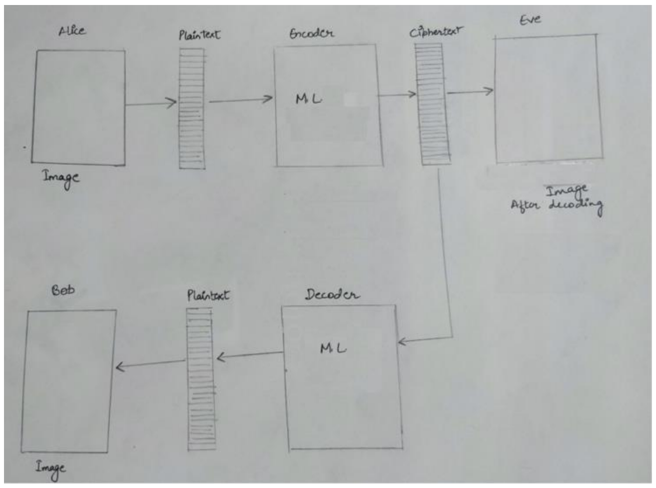
As neural networks are applied to increasingly complex tasks, they are often trained to meet end- to-end objectives that go beyond simple functional specifications. These objectives include, for example, generating realistic images (e.g., (Goodfellow et al., 2014a)) and solving multiagent problems (e.g., (Foerster et al., 2016a;b; Sukhbaatar et al., 2016)). Advancing these lines of work, we can show that neural networks can learn to protect their communications in order to satisfy a policy specified in terms of an adversary.
Somewhere between anonymization methods and homomorphic encryption, we find a novel technique pioneered by Google that uses adversarial neural networks to protect information from other neural models. The research paper detailing this technique was published at the end of 2016 under the title “Learning to Protect Communications with Adversarial Neural Cryptography” and it’s, easily, one of the most fascinating AI papers I’ve read.
The Google team adapted the GAN cryptography architecture in a model in which Alice and Bob still share a key, but here Alice receives A, B, C and produces D-public in addition to a ciphertext; both Bob and Eve have access to Alice’s outputs; Bob uses them for producing an

improved estimate of D, while Eve attempts to recover C. The goal is to demonstrate that the adversarial training permits approximating D without revealing C, and that this approximation can be combined with encrypted information and with a key in order to obtain a better approximation of D.



## 4. Methodology.

As a proof-of-concept, we implemented Alice, Bob, and Eve networks that take N -bit random plain- text and key values, and produce N-entry floating-point ciphertexts, for N = 16, 32, and 64. Both plaintext and key values are uniformly distributed. Keys are not deliberately reused, but may reoccur because of random selection. A classic scenario in security involves three parties: Alice, Bob, and Eve. Typically, Alice and Bob wish to communicate securely, and Eve wishes to eavesdrop on their communications.



Img: The methodology used in the experiment

Thus, the desired security property is secrecy (not integrity), and the adversary is a "passive attacker" that can intercept communications but that is otherwise quite limited: it cannot initiate sessions, inject messages, or modify messages in transit. For us, Alice, Bob, and Eve are all neural networks

## 5. Experiments

During the experiments I categorized the project in 3 separate sections:

i. Converting Image to bytes i.e. when Alice tries to send a message to Bob. So, I converted images into Hexadecimal bytes. I carried over the experiment in Python. So the data used are a black and white image. The black and white image I used is just for this initial research. As when we use image with colors in it we increase the components of RGB. In RGB each components needs to dealt independently for generating bytes.

ii. Converting Bytes to an Image(for Bob): Now Further I am working on the part where an Image needs to be created from any given piece of Bytes Data. So this is done or the part where this is taking place is just before the message reaches to Bob.

iii. Converting Bytes to an Image(for Eve): Now in this part an Image needs to be created from any given piece of Bytes Data similar to Bob's case but this data is been encrypted by Eve. So this is done or the part where this is taking place is just before the message is decoded by Eve and reaches to her.

## 6. Discussion or Analysis.

Now to understand whether the system is learning to hide information properly, we train a separate evaluator that we call "Blind Eve", which is aware of the distribution of Cipher text. Blind Eve tries to guess Cipher text relying only upon this baseline information. If Eve's reconstruction error becomes equal to that of First image, we know that Eve is not successfully extracting information from the public estimate and the ciphertext.
I am still working on the research part and will reach on to some concrete result regarding the Encryption and Decryption of the Adversarial Neural Network.

## 7. Conclusion.

In this paper, I am trying to demonstrate that neural networks can learn to protect communications. It is based only on a secrecy specification represented by the Methodology. In this setting, we model attackers by neural networks; alternative models may perhaps be enabled by reinforcement learning.
There is more to cryptography than encryption. In this spirit, further work may consider other tasks, for example steganography, pseudorandom-number generation, or integrity checks. Finally, neural networks may be useful not only for cryptographic protections but also for attacks. While it seems improbable that neural networks would become great at cryptanalysis, they may be quite effective in making sense of metadata and in traffic analysis.

## 8. References / Bibliography.

**Learning To Protect Communications with Adversarial Neural Cryptography**
Martín- Andersen-David G. - *https://arxiv.org/abs/1610.06918Bottom of Form*