

Autoencoder Neural Networks for Digital Cryptography

Jaime Tenorio

Óbuda University
John von Neumann Faculty of Informatics
Information and Coding Theory
Dr. Márta Takács
May 2023

1 Introduction

1.1 Cryptography

Cryptography is the study of techniques for secure communication, so that third parties cannot read messages sent between the sender and the receiver. An encryption algorithm is a mathematical function that transforms a message into a form that is difficult to predict or duplicate. Several encryption methods have been developed over the years, the most popular and widely used involve the use of a key, which defines how the message is encoded and decoded, for example, the Advanced Encryption Standard (AES) [1]. Cryptography has many applications, such as authentication, confidentiality, integrity, and non-repudiation.

1.2 Autoencoder Neural Networks

Autoencoders are a type of neural networks whose purpose is to learn efficient representations of data by undergoing unsupervised training [2]. When training an autoencoder, the network learns to reconstruct the original data by encoding the information in a latent space, which is a representation that captures the essential features of the data. These neural networks have a symmetric architecture, in other words, their input is the same shape as their output, and the middle layer is the one responsible for the representation of the data. The middle layer is usually called the code layer.

Autoencoder neural networks are used in tasks such as dimensionality reduction, anomaly detection, signal processing, data compression, and cryptography.

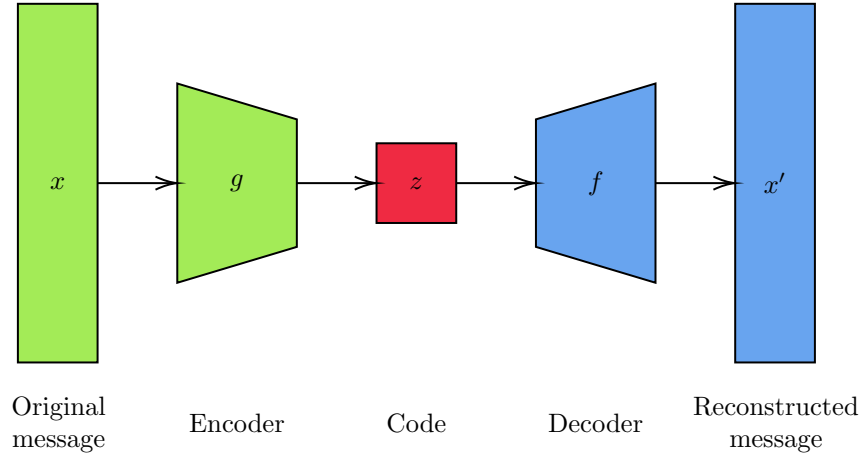


Figure 1: Autoencoder neural network.

A basic autoencoder architecture is shown in 1. The autoencoder may be expressed in mathematical terms as follows:

$$\begin{aligned}
 z &= g(x) \\
 x' &= f(z) \\
 x &\approx x'
 \end{aligned} \tag{1}$$

where x is the input, z is the code, x' is the output, g is the encoder, and f is the decoder.

2 Autoencoders for Cryptography

The nature of the autoencoder architecture makes it a suitable technique for cryptography, since the code layer may be used as an encrypted message. Essentially, a message can be encrypted by passing it through the trained encoder part of the network and extracting the output from the code layer. Then, the encrypted message is decoded by passing it through the decoder part of the network. The encoded message can then be transmitted securely, because the latent representation may not reveal sensitive information, even if it may be intercepted.

One advantage of autoencoder networks for cryptography is that they are versatile. Autoencoders can be adapted to any kind of data, such as images, text, numerical data, etc., and can be designed to learn codes of any size. Another advantage is their ability to filter noise and redundancies while encoding, due to the efficient representation of the data in the latent space.

Autoencoder architectures may vary depending on the cryptographic use case. Some examples include:

- **Fully connected autoencoder.** This is the most basic autoencoder, which consists of fully connected layers. This architecture can be used in general cryptography for encoding and decoding messages.
- **Convolutional autoencoder.** The use of convolutional layers in autoencoder architecture allows for the capture of spatial information, therefore this can be used for image or video cryptography.
- **Recurrent autoencoder.** Recurrent neural networks can learn sequential data efficiently. An autoencoder with this architecture may be useful for text encryption.

3 Case Study

3.1 Implementation and Results

We can demonstrate an autoencoder for cryptography by attempting to encrypt binary representation of characters. Previous studies show the viability of such autoencoders for 8-bit ASCII characters [3], so we will expand on this example and do it for Unicode characters.

Firstly, the training data is determined by generating all Unicode characters, for which we need at least 21 bits. If we define the training data as X , then it can be defined as follows:

$$X = \{x \in \{0, 1\}^{21} \mid x = \text{bin}(i), i \in [0, 1114111]\} \quad (2)$$

As it was discussed previously, the output of the neural network will be of the same dimension as its input. The depth of the neural network depends on the specific application, and the complexity of the system. Many sources suggest that a hidden encoder layer and a hidden decoder layer are sufficient for general purposes [4]. In this case, we will use a hidden layer of 32 neurons for both the encoder and the decoder. The activation function for the hidden layers will be the rectified linear unit (ReLU) function, and the output layer will use the sigmoid function. The sigmoid function is used because it is a good choice for binary classification problems. The ReLU function is used because it is a good choice for hidden layers in general, and it is the most common activation function for autoencoders [5]. The loss function to optimize is binary cross entropy, due to the binary nature of the input and output data. The optimizer used is the Adam optimizer, which is a popular choice for general purpose neural networks. The autoencoder will be trained for 50 epochs.

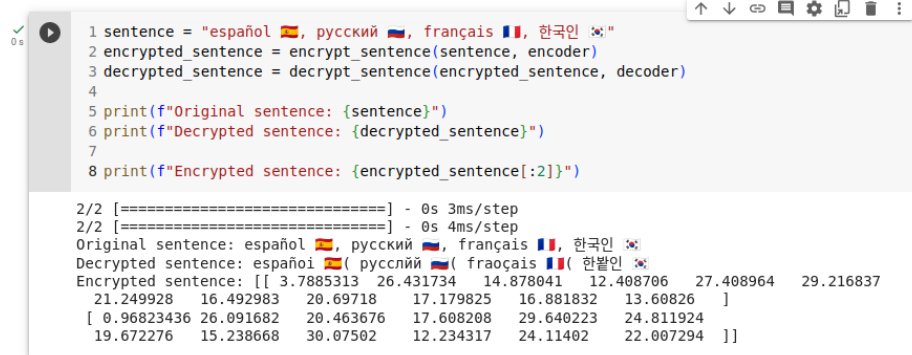
Since the encoding dimension will represent the characters, it is the most important parameter of the autoencoder. To determine an appropriate encoding dimension, we may train multiple autoencoders with different encoding dimensions, and compare their performance. The performance of the autoencoder will be measured by the percentage of characters that are correctly reconstructed. The encoding dimensions to be tested are 8, 12, and 16.

The results of reconstruction efficiency for the different encoding dimensions are shown in Table 1. The results show that the autoencoder with an encoding dimension of 16 is the most efficient, with a reconstruction efficiency of 100%. This means that using this autoencoder we could be capable of encrypting and decrypting Unicode characters in a lossless manner. The other encoding dimensions do not reach 100% efficiency, therefore they are not suitable for this application.

| Encoding Dimension | Reconstruction Efficiency | Final Loss Value |
|--------------------|---------------------------|-------------------------|
| 8 | 4.43% | 0.1715 |
| 12 | 33.83% | 0.0794 |
| 16 | 100% | 7.6340×10^{-5} |

Table 1: Reconstruction efficiency for different encoding dimensions

We may assume that encoding dimensions bigger than 16 will also have 100% reconstruction efficiency, but they will be more computationally expensive. Therefore, we can conclude that the best autoencoder for this application is the one with an encoding dimension of 16. The system can be tested by encoding and decoding text with Unicode characters, as it is shown in Figure 2. In the figure we can see that the autoencoder is capable of reconstructing all characters without error. A preview of the first two character encryptions is shown too.



```

1 sentence = "español 🇪🇸, русский 🇷🇺, français 🇫🇷, 한국인 🇰🇷"
2 encrypted_sentence = encrypt_sentence(sentence, encoder)
3 decrypted_sentence = decrypt_sentence(encrypted_sentence, decoder)
4
5 print(f"Original sentence: {sentence}")
6 print(f"Decrypted sentence: {decrypted_sentence}")
7
8 print(f"Encrypted sentence: {encrypted_sentence[:2]}")

2/2 [=====] - 0s 3ms/step
2/2 [=====] - 0s 4ms/step
Original sentence: español 🇪🇸, русский 🇷🇺, français 🇫🇷, 한국인 🇰🇷
Decrypted sentence: español 🇪🇸( русский 🇷🇺( fraçois 🇫🇷( 한말인 🇰🇷
Encrypted sentence: [[ 3.7885313 26.431734 14.878041 12.408706 27.408964 29.216837
21.249928 16.492983 20.69718 17.179825 16.881832 13.60826 ]
[ 0.96823436 26.091682 20.463676 17.608208 29.640223 24.811924
19.672276 15.238668 30.07502 12.234317 24.11402 22.007294 ]]

```

Figure 2: Encoding and decoding Unicode characters.

3.2 Discussion

Although the network is capable of reconstructing all characters without error, one of the issues is the transmission of the encoded messages. The best performing model has a 16-dimensional code, which means that the encrypted characters consists of 16 neuron activations, which are in turn, 16 floating point numbers. This means that the encrypted message is more than 20 times larger than the original message. This is a significant increase in size, and it is not suitable

for most applications. However, it is possible to reduce the size of the encoded message by using a smaller encoding dimension, at the cost of reconstruction efficiency.

Of course, another option is to utilize a different activation function for the encoding layer, so that it can be discretized more easily, and thus, transmitted. However, this would require a different loss function, and a deeper neural network, since it would be more difficult to reconstruct the original message. This would also increase the computational complexity of the system, which is not desirable.

Nonetheless, this example shows that autoencoders can be used for cryptography, and that they can be used to encrypt and decrypt messages without error. This shows that autoencoders can be used for lossless encryption, although it would take a more sophisticated system to achieve lossless encryption with a reasonable message size.

Furthermore, the complexity of AES, the most popular encryption algorithm, is $O(n)$, where n is the number of bits in the message [1]. This means that the complexity of AES is linear with respect to the message size. On the other hand, the complexity of the autoencoder is $O(n^2)$, where n is the number of neurons in the network. This means that the complexity of the autoencoder is quadratic with respect to the message size.

4 Conclusion

Autoencoder neural networks are powerful and versatile tools that can be used for a variety of applications. It is possible to apply this technology in the field of cryptography. As we discussed previously, the advantages of autoencoders are that they are easy to implement, and they can be trained with either small or large datasets. The results from the case study show significant drawbacks regarding encrypted message size. We conclude that autoencoders can be used for lossless encryption, although it would take a more sophisticated system to achieve lossless encryption with a reasonable message size.

References

- [1] B. A. Forouzan, *Introduction to Cryptography and Network Security*. New York, NY, USA: McGraw-Hill Higher Education, 2008, ISBN: 978-0-07-287022-0.
- [2] M. Kramer, "Autoassociative neural networks," *Computers & Chemical Engineering*, vol. 16, no. 4, pp. 313-328, 1992, Neutral network applications in chemical engineering, ISSN: 0098-1354. DOI: 10.1016/0098-1354(92)80051-A.

- [3] F. Q. Socasi., R. Velastegui., L. Zhinin-Vera., R. Valencia-Ramos., F. Q. Ortega-Zamorano., and O. Chang.,
 “Digital cryptography implementation using neurocomputational model with autoencoder architecture,” in *Proceedings of the 12th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*, INSTICC, SciTePress, 2020, pp. 865–872, ISBN: 978-989-758-395-7.
 DOI: 10.5220/0009154908650872.
- [4] U. Menon, A. R. Menon, A. Hudlikar, A. Sharmila, and P. Mahalakshmi,
 “A hybrid autoencoder architecture for text encryption,” in *2021 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2021, pp. 1–7. DOI: 10.1109/i-PACT52855.2021.9696715.
- [5] Y. Song, S. Hyun, and Y.-G. Cheong, “Analysis of autoencoders for network intrusion detection,” *Sensors*, vol. 21, no. 13, 2021, ISSN: 1424-8220. DOI: 10.3390/s21134294.