Deep Learning-based Cryptanalysis of Different AES Modes of Operation

Milena Gjorgjievska Perusheska[1], Hristina Mihajloska Trpceska[1] and Vesna Dimitrova[1]

[1] "Ss. Cyril and Methodius" University in Skopje, Faculty of Computer Science and Engineering, Skopje, North Macedonia
milena.gjorgjievska-perusheska@students.finki.ukim.mk
hristina.mihajloska@finki.ukim.mk
vesna.dimitrova@finki.ukim.mk

Abstract. With the advent of machine learning and the development of powerful machines, the problem of decryption takes on a new light and opens new avenues for research. The remarkable rise of technologies and algorithms contributes to the widespread use of machine learning in various cases. One of the uses is in cryptanalysis and attack of algorithms used in cryptographic processes. In this paper, we elaborate the idea by using the deep neural network to perform the known-plaintext attack on AES to restore as many bits as possible, on the given plaintext. Moreover, we perform our experiments on different key sizes and different modes of operation on AES. The results show that the deep neural network can restore the bits in the whole data set with a probability of more than 98%, restore two consecutive bytes with more than 70%, and more than half of the plaintext bytes with a probability of 99%.

Keywords: Cryptography, Cryptology, Cryptanalysis, Machine Learning, Deep Neural Network

1 Introduction

In recent years, when talking about Cryptology as a science, the emphasis has been put on using resources and the application of ML (Machine Learning) as a discipline that finds application even when it comes to security. On the one hand, ML is used in the construction of new security solutions, while on the other hand, it is used in various attacks and analyses of current methods.

When it comes to the implementation and use of AES (Advanced Encryption Standard) [1] as a block and symmetric cipher , ML offers potential for expansion research. Furthermore, while cracking a cipher like AES used to be relatively tricky, today's theory shows that the attack surface is far broader than previously thought.

Before being chosen as a standard, this primitive was known as the Rijndael block cipher [2] [3]. Furthermore, the key that is utilized is private and can be 128, 192, or 256 bits in length, ensuring that the security is reliable. The input, output, and mode of operation will all be in 128-bit block size. The operations carried out inside this algorithm are based on two-dimensional arrays substitution and permutation.

Additionally, with its modes of operation, AES allows a new way of thinking about how to make modifications to crack encryption and disclose information. ML is the best match for analyzing these AES modes of operation using DNN (Deep Neural Networks) with the intricacy of work in mind.

AES is one of the most popular block ciphers, and over the years, many known attacks against it have been identified, and their analysis is being improved for future use. However, there is little research done in the field of machine learning and cryptanalysis of AES. On the basis of the structural characteristics of the neural networks and known attacks against AES, the specific experimental system is designed and discussed in this paper. The capacity to learn from prior experience and improve known facts allows some of the attacks to be implemented into a well-constructed DNN and create a model that can be utilized for other analyses.

Additionally, in this paper, the focus is on the analysis of the ECB (Electronic CodeBook), CBC (Cipher Block Chaining), and Ctr (Counter) modes of operation in AES through known plaintext-ciphertext pairs. To be more specific, the pairs are encrypted separately and independently of the modes and key sizes, which are 128-, 192-, and 256-bits long. Furthermore, the study is based on the training and testing of a DNN that provides specific block restoration findings. On the other hand, the experimental findings give details on the subsequent recovered blocks as well as the consequences of using DNN as a decipher. Finally, the proposed DL-based cryptanalysis is a promising step towards a more efficient and automated test for checking the safety not for the AES as a standalone cipher but more for the other primitives that use AES in some mode of operation.

The remainder of this paper is organized as follows: in Section 2, we give a short overview of the usage of DNN in different cryptanalysis scenarios. In Section 3, we describe in more detail the DNN framework that will be used in experiments in the rest of the paper. In order to perform a machine learning approach in this cryptanalysis, we need to set up the experimental environment, generate data, adjust the parameters and obtain the train and test phase, which are all explained in Section 4. Section 5 contains our main experimental results on using the deep neural network to perform the known-plaintext attack on AES in different modes of operation to restore as many bits as possible on the given plaintext. We conclude our paper in Section 6.

2 Related work

The symmetric crypto primitives have been the most attractive target for attacks for a long time. In a general case, the attacks can be divided into two cryptanalysis groups, differential, and linear attacks. The differential cryptanalysis analyzes how differences of plaintext pairs result in differences of the resultant ciphertext pairs and is done by the well-known chosen-plaintext attack [4]. On the other hand, linear cryptanalysis is a statistical method where linear approximations between plaintext bits, ciphertext, and key bits are analyzed. The known-plaintext attack on block ciphers obtains it [5].

Afterward, there was a huge improvement in the area of cryptanalysis, and many other cryptanalytic attacks were developed. In this paper, we focus on the known-plaintext attack where the plaintext can be restored from the ciphertext without knowing the key. In order to skip the part with the mathematical calculations about the linear approximations between the bits, we combine the power of machine learning and neural networks to recognize a given pattern [6]. Pattern recognition is thus equivalent to the issue of decrypting a ciphertext by knowing the plaintext. In this scenario, even with the smaller amount of data (compared to the traditional method) of plaintext and ciphertext bits pairs, the neural network can be well trained and afterward used for testing.

However, traditional cryptanalysis methods have flaws in terms of attack difficulty and the quantity of data necessary for them, which DNN cryptanalysis can address. Whether or not you have the key, DNN analysis can answer a wide range of problems and raise many concerns about other security issues. There are several vulnerabilities in symmetric ciphers, such as block ciphers, that can be exploited for security breaches. Knowing the key or not, the analysis with DNN can solve many different variations of problems and open up many additional questions for other security issues.

A few scientific publications are concentrating on utilizing DNN to identify the mapping relationship between plaintexts, ciphertexts, and the key. The work in [7] presented results about attacking the DES and 3DES, where a known-plaintext attack is based on neural networks. They successfully trained a neural network to retrieve plaintext from ciphertext without any interaction with the keys. Also, there is another attack on nonlinear characteristics of DES by using a neural network [8]. In this work, the authors show the application of a neural cryptanalysis approach to DES S-box. With this approach, they obtained the correct values for some of the key bits. The authors in [9] use backpropagation neural networks to perform cryptanalysis on AES to restore plaintext. Their results show that the neural network can restore the entire byte with more than 40% probability. The authors in [10] use a neural network to predict the key from a dataset used to encrypt a collection of plaintexts acquired using a lightweight block cipher - Simon cipher. The author in [11] trains neural networks to distinguish the output of Speck cipher with a given input difference from random data and exploit differential properties of round-reduced Speck. Another successful attack on lightweight block ciphers, such as Simon and Speck, is done in [12]. The results show that the DL-based cryptanalysis can recover the key bits successfully when the keyspace is restricted to 64 ASCII characters. And the newest developments in [13] demonstrate an analysis of Gohr's deep neural network distinguishers [11] and give possible directions even to improve the existing accuracy of the proposed DNN.

3 DNN framework

Neural networks are a subset of machine learning (ML) that refers to a specific type of multi-layered model that learns data representations by layering smaller statistical components. With the combination of many hidden layers and non-linear activation functions a representation of complicated patterns in datasets is created. A neural network takes an input and sends it through several layers of hidden neurons, represented as functions with unique coefficients that must be learnt, to produce a prediction representing the combined input of all the neurons. This type of capacity is associated with the brain's biological neuron, which interprets incoming signals from the body as a piece of information based on prior knowledge and experience [14]. Various neurons are stacked in layers, and each input is processed via the layers to generate an output. Each layer-to-layer link has a weight that is changed by two propagation methods: backpropagation and feedforward.

A collection of input features and some random weights in a feedforward neural network will be tuned via backward propagation. The difference between anticipated and goal output is computed during backpropagation, and the weight values are updated using a method [15]. Furthermore, backpropagation is represented as computing the derivative. It is essentially the chain rule from calculus, in which a minor change in the set of weights impacts the ultimate loss. During the training phase, the derivative or gradient retains that change.

The weights are separated into sets and input into a hidden layer, which has a second set of weights leading to the anticipated output and loss. Additionally, the impact of changing weights on the concealed layer should be addressed. If there are more weights to consider, the chain rule from output to input will be used to continue the process. A neural network is considered to have completed one iteration when it has read the number of records given by the batch size in both the forward and backward directions. The uniform distribution approach is used to establish the neurons' weights, and they are modified after each iteration. The neurons are given estimated weights and bias values at the conclusion of each iteration. One epoch is defined as the number of iterations that have been done throughout the whole dataset. For each record in the dataset, estimated values for each neuron in the network would have been assigned at the end of one epoch.

4 Model for cryptanalysis

The model learns more and provides better outcomes due to using more hidden layers [16]. The primary objective is to learn from the input and provide an output as near as possible to the plaintexts supplied. Keras [17], a neural network framework that is part of TensorFlow [18], an open-source toolkit for a variety of machine learning applications, is utilized for this.

Experimenting with various machine learning techniques to train and evaluate the model, such as Binary cross-entropy, SGD (Stochastic gradient descent), RELU (Rectified linear activation function), and Sigmoid function, yielded different results and addressed different ways of improving the model to improve the experiment's effectiveness. Furthermore, the approaches are utilized when binary classification difficulties are encountered and have yielded promising results in the past, as seen by their usage in various studies. Scikit-learn is a machine learning software package for the Python language that is used to evaluate data.

The primary issue is that data will be represented in binary format because it involves two values from the matrix, either 0 or 1. Binary cross-entropy is the most common and well-known loss function in machine learning. The computations for these two maximum likelihood estimators are based on distinct assumptions about the dependent variable.

4.1 **Experimental Environment**

The DNN, a sequential model with a linear stack of layers, is developed for this experiment. The input layer features a 128-unit input shape and a RELU activation function. In addition, there are three hidden layers with a dimension of 256 bits and a RELU activation function, as well as an input matrix with a size of [n x 128] and an output matrix with a size of [n x128] and a Sigmoid function. The input and output layers are each 128 bits in size.

Furthermore, the device on which the training and testing part is performed has the configuration represented in Table1.

Table 1. Device specifications

Device part	Capability	
Processor	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz	
System (OS)	Windows 10	
GPU	NVIDIA GeForce GTX 1050 Ti	
Memory	128 GB SSD + 1 TB HDD	
RAM	16.0 GB	

Even though the plaintexts were encrypted with different key sizes and different AES modes of operation, after comparing the time required for training a DNN, the time taken does not change much. The average number of hours required is 0.9. This was done to see if the training duration, the accuracy, or the complexity of prediction were affected.

4.2 **Data Generation (Dataset)**

For the experimental chapter of this research, a dataset comprising pairings of plaintexts and ciphertexts was required. Still, more specifically, ciphertexts were obtained by encrypting the plaintexts with different key sizes to demonstrate if there is a difference or difficulty when training the DNN and comparing the results produced.

The IMDB dataset [3] was utilized to do this, which is made up of movie reviews with individual lengths of no more than 200 words, with about 50K reviews recorded as several lengthy phrases. According to the dataset description, the reviews are unique, which is the major point of interest and will be demonstrated further on.

The reviews dataset will represent the plaintexts and will be utilized for DNN training and testing. In addition, the dataset is prepared for the model using a few basic processes in order to get accurate results. To begin, every input in the encryption procedure must be divisible by 16 (measure in bytes) or 128 (measure in bits) in order to use a 128-bit block of AES encryption.

To avoid further issues with encryption, such as padding with zero bytes, which would confuse the training process and provide unexpected results, each review is reduced to a size that is divisible by 16. In other words, the difference between the plaintext's length and the remaining characters is picked for the following step from a single plaintext.

When considering ECB mode, there is one encryption according to the key, apart from CBC and Ctr modes where there are two different encryptions where the IV (Initialization vector) is also considered. Moreover, one encryption type is when each IV is unique and used separately for every plaintext, and the other is the usage of the same IV for the whole dataset.

Then, keys with sizes of 16 bytes (128 bits), 24 bytes (192 bits), and 32 bytes (256 bits) are generated using cryptographically safe pseudo-random number generators. Additionally, each plaintext is encoded as a binary string and encrypted with each key separately to generate a ciphertext as an output, resulting in three distinct datasets of plaintext and ciphertext pairings for the training process, for each mode of operation.

After that, because the pairings are in byte string format, the model cannot be created without first formatting the plaintexts and ciphertexts into bit arrays. Natural language must be converted into machine language, such as numerical vectors, when employing a machine learning method like DNN.

Furthermore, between plaintexts and ciphertexts, an appropriate bit scale is produced, where the length of the plaintext bits equals the length of the ciphertext bits. This is done in order to prevent issues with dataset size while training the DNN. Following that, all of the plaintext and ciphertext bit arrays are combined into a matrix, or so-called 2D array, with a fixed column size of 128 bits and a row size of about 3 million, with exception when using Ctr mode with the same IV, where the row size is about 1.5 million. This is due to the fact that the encryption is performed on text of the same length, 2^{10}characters.

4.3 **Parameter Selection**

The inputs are pairs of 128-bit blocks of plaintexts and ciphertexts $[(p_1, c_1), \dots (p_n, c_n)]$, from a binary matrix of order $[n\times128]$, with n specified by the dataset's train-test divide of 80 percent and 20 percent. Each bit array of the plaintext and ciphertext pairings corresponds to one neuron in the input layer. Following that, the data is passed through three hidden layers with a 256-dimensional dimension and a RELU activation function. The output layer generates outputs $[o_1, \dots, o_n]$ using a sigmoid function and a 128-dimensional dimension. An optimizer, loss function, and monitoring metrics are all set up before the model is trained. The dataset was trained with Binary cross-entropy loss functions, SGD as an optimizer, and accuracy as the measure.

4.4 **Training Phase**

The input data and target data, which in this case are the plaintexts and ciphertexts, make up the training procedure of the presented model, accounting for 80 percent of the total dataset (3 million blocks). Batch size, epochs, and validation data are also available. Furthermore, the important notion here is that the model must be able to recognize when it is incorrect, which we accomplish by computing some type of loss.

The loss that is computed is determined by the task at hand, although it usually entails minimizing the difference between the expected and actual output. Batch size refers to the number of samples per gradient update, epochs to the number of training cycles, and validation data to the data used to assess the loss and any model metrics at the conclusion of each epoch [19].

Once making a prediction, a history object is returned that stores all the loss values and other metric data in memory for use in visualizations or reports. To put it another way, it's a dictionary that keeps track of training loss and metrics values over time, as well as validation loss and metrics values. The loss function's main objective is to assess how excellent or poor the predicted probabilities are; thus, it should return high values for bad predictions and low values for good ones.

Binary cross-entropy compares the expected distribution of whether the bits will be predicted to the actual distribution and seeks to minimize the discrepancies between the two.

During the training phase, Binary cross-entropy and SGD produce a score representing the average difference between the actual and projected probability distributions for correct prediction. Under the maximum likelihood inference framework, the score is minimized for a perfect cross-entropy, which is the chosen loss function.

As the epoch size grows, the accuracy above shows the properly predicted bits as well as the loss during training. It can be shown that using this combination of optimizer and loss function, the results may be discussed further.

4.5 **Test Phase**

One of the most challenging aspects of training a DNN is determining the right number of epochs and batch sizes. The learning algorithm's major characteristics are batch size and SGD. When training the network, the batch size affects the accuracy of the error gradient estimate. The speed and consistency of the learning process increase as the batch size grows [20].

Furthermore, while dealing with large datasets, it is inevitable that the number of training instances will be large, and the network's weights will be modified to ensure that the model's performance is always improved. The batch size is set to 5000 for this purpose, implying that 5000 samples from the training set will be used to estimate the error gradient before the model weights are changed.

An epoch, on the other hand, denotes that the learning algorithm only makes one run over the training set, which was previously divided into batches of varying sizes. Furthermore, the validation set includes the test set, which contains 20% of the plaintexts and ciphertexts.

5 Experimental Results

The constructed model exceeded expectations and contributed to the thoughts and analyses that will be discussed in this chapter. These outcomes are split into three datasets, each containing ciphertexts obtained by encrypting the given plaintexts with different key sizes and AES modes of operation. Furthermore, the train data was evaluated using the test set, which is 20% of the plaintext and ciphertext data, with an average accuracy of 98.44% for ECB mode, 93.09% for CBC mode with different IV, 98.44% for CBC mode with same IV, 98.29% for CTR mode with different IV and 93.18% for CTR mode with the same IV, reflecting the overall predicted bits in a 128-bit block.

Tables 2-7 provide a representation of the results obtained. Moreover, the main focus was on four types of experiments: restoring the bits in the whole dataset (accuracy), restoring an entire block individually (accuracy score), restoring two consecutive bytes in a block and restoring more than a half of the bytes within the block. There is a ratio between the number of true positives and false positives determined by the precision score. The precision score allows the classifier to avoid labelling a negative sample as positive. The recall score, which preserves the ratio between true positives and false negatives and tends to discover all positive samples (in this example, a matched bit), is another measure used for assessment. Finally, an F1 score (see [21]) is calculated as a weighted average of accuracy and recall.

When considering Table 7, there are results which represent calculations of restoring two consecutive bytes that vary from 33% up to 77.1% and restoring more than half of the bytes in a block in the range of 59.8% up to 99.8%. All the results are according to the complexity of the cipher, its key, and the operation mode used.

Table 2. Results for AES-ECB mode

	AES-ECB-128	**AES-ECB-192**	**AES-ECB-256**
Accuracy	98.81%	98.19%	98.33%
Loss	3.44%	4.98%	4.64%
Accuracy score	32.10%	18%	20.45%
Precision score	98.87%	98.26%	98.42%
Recall score	98.9%	98.35%	98.46%
F1 score	98.88%	98.30%	98.44%

Table 3. Results for AES-CBC mode with different IV

	AES-CBC-128	**AES-CBC-192**	**AES-CBC-256**
Accuracy	98.46%	82.94%	97.87%
Loss	4.31%	4.72%	5.74%
Accuracy score	22.30%	7.44%	1.32%

Precision score	98.54%	83.85%	97.99%
Recall score	98.57%	84.24%	98.02%
F1 score	98.55%	84.03%	98.00%

Table 4. Results for AES-CTR mode with different IV

	AES-CTR-128	**AES-CTR-192**	**AES-CTR-256**
Accuracy	98.47%	98.38%	98.03%
Loss	4.29%	4.52%	5.37%
Accuracy score	23.33%	21.52%	15.63%
Precision score	98.55%	98.47%	98.13%
Recall score	98.58%	98.49%	98.19%
F1 score	98.56%	98.48%	98.16%

Table 5. Results for AES-CBC mode with the same IV

	AES-CBC-128	**AES-CBC-192**	**AES-CBC-256**
Accuracy	98.40%	98.52%	98.40%
Loss	4.45%	4.18%	4.44%
Accuracy score	21.81%	24.53%	22.01%
Precision score	98.45%	98.59%	98.48%
Recall score	98.54%	98.63%	9.52%
F1 score	98.50%	98.61%	98.50%

Table 6. Results for AES-CTR mode with the same IV

	AES-CTR-128	**AES-CTR-192**	**AES-CTR-256**
Accuracy	93.47%	93.03%	93.05%
Loss	16.88%	17.77%	17.77%
Accuracy score	11.98%	6.89%	7.53%
Precision score	93.93%	93.51%	93.53%
Recall score	93.82%	93.40%	93.43%
F1 score	93.87%	93.46%	93.48%

Table 7. Results for two consecutive bytes and more than half of the bytes guessed within a block

Encryption mode	Two consecutive bytes guessed	More than half of the bytes guessed
AES-Same-CTR-128	33%	65.2%
AES-Same-CTR-192	30.9%	59.8%

AES-Same-CTR-256	30.8%	60.1%
AES-Diff-CTR-128	74.8%	99.6%
AES-Diff-CTR-192	73.8%	99.6%
AES-Diff-CTR-256	70.3%	99.3%
AES-Diff-CBC-128	74.7%	99.7%
AES-Diff-CBC-192	46%	68.3%
AES-Diff-CBC-256	68.5%	99.1%
AES-Same-CBC-128	74%	99.6%
AES-Same-CBC-192	75.3%	99.7%
AES-Same-CBC-256	74.1%	99.6%
AES-ECB-128	77.1%	99.8%
AES-ECB-192	75.8%	99.7%
AES-ECB-256	73%	99.6%

Furthermore, Figures 1-15 provide a sampling of the differences between the original and predicted bytes in the block. The first 50 blocks of the data from the complete set for a simplified visual representation are taken as an example. The calculations are based on the encrypted blocks with different key sizes and AES modes of operation. Every result from an encrypted block is separate and not connected with a result obtained in another mode. More accurately, every figure is a representation of the same part of blocks from the dataset (128-bit blocks), in which every individual block consists of 16 bytes. Each byte from each original and predicted block is extracted and compared bit by bit, yielding a value ranging from 0 (no difference in both bytes) to 1 (no match in both bytes), as indicated in the color scheme.

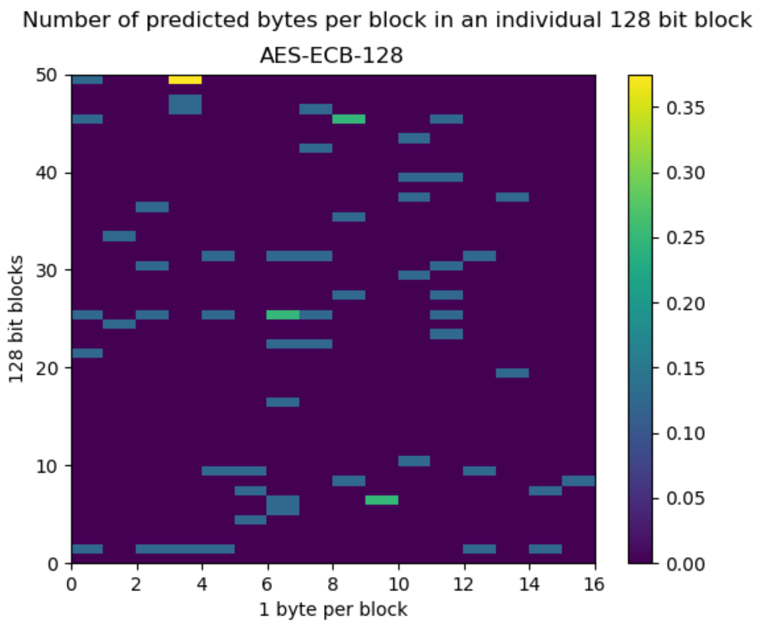

Fig. 1. Histogram representation of predicted bytes per block when using AES-ECB-128

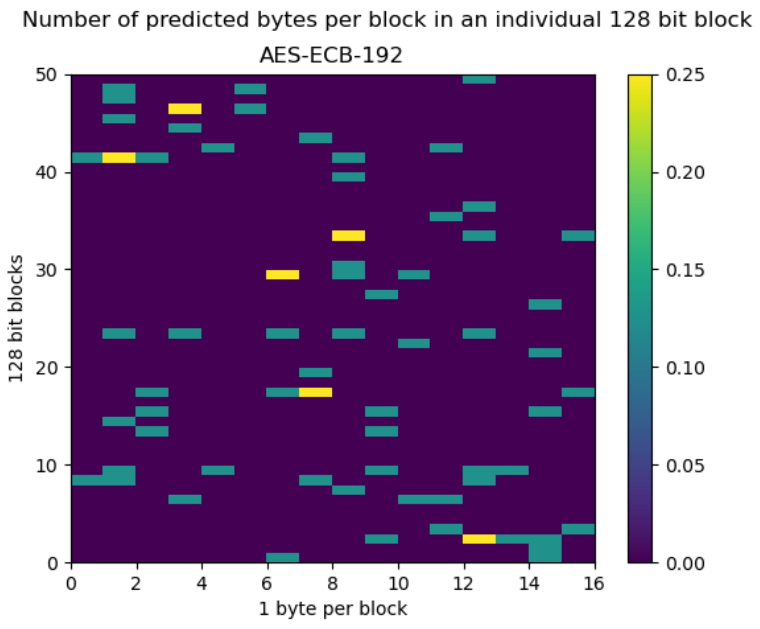

Fig. 3. Histogram representation of predicted bytes per block when using AES-ECB-192

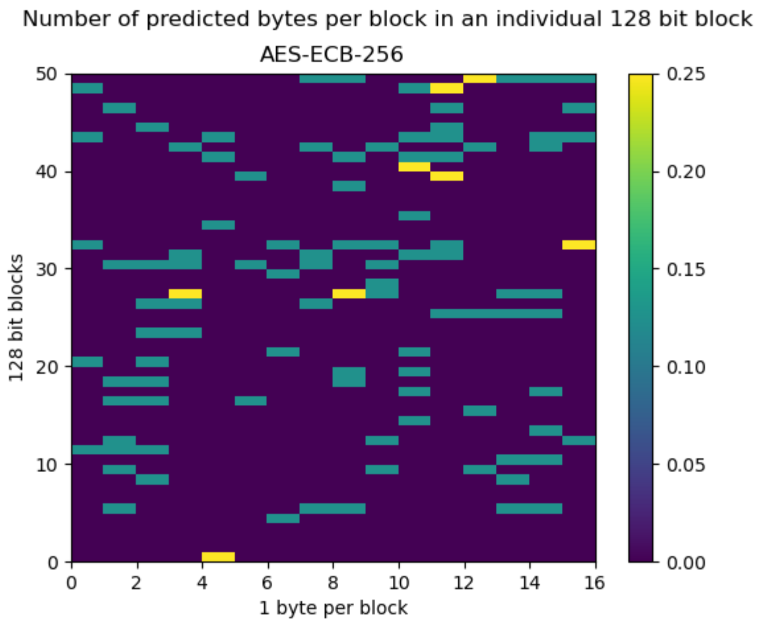

Fig. 3. Histogram representation of predicted bytes per block when using AES-ECB-256

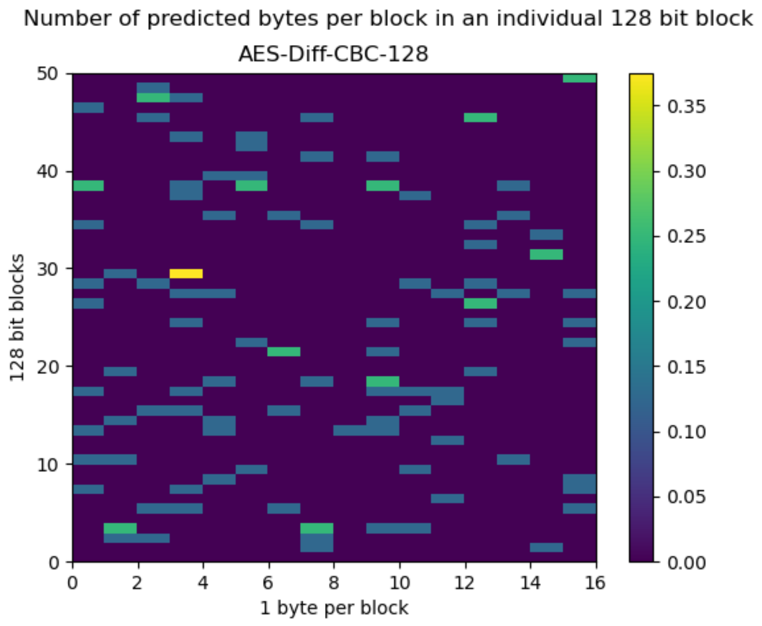

Fig. 4. Histogram representation of predicted bytes per block when using AES-CBC-128 with different IV

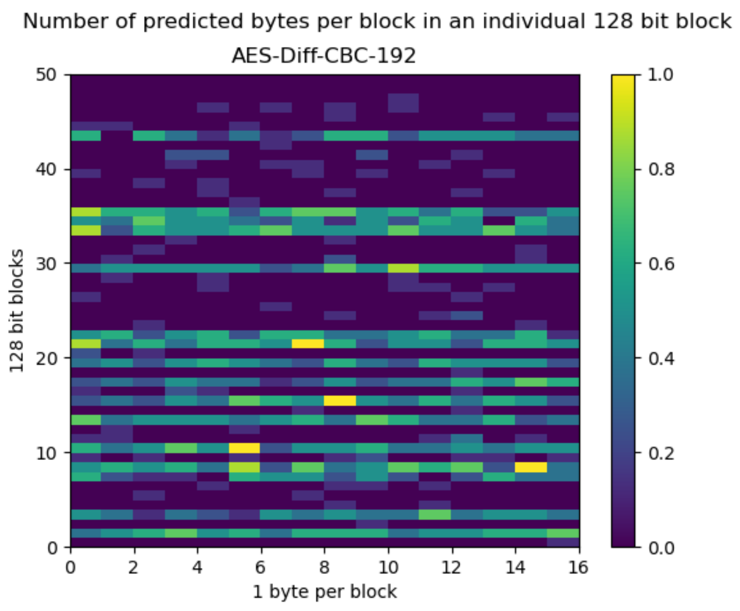

Fig. 5. Histogram representation of predicted bytes per block when using AES-CBC-192 with different IV

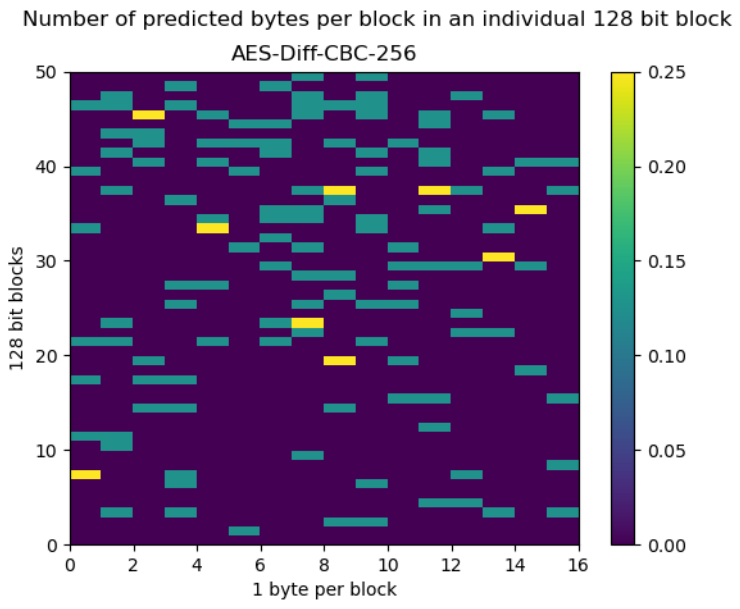

Fig. 6. Histogram representation of predicted bytes per block when using AES-CBC-256 with different IV

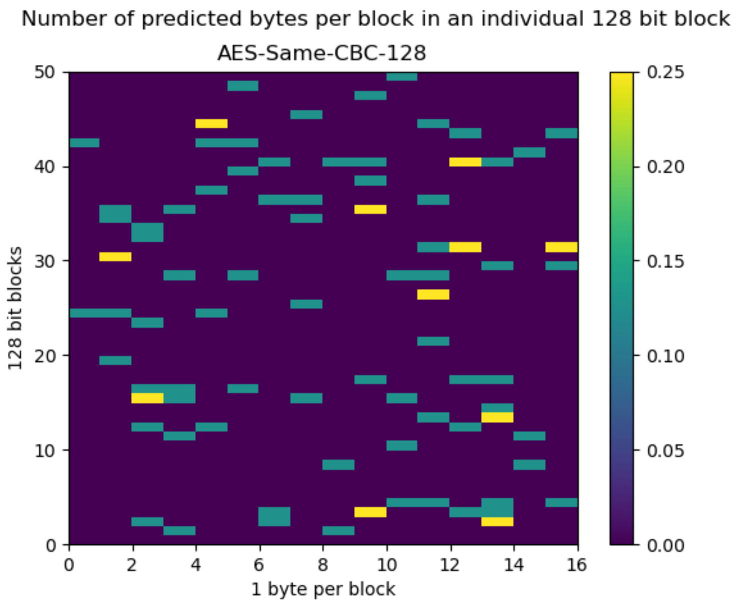

Fig. 7. Histogram representation of predicted bytes per block when using AES-CBC-128 with same IV

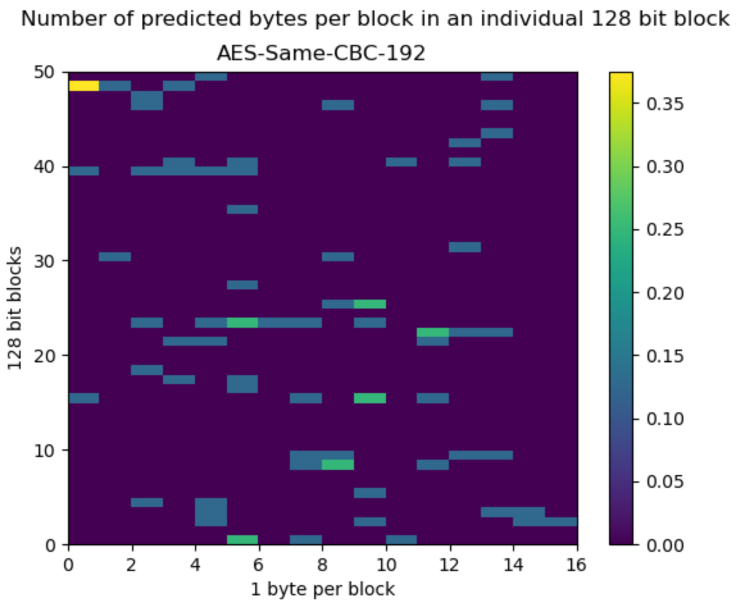

Fig. 8. Histogram representation of predicted bytes per block when using AES-CBC-192 with same IV

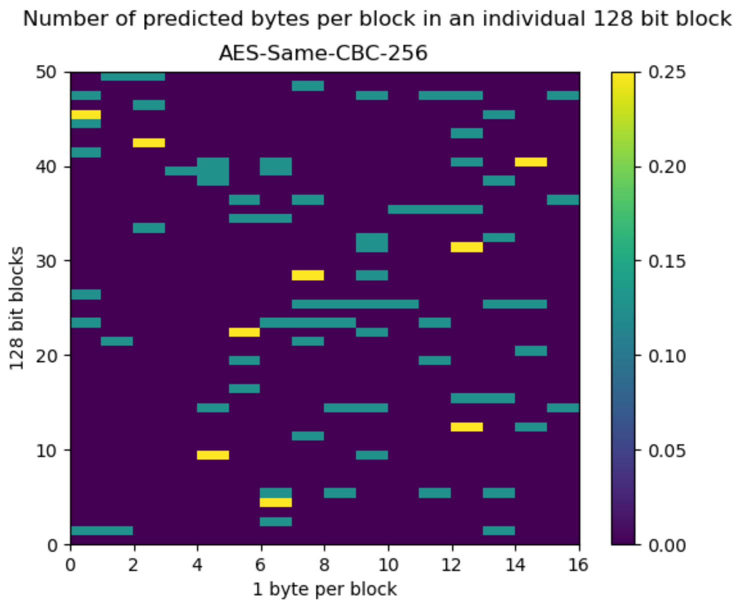

Fig. 9. Histogram representation of predicted bytes per block when using AES-CBC-256 with same IV

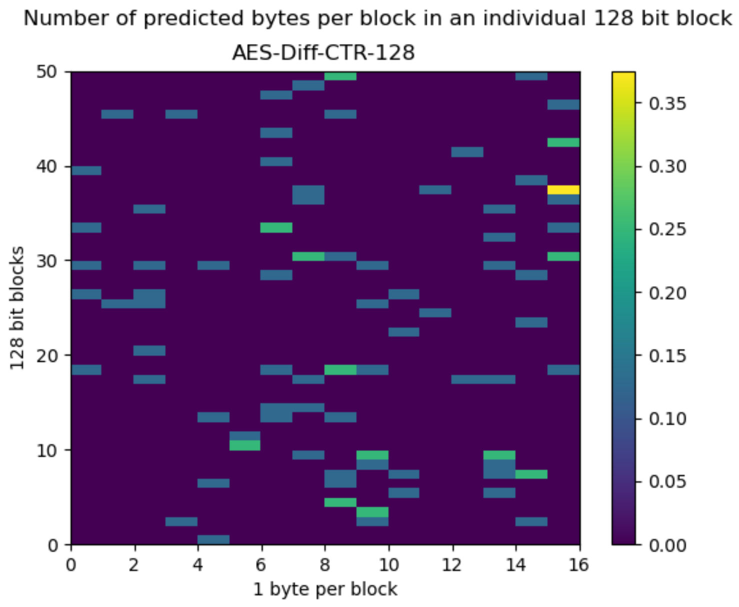

Fig. 10. Histogram representation of predicted bytes per block when using AES-CTR-128 with different IV

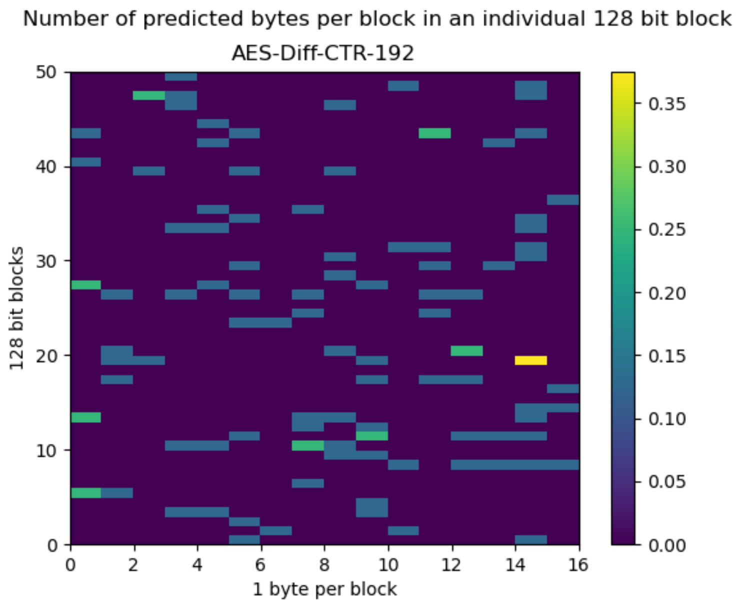

Fig. 11. Histogram representation of predicted bytes per block when using AES-CTR-192 with different IV

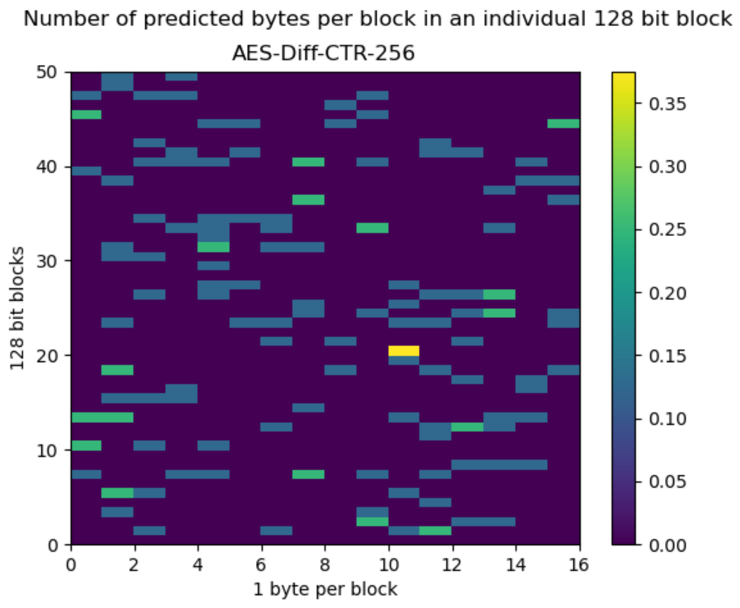

Fig. 12. Histogram representation of predicted bytes per block when using AES-CTR-256 with different IV

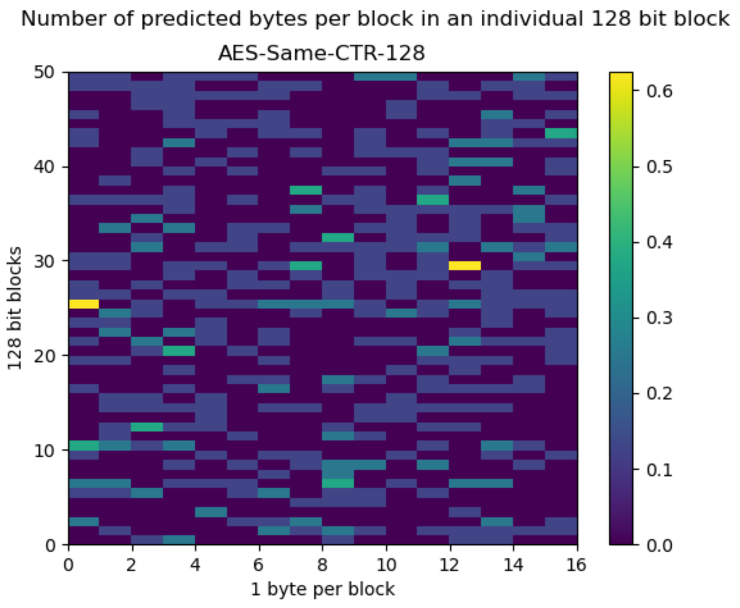

Fig. 13. Histogram representation of predicted bytes per block when using AES-CTR-128 with same IV

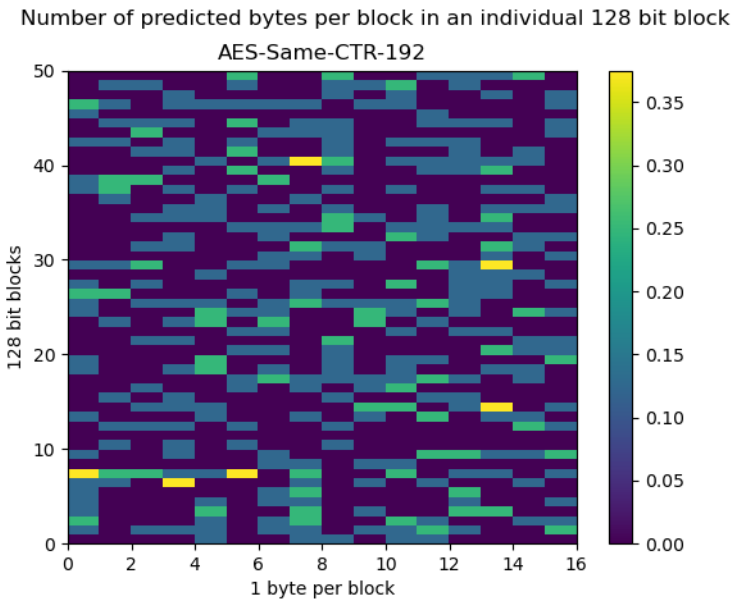

Fig. 14. Histogram representation of predicted bytes per block when using AES-CTR-192 with same IV

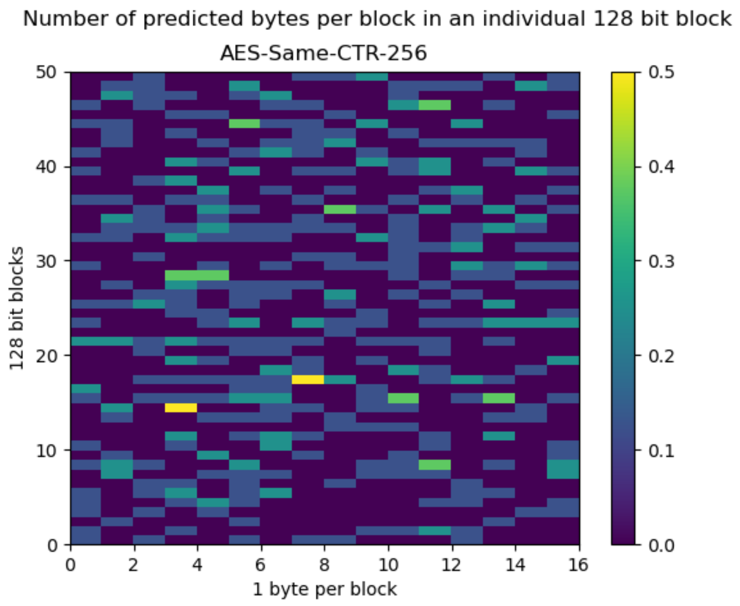

Fig. 15. Histogram representation of predicted bytes per block when using AES-CTR-256 with same IV

The histograms show that the best results are obtained when utilizing the AES-ECB mode of operation for encryption and that the results degrade as the complexity of the encryption method increases.

6 Conclusion

This research was created with the goal of observing how machine learning may be used to analyse algorithms from the subject of cryptography and act based on the obtained data. The experimental part includes a demonstration of AES with ECB, CBC, and Ctr modes of operation as plaintext-ciphertext attacks, as well as cryptanalysis using a neural network. To be more precise, we use the known-plaintext attack on AES with different modes of operation and different key sizes in order to see how the neural network will act and what kind of results can be obtained.

The proposed attack trains a neural network to decrypt ciphertext without knowing the encryption key. The attack successfully reduced the time and found almost 99% of the bits in the whole dataset. Something that is most relevant in cryptanalysis of the symmetric cipher is guessing two consecutive bytes, which can be seen from the experiments reaching more than 70% for some specific AES mode of operation.

Another point worth mentioning is that thanks to new technological developments, neural network training is becoming a simple operation that everyone can comprehend, and the time required is drastically reduced, according to prior records available online.

Different outcomes were produced while training a neural network to attack and give cryptanalysis of cryptographic methods. Having different algorithms and methods for experimenting with the results proves the power of ML and opens up a variety of new ideas for implementing and automating cryptanalysis.

To summarize, the reported results are typical of ML when deep learning models are built and trained in a shorter amount of time. Neural networks may learn from their mistakes by calculating and updating errors, and they can generate output that isn't constrained by the inputs they're given.

References

1. Advanced encryption standard (AES)., Federal Information Processing Standards Publication 197, (2001).
last accessed 2021/08/31
2. The Simon and Speck Families Of Lightweight Block Ciphers (PDF), ePrint, (2013).
3. Kaggle, "IMDB dataset (Sentiment analysis) in CSV format", https://www.kaggle.com/columbine/imdb-dataset-sentiment-analysis-in-csv-format, last accessed 2021/08/20.
4. E. Biham, and A. Shamir, Differential cryptanalysis of DES-like Cryptosystems, Journal of Cryptology, Vol.4, No.1, pp.3-72,1991.
5. M. Matsui, Linear cryptanalysis Method for DES Cipher, in T. Helleseth (Ed.) Advances in Cryptology – EUROCRYPT'93, LNCS 765, pages 386-397, Springer Verlag, 1994.
6. Yegnanarayana, B.: Artificial neural networks for pattern recognition. Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600036, India (1993).
7. Mohammed M. Alani, Neuro-Cryptanalysis of DES and Triple-DES, Department of Computing, Middle East College, Muscat, Sultanate of Oman, p.2-4, 2011.
8. M. Danziger and M. A. A. Henriques, "Improved cryptanalysis combining differential and artificial neural network schemes," in Proceedings of the International Telecommunications Symposium (ITS), pp. 1–5, Vienna, Austria, August 2014.
9. Hu, X. and Zhao, Y., Research on Plaintext Restoration of AES Based on Neural Network. Security and Communication Networks, (2018), pp.1-9.
10. Jayachandiran, K.: A Machine Learning Approach for Cryptanalysis. Department of Computer Science, Golisano College of Computing and Information Sciences, Rochester Institute of Technology, Rochester, NY 14586 (2018).
11. A. Gohr, "Improving attacks on round-reduced speck32/64 using deep learning," Advances in Cryptology-CRYPTO 2019, Springer, Berlin, Germany, pp. 150–179, 2019.
12. So, J., Deep Learning-Based Cryptanalysis of Lightweight Block Ciphers. Security and Communication Networks, (2020), pp.1-11.
13. Benamira A., Gerault D., Peyrin T., Tan Q.Q. (2021) A Deeper Look at Machine Learning-Based Cryptanalysis. In: Canteaut A., Standaert FX. (eds) Advances in Cryptology – EUROCRYPT 2021. EUROCRYPT 2021. Lecture Notes in Computer Science, vol 12696. Springer, Cham.

14. Chio, C., Freeman, D.: Machine Learning & Security: Protecting systems with data and algorithms. First Edition. O'Reilly Media, Incorporated, USA (2018).
15. Meduim, Towards Data Science, The mathematics behind deep learning, https://towardsdatascience.com/the-mathematics-behind-deep-learning-f6c35a0fe077, last accessed 2021/08/26.
16. Gjorgjievska Perusheska, M., Dimitrova, V., Popovska-Mitrovikj, A. and Andonov, S., Application of Machine Learning in Cryptanalysis Concerning Algorithms from Symmetric Cryptography. Lecture Notes in Networks and Systems, (2021), pp.885-903.
17. Keras, "The Functional API", https://keras.io/guides/functional_api/
18. Tensorflow, "Tensorflow Core", https://www.tensorflow.org/overview
19. Machine learning mastery, Difference Between a Batch and an Epoch in a Neural Network https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch, last accessed 2020/08/10.
20. Brownlee, J.: Better Deep Learning. Machine Learning Mastery (2018).
21. Scikit-learn, Precision-Recall, https://scikitlearn.org/0.15/auto_examples/plot_precision_recall.html, last accessed 2021/08/21.