Deep Reinforcement Learning for Profiled Side Channel Attacks

Applying Proximal Policy Optimization for Point of Interest Selection

Universiteit Leiden
The Netherlands

Mick G. D. Remmerswaal
1221620

Universiteit Leiden

Leiden University

Master Thesis

Deep Reinforcement Learning for Profiled Side-Channel Attacks

Applying Proximal Policy Optimization for Point Of Interest Selection

Author:

Mick G.D. Remmerswaal 1221620

August 18, 2022

Universiteit Leiden

Acknowledgments

Dear reader, before you lies the culmination of several months of hard work. This thesis was written for my graduation from the MSc. Computer Science: Artificial Intelligence.
During this period, I have learned a lot of new and exciting things, and certain people have helped me a lot along the way. First, I want to thank Nele Mentens and Nusa Zidaric for being my supervisors for this thesis and providing me with feedback, comments, and helpful tips. Second, I want to extend my gratitude to Ernst Bovelander and Lex Schoonen for allowing me to do my graduation at SGG Brightsight. Working at Brightsight for the duration of my graduation was a pleasure. Finally, I also want to thank Lichao Wu and Sebastien Tiran for their continuous help during my graduation. I learned a lot from you and could not have done this without you.
During my studies, I met several people, but two of them stand out among them, Denis Krol and Eirini Kousathana. Thank you guys so much for the many fun days that we shared. Many thanks to all my other friend for having the patience when I wasn't there, while still being there for me when I needed it.
I am grateful for the unwavering loyalty of my family, Mom & Dad, Uyen & Fried, Evan & Shane, Jelle & Sandra, Tristan & Anouk, Mai & John, Tina & Nikos. Thank you all for the endless support you have given me over the years. Ik zei toch dat het goed kwam!

Mick G.D. Remmerswaal
Leiden, July 2022

Universiteit Leiden

Abstract

Under the worst-case attack assumption, profiling side-channel analysis is recognized as one of the most potent attack methods for cryptographic implementations. The success of such attacks relies on the effectiveness of the profiling model in retrieving the leakage information. Previous research indicates that a proper selection of the input features could significantly increase the attack performance. However, due to the countermeasures and technology nodes' advances, such features become more difficult to extract with the conventional approaches. In extreme cases, manual inspection has to be performed, where extensive device knowledge is needed. To this day, an optimal feature selection method has not yet been found. This thesis proposes an automatic framework based on Proximal Policy Optimization to find, select and scale down features. The input features are first grouped in small regions. The best candidates selected by the framework are further scaled down with an online-optimized Dimensionality Reduction Neural Network. Finally, the framework rewards the performance of these features with the results of a Template Attack. Based on the experimental results, the proposed framework is able to extract features that lead to near state-of-the-art performance on several commonly used datasets.

Universiteit Leiden

Contents

Universiteit Leiden

List of Figures

Universiteit Leiden

List of Tables

Acronyms

AE AutoEncoder. 18

AES Advanced Encryption Standard. v, vii, 1–3, 6, 34, 36–38, 40–44, 46, 48, 51

ANN Artificial Neural Network. v, 6–8, 12, 25, 26, 44, 54

CHES Conference on Cryptographic Hardware and Embedded Systems. 34

CNN Convolutional Neural Network. 8, 9, 18, 29, 30, 55

DL Deep Learning. 1, 2, 6, 7, 18, 19, 25, 27

DPA Differential Power Analysis. 1, 4

DQN Deep Q-Network. 12

DRL Deep Reinforcement Learning. 12

GE Guessing Entropy. 2, 6, 36, 38, 50, 56, 57

HD Hamming Distance. 6

HW Hamming Weight. 6, 42

IAM Invalid Action Masking. 31, 32

ID Identity. 6

KL Kullback-Leibler. 15–17

LDA Linear Discriminant Analysis. v, vii, 24, 25, 42, 43, 45, 54, 55, 57

LMM Lagrange Multiplier Method. 23

LSTM Long Short-Term Memory. 18

MDP Markov Decision Process. 10

ML Machine Learning. 1, 2, 6, 7

MLP Multi-Layer Perceptron. 8, 18, 30

NIST National Institute of Standards and Technology. 3

NN Neural Network. 18

PCA Principal Component Analysis. v, vii, 22, 24, 25, 40–42, 45, 54, 55, 57

POI Points Of Interest. vii, 1, 2, 5, 18–22, 24–27, 29, 35, 37–47, 54–57

PPO Proximal Policy Optimization. vii, 17, 30, 32, 54

RELU Rectified Linear Unit. 7

RL Reinforcement Learning. v, 1, 2, 7, 10–12, 15, 18, 19, 28, 56, 57

RSA Rivest-Shamir-Adleman. v, 3, 4

SCA Side-Channel Analysis. 1–3, 5, 6, 18–21

SCARL Side-Channel Analysis with Reinforcement Learning. 19

SELU Scaled Exponential Linear Unit. 7

SNR Signal-to-Noise Ratio. v, vii, 19, 21, 22, 28, 38–40, 45, 49, 51, 52, 54–57

SOD Sum of Differences. 20

SOSD Sum Of Squared Differences. v, 20, 21

SOST Sum Of Squared T-Differences. v, vii, 20, 21, 36–38, 45, 54–57

SPA Simple Power Analysis. v, 1, 3, 4

SR Succes Rate. 6

TA Template Attack. 1, 2, 5, 18–20, 27, 29, 35, 46, 50, 56, 57

TRPO Trust Region Policy Optimization. v, 15–17

1 Introduction

Since the introduction of circuitry in plastic in 1960 by two German engineers, more smart cards have been in use. From authorization cards and debit cards to the OV-chipkaart, embedded circuitry in plastic has become very common. Since these smart cards can contain sensitive information, they often carry cryptoprocessors to ensure the security of this sensitive information. However, where there is sensitive information, there are bad actors and criminals after it.
Since the introduction of the Advanced Encryption Standard (AES) [12], brute-force hacking, the act of trying every possible combination of password or secret key, is not possible in any feasible time [53]. To stay ahead of the curve, researchers tried other methods and found that devices unintendedly leak information about the processing of the data. These unintended leakages can consist of the power consumed by the device, the electromagnetic emission from the device, or the timing it takes to execute operations.
These so-called side-channels gave rise to Side-Channel Analysis (SCA) [23]. In the beginning SCA was based on visual analysis like Simple Power Analysis (SPA) [23], later more statistical approaches were introduced, such as Differential Power Analysis (DPA) [23] and the Template Attack (TA) [9]. Especially the Template Attack proved to be a very well-performing attack strategy.
The Template Attack contains two phases: a profiling phase and an attack phase. In the profiling phase, the attacker creates profiles or templates of the leakage information based on a similar or identical device under the attacker's control. These templates are characterized by a multivariate normal distribution built by a mean vector and a covariance matrix, which is the optimal way to describe the templates [32]. The Template Attack (TA) [9] gave way to more advanced techniques for profiling attacks. Machine Learning (ML) and Deep Learning (DL) have been used for SCA since its introduction [37][6][17][18][31].
These techniques proved very strong, and the field of SCA shifted its attention to these techniques. As a result, TA became more of a baseline to compare to than the preferred method of attack, even though they are the optimal method from an information-theoretic point of view [9]. The main drawback of TA is their reliance on proper preprocessing of the measurements because they often contain redundant or uninformative features [37].
This preprocessing is done with a method called Points Of Interest (POI) selection. POI selection tries to capture the most relevant features from within the measurements and uses these to mount an attack. Because of the shift from TA to ML and DL methods, research in POI selection has diminished. Although the research has diminished, a few researchers have proved that with proper POI selection, TA can still outperform ML and DL methods [26][56].
The main issue with proper POI selection is the need for device knowledge and, in extreme cases, the manual selection of POIs. To this day, an optimal strategy for POI selection is not known [32]. Finding an optimal strategy to reach a goal is one of the strengths of Reinforcement Learning (RL). RL has already been employed in the field of SCA and produced comparable state-of-the-art performance when optimizing network architectures for Deep Learning attacks [41].
Knowing that RL can be employed to search for optimal network architectures, it may also be employed to find an optimal strategy for identifying optimal Points Of Interest. With this in mind, two research questions will be answered in this work.

Research Questions

1. What are the current Points Of Interest selection techniques, and which one performs

the best, according to the Guessing Entropy?

1.a What are the current state-of-the-art techniques used to select Points Of Interest?

1.b What are the most prominent pros and cons of each method?

2. Using Reinforcement Learning is it possible to create a Points Of Interest selection method?

2.a With the help of Reinforcement Learning techniques, is it possible to identify possible points of interest from within a trace?

2.b Is it possible to create a Dimensionality Reduction Neural Network with the help of Reinforcement Learning?

3. Can Reinforcement Learning improve Points Of Interest selection in comparison to the current state-of-the-art, measured in the Guessing Entropy?

Sub-question 1.a is answered in Section 4, where an introduction to five state-of-the-art methods is given. Sub-question 1.b is answered in Section 8.6, where each method is benchmarked on several commonly used datasets.
Research question 2.a and 2.b are answered in Section 5, where a proposed framework is introduced which automatically finds, selects, and scales down Points Of Interest.
Finally, research question 3 is answered in Section 8, where the proposed framework is benchmarked on several commonly used datasets.
This work has several sections, it begins with some background information about the Advanced Encryption Standard, Side-Channel Analysis, Machine Learning, Deep Learning and Reinforcement Learning in Section 2. Then Section 3 outlines related work in SCA with a focus on Points Of Interest selection. The current methods used are outlined in Section 4 and the proposed RL-driven framework is introduced in Section 5. Experiments are run to benchmark the proposed framework and are outlined in Section 7. The results and discussion of these experiments can be found in Section 8 and finally a conclusion and a discussion about future work is made in Section 9.

2 Background

This section outlines the background information research in this document. It starts with the basics of Cryptography and an introduction to the Advanced Encryption Standard. Then an introduction to Side-Channel Analysis, with an emphasis on Template Attacks and Points Of Interest selection. Eventually, this section ends with information about various techniques of Machine Learning, Deep Learning and Reinforcement Learning.

2.1 Cryptography

Cryptography has its roots in the Greek words *kryptos*, secret, and *graphein*, to write, and the act of writing secrets can be traced back to ancient Roman and Greek times. Modern cryptography can be distinguished between two methods to encrypt and decrypt data, symmetric and asymmetric cryptography. The data that has to be encrypted is called the plaintext and the data that is already encrypted is called the ciphertext. Symmetric cryptography uses one secret, or private, key shared by two entities exchanging secret data. The key acts as both the encryption and decryption key for the data. An example of a symmetric cryptographic algorithm is the Advanced Encryption Standard (AES) [12]. Asymmetric cryptography uses

two keys, a private key and a public key. The public key, as the name suggests, is publicly available. The public key encrypts the data, while the private key decrypts it. An example of an asymmetric cryptographic algorithm is the Rivest-Shamir-Adleman (RSA) algorithm [44].

2.1.1 Advanced Encryption Standard

The Advanced Encryption Standard (AES) was dubbed as such by the National Institute of Standards and Technology (NIST) in 2001, its official name is the Rijndael Block Cipher [12]. The AES algorithm comes in three variants describing the key length used, AES-128, AES-192, and AES-256. The cipher encrypts or decrypts blocks of 16 bytes every round and does so for 10, 12, or 14 rounds, depending on the variant. The 16 bytes are represented in a 4×4 matrix. AES is defined by four round operations shown in Table 1.

Operation	Description
SubBytes	Each byte of the state is substituted with another byte using a non-linear lookup table called the S-Box.
ShiftRows	Rows 2,3 and 4 in the state are shifted to the left by 1, 2 and 3 bytes respectively.
MixColumns	The columns of the matrix are mixed using a linear transformation
AddRoundKey	The 16-byte state is XOR'ed with the current 16-byte round key, each round a different round key is computed according to the keyschedule.

Table 1: The four round operations of AES with their respective descriptions.

2.2 Side-Channel Analysis

With the implementation of cryptographic algorithms on devices, a new security risk was introduced. Often these devices are embedded systems, such as smart cards [10]. Since these devices do not have access to enough computing power to hold the most advanced cryptographic algorithms, the implementation of the device itself is tested as well. Breaking such a device is akin to extracting the secret key the device is using. Over the years, several attacks were devised to assist with recovering the secret key from cryptographic devices. Two attack categories can be distinguished, active and passive attacks. In active attacks, an attacker tampers with the device, its inputs, or its environment. In passive attacks the attacker analyzes the physical properties of the device and exploits any faults within those systems. One of the most common passive attacks is Side-Channel Analysis (SCA).
SCA is the act of exploiting leakages of information from unintended channels. The most analyzed side-channels are power consumption, electromagnetic emission, and operation execution time. The attacks are, respectively, called Power Analysis Attacks [23], Electromagnetic Attacks [39] and Timing Attacks [13].

2.2.1 Non-profiled Attacks

Non-profiled attacks are the most rudimentary form of Side-Channel Analysis. The most common non-profiled attack is the Simple Power Analysis (SPA) [23]. In this attack, the attacker measures the power consumption, often called traces, of a device under attack while the encryption process is running. SPA then exploits key-dependent operational differences seen within traces and extracts the secret key. An example of SPA on the RSA cipher is shown in Figure 1.

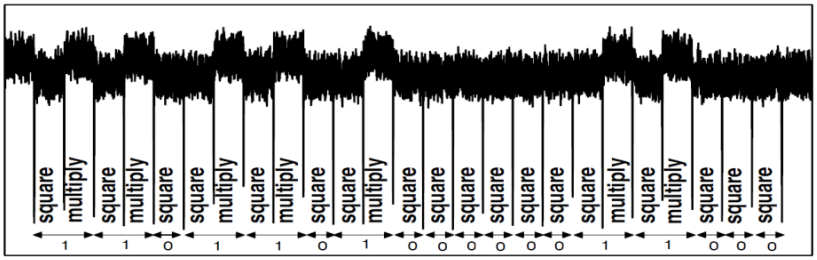


Figure 1: An example of SPA on the RSA cipher. Since the secret key dictates the operations used in the algorithm, it is easy to exploit by measuring the power consumption [27].

Another popular non-profiled attack is Differential Power Analysis (DPA) [23]. DPA is the technique of interpreting the dependency of data encrypted or decrypted on the power consumption. Since the secret key is used during encryption or decryption, the data used can influence in conjunction with the key can influence the power consumption. Where SPA mainly focuses on the dependency of the operations of a cryptographic algorithm on power consumption, DPA focuses on the data being fed into a cryptographic algorithm. In contrast with SPA, DPA requires a larger amount of measurements
DPA constitutes five steps [32]. The first step consists of selecting a point in the algorithm to attack. This point is an intermediate result of encryption or decryption and can be modeled as the function $f(d_i, k_i)$. Here d_i denotes i-th byte of the plain- or ciphertext and k_i the i-th byte of the secret key, often called a subkey.
The second step is to measure the power consumption while the device encrypts or decrypts $\mathbf{D}$ different known data blocks. For each d_i, from $\mathbf{d}$, a power trace $\mathbf{t}'_i = (t_{i,1}, ..., t_{i,T})$ is recorded, where T denotes the length of each trace. Traces will be collected in $\mathbf{T}$ for further use.
Step three consists of creating hypothetical intermediate values from the selected data blocks $\mathbf{d}$ and every possible key guess in $\mathbf{k}$. The result will be matrix $\mathbf{V}$ consisting of $v_{i,j} = f(d_i, k_j)$. Since $\mathbf{k}$ consists of every possible key guess, it also contains k^*, the key used by the device.
Step four is creating a mapping from these hypothetical intermediate values $\mathbf{V}$ to a set of hypothetical power consumptions $\mathbf{H}$. This mapping is done with a Hypothetical Leakage Model, $h_{i,j} = \mathrm{LM}(v_{i,j})$. The leakage models used in this thesis are explained in a later section.
The fifth and final step of DPA is to compare the hypothetical power consumptions $\mathbf{H}$ with the actual power consumptions $\mathbf{T}$. This will result in matrix $\mathbf{R}$ with

$$r_{i,j} = \frac{\sum_{d=1}^{D}((h_{d,i} - \bar{h}_i) * (t_{d,j} - \bar{t}_j))}{\sqrt{\sum_{d=1}^{D}(h_{d,i} - \bar{h}_i)^2 * \sum_{d=1}^{D}(t_{d,j} - \bar{t}_j)^2}} \tag{1}$$

Here $r_{i,j}$ denotes the correlation coefficient between column $\bar{h}_i$ and column $\bar{t}_j$. Since it is known which values are used to create $\bar{h}_i$, namely $h_{i,j} = \mathrm{LM}(v_{i,j})$ and $v_{i,j} = f(d_i, k_j)$, key k_j must be the key k^* used by the device.

2.2.2 Profiled Attacks

Profiled attacks introduce an extra phase in the attack, the profiling phase. In this phase, traces are measured from a copy of the device under attack and, based on these traces, profiles are made. These profiles can then be applied to the device under attack for the attacking phase.

Template attacks (TA), as proposed in [9], are the most potent profiling attack from an information-theoretic point of view. These attacks take into account that traces can be fully defined according to a multivariate normal distribution with a mean vector and a covariance matrix $(\bar{m}, \mathbf{C})$ [32], henceforth referred to as a template. Since a device is available to create the templates, an attacker is in control of the parameters used for the template, namely the plain- or ciphertext d_i and the key k_i. These templates are created for a specific combination and can be grouped. This way the attacker obtains a template for each group $h_{d_i,k_i} = (\bar{m}, \mathbf{C})_{d_i,k_i}$.
Given a key k_j and a trace $\bar{t}_i$ the conditional probability $p(k_j|\bar{t}_i)$ can be calculated using Bayes Theorem, which is shown in Equation 2, and the Law of Total Probability, as shown in Equation 3.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

$$P(A) = P(A|B)P(B) + P(A|1-B)P(1-B) \tag{3}$$

Using the prior probability $p(k_l)$ and the probability $p(\bar{t}_i|k_l)$ for every key guess $k_1, ..., k_K$ in the search space, Equation 2 transforms into its TA equivalent shown in Equation 4. An extension to multiple traces is shown Equation 5.

$$p(k_j|\bar{t}_i) = \frac{p(\bar{t}_i|k_j)p(k_j)}{\sum_{l=1}^{K}(p(\bar{t}_i|k_l)p(k_l))} \tag{4}$$

$$p(k_j|\mathbf{T}) = \frac{(\prod_{i=1}^{T} p(\bar{t}_i|k_j))p(k_j)}{\sum_{l=1}^{K}((\prod_{i=1}^{T} p(\bar{t}_i|k_l))p(k_l))} \tag{5}$$

The probability $p(\bar{t}_i|k_l)$ can be transformed to $p(\bar{t}_i|h_{d_l,k_l})$ since the attacker has access to the device. Using the template $h_{d_l,k_l} = (\bar{m}, \mathbf{C})_{d_l,k_l}$ the probability can be calculated Equation 6.

$$p(\bar{t}|(\bar{m}, \mathbf{C})_{d_i,k_i}) = \frac{exp(-\frac{1}{2}(\bar{t}-\bar{m})^T\mathbf{C}^{-1}(\bar{t}-\bar{m}))}{\sqrt{(2\pi)^T det(\mathbf{C})}} \tag{6}$$

This is done for each template over multiple traces. Essentially, the key guess k_l with the highest probability must be k^*.

2.2.3 Point Of Interest Selection

Under the worst-case attack assumption, profiling Side-Channel Analysis is recognized as one of the most potent attack methods for cryptographic implementations. The success of such attacks relies on the effectiveness of the profiling stage and its ability to properly retrieve leakage information from the traces. Therefore the profiling stage commonly undergoes a preprocessing step called Points Of Interest Selection.
Points Of Interest (POI) selection is the method of distinguishing between relevant, irrelevant or redundant features within the traces [37]. Three approaches to POI Selection can be made

- Feature Selection methods,
- Dimensionality Reduction methods, and

- Deep Learning based methods.

Feature Selection methods create a subset of the input features and uses these as the attack features. Dimensionality Reduction methods transform the original features to a new subspace of features and uses the subspace for the attack. Deep Learning methods uses Artificial Neural Networks to transform the features into a new set of features.

2.2.4 Hypothetical Leakage Models

Inferring the correct key or correct partial key is done by approximating the leakage. This is done by modeling a hypothetical leakage model. The idea behind this is that while the cryptographic algorithm is executed on the chip, there will be a correlation between the intermediate values and the power consumption or electromagnetic emission.
Commonly three leakage models are defined, the Hamming Weight (HW) model, the Hamming Distance (HD) model and the Identity (ID) model.
The HW model, models the assumption that a 1-bit in the intermediate value correlates with the leaking side-channel. The HD model models the assumption that the bit-flips between an intermediate value before a certain operation and after, correlates with the leaking side-channel. For the ID model, the assumption that the intermediate values directly correlates with the leaking side-channel is modelled.
The HW is defined as the sum of the bits in a binary number that equal 1. In the case of AES, each key byte is a number between 0 and 2^8, e.g. HW(84) = HW(01010100) = 3.
The HD is defined as the amount of bits that flipped between two binary numbers. For example, a=10100100 and b=01110110, then the HD(a,b)=$\frac{10100100}{01110110} = 4$
The ID model does nothing else than use the intermediate values themselves, e.g. ID(128) = 128.

2.3 Side-Channel Analysis Metrics

Two metrics are defined to compare different SCA attacks on their performance, the Guessing Entropy (GE) and the Succes Rate (SR) [50]. Both of these metrics are predicated on a guess vector $\mathbf{g} = [g_1, ..., g_{|K|}]$. Here $|K|$ denotes the search space of the key, in the case of AES $|K| = 256$. The guess vector $\mathbf{g}$ is an ordering of how likely the attack guesses key values. Only the Guessing Entropy is used in this thesis.

2.3.1 Guessing Entropy

The Guessing Entropy or Guess Entropy is the average ranking of the correct key k^* among the other key guesses, where the averaging is done over multiple attacks. The length of the ranking vector denotes the amount of traces used for the attack. An attack is said to break the target if it achieves a GE of 0. If the target of the attack is not the full key but only one byte, it is commonly referred to as Partial Guessing Entropy. Often these terms are interchangeable, and this will be done in this work as well.

2.4 Machine Learning

Machine Learning (ML) is one of the many applications of Artificial Intelligence. Its emergence can be traced back to the year 1943 in the paper by McCoullough and Pitts [34]. In this paper, the authors mathematically map out how the human cognitive works.

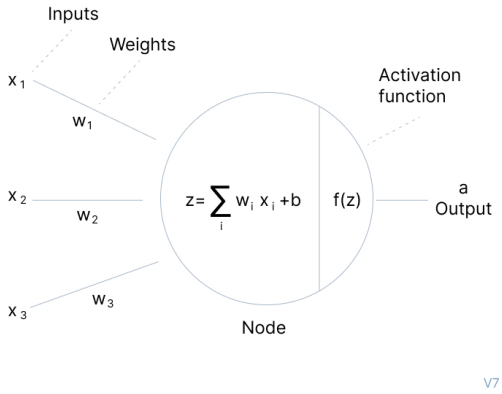


Figure 2: A neuron with three inputs and one output [38].

A few years later in 1950, Alan Turing devised a test to measure the amount of intelligence a system has. Using questions and answers, if the machine is able to convince a human examiner that it is a human, the test is successful, and the machine is deemed intelligent.
Generally, Machine Learning is be divided into categories: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.
In Supervised Learning, the data has pre-assigned labels. These labels often denote a class to which the data belong to. For example, an image of a dog having the class *dog*. Supervised Learning is the task of distinguishing features from the data that makes them fall into a particular class or label.
Unsupervised Learning is used on data that does not have assigned labels. The general idea is that techniques learn from patterns within the data itself.
Reinforcement Learning will be discussed in Section 2.6.

2.5 Deep Learning

Deep Learning (DL) refers to the extension of the work by McCoullough and Pitts and their mathematical approach to map the human cognitive. It is a form of ML that creates a so-called Artificial Neural Network (ANN). ANNs are built by consecutive layers of neurons and activation functions. Neurons consist of n input connections $\bar{x} = [x_1, ..., x_n]$, weights $\bar{w} = [w_1, ..., w_n]$, a bias b and an activation function $f(z)$. The neuron calculates its outputs by summing up the element-wise multiplication of x and w and the bias. The result runs through the activation function and is output to further layers of neurons. This can be defined with the following equations:

$$z = \sum_{i}^{n} w_i x_i + b \tag{7}$$

$$a = f(z) \tag{8}$$

A graphical representation of a neuron with $n = 3$ inputs can be found in Figure 2.
Activation functions are used to introduce non-linearity into the neural networks. The most common activation functions are the Sigmoid function (Equation 9), the Rectified Linear Unit (RELU) (Equation 10), the Scaled Exponential Linear Unit (SELU) (Equation 11) and

the Softmax function (Equation 12).

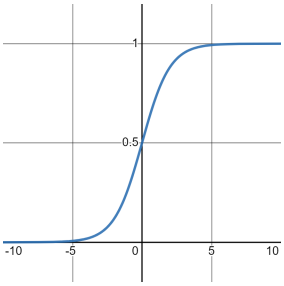
$$f(z) = \frac{1}{1 + e^{-z}} \tag{9}$$

$$f(z) = max(0, x) \tag{10}$$

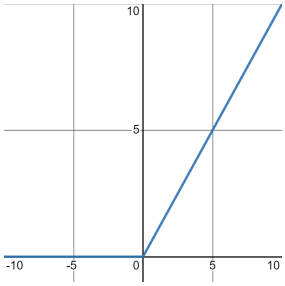
$$f(\alpha, z) = \lambda \begin{cases} \alpha(e^z - 1) & \text{if } z \leq 0 \\ z & \text{if } z > 0 \end{cases} \tag{11}$$

$$f(z_i) = \frac{exp(z_i)}{\sum_j^n (exp(z_j))} \tag{12}$$

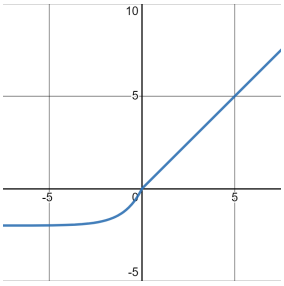
Graphical representations of each activation functions are found in Figure 3.



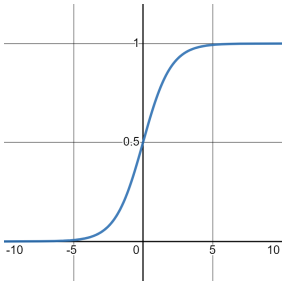
(a) The Sigmoid Function



(b) The ReLU function



(c) The SeLU function



(d) The Softmax Function

Figure 3: Graphical representations of each activation function [4].

2.5.1 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is an ANN built up by multiple layers of multiple neurons. Each MLP begins with one input layer followed by several hidden layers and finishes with one output layer. For example, a simple MLP consisting of four layers can be seen in Figure 4.
Since every neuron is connected to every neuron in the next layer, they are known as Fully Connected Layers. MLPs are mostly used for classification problems, for example breast cancer risk estimation [3], automatic weld defect classification [28] and identifying and classifying of sonar targets [22].

2.5.2 Convolutional Neural Networks

A CNN is a type of ANN specialized for image analysis. Its main feature is the Convolutional Layer. This layer also multiplies weights by its inputs but instead only takes a small area

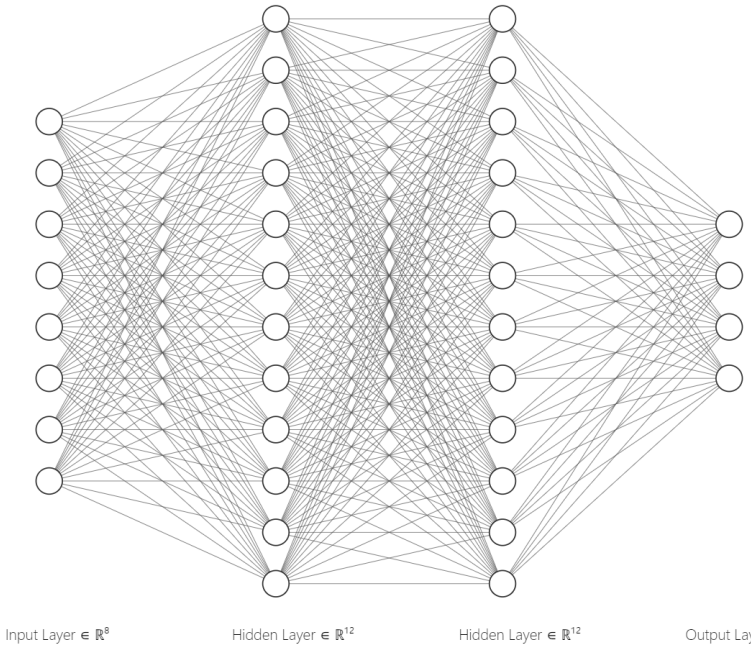
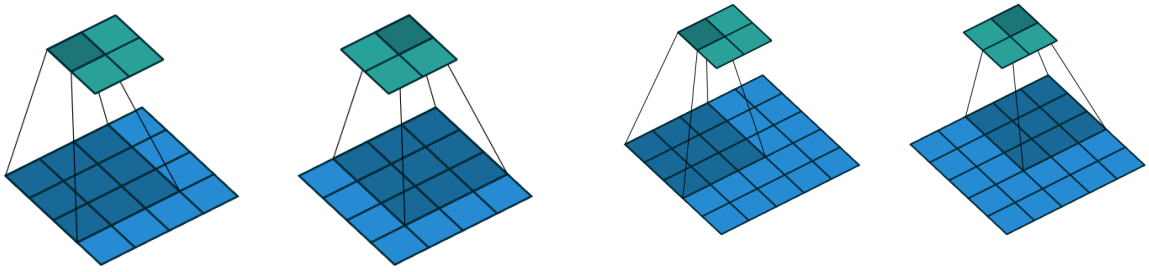


Figure 4: A Multi-Layer Perceptron consisting of an input layer with eight neurons, two hidden layers of 12 neurons and a four neuron output layer.

into account instead of the entire input vector. The kernel size defines this area. Next to the kernel size, a Convolutional Layer has two more important parameters that can be set, the stride and the padding. The stride defines the step the layer takes before applying the kernel again to an area. The padding is used to pad areas if the kernel is larger than the current input and is mainly used to retain the original size.
Figure 5a gives an example of a 3x3 kernel and a stride of 1. Figure 5b gives an idea of how the stride can influence the outcome of the output dimension.



(a) A 3x3 kernel with a stride of 1 moves over a 4x4 image.

(b) A 3x3 kernel with a stride of 2 moves over a 5x5 image.

Figure 5: A example how Convolutional layers work [14].

Next to Convolutional Layers, Pooling layers are an important part of CNN. The two most used Pooling layers are Average Pooling and Max Pooling. As the name suggests, Average Pooling takes the average of the designated area, while Max Pooling takes the maximum of a designated area.

2.6 Reinforcement Learning

Reinforcement Learning (RL) [51] is the act of learning through taking actions from observations made within an environment while being given an increasing reward for correct actions taken. It is a class of learning algorithms that learns unsupervised but is not actively seeking patterns in the data presented.
In Reinforcement Learning, an Agent-Environment interaction is built around the problem to be solved, and a graphical representation can be found in Figure 6.

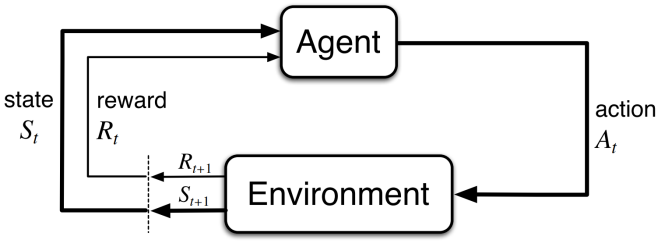


Figure 6: A graphical representation of a generic RL environment [51].

The Actor or Agent makes observations from the Environment called states. On time step t, the Agent receives state S_t from the Environment and acts by following a certain Policy π or transition probability T, by taking action A_t. The Environment takes the taken action into account and according to a reward function $f(S_t, A_t)$ gives a reward R_t and returns a new state S_{t+1}. When the Agent reaches a predetermined terminal state, the Environment sends a done signal to the Agent. From there on out a new sequence of states, actions and rewards begins.
These sequential problems can be formalized as a Markov Decision Process (MDP) [30] and form the basis for a mathematical approach to RL. The definition of an MDP for RL is done by forming the 5-tuple (S, A, T_a, R_a, γ). Where S defines a finite set of legal states of the Environment where s_0 denotes the initial state. A defines the set of all actions that can be taken in a state. Important to note is that this can change based on the state. If this is the case, A becomes A_s. The transition probability is modelled as $T_a(s, s') = \Pr(s' = s_{t+1}|s_t = s, a_t = a)$ and is the probability that, given a state s and taking action a, the next state is s'. $R_a(s, s')$ is the reward after transitioning from state s to s' after taking action a. Finally, γ represents the discount factor, a value that dictates the difference in the present and future rewards.
As mentioned before, the Agent takes actions according to some transition probability T and is the main focus of the learning process. This function is a conditional probability distribution. An example is given in Table 2, which dictates the mapping from a state S to a probability distribution over all actions in S:

$$T = S \rightarrow p(A) \tag{13}$$

Reinforcement learning algorithms have several defining attributes. They can be model-based or model-free, meaning they either know or do not know the exact transition probability from one state to another. They can be on-policy or off-policy, meaning that they either calculate the discounted future rewards based on the state-action pairs following the current policy or the greedy approach, where all actions are taken into account. The final defining attribute is if the algorithm is value-based, where the algorithm approximates an optimal value function that maps state-action pairs to reward values, or policy-based, where the algorithm directly

s	$\pi(a = up\|s)$	$\pi(a = down\|s)$	$\pi(a = left\|s)$	$\pi(a = right\|s)$
1	0.1	0.1	0.0	0.8
2	0.5	0.3	0.2	0.0
3	0.0	1.0	0.0	0.0
4	0.7	0.1	0.2	0.0

Table 2: An example of a policy on a discrete state and action space, which can be stored as a table.

optimizes the policy and approximates the optimal policy that way.

2.6.1 Q-Learning

One of the first Value-based Reinforcement Learning algorithms is Q-learning [54]. It is a model-free, off-policy, value-based RL algorithm. It instantiates a lookup table, the Q-table, filled with values for each state-action pair. The Agent does not know the exact transition probability T and therefore cannot take actions based on T. To reflect this, T is often interchanged with a policy π and the Agent takes actions based on this policy π. At each time step, the algorithm looks up the state inside the Q-table and takes the action with the highest value.
In Q-learning the policy π is defined as taking the action corresponding with the maximum Q-value. To ensure proper exploration of the environment, an action sampling method is used called ϵ-greedy sampling, as depicted in Equation 14. The method defines ϵ as the probability of taking a random action from the action space. Where $\epsilon = 1.0$ is akin to pure random action sampling and $\epsilon = 0.0$ is akin to deterministically taking the action with the highest Q-value.

$$a_s = \begin{cases} \underset{a \in A}{\text{argmax}}\ Q(s,a) & 1-\epsilon \\ \text{random } a & \epsilon \end{cases} \tag{14}$$

After taking an action and receiving rewards, the Q-table is updated via the Bellman Equation [5]

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]. \tag{15}$$

In essence, the Q-value associated with a state-action pair (s_t, a_t) is updated with future discounted rewards when taking the next state s_{t+1} into account. Here γ, with $0 \leq \gamma \leq 1$, denotes the discount factor and α, with $0 \leq \alpha \leq 1$, denotes the learning rate.

2.7 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) is the class of RL algorithms that make use of Artificial Neural Networks.

2.7.1 Q-Networks

The work by Mnih et al. [35] describes the development of the Deep Q-Network (DQN) algorithm in their efforts to create a single algorithm capable of solving a wide range of challenging tasks. Instead of a Q-table that stores the values that map states to actions, each state runs through a neural network and outputs a value for each action. Then according to the same ϵ-greedy strategy, samples an action.
To stabilize the network's learning and emulate the learning of past experiences in humans, the authors employ Experience Replay [29]. The algorithm initializes a data set D of past experiences and at each time step adds $e_t = (s_t, a_t, r_t, s_{t+1})$ to D. Then during the application of the Q-learning updates, a batch of experience is sampled at random from D and used for the updates.
Each timestep t the network is trained by minimizing the loss function L_t of the network with respect to the weights θ, with the following equation

$$L_t(\theta_t) = \mathbb{E}_{s,a\sim p(\cdot)}[(y_t - Q(s,a;\theta_t))^2] \tag{16}$$

with

$$y_t = \begin{cases} r_t & \text{for terminal } s_{t+1} \\ r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta_{t-1}) & \text{for non-terminal } s_{t+1} \end{cases} \tag{17}$$

Here the expectation to be minimized is the squared difference in future discounted rewards y_t, which are calculated with the previous parameters θ_{t-1}, and the current rewards $Q(s,a;\theta_t)$. Note that the expectation is calculated given a known state s and action a according to a probability over all actions $a \sim p(\cdot)$.

2.7.2 Policy Gradient Theorem

In contrast with value-based RL algorithms, such as the previously mentioned Q-Learning and DQN algorithms, policy-based methods directly approximate the optimal policy π^*. Recall that the policy dictates the action a taken in a state s: $\pi(a_t = a|s_t = s)$. In policy-based algorithms, the policy is modelled as a parameterized function with respect to parameters θ: $\pi_\theta(a_t = a|s_t = s)$. To maximize the policy is akin to maximizing θ and results in an objective function $J(\theta)$ to be maximized and is defined as [51]

$$J(\theta) = \mathbb{E}[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta]. \tag{18}$$

Since this is a maximization problem, gradient ascent is used to find the optimal parameters. Starting from Equation 18, expanding the expectation gives the following equation

$$J(\theta) = \mathbb{E}[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta], \tag{19}$$

$$= \sum_{t=i}^{T-1} P(s_t, a_t|\tau) r_{t+1}. \tag{20}$$

Here i dictates a starting point in a sequence of state-action pairs, $P(s_t, a_t|\tau)$ is the probability of the state-action pair in state-action sequence τ.
Equation 20 is the starting point in finding the gradient to maximize $J(\theta)$. Differentiating with respect to θ

$$\text{using } \frac{d}{dx}\mathbf{log} f(x) = \frac{f'(x)}{f(x)}, \tag{21}$$

$$\tag{22}$$

$$\nabla_\theta J(\theta) = \sum_{t=i}^{T-1} \nabla_\theta P(s_t, a_t|\tau) r_{t+1}, \tag{23}$$

$$= \sum_{t=i}^{T-1} P(s_t, a_t|\tau) \frac{\nabla_\theta P(s_t, a_t|\tau)}{P(s_t, a_t|\tau)} r_{t+1}, \tag{24}$$

$$= \sum_{t=i}^{T-1} P(s_t, a_t|\tau) \nabla_\theta \mathbf{log} P(s_t, a_t|\tau) r_{t+1}, \tag{25}$$

$$= \mathbb{E}[\sum_{t=i}^{T-1} \nabla_\theta \mathbf{log} P(s_t, a_t|\tau) r_{t+1}]. \tag{26}$$

During learning the algorithm uses random batches of episodes, therefore instead of computing an expectation of the gradient it can be replaced which results in

$$\nabla_\theta J(\theta) \sim \sum_{t=i}^{T-1} \nabla_\theta \mathbf{log} P(s_t, a_t|\tau) r_{t+1}. \tag{27}$$

Dissecting $\nabla_\theta \mathbf{log} P(s_t, a_t|\tau)$ by laying out the sequence τ by definition results in

$$P(s_t, a_t|\tau) = P(s_0, a_0, ..., s_t, a_t|\pi_\theta) \tag{28}$$

$$= P(s_0)\pi_\theta(a_0|s_0)P(s_1|s_0, a_0)\pi_\theta(a_1|s_1)...P(s_t|s_{t-1}, a_t\pi_\theta(a_t|s_{t-1}). \tag{29}$$

Taking the log on both sides gives

$$\begin{aligned}\mathbf{log} P(s_t, a_t|\tau) =& \mathbf{log} P(s_0) \mathbf{log} \pi_\theta(a_0|s_0) \mathbf{log} P(s_1|s_0, a_0) \\ & \mathbf{log} \pi_\theta(a_1|s_1)...\mathbf{log} P(s_t|s_{t-1}, a_t) \mathbf{log} \pi_\theta(a_t|s_{t-1}).\end{aligned} \tag{30}$$

Differentiating with respect to θ, note that $P(s_t|s_{t-1}, a_{t-1})$ is not dependent on θ and thus equals 0, yields:

$$\nabla_\theta \mathbf{log} P(s_t, a_t|\tau) = 0 + \nabla_\theta \mathbf{log}\pi_\theta(a_0|s_0) + 0 + \nabla_\theta \mathbf{log}\pi_\theta(a_1|s_1) + ...+ \nabla_\theta \mathbf{log}\pi_\theta(a_{t-1}|s_{t-1}) + 0 + \nabla_\theta \mathbf{log}\pi_\theta(a_t|s_t), \tag{31}$$

$$= \sum_{t'=0}^{t} \nabla_\theta \mathbf{log}\pi_\theta(a_{t'}|s_{t'}). \tag{32}$$

Plugging this in the original equation, Equation 27, and expanding results in

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} r_{t+1}(\sum_{t'=0}^{t} \nabla_\theta \mathbf{log}\pi_\theta(a_{t'}|s_{t'})), \tag{33}$$

$$= r_1(\sum_{t'=0}^{0} \nabla_\theta \mathbf{log}\pi_\theta(a_{t'}|s_{t'})) + r_2(\sum_{t'=0}^{1} \nabla_\theta \mathbf{log}\pi_\theta(a_{t'}|s_{t'})) + ...+ r_{T-1}(\sum_{t'=0}^{T-1} \nabla_\theta \mathbf{log}\pi_\theta(a_{t'}|s_{t'}), \tag{34}$$

$$= r_1 \nabla_\theta \mathbf{log}\pi_\theta(a_0|s_0) + r_2(\nabla_\theta \mathbf{log}\pi_\theta(a_0|s_0) + \nabla_\theta \mathbf{log}\pi_\theta(a_1|s_1)) + ...+ r_T(\nabla_\theta \mathbf{log}\pi_\theta(a_0|s_0) + ... + \nabla_\theta \mathbf{log}\pi_\theta(a_{T-1}|s_{T-1})), \tag{35}$$

$$= \nabla_\theta \mathbf{log}\pi_\theta(a_0|s_0)(r_1 + ... + r_T) + \nabla_\theta \mathbf{log}\pi_\theta(a_1|s_1)(r_2 + ... + r_T) + ...+ \nabla_\theta \mathbf{log}\pi_\theta(a_{T-1}|s_{T-1})(r_T), \tag{36}$$

$$= \sum_{t=0}^{T-1} \nabla_\theta \mathbf{log}\pi_\theta(a_t|s_t)(\sum_{t'=t+1}^{T} r_{t'}). \tag{37}$$

Adding the discount factor γ and expanding the same way as Equations 33-37 yields:

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta \mathbf{log}\pi_\theta(a_t|s_t)(\sum_{t'=t+1}^{T} \gamma^{t'-t-1} r_{t'}). \tag{38}$$

Simplifying the equation, by replacing $\sum_{t'=t+1}^{T} \gamma^{t'-t-1} r_{t'}$ with G_t, gives the final equation

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta \mathbf{log}\pi_\theta(a_t|s_t) G_t. \tag{39}$$

2.7.3 Actor-Critic Architecture

Two issues arise when using policy gradients, noisy gradients and high variance [52]. This is because the trajectories τ can deviate from each other to great degrees during training. For example, let there be three trajectories (τ_1, τ_2, τ_3) for each $\nabla_\theta \mathbf{log}\pi_\theta(\tau)$ was calculated and equals $(0.4, 0.7, 0.1)$. Each of the respective G_t's were calculated and returned $(200, 201, 202)$. Calculating the variance $Var(0.4 * 200, 0.7 * 201, 0.1 * 202)$ yields 3642.16, this high variance can lead to unstable learning.
A counter to this was introduced by Williams in 1992 [55]. This work introduces a baseline $b_t(s_t)$ to be subtracted from the policy gradient

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta \mathbf{log} \pi_\theta(a_t|s_t)(G_t - b_t(s_t)). \tag{40}$$

One common method is choosing the estimate of the value function $V(s_t)$ as the baseline. Since the baseline only depends on the state, it will not impact the gradient of the policy. The idea behind this is that the algorithm constantly checks if a specific action a_t is better or worse than the average action, given the state s_t. This is more commonly known as the advantage function

$$A(a_t, s_t) = Q(a_t, s_t) - V(s_t). \tag{41}$$

This approach forms the basis of the actor-critic architecture, where the policy π is seen as the actor and the baseline b_t is seen as the critic [51].
Adding the advantage function as the baseline, the new gradient becomes

$$\nabla_\theta J(\theta) \sim \sum_{t=0}^{T-1} \nabla_\theta \mathbf{log} \pi_\theta(a_t|s_t) A(s_t, a_t). \tag{42}$$

2.7.4 Trust Region Policy Optimization

Trust Region Policy Optimization (TRPO) by Schulman et al. [47] introduces an algorithm for smoother policy learning. It does so by applying a Kullback-Leibler (KL) Divergence [24] on parameter updates in the policy.
The KL Divergence of a function $q(x)$ from function $p(x)$ is denoted as $D_{KL}(p(x), q(x))$ and is a measurement of information loss when $q(x)$ is used to approximate $p(x)$. It is calculated with the following equation

$$D_{KL}(p(x), q(x)) = \int_{-\infty}^{\infty} p(x) \mathbf{ln} \frac{p(x)}{q(x)} dx. \tag{43}$$

Instead of directly applying the policy gradient, as depicted in Equation 39, to update the parameters, TRPO uses a surrogate loss function to update its parameters. The surrogate loss has its roots in Importance Sampling[49].
The idea behind Importance Sampling is to estimate the expected value of a function $f(x)$, where x follows a distribution $p(x)$. Then, instead of sampling x from p, it is sampled from q

$$\mathbb{E}_p[f(x)] = \mathbb{E}_q[\frac{f(x)p(x)}{q(x)}]. \tag{44}$$

If $q(x)$ is sufficiently close to $p(x)$ then the estimation is sufficiently accurate.
The application for Importance Sampling in RL, is that it is very inefficient to recalculate the expected rewards for a trajectory τ for each update done to the parameters θ from the old parameters θ_{old}. Therefore, instead of recalculating the expected rewards, the expected rewards from the trajectory τ under θ_{old} are used.

Figure 7: An intuitive depiction of TRPO; on the left normal gradient ascent, where a large step in the direction of the yellow arrow is taken. On the right an applciation of TRPO where the trust region is depicted as the yellow circle and a small update step as the blue dot [21].

The surrogate loss used by TRPO is constraint by a KL Divergence constraint such that it the new policy does not drift to far apart from the old policy

$$\max_{\pi} L(\pi) = \mathbb{E}_{\pi_{old}}[\frac{\pi(a|s)}{\pi_{old}(a|s)} A^{\pi_{old}}(s,a)], \text{ subject to} \tag{45}$$

$$\mathbb{E}[KL(\pi, \pi_{old}) \leq \epsilon]. \tag{46}$$

Note that $A^{\pi_{old}}(s,a)$ is the advantage function as depicted in Equation 41.

2.7.5 Proximal Policy Optimization

In 2017, Schulman et al. introduced Proximal Policy Optimization (PPO) [48], which builds upon their earlier work in TRPO and resulted in an algorithm that is simpler to implement, more general, and has better computational complexity.
Instead of using a constraint on the KL Divergence between the new and old policy, a clipping of the ratio was proposed

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\mathbf{min}(r_t(\theta)\hat{A}_t, \mathbf{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]. \tag{47}$$

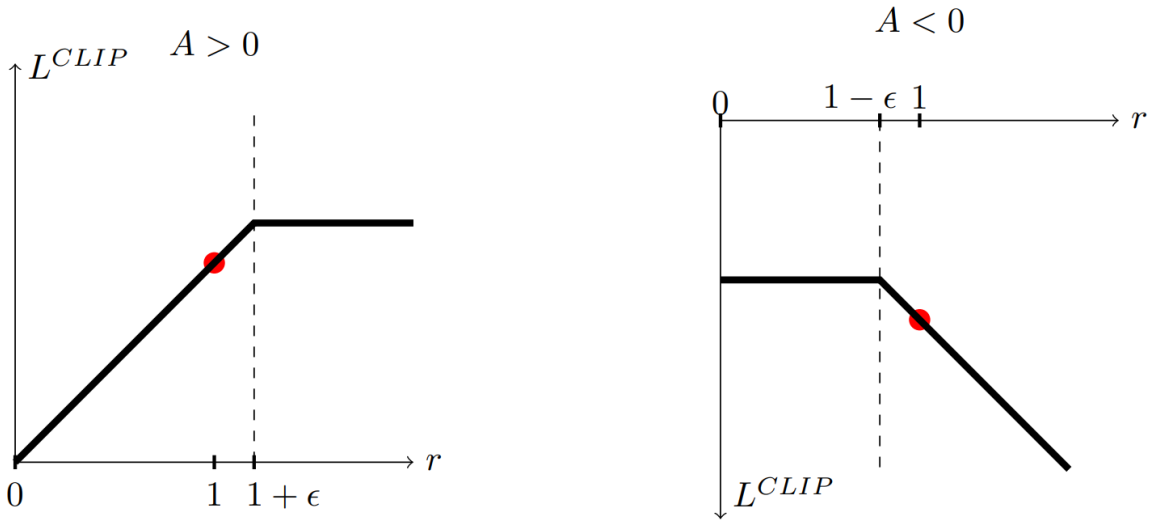


Figure 8: A graphical representation of the clipping of the gradient. The red dot represent the starting point [48].

Figure 8 gives an overview of one optimization step, starting from the red dot where $r(\theta) = 1.0$. In the left image, the advantage was returned positive, meaning that the action taken yielded better rewards than the expected average rewards. In the right image, the advantage was returned negative, meaning that the action taken yielded worse rewards than the expected average rewards.
Notice that in the left image, the loss function flattens out when $r_t(\theta)$ gets too high. This happens when the action taken is a lot more likely in the current policy π than it was in the old policy π_{old}. The algorithm clips the loss function to not overdo the gradient step and limit the effect of the update.
In the right image, the loss function flattens on the left side. This corresponds to an action taken that is much less likely to happen in the current policy than in the old policy. The same principle applies, clipping the loss function not to overdo the gradient step and thereby limiting the influence of the gradient update.
In the left image, the unflattened area corresponds to actions that returned better rewards but were less likely to be chosen under the current policy than the old policy. Therefore a much lower loss would be given since the policy moved in the wrong direction. The same idea applies to the right image; an action that returned worse rewards was more likely to be chosen under the current policy than the old one. Again the gradient moved in the wrong direction, and a larger negative loss was returned.

3 Related Work

After the introduction to Side-Channel Analysis by Kocher et al. [23], the important work by Chari et al. [9] introduced Template Attack (TA)s, which would drive the research in the SCA community for many years. Although being the most potent attack from an information-theory standpoint, its assumptions can be somewhat daunting and sometimes impossible (unlimited traces). Years later, more advanced methods were devised, such as the Stochastic Models presented in the work of Schindler et al. [46], which aims to reduce the amount of traces needed for profiling significantly. Further work was done by Choudhary and Kuhn [11], where the authors introduced pooling the covariance matrices used in the profiling phase and attaining a significant speed-up of the attack. These methods represented the state-of-the-art for several years, mainly due to the strength of performance and the fact that no hyperparameter tuning was needed.

3.1 Machine Learning and Deep Learning in SCA

A few years later, Machine Learning was introduced in the SCA community. Lerman et al. [25] aimed to find the limits of machine learning when conducting attacks. In their work, five classification models were tested: Support Vector Machines, Random Forests, Template Attacks, Stochastic Attacks, and Multivariate Regression Analysis. They found that the combination of a Support Vector Machine with a Correlation Power Analysis performed the best. The work by Martinasek et al. [33] proposed the first Neural Network (NN) for key recovery. The method is divided into three phases, a pattern preparation for each secret key, a training phase of the network, and a classification phase of the key estimates. The authors use a standard Multi-Layer Perceptron with three layers. It achieved promising, being able to achieve a 90% accuracy of the first byte with only one trace. Gilmore et al. [17], building upon the work of Lerman et al. [25], proposed the use of a Neural Network instead of a Support Vector Machine. The NN was able to provide a decrease in traces needed to identify the mask used and was able to achieve a 91.75% accuracy in the key recovery phase.
Maghrebi et al. [31] continued this line of research by introducing several Deep Learning techniques to the field of Side-Channel Analysis. They experimented with four DL-based attacks, an AutoEncoder (AE), a Convolutional Neural Network (CNN), a Long Short-Term Memory (LSTM) network, and the Multi-Layer Perceptron (MLP). The authors concluded that these methods are able to target different aspects of SCA, such as implicit feature extraction through AEs and CNNs and exploiting time dependencies in traces with LSTMs. Although the performance of these techniques is strong, one downside of these techniques is their reliance on finely tuned hyperparameters. Maghrebi et al. [31] mentioned that the performance fluctuates by tuning the network with Genetic Algorithms. In 2020, Benadjilla et al. [6] made an exhaustive search for the best-performing CNN on their ASCAD data set. Recently Rijsdijk et al. [41] introduced Reinforcement Learning to automatically search for the best performing CNN architecture for several databases. The authors specifically search for architectures that are relatively small yet retain state-of-the-art performance.

3.2 Point Of Interest selection

Over the years and with the emergence of these strong techniques, Template Attacks became more of a baseline to compare techniques with than the preferred method of attacking devices. However, they stay the most optimal from an information-theoretic point of view. In 2015 Lerman et al. [26] even concluded that, with proper Points Of Interest selection, Template Attacks outperform Machine Learning attacks.

Over the years, several techniques have been researched to reduce the complexity of Template Attacks. One of the first works was in 2006, where Archambeau et al. [2] introduced Principal Component Analysis to create a principal subspace. The principal subspace reduced the dimension of the traces by 99.99% and resulted in being able to correctly classify 93.3% of the traces.
Picek et al. [37] explored many different Points Of Interest selection methods used frequently in Side-Channel Analysis. The authors made a distinction between Points Of Interest selection techniques and divided them into three categories: feature selection, where a subset of all points/features is selected, dimensionality reduction, where a technique transforms the original features into new features, and deep learning techniques, where feature extraction is done implicitly due to the nature of deep learning attacks. The authors concluded that feature selection is a very important step in attacks where the data is noisy and contains various countermeasures
In Perin et al.'s work [36], the authors explore different setups of Points Of Interest for the preprocessing of DL attacks. Three different approaches are mentioned: Refined Points Of Interest, taking only the points with the highest Signal-to-Noise Ratio (SNR) peaks into account, Optmized Points Of Interest, a predefined interval or concatenation of multiple intervals that include the main SNR peaks, and Non-Optimized Points Of Interest, where points with high and low SNR peaks are selected.
More recently, Wu et al. [56] used a Machine Learning technique called Similarity Learning to show that with proper feature engineering, Template Attacks remain feasible and are even able to outperform current state-of-the-art Deep Learning techniques.
Rioja et al. [42] introduced an automated DL tuner based on Estimation of Distribution Algorithms (EDAs). EDAs are evolutionary methods based on an implicit distribution model instead of the more common mutation and crossover operators. The authors concluded that this implementation could automatically choose good-performing POIs and therefore reduce the need for human intervention.
The first instance of Reinforcement Learning applied in Side-Channel Analysis concerning Points Of Interest selection, to the best of the writer's knowledge, is Side-Channel Analysis with Reinforcement Learning (SCARL). In this paper Ramezanpour et al. [40] introduce an algorithm that preprocesses the data with an AutoEncoder and, with the help of a self-supervised Actor-Critic model, is able to cluster features based on the inter-cluster difference on the mean.
As stated in [37], the proper Points Of Interest selection research has seen very few attempts. This is mainly due to the fact that simple techniques were considered sufficient enough. It is evident to see that this step has become less important since most research is benchmarked on predefined POI intervals.
This thesis attempts to reintroduce POI selection as an important topic to research. This is done by introducing a novel Reinforcement Learning driven framework for Points Of Interest selection. The framework is benchmarked on commonly used datasets to see if it is able to find Points Of Interest. A comparison of the performance is made with several currently used POI methods to see if the framework can improve POI selection.

4 Current Methods for Point Of Interest selection

Profiling Side-Channel Attacks are still regarded as one of the most potent forms of attacks on cryptographic algorithms. But the success of these attacks relies on the effectiveness of the profiling stage, as explained in Section 2.2.3. As mentioned before, traces contain important, redundant, or irrelevant features. It is important to distinguish and separate these features, since using features that do not contain leakage information, is bound to lead to the wrong results.
The attack of choice in this thesis is the Template Attack, as explained in Section 2.2.2. It is shown in various works that with proper Points Of Interest selection, this attack performs better than without [26][9][37][56].
Three categories of POI selection methods are explored in this thesis [37]:

- **Feature Selection**: A method that extracts the most important raw features within a trace by returning a subset of the whole trace.
- **Dimensionality Reduction**: A method of transforming the original n features in a set of k new features.
- **Deep Learning Techniques**: A method of using a Deep Learning model to extract features. Based on a loss function these methods are able to distinguish good features from bad.

This section introduces each of the methods explored in this thesis. Each of their results is discussed in Section 8.

4.1 Feature Selection Methods

The selection methods outlined in this section are able to select a subset S of interesting points directly from the data set D. Two methods will be analyzed, Sum of Squared T-Differences and Signal-to-Noise Ratio.

4.1.1 Sum Of Squared T-Differences

Chari et al. [9], in their work on Template Attacks, offered a solution to feature extraction. The Sum of Differences (SOD), as depicted in Equation 48, is the sum of the difference between the mean of point i and point j. The mean of each point is calculated over a set. In the case of SCA these sets are the groupings based on the intermediate values.
Later Gierlichs et al. [16] made an adaption of Chari et al.'s work. The authors proposed the Sum Of Squared Differences (SOSD), as seen in Equation 49. In the same work they proposed the normalized version of SOSD, the Sum Of Squared T-Differences (SOST). The authors adapted their own work by applying the T-test to distinguish noisy signals, which can be found in Equation 50.

$$\bar{c} = \sum_{i,j=1}^{K} \mu_i - \mu_j \tag{48}$$

$$\bar{c} = \sum_{i,j=1}^{K} (\mu_i - \mu_j)^2 \tag{49}$$

$$\bar{c} = \sum_{i,j=1}^{K} \left(\frac{\mu_i - \mu_j}{\sqrt{\frac{\sigma_i^2}{n_i} + \frac{\sigma_j^2}{n_j}}}\right)^2 \tag{50}$$

The difference between SOSD and SOST is striking and an example can be found in Figure 9.

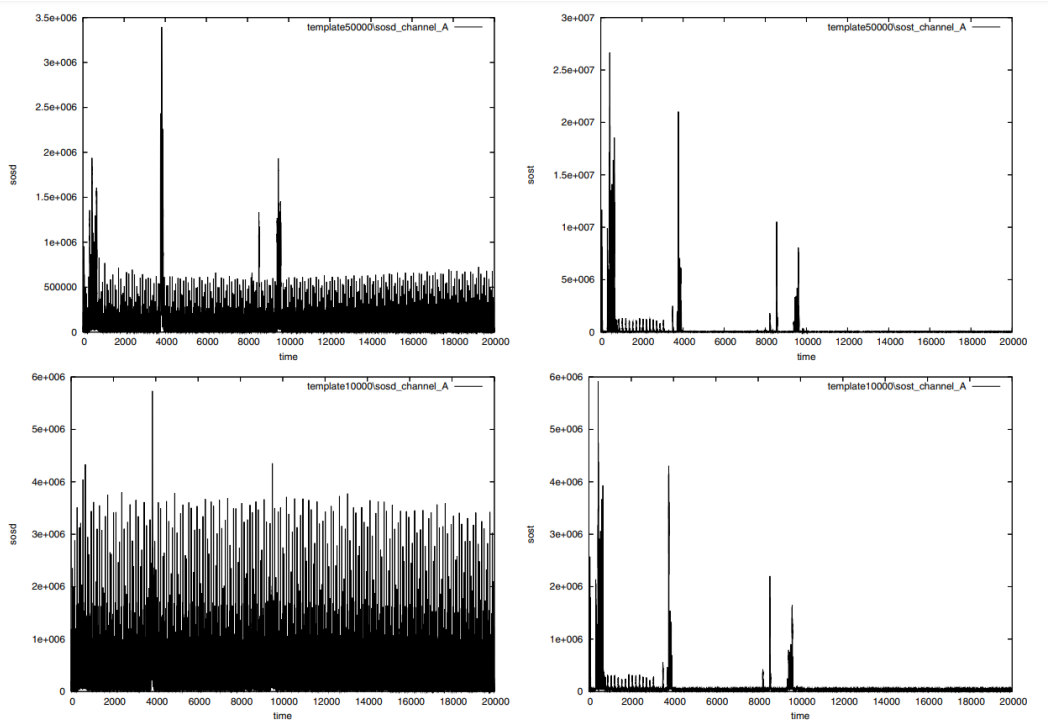


Figure 9: On the left is an image of SOSD, on the right of SOST. It is clear to see that SOST is able to remove the noise and accentuate the peaks [16].

Using SOST for Points Of Interest selection means using the top-n features corresponding to to highest SOST coefficients from $\bar{c}$.

4.1.2 Signal-to-Noise Ratio

Signal-to-Noise Ratio (SNR) [45][32] is a measurement to compare the amount of the desired signal against the unwanted amount of noise. SNR is calculated as $\frac{Signal}{Noise}$, an alternative is to calculate the ratio of mean to standard deviation of the measurements, $\frac{\mu}{\sigma}$.
To use this technique in Side-Channel Analysis, each trace is first grouped based on the leakage model. The mean and variance is calculated for each group. Then, the mean of the group variances and the variance of the group means is calculated. Finally the SNR is calculated as $\frac{Var(\mu_{groups})}{Mean(\sigma_{groups})}$.

The SNR is projected on each feature from the trace. Using SNR for POI selection is done by taking the top-n highest peaks from the trace and using the corresponding features for an attack.

4.2 Dimensionality Reduction Methods

The methods outlined in this section are able to create a mapping from a set of n features to a set of k features. It is important to note that these k features are not equal to the n original features.

4.2.1 Principal Component Analysis

Principal Component Analysis (PCA) [19] is a dimensionality reduction method that decomposes the data into linear combinations. It tries to find the set of linear combinations that separates the data the most. In essence, it finds the linear combinations with the most significant variance in decreasing order, often a top-n approach is used with n being the number of principal components to find.
Given a dataset $\mathbf{X}$ with dimension $n \times p$, the goal of PCA is to find a $n \times k$ representation $\mathbf{Z}$ of $\mathbf{X}$, while $k << p$. It does so by finding a projection matrix $\mathbf{W}$ such that

$$\mathbf{Z} = \mathbf{W}^T\mathbf{X} \tag{51}$$

For the sake of simplicity, consider a one dimensional projection with $k = 1$. Then the projection matrix $\mathbf{W}$ becomes vector $\bar{w}$ and the representation matrix $\mathbf{Z}$ becomes vector $\bar{z}$. Equation 51 the becomes

$$\bar{z}_n = \bar{w}^T\bar{x}_n. \tag{52}$$

This essentially describes a projection of $\mathbf{X}$ on a line $\bar{z}$ where z_n is the position of x_n on that line. Since the goal is to retain the maximum variance of $\mathbf{X}$ through $\bar{z}$, it is equivalent to find the maximum variance of $\bar{z}$. This is done by calculating the sum of squares of the projection and dividing by the number of samples

$$V = \frac{1}{N}\sum_{n=1}^{N}\bar{z}_n^2, \tag{53}$$

$$= \frac{1}{N}\sum_{n=1}^{N}(\bar{w}^T\bar{x}_n)^2. \tag{54}$$

Using the fact that $A^T B = B^T A$, Equation 54 becomes

$$V = \frac{1}{N}\sum_{n=1}^{N}(\bar{w}^T\bar{x}_n)^2, \tag{55}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\bar{w}^T\bar{x}_n\bar{w}^T\bar{x}_n, \tag{56}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\bar{w}^T\bar{x}_n\bar{x}_n^T\bar{w}, \tag{57}$$

$$= \bar{w}^T(\frac{1}{N}\sum_{n=1}^{N}\bar{x}_n\bar{x}_n^T)\bar{w}. \tag{58}$$

To simplify further, the inner term can be replaced with the covariance matrix $\mathbf{C}$,

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}\bar{x}_n\bar{x}_n^T, \tag{59}$$

Equation 58 becomes

$$V = \bar{w}^T\mathbf{C}\bar{w}. \tag{60}$$

Since $\mathbf{C}$ is static, maximizing the variance is akin to maximizing the weight vector w and this can be approached as an optimization problem. To ensure that the variance does not explode, a constraint is applied that the norm of the weight vector cannot exceed unit length

$$\bar{w}^T\bar{w} = \|\bar{w}\|^2 = 1. \tag{61}$$

Since this has become a constrained optimization problem it can be solved by applying theLagrange Multiplier Method (LMM). LMM is setup using the function

$$L(x,\lambda) = f(x) + \lambda g(x). \tag{62}$$

Where $f(x)$ denotes the function to be maximized or minimized and $g(x)$ denotes the equality constraint.
Using LMM the optimization problem is set up as

$$L(\bar{w},\lambda) = \bar{w}^T\mathbf{C}\bar{w} + \lambda(\bar{w}^T\bar{w} - 1). \tag{63}$$

Since the eigenvectors of the covariance matrix point into the direction of the largest variance within the data, finding the largest eigenvalue corresponds to finding the largest variance. As can be seen from Equation 63, the weight vector corresponds to the eigenvector and λ to the eigenvalue.
To solve Equation 63 for the eigenvector $\bar{w}$ it is differentiated with respect to $\bar{w}$ and set to 0

$$\frac{\partial L}{\partial \bar{w}} = 2\bar{w}^T\mathbf{C} - 2\lambda\bar{w}^T = 0, \tag{64}$$

$$=> \bar{w}^T\mathbf{C} = \lambda\bar{w}^T. \tag{65}$$

To solve Equation 63 for the eigenvalue λ it is differentiated with respect to λ and set to 0

$$\frac{\partial L}{\partial \lambda} = \bar{w}^T\bar{w} - 1 = 0, \tag{66}$$

$$=> \bar{w}^T\bar{w} = 1 \tag{67}$$

Plugging these in Equation 60 it becomes

$$V = \bar{w}^T\mathbf{C}\bar{w}, \tag{68}$$

$$= \lambda\bar{w}^T w, \tag{69}$$

$$= \lambda \tag{70}$$

So to reiterate, the variance is equivalent to the eigenvalue associated with the eigenvector that spans $\mathbf{Z}$. Thus finding the largest eigenvalue is equivalent to finding the largest variance. So PCA essentially makes an eigenvalue decomposition to find the largest eigenvalue that corresponds to the eigenvector that spans the projection.
It is possible to extrapolate this to a k-dimensional subspace $\mathbf{Z}$ which constructs k eigenvalues for each V_i for $i \in [1...k]$.
For POI selection, the top-n largest eigenvalues are selected, and with their corresponding eigenvectors a transformation matrix is constructed. The transformation matrix is then used to transform each trace from the attack set and these are subsequently used for an attack.

4.2.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA), also known as Fisher's Linear Discriminant named after founder R.A. Fisher [15], is a classification and dimensionality reduction method. LDA is closely related to PCA in that it also looks for a linear combination of variables to explain the differences in the data. The difference is that LDA takes class differences into account, whereas PCA ignores these.
LDA is calculated with two variance matrices, often called scatter matrices. The intra- or within-class scatter matrix $\mathbf{S}_w$, which denotes the total variance within a class, and the inter- or between-class matrix $\mathbf{S}_b$, which denotes the total variance between the classes.
In the first step to calculate the scatter matrices, several means are calculated. The data is grouped according to their class and for each class c a mean $\bar{\mu}_c$ is calculated. The last mean needed is an overall mean $\bar{\mu}_o$ that is calculated over the whole dataset.
The within-class scatter matrix is calculated by summing over the individual class variance matrices with

$$\mathbf{S}_w = \sum_{i=1}^{c} \mathbf{S}_i. \tag{71}$$

Each individual variance matrix S_i is calculated with

$$\mathbf{S}_i = \sum_{j=1}^{N} (\bar{x}_j - \bar{\mu}_i)(\bar{x}_j - \bar{\mu}_i)^T \tag{72}$$

Since the summation is done over each variable $\bar{x}$, if there exist a significant class imbalance the summation is skewed towards overrepresented classes. To counter this, each individual scatter matrix $\mathbf{S}_i$ is normalized according to the amount of samples in the class

$$\frac{1}{N_i}\mathbf{S}_i. \tag{73}$$

The between-class scatter matrix is calculated by summing the difference between the class means $\bar{\mu}_c$ and the aforementioned overall mean $\bar{\mu}_o$

$$\mathbf{S}_b = \sum_{i=1}^{c} N_i(\bar{\mu}_i - \bar{\mu}_o)(\bar{\mu}_i - \bar{\mu}_o)^T \tag{74}$$

As with PCA, as explained in Section 4.2.1, the largest variance corresponds to the largest eigenvector. Instead of using a eigenvalue decomposition of the covariance matrix, LDA solves the generalized eigenvalue problem of the matrix $\mathbf{S}_w^{-1}\mathbf{S}_b$. This results in the c eigenvalues with their corresponding eigenvectors.
At last, LDA performs a projection on a k-dimensional plane by multiplying the data with weight matrix $\mathbf{W}$. The weight matrix $\mathbf{W}$ is constructed by the eigenvectors corresponding to the top-k eigenvalues.
For LDA the same holds as was done with PCA. The top-n eigenvalues are selected and their eigenvectors are used for the transformation matrix. This matrix is used to transform each attack trace and subsequently the attack traces are used for an attack.

4.3 Deep Learning Method

Deep Learning Methods use an Artificial Neural Network to transform the original features in a new set of features. All Deep Learning methods use a loss function to learn how well the network performs.

4.3.1 Triplet Network

With the recent shift from statistical analysis for POI selection to Machine Learning techniques, Wu et al. [56] introduced the Triplet Network, as depicted in Figure 10, for feature extraction.
The network name comes from the way the input is set up. A triplet consists of an anchor a, a positive p, and a negative n. The anchor and the positive share the same label, while the negative has another label. All three are run through the network, and the loss is calculated as shown in Equation 75.

$$loss = \mathbf{max}(dist(a,p) - dist(a,n) + margin, 0) \tag{75}$$

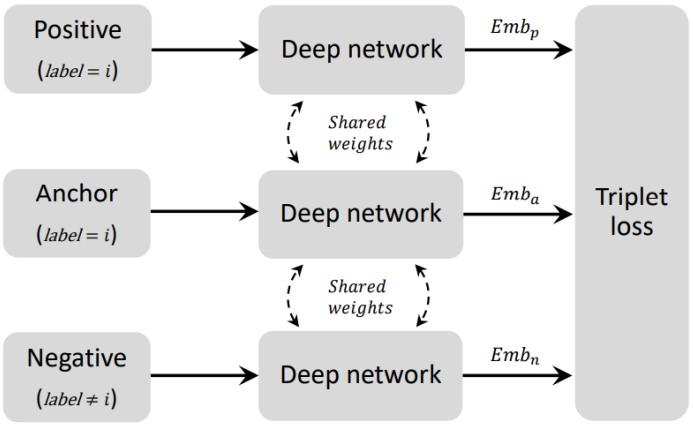


Figure 10: Triplet Network; three identical ANNs using different inputs to produce three embeddings [56].

In the loss function *dist* denotes the Euclidean distance between two points in a space. Based on the loss function three types of triplets can be distinguished

- Easy triplets, where $dist(a, p) + margin < dist(a, n)$,
- Hard triplets, where $dist(a, n) < dist(a, p)$,
- Semi-hard triplets, where $dist(a, p) < dist(a, n) < dist(a, p) + margin$.

the $margin$ describes the boundary between the three types of triplets Training on easy triplets can lead to low loss values since p and n are easily separable. But, this could also lead to a local optimum. Training on hard triplets could lead to a collapsing model, since the distance from a to n is less than from a to p. Training on semi-hard triplets introduces learning difficulties set by the boundary value $margin$, this results in the model extracting more representative features [56].
The network is used for POI selection in a very simple way. Once the network is trained on the profiling set, it is used on the attack set. The output of the network is then used to perform an attack.

5 Proposed framework

This section explains the proposed framework and the algorithms working within. In general, the framework provides a fire-and-forget method, devised as an alternative to manually selecting points of interest. Furthermore, the framework automatically scales down the dimensions further by providing an optimized dimensionality reduction network tailored to the selected points, based on the Triplet Network as explained in Section 4.3.1. Since this framework combines the conventional POI selection method and DL-based method in an automatic manner, it essentially reduces the amount of work and domain knowledge that is needed for proper Points Of Interest selection.

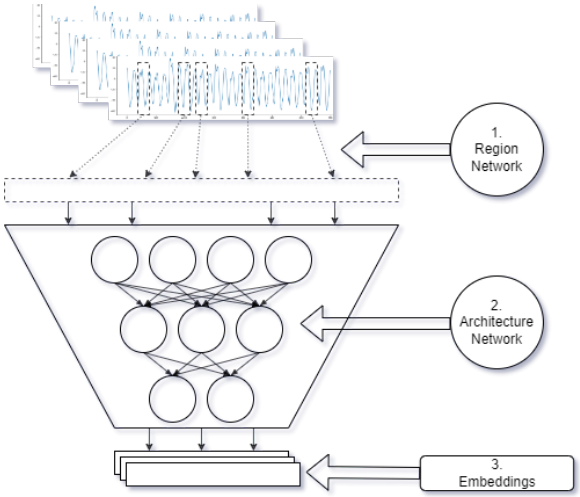


Figure 11: Graphical overview of proposed framework

An overview of the framework is shown in Figure 11. Each episode, the framework executes two inner loops that run until terminal conditions are met. The first inner loop is the region selection phase, depicted with 1 in Figure 11, the second inner loop is the dimensionality reduction phase, depicted with 2 in Figure 11. The output of the dimensionality reduction network produces embeddings, depicted with 3 in Figure 11. These are then used for a Template Attack, which determines the performance of the chosen regions and network. An algorithmic overview is shown in Algorithm 1.

Algorithm 1 Proposed framework Framework

```
region_ppo ← build_PPO_net()
dim_red_ppo ← build_PPO_net()
for ep ∈ episodes do
    selected_regions ← select_regions(region_ppo)          ▷ Displayed in Algorithm 2
    dim_red_network ← create_network(dim_red_ppo)          ▷ Displayed in Algorithm 3
    processed_traces ← process(traces, selected_regions)
    batches ← generate_batches(processed_traces, batch_size)
    train_network(dim_red_network, batches)
    ranks ← perform_attack(dim_red_network)
    reward ← calculate_reward(ranks)
    train_network(region_ppo, dim_red_ppo)
```

5.1 The Environment

The framework operates within an environment that has access to the dataset. This includes the profiling traces, attack traces, plain- and ciphertexts, and profiling keys.
The environment aggregates the features in regions of length n. This value n is determined by the trace length and the amount of regions available as $n = length/regions$. The amount of regions available is a hyperparameter that needs to be set beforehand.
Aggregations of the features is done to ensure that leakages spanning multiple points are selected in one go, thereby spanning a greater range of possible leakage points. An example of leakages spanning multiple points close together is shown in Figure 12.

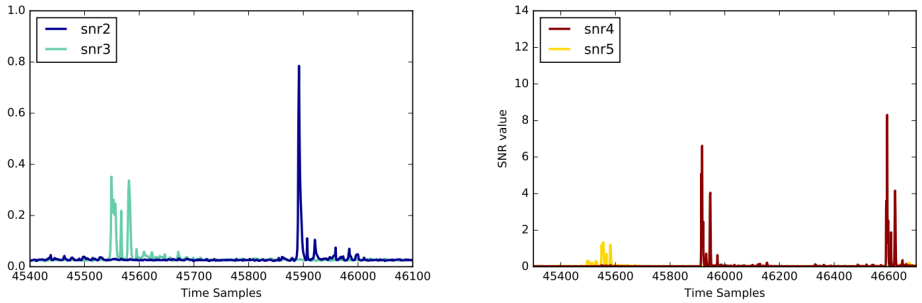


Figure 12: The SNR of the unmasked ASCAD dataset showing the leakage spanning multiple points in a trace [6].

At each episode, multiple regions are selected. To narrow down the selection of the regions to only the well-performing regions, a current maximum amount of regions r_{cur} is established. At each episode, the environment reduces r_{cur} with Exponential Decay. This is done to explicitly induce an exploration-vs-exploitation dichotomy. Furthermore, since the search space of features can be rather large, reducing the amount of the to-be-selected regions provides a speed up of the framework.
To kick-start the learning process of distinguishing between well-performing regions and bad-performing regions, a percentage of the total amount of regions is considered as the initial maximum amount of regions to-be-selected r_{max}. For example, if the total amount of regions is 1000 and the environment is setup with 50% initial regions, $r_{max} = 500$.
Since Exponential Decay reduces to 0 given enough time, a minimum amount of to-be-selected regions r_{min} is defined. This again is a percentage of the total amount of regions available. Equation 76 gives the Exponential Decay function.

$$r_{cur} = \max(\lfloor r_0 e^{-\lambda ep} \rfloor, r_{min}) \tag{76}$$

Here λ denotes the decay factor and ep denotes the current episode of the framework. An example of the decay function with $\lambda = 0.002$, $r_{max} = 1000$ and $r_{min} = 100$ is shown in Figure 13.
Finally, the environment consists of the total amount of episodes the framework will run for and the batch size of traces that are analyzed when selecting regions.

5.2 Reward Function

The framework follows the general Reinforcement Learning loop as is depicted in Figure 6. For the framework to learn, a reward function is needed. This reward function is an adapta-

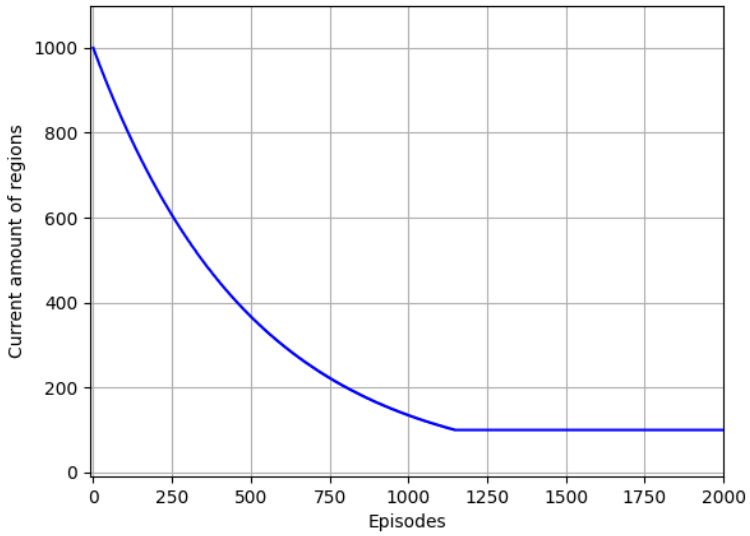


Figure 13: An example of the Exponential Decay function in Equation 76. Here $\lambda = 0.002$, $r_{max} = 1000$ and $r_{min} = 100$.

tion of the reward function found in [41]. Two adaptions were made, the first was to remove the notion of the accuracy metric. In [41], the goal was to correctly classify key guesses with a CNN, in this thesis that is not the case. Since there are no labels associated with Points Of Interest, any form of classification metric are not available. The second adaption was to remove the reward for the size of the networks created by the proposed method in [41]. In this thesis there exists no goal to produce as less as possible Points Of Interest, just the best performing, therefore this part was removed. The reward function used in this work is shown in Equation 77.

$$r = \frac{t' + GE'_{10} + 0.5GE'_{50}}{2.5} \tag{77}$$

$$t' = \frac{t_{max} - min(t_{max}, GE_{k^*})}{t_{max}} \tag{78}$$

$$GE'_{10} = \frac{128 - min(GE_{10}, 128)}{128} \tag{79}$$

$$GE'_{50} = \frac{128 - min(GE_{50}, 128)}{128} \tag{80}$$

To calculate the reward, the framework performs a Template Attack with the selected and reduced points of interest. The template attack returns the Guess Entropy vector and is used as GE_{k^*}. Here t_{max} denotes the amount of attack traces used for the attack. Furthermore, GE_{10} resembles the Guess Entropy when 10% of the maximum amount of traces are used, GE_{50} resembles the Guess Entropy when 50% of the traces are used.
These metrics were chosen to incentivize the learning to focus on reducing the amount of traces needed to converge to the correct key guess. The last two metrics are added to make sure that, although complete convergence was not achieved, the actions taken are not disregarded as completely wrong.

5.3 Phase 1: Feature Selection

This phase of the framework is responsible for the selection of multiple regions from the traces. It is based on the Proximal Policy Optimization algorithm, explained in Section 2.7.5. The architecture of the PPO network is a neural network with five layers. The architecture of the network is found in Table 3. The average pooling layer is added to reduce the number of inputs the network has to take into account, thereby speeding up the process. Very early in the experimentation, it became apparent that a Multi-Layer Perceptron performed better than a Convolutional Neural Network. Therefore it was decided to do all experiments with MLPs, CNNs were not further experimented with.

Region PPO Network	
Avg Pool layer:	Kernel=3, Stride=2
FC layer:	Neurons=256
Activation Layer	ReLU
FC layer:	Neurons=256
Activation Layer	ReLU
FC layer:	Neurons=128
Activation Layer	ReLU
FC layer:	Neurons=64
Activation Layer	ReLU
FC layer:	Neurons=action space size

Table 3: Network Architecture for the Region Selection PPO network.

At each iteration, the environment provides the network with a subset of the profiling traces. The subset size is dictated by the batch size provided in the environment. This is done to diversify the input and ensure that the algorithm is not taking actions based on only a single trace. In each iteration, one region of all possible regions is selected until it has reached the amount of regions to select.
Each trace in the network is run through the network and outputs raw network outputs, often called logits. Since the network is set up to take only one action for multiple inputs, the logits are summed.
After summation the new logits are used to create a categorical distribution,according to Equation 12. From the categorical distribution one action is sampled. This action represents the selected region for that iteration.
The PPO algorithm has the Actor-Critic architecture, as explained in Section 2.7.3. This means that a critic-value is calculated with the same network, but outputs only one value. This value can be interpreted as a score for how well the network is performing. Since the network takes in a batch of traces, there is also a batch of critic values. In this phase, the critic value is averaged instead of summed, as this is interpreted as the average state of the network. The algorithm's general layout is shown in Algorithm 2.

Invalid Action Masking To ensure that regions are not duplicated, Invalid Action Masking (IAM) [20] is applied. IAM is the method of replacing certain logits with a large negative number, such that it defaults to a probability of practically 0 when creating a categorical distribution. For example, assume an environment is defined by two states $[s_0, s_1]$ and four actions, $[a_0, a_1, a_2, a_3]$, are available in s_0. Assume that s_1 is the terminal state and that the default reward is 1. Furthermore, a policy π_θ is defined with $\theta = [1.0, 1.0, 1.0, 1.0]$, assume that it directly returns θ as the output logits. Then in s_0, the policy produces

$$\begin{aligned}
\pi_\theta(\cdot|s_0) &= [\pi_\theta(a_0|s_0), \pi_\theta(a_1|s_0), \pi_\theta(a_2|s_0), \pi_\theta(a_3|s_0)], && (81)\\
&=> softmax([l_0, l_1, l_2, l_3]), && (82)\\
&= [0.25, 0.25, 0.25, 0.25]. && (83)
\end{aligned}$$

Now IAM is applied by replacing the invalid action's logit by a large negative number M, e.g. $M = -10^8$, assume action a_2 is invalid then

$$\begin{aligned}
\pi'_\theta(\cdot|s_0) &= \textbf{mask}([\pi_\theta(a_0|s_0), \pi_\theta(a_1|s_0), \pi_\theta(a_2|s_0), \pi_\theta(a_3|s_0)]), && (84)\\
&=> softmax([l_0, l_1, M, l_3]), && (85)\\
&= softmax([1.0, 1.0, -10^8, 1.0]), && (86)\\
&= [0.33, 0.33, 0.0, 0.33]. && (87)
\end{aligned}$$

Since the mask is only dependent on the state and has no bearing on the parameters of the policy, it produces a valid policy gradient. For the complete proof see [20].

Algorithm 2 Region Selection Algorithm

Require: $env, region\_model$
  $phase \leftarrow$ **region**
  $obs \leftarrow$ `reset`$(env, phase)$
  while not $done$ **do**
    $logits, val \leftarrow$ `region_model`(obs)
    $logits \leftarrow \sum logits$
    $val \leftarrow mean(val)$
    $mask \leftarrow \bar{0}$ ▷ Length of mask is determined by logits
    $regions \leftarrow$ `get_selected_regions`(env)
    for each $r \in regions$ **do**
      $mask[r] \leftarrow 1$
    $logits \leftarrow$ `apply_mask`$(logits, mask)$
    $dist \leftarrow$ `create_categorical_ditribution`$(logits)$
    $action \leftarrow$ `sample`$(dist)$
    $log\_prob \leftarrow$ `get_log_probability`$(dist, action)$
    $next\_obs, done \leftarrow$ `step`$(env, action)$
    $train\_data \leftarrow$ `setup_train_data`$(obs, action, val, log\_prob, mask)$
    $obs \leftarrow next\_obs$
    if $done$ **then**
      `break_while`

5.4 Phase 2: Dimensionality Reduction

In the second phase of the framework, the algorithm iteratively builds up a Neural Network similar to the Triplet Network, as explained in Section 4.3.1. As with the previous phase, this phase uses a Proximal Policy Optimization algorithm to find the best network for selected regions. The architecture of the PPO network can be found in Table 4.

Dimensionality Reduction PPO Network	
FC layer	Neurons=32
Activation layer	ReLU
FC layer	Neurons=32
Activation layer	ReLU
FC layer	Neurons=32
Activation layer	ReLU
FC layer	Neurons=action space size

Table 4: Network Architecture for the Dimensionality Reduction PPO network.

The architecture of the PPO network is chosen to reflect the significant difference in state size from the Region Network shown in Table 3. The choice to omit any convolutional layer is made because the input consists of variables that are strongly independent of each other. The states built by the environment constitute of the current amount of layers present in the network, the type of layer selected in the previous step, the output shape of the layer selected in the previous step, and if the algorithm has achieved a terminal state. A general layout of the algorithm is shown in Algorithm 3.
The action space of Algorithm 3 has been masked with IAM as well. In addition, several restrictions have been put up to guide the algorithm in building valid networks. An overview of these restrictions is found in Table 5 and a graphical overview of the state transitions is shown in Figure 14.

State	**Restrictions**
Null State	Can only transition to Pooling, Convolutional or Terminal layers
Pooling layer	Cannot transition to other Pooling layers
Convolutional layer	No restrictions
Activation layer	Cannot transition to other Activation layers
Fully Connected layer	No restrictions
Terminal layer	Can only transition to optimizers

Table 5: On overview of each state and the transition restrictions

It is important to note that not only are state transitions restricted, but the hyperparameters belonging to those states as well. Since the network's purpose is to reduce the dimensions, the outputs of the following state cannot exceed the inputs to that state, e.g. if the output of the current state is of dimension 64, every action that leads to a new layer with an output dimension larger than 64 is determined invalid and is masked as such.

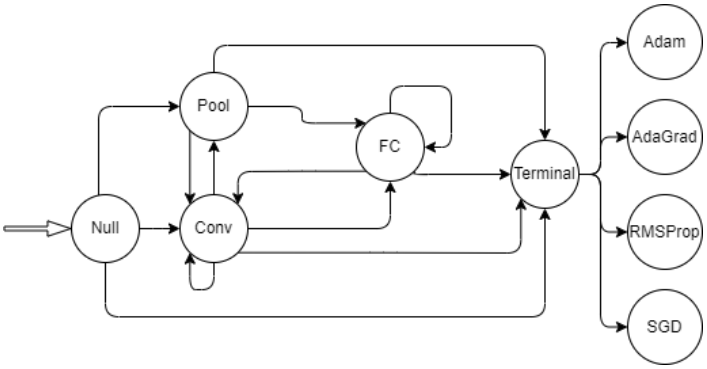


Figure 14: Graphical overview of the state transitions, note that for ease of viewing, activation layers are removed.

Algorithm 3 Dimensionality Reduction Algorithm

Require: $env, network\_model$
  $phase \leftarrow$ **network**
  $obs \leftarrow$ `reset`$(env, phase)$
  while not $done$ **do**
    $logits, val \leftarrow$ `network_model`(obs)
    $mask \leftarrow$ `determine_mask`(obs)
    $logits \leftarrow$ `apply_mask`$(logits, mask)$
    $dist \leftarrow$ `create_categorical_ditribution`$(logits)$
    $action \leftarrow$ `sample`$(dist)$
    $log\_prob \leftarrow$ `get_log_probability`$(dist, action)$
    $next\_obs, done \leftarrow$ `step`$(env, action)$
    $train\_data \leftarrow$ `setup_train_data`$(obs, action, val, log\_prob, mask)$
    $obs \leftarrow next\_obs$
    if $done$ **then**
      `break_while`

Several hyperparameters are made available to be chosen by the algorithm. An overview of each layer and the respective hyperparameters are shown in Table 6.

Layer	**Hyperparameters**	
Fully Connected layer	Neurons : [256, 128, 64, 32, 16, 8]	
Convolutional layer	Kernel : [256, 128, 64, 32, 16, 8]	Stride : [16, 8, 4, 2, 1]
Pooling Layer	Kernel : [256, 128, 64, 32, 16, 8]	Stride : [16, 8, 4, 2, 1]
Activation Layer	Type : ReLU, Tanh, SeLU	
Embeddings Layer	Neurons : [64, 32, 16, 8]	
Optimizer	Type : Adam, AdaGrad, RMSProp, SGD	LR: [1e-4, 3e-4, 1e-3, 3e-3, 1e-2, 3e-2]

Table 6: Hyperparameter overview of the possible combinations of layers.

After the selection of the optimizer the network is build witht he selected layers and hyperparameters and trained using the loss explained in Equation 75. Training is done for 25 episodes with a batchsize of 32 and a margin of 0.4.

6 Datasets

Four datasets will be used to benchmark the current methods and the proposed framework, the ASCAD dataset containing the fixed key and variable key dataset, the CHES_CTF dataset, and the AES_HD dataset.

6.1 ASCAD Fixed and Variable

The ASCAD Dataset [6] is created by acquiring EM traces from an AES implementation on the WB Electronics 64 kbit ATMega chipcard. The chip card contains an ATMega8515 controller running an AES implementation. The chip card itself has no hardware security implementation. Therefore two countermeasures are implemented in the AES implementation, affine masking [8] and operation shuffling [43]. The dataset has the following leakage model:

$$y(k^*, pt) = SBox(k^* \oplus pt)$$

The complete ASCAD dataset consists of two actual datasets, ASCAD_F and ASCAD_R. With ASCAD_F, the traces are acquired with a fixed key. With ASCAD_R, for only 33% of the traces the keys are fixed and for the other 66%, the keys are random. Both dataset consist of 45000 profiling traces and 5000 attack traces. Originally the datasets contains raw traces of 100000 features. The authors made a predefined window of 700 features for the fixed key dataset and 1400 features for the variable key dataset.
To determine the effectiveness of the current state-of-the-art methods and the proposed framework on higher dimensional data, the dimensions are increased to 5000 points. This is done by accessing the raw 10000 feature dataset and running `ASCAD_generate.py` with a window of 5000 around the center point of each window. The code and raw traces publicly available.

6.2 AES_HD Dataset

The AES_HD dataset [7] is a dataset created by measuring EM emission from an unprotected Xilinx Virtex-5 FPGA. The leakage model is defined as follows:

$$y(t_{b_1}, t_{b_2}, k^*) = SBox^{-1}(t_{b_1} \oplus k^*) \oplus t_{b_2}$$

The dataset consists of 45000 profiling traces and 5000 attack traces with each 1250 features.

6.3 CHES_CTF Dataset

The CHES_CTF data set is a data set created for the annual Capture-The-Flag event organized by the Conference on Cryptographic Hardware and Embedded Systems (CHES). The traces are taken from a 32-bit STM Controller running a masked AES-128 encryption algorithm. This data set originates from the year 2018 and is publicly available [1].
From the data set, 45000 profiling and 5000 attack traces are retrieved. All traces have a dimension of 2200 features.

7 Experiments and Experimental Setup

All methods explained in Section 4 are run on the datasets mentioned in Section 6. Each of these methods has a minimum and a maximum amount of Points of Interest that will be

experimented with, an overview is given in Table 7.

Method	Number of POIs
SOST	[2, 4, 8, 16, 32, 64]
SNR	[2, 4, 8, 16, 32, 64]
PCA	[2, 4, 8, 16, 32, 64]
LDA	[2, 3, 4, 5, 6, 7, 8]
Triplet	[2, 4, 8, 16, 32, 64]

Table 7: Points of Interest for each method.

For Wu et al.'s Triplet Network, the hyperparameters chosen are according to the authors' choice as explained in their paper [56].
The proposed framework is benchmarked on the same four datasets. The method runs for a total of 2000 episodes for each dataset to give the algorithm sufficient time to understand the data. The batch size of the traces in the Region Selection phase is set to 32. This is done to reduce the network's computational requirements when increasing the traces' dimensions. At the end of the learning process, the best-performing regions and networks are used to test their performance. This is done by mounting 100 Template Attacks and recording the Guess Entropy.
The framework automatically selects the best performing regions while the amount of regions to be selected exponentially decays over time. Two settings are run to see the impact of starting and ending with a higher number of regions. The first setting selects 50% of the total amount of regions and reduces the amount to 1%. The second setting begins with 75% of the total amount of regions and reduces the amount to 10%.
An ablation study will be conducted to gain insight into the strength of the framework. In this ablation study, the dimensionality reduction network will be omitted during the attack phase to see the impact of using only the best-performing regions.
The traces of each dataset are divided into regions of length 5, this corresponds to the following amount of regions per dataset

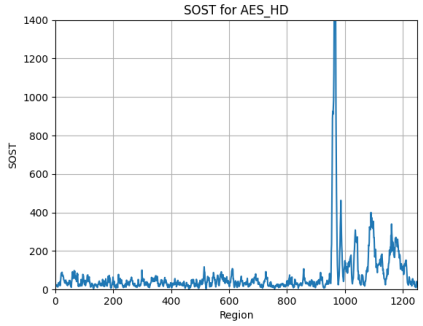
- AES_HD: 250,
- CHES_CTF: 440,
- ASCAD_F: 1000,
- ASCAD_R: 1000.

8 Results and Discussion

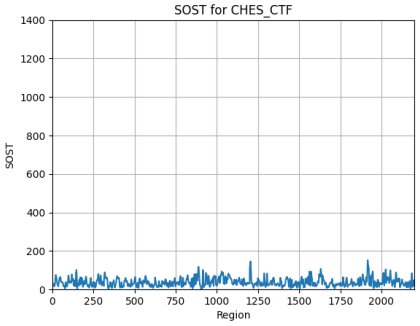
This section displays and discusses the results of the application of the various methods explained in Sections 4 and 5 on each dataset. The results of the current methods are shown and discussed first. Afterward, the strengths and weaknesses of these methods are discussed. Finally, this section ends with the results of the proposed framework.

8.1 SOST

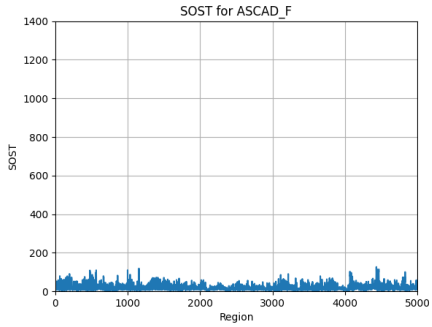
As explained in Section 4.1.1, SOST calculates the squared difference of each point in the whole trace. To give an impression of how SOST selects points of interest, the projection is shown in Figure 15. From the projection on the AES_HD dataset in Figure 15a, a large peak around dimension 950 can be observed. This means that SOST was able to find significant coefficients around dimension 950, which can be interpreted as a significant leakage from the AES_HD dataset. This probably is because the AES_HD dataset is unprotected, as mentioned in Section 6.2. In the Figures 15b-15d no peaks are observed. Meaning that there were no significant coefficients found and thus that SOST cannot detect leakage within these datasets. These results probably stem from the fact that these datasets are masked.



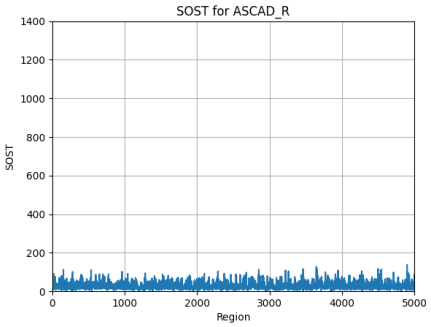
(a) SOST on AES_HD.



(b) SOST on CHES_CTF.



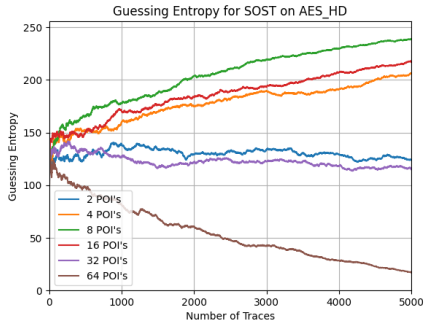
(c) SOST on ASCAD_F.



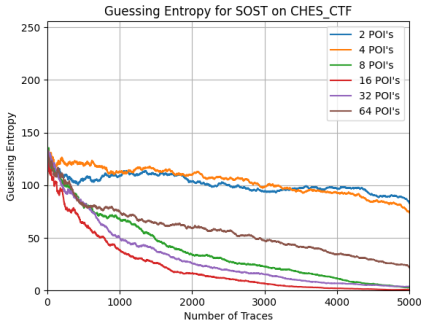
(d) SOST on ASCAD_R.

Figure 15: The projection of the SOST on the features of each dataset.

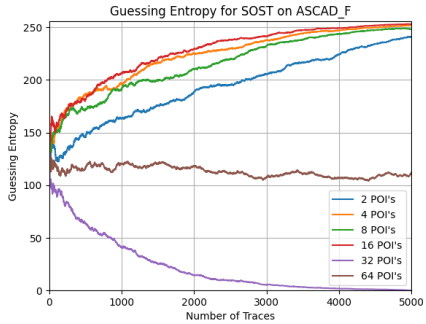
From this projection, the top-n maximum points are used as features for the attack. The amount of points for the attacks is depicted in Table 7. The GE is calculated for each of the datasets over 100 attacks and is displayed in Figure 16.
Although the projection in Figure 15a showed the detection of a significant leakage by SOST, the results from the attack, as depicted in Figure 16a, shows very conflicting results. It seems that SOST cannot break the AES_HD dataset within the allotted amount of attack traces.



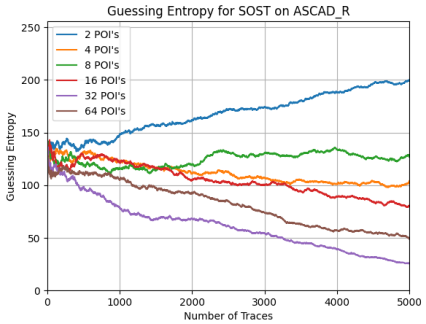
(a) GE of SOST on AES_HD.



(b) GE of SOST on CHES_CTF.



(c) GE of SOST on ASCAD_F.



(d) GE of SOST on ASCAD_R.

Figure 16: Performance of SOST on all data sets averaged over 100 attacks.

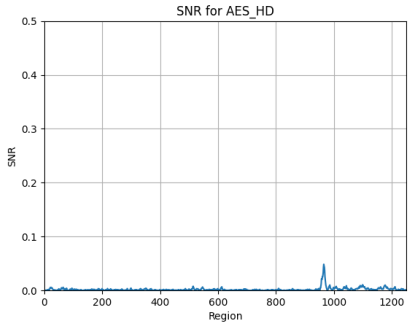
Although, it is observed that with more attack traces, SOST with 64 POIs would be able to break the target.
The results in Figure 16b and Figure 16c show that both the CHES_CTF and ASCAD_F dataset can be broken when using SOST. This again is conflicting with the results shown in their projections in Figure 15b and Figure 15c.
It is observed from Figure 16a that the best performing amount of POIs for the AES_HD dataset is 64. The results indicate a GE_0 of more than the allotted amount of attack traces. For the CHES_CTF the best performing POIs was 16 and resulted in a GE_0 of 4510, as observed in Figure 16b. For ASCAD_F and ASCAD_R, the best performing POIs is 32. It seems that for both datasets, more POIs perform better than lower amounts of POIs. For ASCAD_F this resulted in a GE_0 of 4522, as observed in Figure 16c. For ASCAD_R, as observed in Figure 16d, the results indicate a GE_0 of more than the allotted amount of attack traces. An overview of the best performing amount of POIs is found in Table 8.

Dataset	Best Performing amount of POIs	GE_0
AES_HD	64	> 5000
CHES_CTF	16	4510
ASCAD_F	32	4522
ASCAD_R	32	> 5000

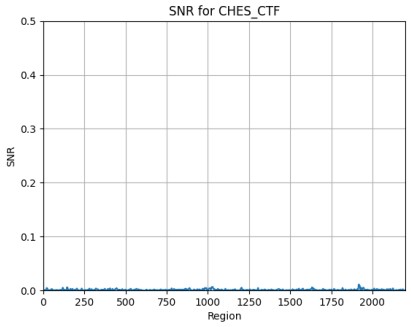
Table 8: The GE of the best performing number of POIs on each dataset for SOST.

8.2 SNR

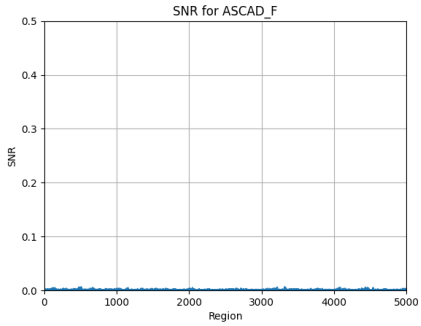
The SNR is the ratio between the relevant signal and the irrelevant noise inside the data. As with SOST, the SNR can also be projected on each feature. The SNR for each dataset is shown in Figure 17.



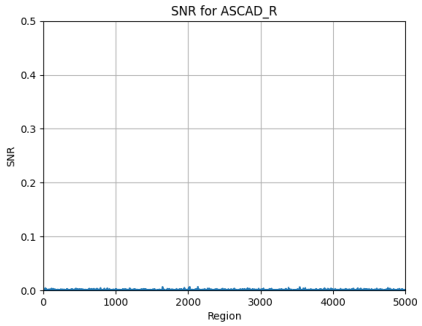
(a) SNR of the AES_HD dataset.



(b) SNR of the CHES_CTF dataset.



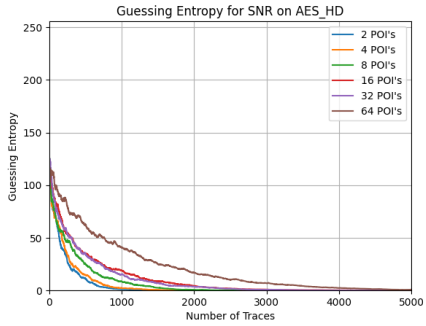
(c) SNR of the ASCAD_F dataset.



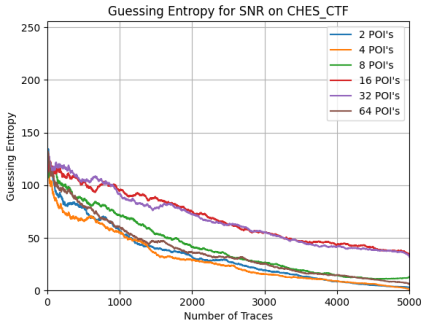
(d) SNR of the ASCAD_R dataset.

Figure 17: The projection of the SNR on the features of each dataset.

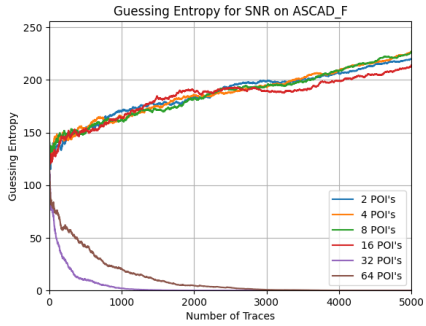
From Figure 17a, a small peak can be observed around feature 950. This means that SNR can detect a leakage from the AES_HD dataset. The same reasoning mentioned in Section 8.1 can be applied here. This detection is probably because the AES_HD dataset is unprotected. The SNR from the Figures 17b-17d show no significant peaks. Again, as mentioned in Section 8.1, this is probably because these datasets are protected with masking. The GE is calculated for each of the datasets over 100 attacks and is displayed in Figure 18.
It is observed, in Figure 18a, that SNR is able to break the target of the AES_HD dataset, which is in line with the fact that SNR was able to detect leakage information, as was shown in Figure 17a. From Figure 18b, it is observed that SNR is not able to to break



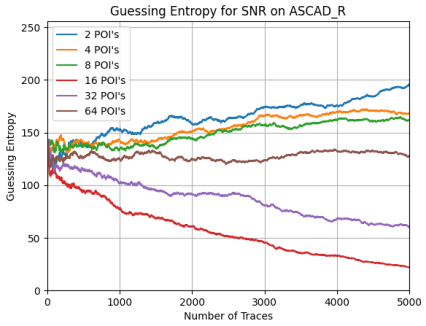
(a) GE of SNR on AES_HD.



(b) GE of SNR on CHES_CTF.



(c) GE of SNR on ASCAD_F.



(d) GE of SNR on ASCAD_R.

Figure 18: Performance of SNR on all data sets averaged over 100 attacks.

the target for the CHES_CTF dataset. Although, the results indicate that with more traces, CHES_CTF would have been broken as well. The results on the ASCAD_F dataset, displayed in Figure 18c, show that SNR is able to break the target. This result is not entirely in line with what was observed in Figure 17c since no leakage could be detected by SNR. Since there was no leakage detected by SNR and the results show that with more POIs it is still able to break the target, it could be that by chance, the correct features were selected. The ASCAD_R dataset, which can be seen in Figure 18d, is not able to be broken with SNR. Although, the results indicate that with more attack traces, the ASCAD_R dataset would have been broken as well. Again the results indicate that, on average, higher amounts of POIs give better results.

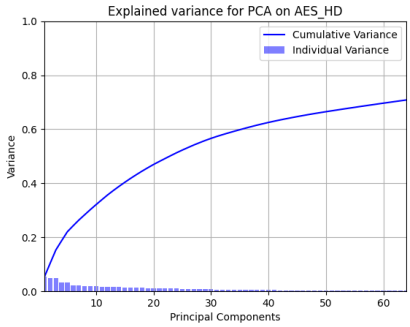
The best performing amount of POIs, according to the results in Figure 18a, for the AES_HD dataset is 2 with a GE_0 of 1094. It is interesting to see that the results indicate that more POIs is detrimental to the performance. For the CHES_CTF dataset, the best performing amount of POIs is 4 with a GE_0 of more than the allotted amount of attack traces, as can be observed from Figure 18b. As mentioned before, the results indicate that with more attack traces, all amounts of POIs would be able to break the target. As with the AES_HD dataset, it seems that for CHES_CTF, lower amounts of POIs perform better. For the ASCAD_F dataset, the best performing amount of POIs is 32 with a GE_0 of 1184. As mentioned before, this result may be due to chance. An increase in selected POIs could lead to selecting the correct combination of features. The results in Figure 18d show that the best performing amount of POIs for the ASCAD_R dataset is 16 with a GE_0 of more than the allotted amount of attack traces. Again, on average, an increase of POIs shows an increase in performance. An overview of the best performing amount of POIs is found in Table 9.

Dataset	Best Performing amount of POIs	GE_0
AES_HD	2	1094
CHES_CTF	4	> 5000
ASCAD_F	32	1184
ASCAD_R	16	> 5000

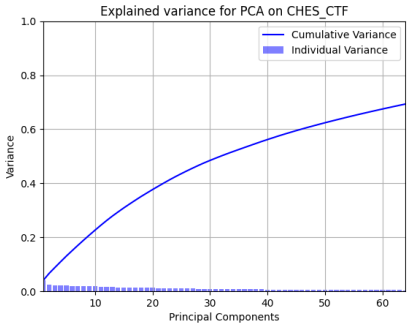
Table 9: The GE_0 of the best performing number of POIs on each dataset for SNR

8.3 PCA

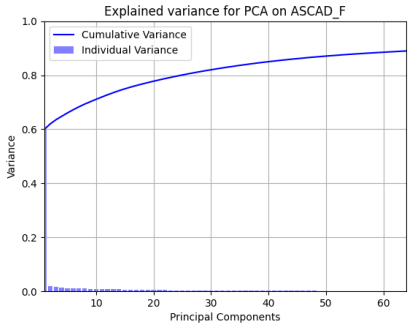
As explained in Section 4.2.1, PCA constructs Principal Components that contain a certain percentage of the variance in the data. This variance can be plotted for each Principal Component, these plots are called scree plots. A scree plot for each dataset is displayed in Figure 19



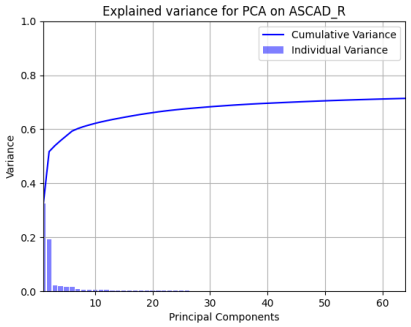
(a) Scree plot of PCA on AES_HD



(b) Scree plot of PCA on CHES_CTF



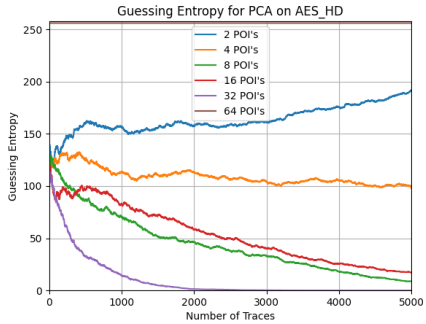
(c) Scree plot of PCA on ASCAD_F



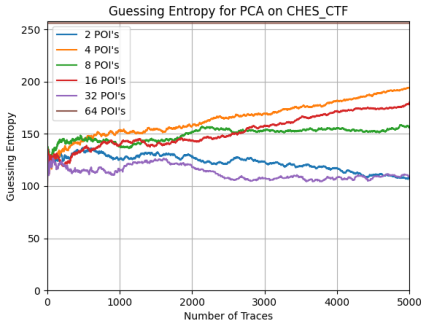
(d) Scree plot of PCA on ASCAD_R

Figure 19: Scree plot of PCA on all datasets.

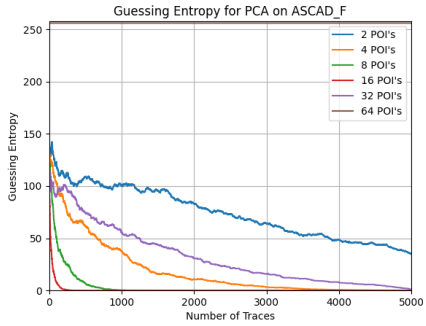
From the scree plot in Figure 19a, it is observed that PCA has issues decomposing the dataset, where the first 10 Principal Components only hold 34% of the explained variance. The same is observed in Figure 19b for the CHES_CTF dataset, where the first 10 Principal Components hold even less of the explained variance, only 22%. PCA has fewer issues decomposing the ASCAD dataset. From the scree plot for ASCAD_R in Figure 19c, it is observed that already 72% of the explained variance resides in the first 10 Principal Components. The scree plot for ASCAD_R in Figure 19d shows an explained variance of 62% for the first 10 Principal Components.



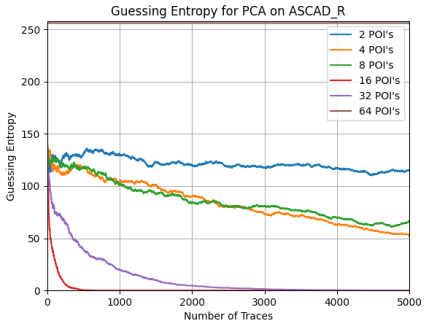
(a) GE of PCA on AES_HD



(b) GE of PCA on CHES_CTF



(c) GE of PCA on ASCAD_F



(d) GE of PCA on ASCAD_R

Figure 20: Performance of PCA on all data sets averaged over 100 attacks.

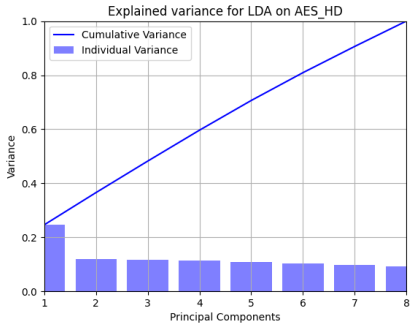
Figure 20a displays that PCA is able to break the AES_HD dataset. In contrast, the CHES_CTF could not be broken with PCA. These results correlate with what was depicted in their respective scree plots in Figure 19a and Figure 19b. Both datasets have a hard time being decomposed. The results for the ASCAD_R dataset, shown in Figure 20c, show that PCA can break it relatively easily. This is in line with the scree plot in Figure 19c, where it was observed that the ASCAD_F dataset could be decomposed fairly well. The same holds for the ASCAD_R dataset, as observed in Figure 20d, where PCA can break the dataset relatively easily. It is strange to see that that PCA completely breaks when using 64 POIs. This is likely due to selecting such low variance Principal Components that the transformation matrix is muddled and cannot transform the features concisely.
For the AES_HD dataset the best performing amount of POIs, as observed from Figure 20a, is 32 with a GE_0 of 2178. These 32 Principal Components correspond to 58% of the total explained variance. As observed from Figure 20b, the best performing amount of POIs is 32 with a GE_0 of more than the allotted amount of traces. These 32 Principal Components correspond to 69% of the total explained variance. Figure 20c shows that the best performing amount of POIs for the ACSCAD_F dataset is 16 with a GE_0 of 203. The 16 Principal Components together hold 76% of the total variance. ASCAD_R, as is depicted in Figure 20d, also indicate that the best performing amount of POIs is 16 with a GE_0 of 447. For ASCAD_R, these 16 Principal Components hold 72% of the total variance. The best performing number of Principal Components can be seen in Table 10.

Dataset	Best Performing amount of POIs	GE_0
AES_HD	32	2178
CHES_CTF	32	> 5000
ASCAD_F	16	203
ASCAD_R	16	447

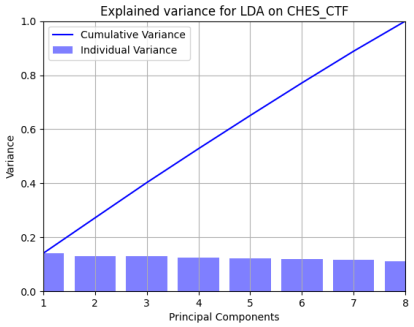
Table 10: The GE_0 of the best performing number of POIs on each dataset for PCA

8.4 LDA

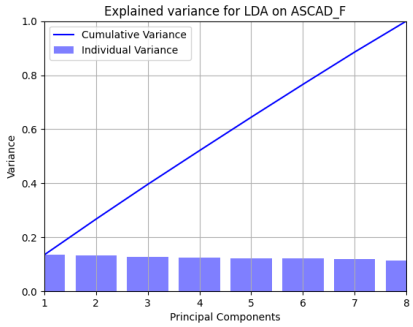
As with PCA, LDA creates a decomposition into eigenvalues as well, as explained in Section 4.2.2. Since this is a supervised method, the maximum number of components equals the number of classes minus one. In the case of the Hamming Weight, the number of classes is nine; thus, only eight components can be created. As with PCA, each component can be depicted in a scree plot. Each of the scree plots for each dataset is shown in Figure 21.



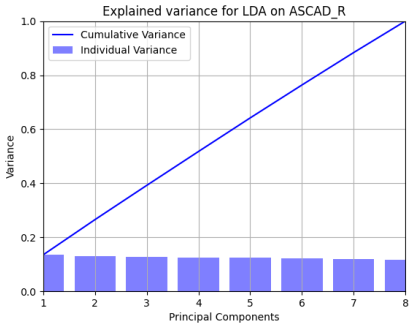
(a) Scree plot of LDA on AES_HD



(b) Scree plot of LDA on CHES_CTF



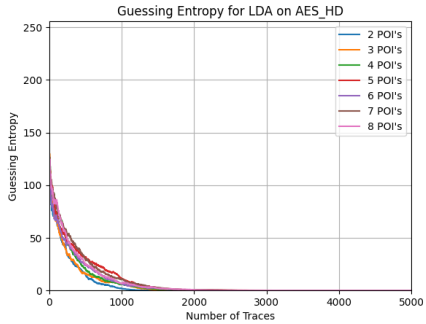
(c) Scree plot of LDA on ASCAD_F



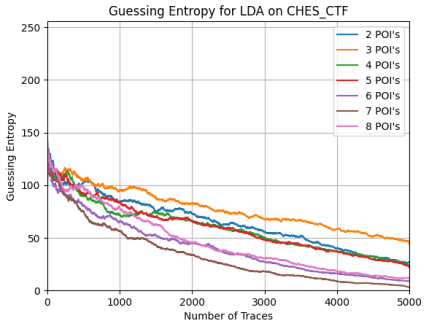
(d) Scree plot of LDA on ASCAD_R

Figure 21: Scree plot of LDA on all datasets.

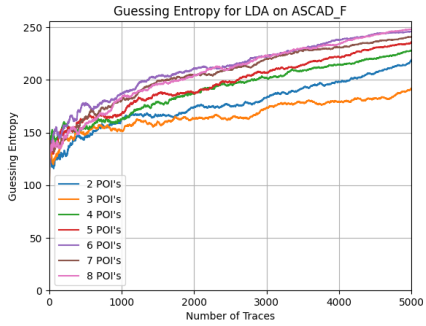
It can be seen from Figure 21a that the AES_HD dataset is the only dataset that has a higher starting explained variance than the other datasets. This might be due to the fact that the AES_HD dataset is unprotected. For the other datasets, as depicted in Figures 21b-21d, the explained variance is equally distributed over all components.
In Figure 22a it is observed that LDA is able to break the AES_HD dataset. This is likely due to the observed results from the scree plot in Figure 21a. A higher explained variance in the first component seems to affect the performance positively. Especially considering the



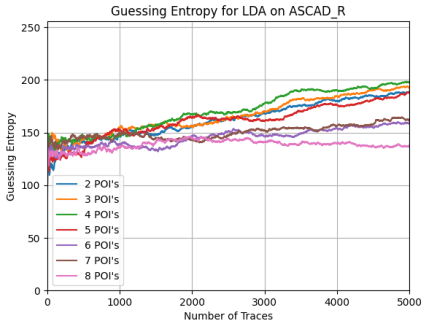
(a) GE of LDA on AES_HD



(b) GE of LDA on CHES_CTF



(c) GE of LDA on ASCAD_F



(d) GE of LDA on ASCAD_R

Figure 22: Performance of LDA on all data sets averaged over 100 attacks.

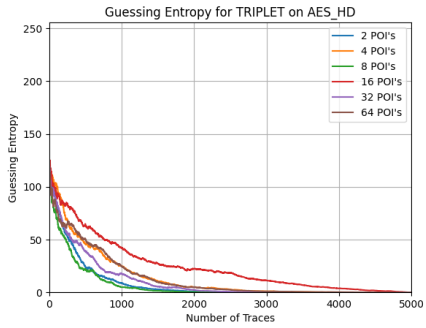
results observed for the other datasets in Figure 22b-22d. It is apparent that none of the datasets are broken with LDA. Although, it seems that the CHES_CTF dataset would be able to be broken when given more attack traces, as observed in Figure 22b.
From Figure 22a it can be seen that the best performing amount of POIs for the AES_HD dataset is 2 with a GE_0 of 1104. It is also observed that all amounts of POIs can break the target. For the CHES_CTF, the best performing POIs is 7, as is observed in Figure 22b, with a GE_0 of more than the allotted amount of attack traces. For ASCAD_R and ASCAD_F it is apparent, form their respective Figures 22c and 22d, that no amount of extra traces would break the target. Their best performing amount of POIs are 3 and 8 with a GE_0 of more than the allotted amount of attack traces. An overview of the best performing amount of POIs is found in Table 11.

Dataset	**Best Performing amount of POIs**	**GE_0**
AES_HD	2	1104
CHES_CTF	7	> 5000
ASCAD_F	3	> 5000
ASCAD_R	8	> 5000

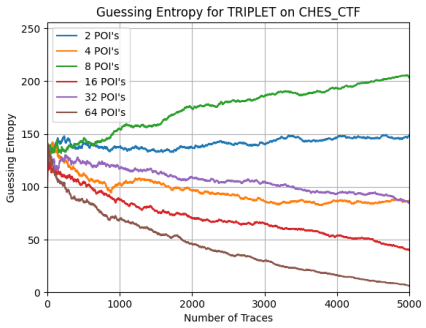
Table 11: The GE_0 of the best performing number of POIs on each dataset for LDA

8.5 Wu et al. Triplet Network

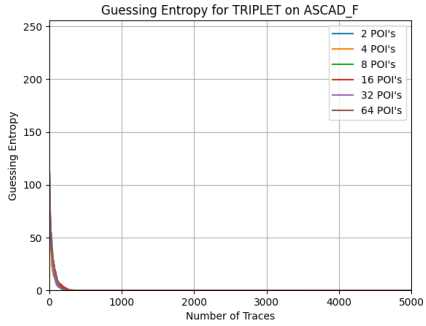
The Triplet Network is a Deep Learning method, meaning that it uses a ANN to transform the features. The triplet network learns to differentiate between good and bad features by minimizing a loss function. The loss function uses the distance between traces with the same label and the distance between traces with differing labels. The Triplet Network represents the current state-of-the-art in Points Of Interest selection and this can be clearly seen in the results in Figure 23.



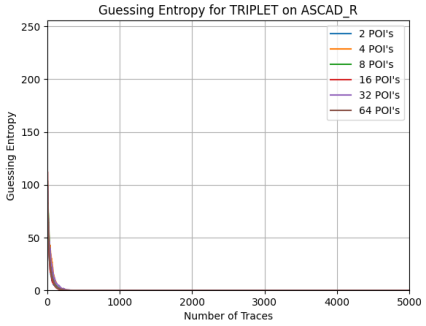
(a) Triplet on AES_HD.



(b) Triplet on CHES_CTF.



(c) Triplet on ASCAD_F.



(d) Triplet on ASCAD_R.

Figure 23: Performance of the Triplet Network on all data sets averaged over 100 attacks.

From the results in Figure 23a, it is observed that the Triplet Network can break the AES_HD dataset. From Figure 23b, it can be seen that the Triplet Network can not break the CHES_CTF dataset within the allotted amount of attack traces. It does seem that given more traces, the CHES_CTF dataset would also be broken. This is likely due to the network not being fine-tuned for this dataset. Both the ACAD_F and ASCAD_R dataset are easily broken by the Triplet Network as depicted in Figure 23c and Figure 23d.
In Figure 23a it is shown that the best performing amount of POIs for the AES_HD dataset is 8 with a GE_0 1668. The results also indicate that every amount of POIs can be used to break the target. For the CHES_CTF, according to Figure 23b, the best performing amount of POIs is 64 with a GE_0 of more than the allotted amount of attack traces. For the ASCAD_F and ASCAD_R datasets, the best performing amount of POIs is 32 and 16 with a GE_0 of 194 and 165, respectively. It is interesting to see that the amount of POIs has almost no bearing on the performance, as depicted in Figure 23c and Figure 23d. An overview of the best performing amount of POIs is shown in Table 12.

Dataset	Best Performing amount of POIs	GE_0
AES_HD	8	1668
CHES_CTF	64	> 5000
ASCAD_F	32	194
ASCAD_R	16	165

Table 12: The GE_0 of the best performing number of POIs on each dataset for the Triplet Network

8.6 Discussion of the Current Methods

Each method was benchmarked on the four datasets, and the result varied wildly. Of the five methods, none were able to break each target consistently. However, this does not mean that all these methods should be disregarded.
From the results discussed in Section 8.1, it was observed that SOST was able to detect leakages on the AES_HD dataset but was inconstant in being able to break the target. While no leakages were detected for the ASCAD_F and CHES_CTF datasets, SOST was able to break the target. It seems that SOST suffers from being a very inconsistent method to perform POI selection.
SNR suffers less from this inconsistency, as discussed in Section 8.2. The leakage detected on the AES_HD dataset resulted in breaking the target well within the allotted amount of attack traces. However, the ACSCAD_F dataset was again broken, despite no leakage detected by SNR. Both Feature Selection methods seem inconsistent when dealing with the dataset.
PCA performs well on all datasets except the CHES_CTF dataset, as discussed in Section 8.3. This was especially apparent from the scree plots made for each dataset. From the scree plots, it was observed that PCA had difficulties decomposing the CHES_CTF dataset the most, resulting in the worst performance of the four datasets. The other datasets were fairly easily broken, indicating the strength of PCA.
In contrast with PCA, LDA performed very poorly. As discussed in Section 8.4, LDA was only able to break one dataset, the AES_HD dataset. It seems that datasets that are protected by masking are not able to be decomposed very well by LDA. Although, when used on unprotected datasets, such as the AES_HD dataset, it outperforms PCA. It seems that LDA suffers the most from the fact that the transformation of the original traces results in information loss.
The Triplet Network represents the current state-of-the-art when used for POI selection. This was indeed observed from the results discussed in Section 8.5. The only dataset not broken by the Triplet Network was the CHES_CTF dataset. All other datasets were broken well within the allotted amount of attack traces. It even outperformed PCA on all three datasets. As mentioned before, this is the main weakness of the Triplet Network. If the network is not adequately fine-tuned for the dataset, it cannot break the target. However, it is to be expected that if tuned correctly, it would be able to break the CHES_CTF dataset.
An overview of the performance of each method on each dataset is shown in Table 13. The best performing POI selection method for that dataset is depicted in bold.

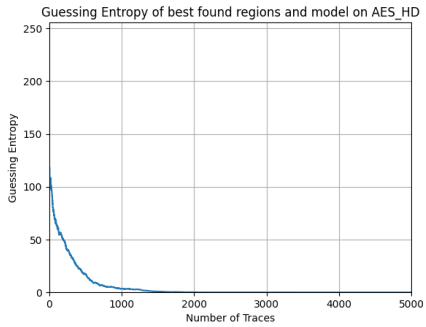
Dataset	SOST	SNR	PCA	LDA	Triplet
AES_HD	-	**1094**	2513	1104	1664
CHES_CTF	**4510**	-	-	-	-
ASCAD_F	4522	1184	203	-	**194**
ASCAD_R	-	-	452	-	**164**

Table 13: A summary of the results of each method on each of the four datasets.

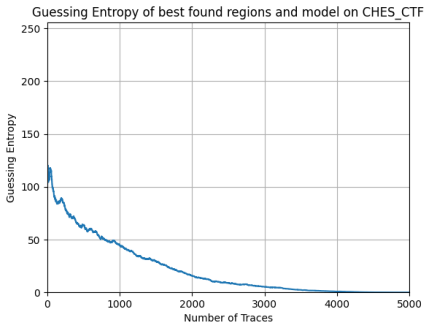
8.7 Proposed framework

The proposed framework was run on each dataset for a total of 2000 episodes, in which, during training, the network found, selected, and scaled-down various amounts of Points Of Interest. During training, one set of regions and one neural network performed the best out of all others for each dataset. The best-found set of regions and networks for each dataset is found in Appendix A.

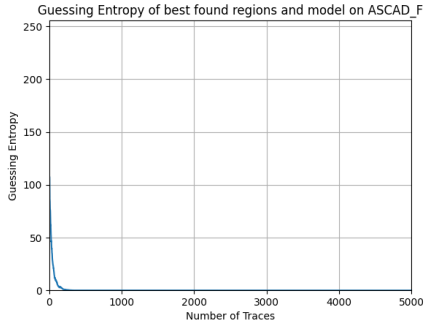
Performance of best-found regions and networks on each dataset After training, the best-found regions and networks were benchmarked for 100 Template Attacks on their respective dataset. The results of the attacks are found in Figure 24.



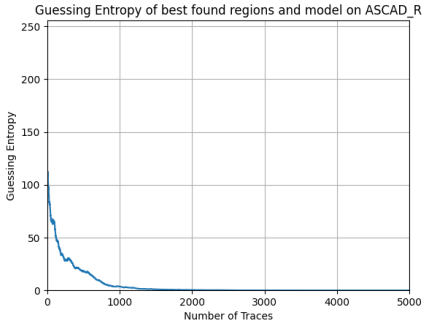
(a) GE of best found regions and model on AES_HD.



(b) GE of Best found regions and model on CHES_CTF.



(c) GE of Best found regions and model on ASCAD_F.



(d) GE of Best found regions and model on ASCAD_R.

Figure 24: Performance of the Proposed framework on all data sets averaged over 100 attacks.

As seen from Figure 24a, the proposed framework can break the target of the AES_HD dataset, which means that an unprotected dataset can be broken with the proposed frame-

work. The proposed framework can break the CHES_CTF dataset, as is observed in Figure 24b. Both ASCAD_F and ASCAD_R can be broken well within the allotted amount of attack traces, as can be seen in Figure 24c and Figure 24d. This means that the proposed framework can break even a protected dataset. Table 13 extends Table 14 by adding the exact GE_0 of the proposed framework for each dataset.

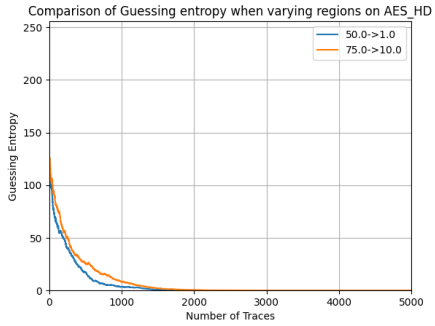
Dataset	SOST	SNR	PCA	LDA	Triplet	Proposed Framework
AES_HD	-	**1094**	2513	1104	1664	1430
CHES_CTF	4510	-	-	-	-	**3962**
ASCAD_F	4522	1184	203	-	194	**193**
ASCAD_R	-	-	452	-	**164**	1499

Table 14: A summary of the results of each method on each of the four datasets.

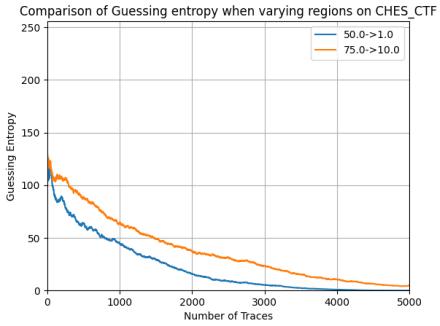
As seen from Table 14, the proposed framework is the only method that can break all four datasets consistently—even achieving state-of-the-art results on the CHES_CTF and ASCAD_F datasets.

Impact of varying regions sizes Two settings were tested to see the impact of having access to more initial regions and more minimum regions, as mentioned in Section 7. Both settings were trained for 2000 episodes, and the best-performing regions and networks were selected to benchmark their respective datasets. A comparison between both settings is shown in Figure 25.
The results from Figure 25a indicate that having access to more initial regions and more minimum regions is detrimental to the performance on the AES_HD dataset. The same results are observed for the CHES_CTF dataset, in Figure 25b, and the ASCAD_F dataset, as shown in Figure 25c. Only the ASCAD_R dataset, as observed in Figure 25d, performs better with more initial and minimum regions.

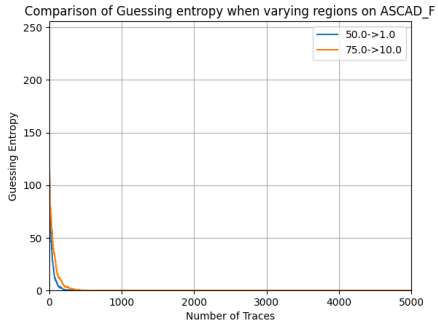
Insight in the learning process To gain insight into the proposed framework's learning process, the rewards of the training over the course of 2000 episodes are depicted in Figure 26 for the AES_HD and CHES_CTF, and in Figure 27 for ASCAD_F and ASCAD_R. The rewards are aggregated per 10 episodes to make them less cluttered. The original figures are shown in Appendix B.
The proposed framework seems to have difficulty propagating what was learned throughout the various episodes. This phenomenon can be clearly seen for the AES_HD dataset in Figure 26a and Figure 26b, the highest rewards are found at the beginning of the learning process. This indicates that the proposed framework might not be able to learn where the most important POIs are. Although, the performance of the found regions and network, as observed from Figure 24a, tells that the proposed network can break the target.
For the CHES_CTF dataset, this is less obvious, as seen in Figure 26c and Figure 26d. Here the framework can hold the average reward fairly steady throughout the learning process. However, this seems to indicate that the framework struggles to learn the correct combination of regions and network.
For the ASCAD_F dataset, this phenomenon was observed less, which is most apparent in Figure 27b. It is observed that here the average rewards returned are greater at the end of the episodes than in the beginning. This might be due to the peak in average reward at the beginning of the learning process. This indicates that if the framework explores a very good



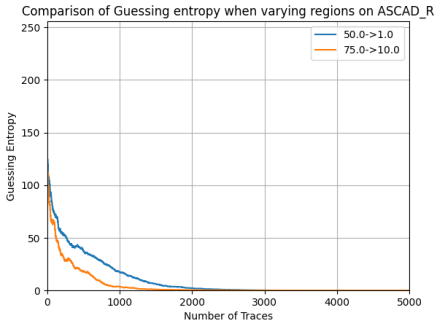
(a) Impact of regions on AES_HD.



(b) Impact of regions on CHES_CTF.



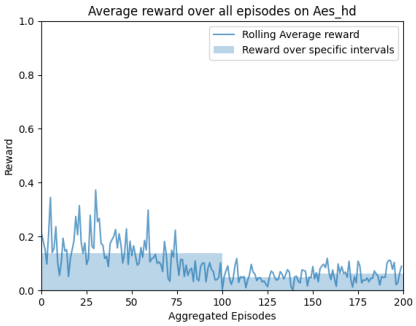
(c) Impact of regions on ASCAD_F.



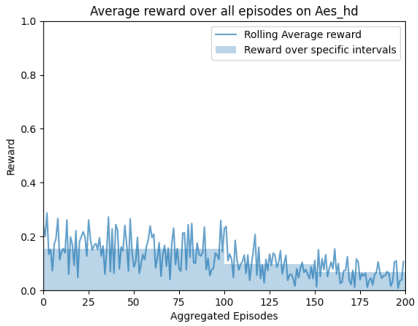
(d) Impact of regions on ASCAD_R.

Figure 25: Comparison of performance when increasing the initial regions and the minimum regions.

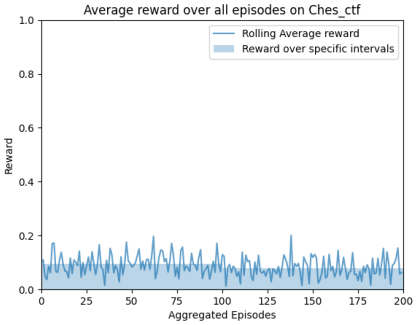
set of regions and networks at the beginning of the learning process, the framework can gain significant momentum.
The ASCAD_R dataset also suffered less from this phenomenon, which is apparent in the results in Figure 27c. Here the results indicate that the higher average rewards tend to be at the end of the training instead of the beginning.



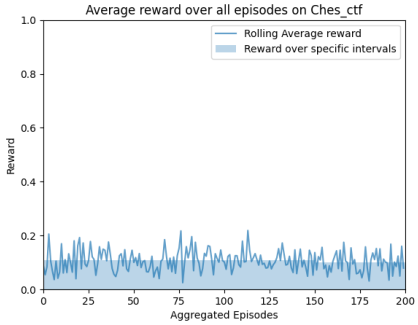
(a) Rewards averaged over 10 episodes for all 2000 episodes on AES_HD 50.0% to 1.0%.



(b) Rewards averaged over 10 episodes for all 2000 episodes on AES_HD 75.0% to 10.0%.



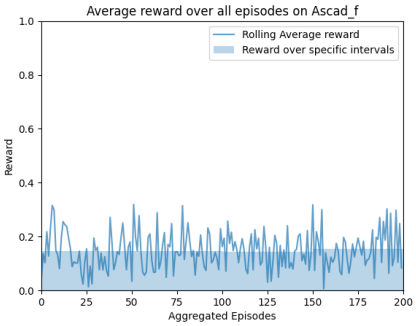
(c) Rewards averaged over 10 episodes for all 2000 episodes on CHESC_CTF 50.0% to 1.0%.



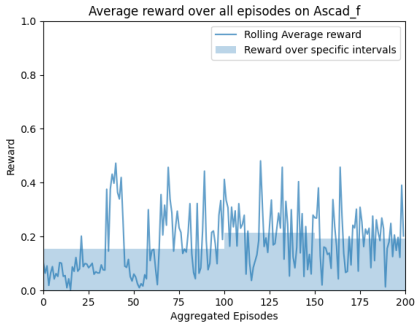
(d) Rewards averaged over 10 episodes for all 2000 episodes on CHESC_CTF 5.0% to 10.0%.

Figure 26: Rewards throughout the episodes on each dataset.

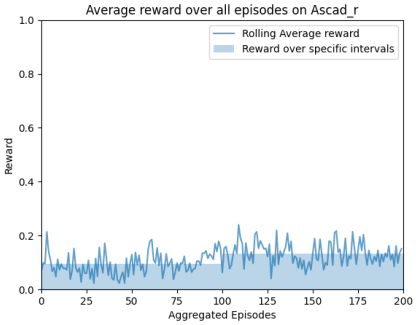
Average reward of the regions selected Each region can be selected multiple times throughout the learning process. To display the overlap of the selected regions with the SNR, both have been displayed in Figure 28 for the AES_HD and CHES_CTF datasets and in Figure 29 for the ASCAD_F and ASCAD_R datasets. To see the impact of increasing the available regions, a distinction is made between the settings when displaying the figures. The region with the highest average reward is indicated in red.
For the AES_HD dataset, it is observed from Figure 28a that the framework is able to detect the leakage when compared to the SNR. Access to more regions indicates, as is displayed in Figure 28b, a detrimental effect on the average reward. For the CHES_CTF it is apparent that the SNR does not overlap with the best performing region, as indicated in Figure 28c and Figure 28d. This is likely because the dataset is protected, and thus the SNR cannot detect a leakage, as was already discussed in Section 8.2. Since the results in Figure 24b indicate that the framework can break the target, it might be that the leakage lies around the highest point. Again it is observed in Figure 28d that having access to more regions is detrimental to the overall performance of the framework.
For the ASCAD_F dataset, the impact of having access to more regions is striking. It is observed in Figure 29a that the average reward of all regions is around 0.4 while the average reward in Figure 29b is around 0.1. It is also interesting to see that the best performing regions lie very far apart.



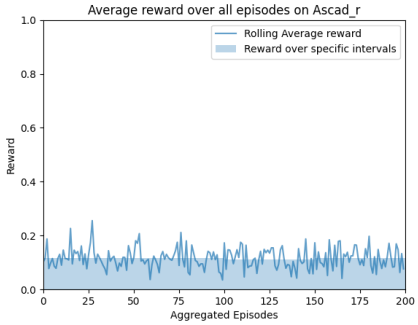
(a) Rewards averaged over 10 episodes for all 2000 episodes on ASCAD_F 50.0% to 1.0%.



(b) Rewards averaged over 10 episodes for all 2000 episodes on ASCAD_F 75.0% to 10.0%.



(c) Rewards averaged over 10 episodes for all 2000 episodes on ASCAD_R 50.0% to 1.0%.

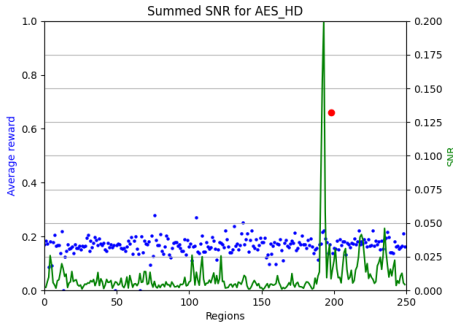


(d) Rewards averaged over 10 episodes for all 2000 episodes on ASCAD_R 75.0% to 10.0%.

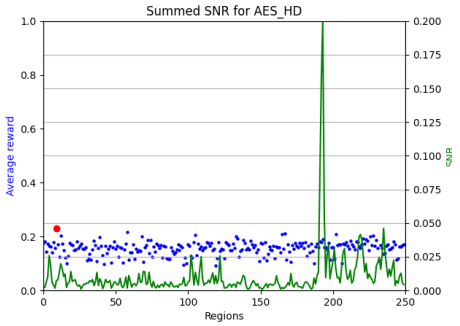
Figure 27: Rewards throughout the episodes on for ASCAD_F and ASCAD_R.

The average reward per regions for the ASCAD_R dataset seems to be roughly the same, as observed in Figure 29c and Figure 29d. Here the best performing regions seem to overlap, seemingly indicating that having access to more regions does not impose a significant detrimental effect. Although, the best performing region in Figure 29c has a higher average reward compared to the best performing region in Figure 29d.

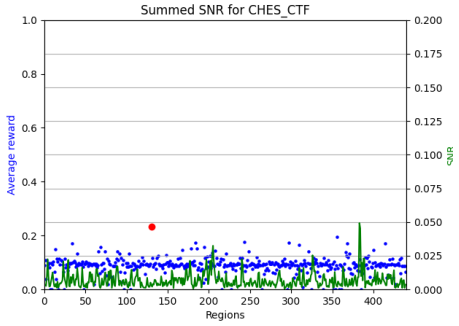
Ablation study An ablation study was conducted to see the effect of scaling down the selected regions with the neural network. Each selected region was isolated and used as the features for 100 TAs. The GE for the performance of the regions is displayed in Figure 30. The difference in results is striking when compared to the results of the full usage of the framework, as displayed in Figure 24. Especially for the AES_HD dataset, as observed in Figure 30a, where the regions alone cannot break the target, let alone provide any meaningful key guess. The results for the CHES_CTF dataset indicate that the regions selected with access to more regions alone cannot provide any meaningful key guess, as seen in Figure 30b. The results indicate that the regions selected, when having access to fewer regions, would be able to break the target when given more attack traces.
For the ASCAD_F dataset, both settings provide meaningful key guesses, but, as the results indicate in Figure 30c, the regions alone cannot break the target. There is also no indication that they would if given more traces. In Figure 30d, it is observed that, again, the regions alone cannot break the target. Even resulting in no meaningful key guess when having access



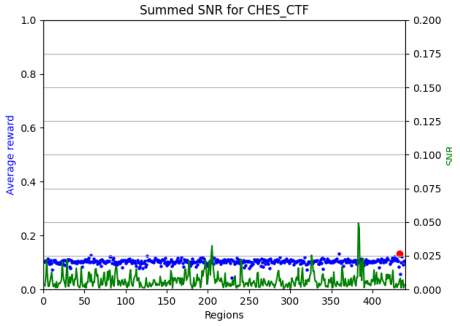
(a) Average reward per selected region on AES_HD, using 50% initial regions and 1% minimum regions



(b) Average reward per selected region on AES_HD, using 75% initial regions and 10% minimum regions



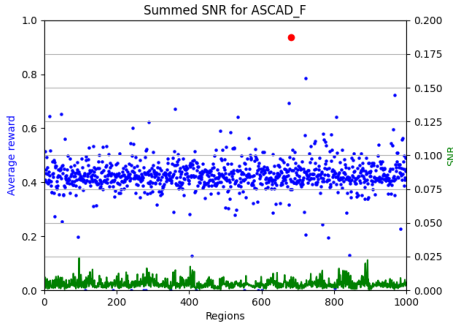
(c) Average reward per selected region on CHES_CTF, using 50% initial regions and 1% minimum regions



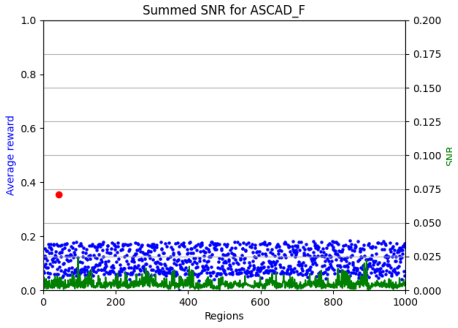
(d) Average reward per selected region on CHES_CTF, using 75% initial regions and 10% minimum regions

Figure 28: Average reward of each selected region versus the SNR per setting for AES_HD and CHES_CTF. The red dot indicates the highest rewarding region.

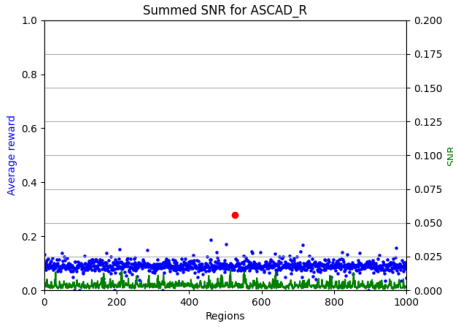
to more regions.



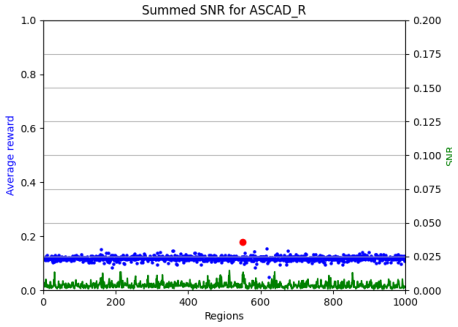
(a) Average reward per selected region on ASCAD_F, using 50% initial regions and 1% minimum regions



(b) Average reward per selected region on ASCAD_F, using 75% initial regions and 10% minimum regions

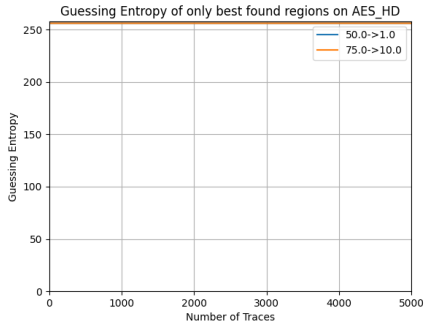


(c) Average reward per selected region on ASCAD_R, using 50% initial regions and 1% minimum regions

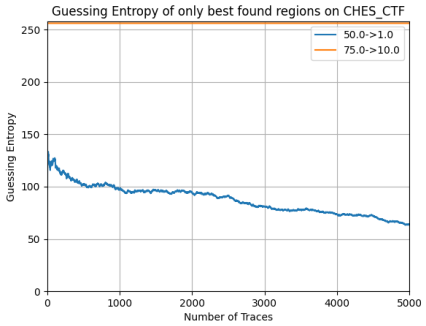


(d) Average reward per selected region on ASCAD_R, using 75% initial regions and 10% minimum regions

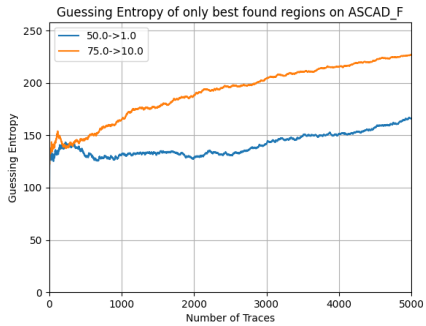
Figure 29: Average reward of each selected region versus the SNR per setting for ASCAD_F and ASCAD_R. The red dot indicates the highest rewarding region.



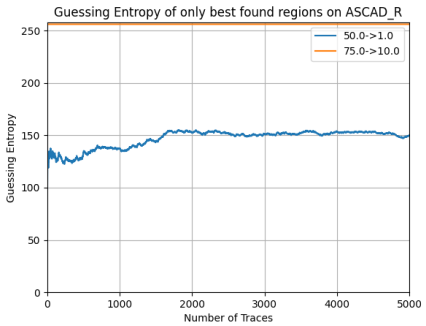
(a) GE of only selected regions on AES_HD.



(b) GE of only selected regions on CHES_CTF.



(c) GE of only selected regions on ASCAD_F.



(d) GE of only selected regions on ASCAD_R.

Figure 30: Performance of only the selected regions on all data sets averaged over 100 attacks.

9 Conclusion

This thesis explored various current state-of-the-art methods of Points Of Interest selection. Each method was introduced in Section 4. Furthermore, this thesis proposed a novel framework based on Proximal Policy Optimization, introduced in Section 5. The current methods and framework were benchmarked on four commonly used datasets, the AES_HD dataset, the CHES_CTF dataset, the ASCAD_F, and the ASCAD_R dataset. The dataset were introduced in Section 6. To see the performance of the methods and framework, several experiments were conducted. The experiments and experimental setup were outlined in Section 7. Each of the results of the methods and framework was discussed in Section 8. Next to an overview of the current methods, this thesis introduces a novel Points Of Interest selection method based on Proximal Policy Optimization. The framework was also benchmarked on the four datasets, and the results are discussed in Section 8.7.
Three main research questions were created at the beginning of this thesis. Research questions 1 and 2 are divided into sub-question, which will be answered first before answering the main research questions.

1.a What are the current state-of-the-art techniques used to select Points Of Interest? In Section 4, an introduction was given to five methods commonly used within Points Of Interest selection. Three categories of POI selection were explored in this thesis:

- Feature Selection methods
- Dimensionality Reduction methods
- Deep Learning methods

Feature Selection methods select a subset of the original features as Points Of Interest. These Points Of Interest are then used to mount an attack. For Dimensionality Reduction methods, the original traces are transformed in a new set of features. These new features are used as Points Of Interest to mount an attack. Deep Learning methods use an Artificial Neural Network to transform the features in a new set of features. The new features represent the Points Of Interest and are used to mount an attack.

The Feature Selection methods explored in this thesis were Sum Of Squared T-Differences and Signal-to-Noise Ratio. Sum Of Squared T-Differences calculates coefficients for each feature from the mean of the traces and the variance within the traces. The top-n features are then selected as Points Of Interest. Signal-to-Noise Ratio calculates the ratio of the relevant signal and the unwanted noise for each feature within a trace. The top-n features with the highest Signal-to-Noise Ratio are used as Points Of Interest.

For the Dimensionality Reduction methods Principal Component Analysis and Linear Discriminant Analysis were explored. Principal Component Analysis performs an eigenvalue decomposition to create a transformation matrix. The transformation matrix consists of eigenvectors corresponding to the top-n largest eigenvalues. The features from the traces are then transformed with the transformation matrix, and the new set of features is used as Points Of Interest. Linear Discriminant Analysis performs a supervised decomposition of the data into an optimized linear combination of variables. This linear combination is based on two matrices that consist of the variance within the classes and between the classes. Since Linear Discriminant Analysis is supervised, it can only have as many linear components as there are classes minus one. Linear Discriminant Analysis construct a transformation matrix from the linear

components. The transformation matrix is then used to transform the features within the traces into a new set of traces, which are subsequently used as Points Of Interest.

For the Deep Learning method, the current state-of-the-art method the Triplet Network was explored. The Triplet Network consists of a Convolutional Neural Network, which is used to extract relevant features. In addition, the Triplet Network uses a loss function to minimize the distance between traces that share labels while, at the same time, maximizing the distance between traces that have differing labels. Once the network is trained, the traces are run through the network, which produces new features. These new features are then used as Points Of Interest.

1.b What are the most prominent pros and cons of each method? In Section 8, each of the five methods was benchmarked on four commonly used datasets, the AES_HD dataset, the CHES_CTF dataset, the ASCAD_F dataset, and the ASCAD_R dataset. The results show that most methods have trouble breaking the CHES_CTF dataset, while the AES_HD dataset and ASCAD_F dataset seems to be broken easily.

Out of all methods, the worst performing method was LDA. This method was only able to break the AES_HD dataset. The amount of POIs was 2 and achieved a GE_0 of 1104. Since the CHES_CTF, ASCAD_F, and ASCAD_R datasets are protected and cannot be broken by LDA, it can be concluded that LDA is not able to successfully decompose the data in protected datasets, which represents the main con of this method. Its strength, however lies with breaking unprotected datasets, although it was not the best performing method on the AES_HD dataset in contrast with SNR which achieved a GE_0 of 1094 with 2 POIs as well.

PCA was able to break the AES_HD dataset with 32 POIs with a GE_0 of 2178. PCA was also able to break the ASCAD_F and ASCAD_R datasets with 16 POIs, achieving a GE_0 of 203 and 447, respectively. Although PCA was not able to attain the best performance on any datasets, it was one of two to break three datasets next to the Triplet Network. This represents the strength of PCA, a well-rounded method that performs well on protected and unprotected datasets. The main weakness is that it is outclassed on all fronts by other methods.

Both SOST and SNR behaved erratically and inconsistent. Both methods broke the ASCAD_F dataset while showing no detection of leakage. Even worse, SOST was able to detect the leakage on the AES_HD dataset but was subsequently unable to break the target. From the results, it can be concluded that SOST is too inconsistent to be used for POI selection. SNR can perform the best on the AES_HD dataset, with only 2 POIs, it achieved a GE_0 of 1094. Therefore, it can be concluded that SNR is a strong method when used on unprotected datasets.

Representing the current state-of-the-art, the Triplet Network can break the AES_HD, ASCAD_F, and ASCAD_R datasets. It provides the best performance on both the ASCAD_F and ASCAD_R datasets, with 32 and 16 POIs, it achieved a GE_0 of 194 and 165, respectively. It was unable to break the CHES_CTF dataset. However, it should be noted that the Triplet Network was not fine-tuned for this dataset. This phenomenon does represent the main con of the Triplet Network, if the Triplet Network is not fine-tuned to a dataset, it seems to be unable to break it within the allotted amount of traces in this thesis. Although when fine-tuned, it provides first-in-class performance on almost all datasets. An interesting observation was that the Triplet Network is not able to be harmonized for all datasets since the best performing amount of POIs for each dataset is different.

1. What are the current Points Of Interest selection techniques, and which one performs the best, according to the Guessing Entropy? In Section 8, each method was benchmarked on the four datasets, the AES_HD dataset, the CHES_CTF dataset, the ASCAD_F dataset, and the ASCAD_F dataset. For the AES_HD dataset, the best performing method was SNR, which was able to break the target with only 2 POIs within 1094 traces. The CHES_CTF was the hardest dataset to be broken. Only SOST was able to break the target and needed 16 POIs, achieving a GE_0 of 4510. The ASCAD_F dataset was easily broken, and the best performing method was the Triplet network. It was able to break the target within 194 traces and needed 16 POIs. ASCAD_R was harder to be broken, but the Triplet network was able to break the target with 16 POIs, achieving a GE_0 of 164. From the results, it is apparent that not one method is able to score the performance on all datasets consistently. Therefore it can be concluded that not one method is the best, alluding to the often stated *No Free Lunch Theorem*.

2.a With the help of RL techniques, is it possible to identify possible points of interest from within a trace? From the results in Section 8.7, it was observed that the proposed framework, as a whole, is the only method to break every dataset within the allotted amount of attack traces. However, an ablation was conducted where the second part of the framework, the scaling down of the selected regions by a dimensionality reduction network, was omitted. In this ablation study, the results indicate that without this network, the regions cannot break any dataset on their own. This represents a double-edged sword. Since the framework can break the target as a whole, it indicates that the selected regions do represent valid POIs. Therefore it can be concluded that the framework can indeed select possible POIs from within a trace.

2.b Is it possible to create a Dimensionality Reduction Neural Network with the help of Reinforcement Learning? In Section 8.7, the results show that the framework can break all datasets. With the conducted ablation study, where the dimensionality reduction network was removed, it was shown that the results achieved by the proposed framework come from the combination of selected regions and the dimensionality reduction network. Therefore it can be concluded that it is possible to create a Dimensionality Reduction Neural Network with the help of RL.

2. Using Reinforcement Learning is it possible to create a Points Of Interest selection method?
The proposed framework was introduced in Section 5. The framework consists of two algorithms based on Proximal Policy Optimization. Together the algorithms are set up to find, select, and scale down POIs. The framework analyzes the traces and designates regions of features. Based on the regions, the framework constructs a neural network to provide a scaled-down version of the selected regions. With these scaled-down features Template Attacks are mounted, and based on the performance, rewards are given. The framework automatically adapts to the rewards given, thereby finding the best-performing regions and networks tailored to each dataset. The results in Section 8.7 show that the POIs found, selected, and scaled down by the framework perform very well when used in combination with Template Attacks. Since previous research indicates that Template Attacks perform better with proper POI selection and subsequently observing the results from the proposed framework, it can be concluded that, yes, it is possible to create a Points Of Interest selection method with Reinforcement Learning.

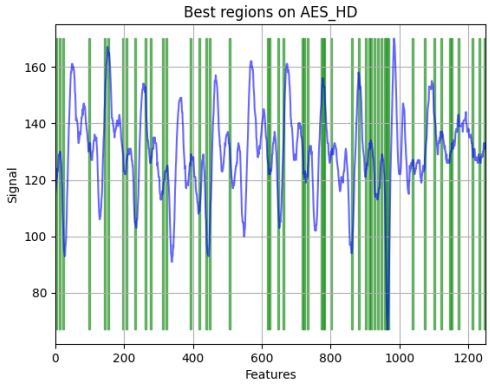
3. Can Reinforcement Learning improve Points Of Interest selection in comparison to the current state-of-the-art, measured in the Guessing Entropy? The framework is able to find, select, and scale down Points Of Interest. With these scaled-down features, multiple Template Attacks were mounted on the AES_HD dataset, the CHES_CTF dataset, the ASCAD_F dataset, and the ASCAD_R dataset. For each dataset the Guessing Entropy (GE) was shown in Section 8.7. The results show that the framework can break all datasets where the current state-of-the-art methods did not. The proposed framework achieved a GE_0 of 1430 for the AES_HD dataset. Compared with SOST, which was not able to break the target, SNR, which achieved a GE_0 of 1094, PCA, which achieved a GE_0 of 2513, LDA, which achieved a GE_0 of 1104, and the Triplet network, which achieved a GE_0 of 1664, the proposed framework was the third best method out of all five.
For the CHES_CTF dataset, the proposed framework was, together with SOST, one of two methods to break the target, achieving a GE_0 of 3962. Compared with SOST, which achieved a GE_0 of 4510, SNR, which was not able to break the target, PCA, which was not able to break the target, LDA, which was not able to break the target, and the Triplet network, which was not able to break the target, the proposed framework achieved state-of-the-art performance.
On the ASCAD_F dataset, the proposed network achieved a GE_0 of 193. Compared with SOST, which achieved a GE_0 of 4522, SNR, which achieved a GE_0 of 1184, PCA, which achieved a GE_0 of 203, LDA, which was not able to break the target, and the Triplet network, which achieved a GE_0 of 194, the proposed framework achieved state-of-the-art performance.
For the ASCAD_R, the proposed framework achieved a GE_0 of 1499, Compared with SOST, which was not able to break the target, SNR, which was not able to break the target, PCA, which achieved a GE_0 of 451, LDA, which was not able to break the target, and the Triplet network, which achieved a GE_0 of 164, the proposed framework was the third best method out of all five.
Not only was the proposed framework able to break all targets, but it was also able to achieve state-of-the-art performance for the CHES_CTF and ASCAD_F datasets. From the results, it can be concluded that, yes, Reinforcement Learning is able to improve Points Of Interest selection compared to the current state-of-the-art.

9.1 Future work

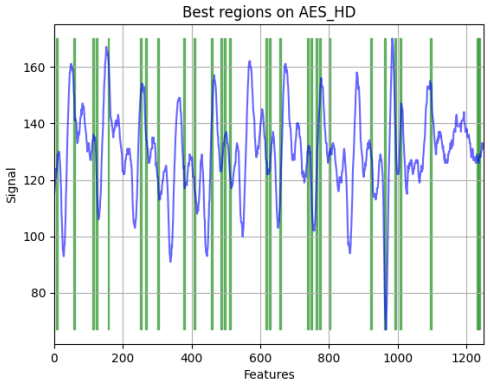
Due to the stochastic nature of the framework, it is reliant on probabilities. These probabilities can be unfavorable during the training of the framework and therefore hamper the learning process. For example, if the number of actions equals 1000 and assuming a uniform distribution over these actions, each action has a probability of being selected of $\frac{1}{1000}$. This is very low, and the main limitation is that the probabilities stay low even after training.
For future work, this is one of the first improvements that have to be made. A proposal is to extend the masking to consider the previous performance of the selected actions. If previous rewards indicate that certain regions do not perform above a threshold, these regions are removed from the action space. This could lead to an improvement of the learning process and an increase in rewards over time.
Also, it would be interesting to benchmark different leakage models to diversify the results and see if the proposed framework can work on multiple leakage models.
Moreover, various countermeasures are known to be employed to induce hurdles for Points Of Interest selection. Methods such as desyncing and introducing extra noise would be interesting to look at.

A Best performing regions and networks

A.1 Region and network for AES_HD



(a) The selected regions for AES_HD with 50% initial regions and 1% minimum regions.



(b) The selected regions for AES_HD with 75% initial regions and 10% minimum regions.

Figure 31: The selected regions for the AES_HD dataset.

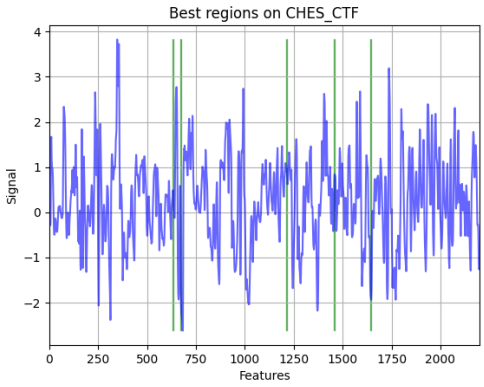
Dimensionality Reduction Network	
Avg Pool layer:	Kernel=32, Stride=8
FC layer:	Neurons=32

Table 15: Network Architecture of the best performing network for AES_HD 50% initial regions and 1% minimum regions.

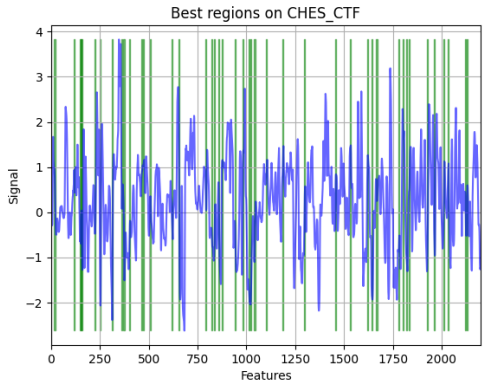
Dimensionality Reduction Network	
Avg Pool layer:	Kernel=4, Stride=4
Convolutional layer:	Kernel=8, Stride=2
FC layer:	Neurons=32

Table 16: Network Architecture of the best performing network for AES_HD 75% initial regions and 10% minimum regions.

A.2 Region and network for CHES_CTF



(a) The selected regions for CHES_CTF with 50% initial regions and 1% minimum regions.



(b) The selected regions for CHES_CTF with 75% initial regions and 10% minimum regions.

Figure 32: The selected regions for the CHES_CTF dataset.

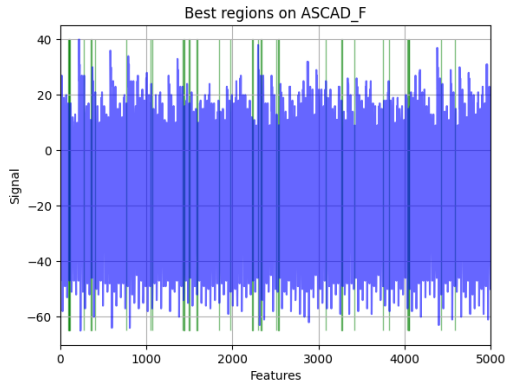
Dimensionality Reduction Network	
Convolutional layer:	Kernel=8, Stride=16
Activation layer:	Type=ReLU
FC layer:	Neurons=32

Table 17: Network Architecture of the best performing network for CHES_CTF 50% initial regions and 1% minimum regions.

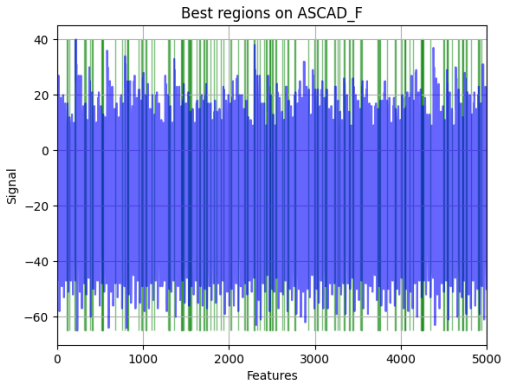
Dimensionality Reduction Network	
Convolutional layer:	Kernel=8, Stride=4
FC layer:	Neurons=32

Table 18: Network Architecture of the best performing network for CHES_CTF 75% initial regions and 10% minimum regions.

A.3 Region and network for ASCAD_F



(a) The selected regions for ASCAD_F with 50% initial regions and 1% minimum regions.



(b) The selected regions for ASCAD_F with 75% initial regions and 10% minimum regions.

Figure 33: The selected regions for the ACAD_F dataset.

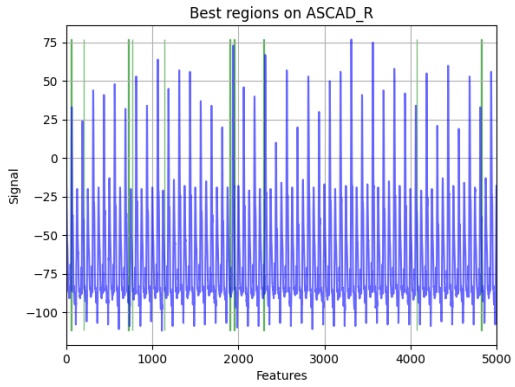
Dimensionality Reduction Network	
Convolutional layer:	Kernel=8, Stride=8
FC layer:	Neurons=16

Table 19: Network Architecture of the best performing network for ASCAD_F 50% initial regions and 1% minimum regions.

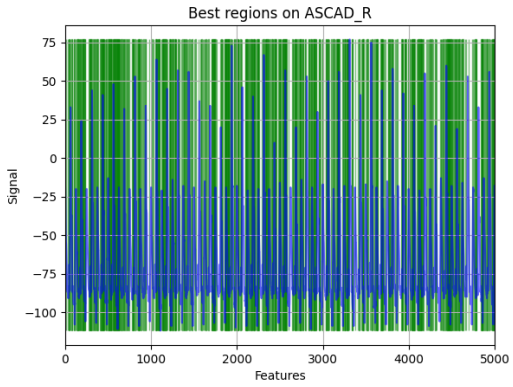
Dimensionality Reduction Network	
Convolutional layer:	Kernel=32, Stride=4
FC layer:	Neurons=64

Table 20: Network Architecture of the best performing network for ASCAD_F 75% initial regions and 10% minimum regions.

A.4 Region and network for ASCAD_R



(a) The selected regions for ASCAD_R with 50% initial regions and 1% minimum regions.



(b) The selected regions for ACSCAD_R with 75% initial regions and 10% minimum regions.

Figure 34: The selected regions for the ASCAD_R dataset.

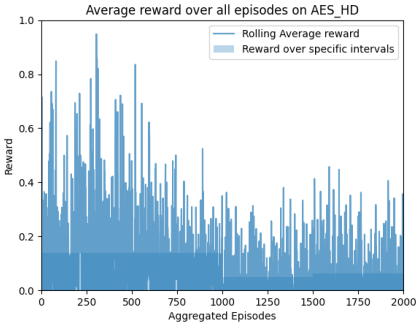
Dimensionality Reduction Network	
Convolutional layer:	Kernel=8, Stride=2
Convolutional layer:	Kernel=16, Stride=2
Activation layer:	Type=Tanh
FC layer:	Neurons=64

Table 21: Network Architecture of the best performing network for ASCAD_R 50% initial regions and 1% minimum regions.

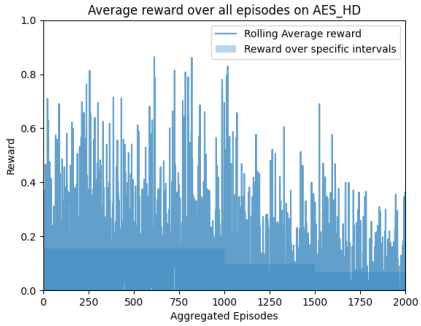
Dimensionality Reduction Network	
Convolutional layer:	Kernel=16, Stride=1
Convolutional layer:	Kernel=16, Stride=1
Convolutional layer:	Kernel=16, Stride=1
Convolutional layer:	Kernel=16, Stride=1
Convolutional layer:	Kernel=16, Stride=1
Convolutional layer:	Kernel=16, Stride=1
Convolutional layer:	Kernel=16, Stride=1
Convolutional layer:	Kernel=16, Stride=1
Convolutional layer:	Kernel=16, Stride=1
FC layer:	Neurons=32

Table 22: Network Architecture of the best performing network for ASCAD_R 75% initial regions and 10% minimum regions.

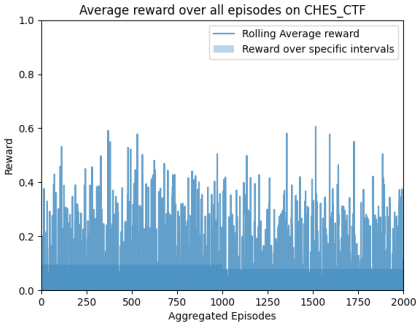
B Reward fluctuations



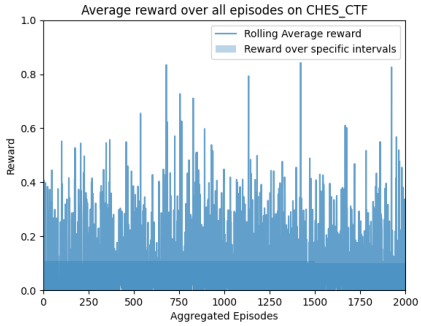
(a) Rewards for all 2000 episodes on AES_HD 50.0% to 1.0%.



(b) Rewards for all 2000 episodes on AES_HD 75.0% to 10.0%.

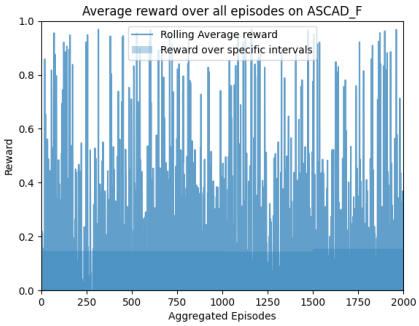


(c) Rewards for all 2000 episodes on CHESC_CTF 50.0% to 1.0%.

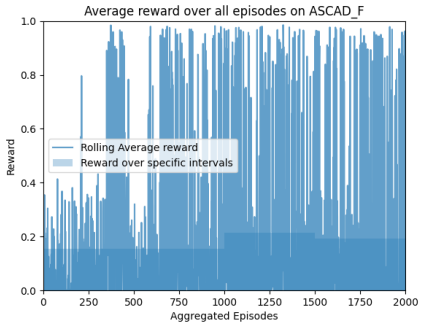


(d) Rewards for all 2000 episodes on CHESC_CTF 5.0% to 10.0%..

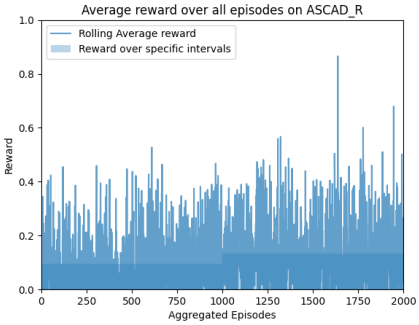
Figure 35: Rewards throughout the episodes on each dataset.



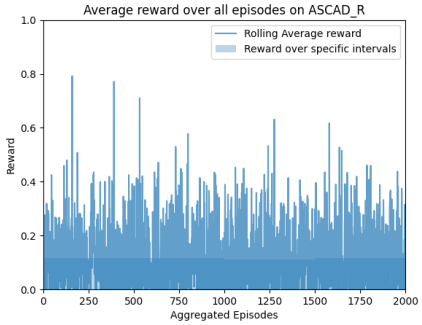
(a) Rewards for all 2000 episodes on ASCAD_F 50.0% to 1.0%.



(b) Rewards for all 2000 episodes on ASCAD_F 75.0% to 10.0%.



(c) Rewards for all 2000 episodes on ASCAD_R 50.0% to 1.0%.



(d) Rewards for all 2000 episodes on ASCAD_R 75.0% to 10.0%.

Figure 36: Rewards throughout the episodes on for ASCAD_F and ASCAD_R.

References

[1] Ches ctf - dataset, 2022.

[2] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater. Template attacks in principal subspaces. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 1–14. Springer, 2006.

[3] T. Ayer, O. Alagoz, J. Chhatwal, J. W. Shavlik, C. E. Kahn Jr, and E. S. Burnside. Breast cancer risk estimation with artificial neural networks revisited: discrimination and calibration. *Cancer*, 116(14):3310–3321, 2010.

[4] P. Baheti. Activation functions in neural networks [12 types use cases], 2022.

[5] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[6] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas. Deep learning for side-channel analysis and introduction to ascad database. *Journal of Cryptographic Engineering*, 10(2):163–188, 2020.

[7] S. Bhasin, D. Jap, and S. Picek. AES HD dataset - 50 000 traces. AISyLab repository, 2020. https://github.com/AISyLab/AES_HD.

[8] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Annual International Cryptology Conference*, pages 398–412. Springer, 1999.

[9] S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.

[10] Z. Chen and M. M. Hendrickson. *Java card technology for smart cards: architecture and programmer's guide*. Addison-Wesley Professional, 2000.

[11] O. Choudary and M. G. Kuhn. Efficient template attacks. In *International Conference on Smart Card Research and Advanced Applications*, pages 253–270. Springer, 2013.

[12] J. Daemen and V. Rijmen. Aes proposal: Rijndael. 1999.

[13] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems. A practical implementation of the timing attack. In *International Conference on Smart Card Research and Advanced Applications*, pages 167–182. Springer, 1998.

[14] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[15] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[16] B. Gierlichs, K. Lemke-Rust, and C. Paar. Templates vs. stochastic methods. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 15–29. Springer, 2006.

[17] R. Gilmore, N. Hanley, and M. O'Neill. Neural network based attack on a masked implementation of aes. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 106–111. IEEE, 2015.

[18] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293–302, 2011.

[19] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[20] S. Huang and S. Ontañón. A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171*, 2020.

[21] J. Hui. Rl - trust region policy optimization (trpo) explained, 2022.

[22] M. Khishe and A. Safari. Classification of sonar targets using an mlp neural network trained by dragonfly algorithm. *Wireless Personal Communications*, 108(4):2241–2260, 2019.

[23] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[24] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[25] L. Lerman, G. Bontempi, and O. Markowitch. A machine learning approach against a masked aes. *Journal of Cryptographic Engineering*, 5(2):123–139, 2015.

[26] L. Lerman, R. Poussier, G. Bontempi, O. Markowitch, and F.-X. Standaert. Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 20–33. Springer, 2015.

[27] Z. Li. Simple power analysis, Mar 2021.

[28] T. Lim, M. Ratnam, and M. Khalid. Automatic classification of weld defects using simulated data and an mlp neural network. *Insight-Non-Destructive Testing and Condition Monitoring*, 49(3):154–159, 2007.

[29] L.-J. Lin. *Reinforcement learning for robots using neural networks*. Carnegie Mellon University, 1992.

[30] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[31] H. Maghrebi, T. Portigliatti, and E. Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

[32] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[33] Z. Martinasek and V. Zeman. Innovative method of the power analysis. *Radioengineering*, 22(2):586–594, 2013.

[34] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[36] G. Perin, L. Wu, and S. Picek. Exploring feature selection scenarios for deep learning-based side-channel analysis. *Cryptology ePrint Archive*, 2021.

[37] S. Picek, A. Heuser, A. Jovic, and L. Batina. A systematic evaluation of profiling through focused feature selection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(12):2802–2815, 2019.

[38] P. Pratikbais. Activation functions in neural network, Dec 2021.

[39] J.-J. Quisquater and D. Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *International Conference on Research in Smart Cards*, pages 200–210. Springer, 2001.

[40] K. Ramezanpour, P. Ampadu, and W. Diehl. Scarl: side-channel analysis with reinforcement learning on the ascon authenticated cipher. *arXiv preprint arXiv:2006.03995*, 2020.

[41] J. Rijsdijk, L. Wu, G. Perin, and S. Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 677–707, 2021.

[42] U. Rioja, L. Batina, J. L. Flores, and I. Armendariz. Auto-tune pois: Estimation of distribution algorithms for efficient side-channel analysis. *Computer Networks*, 198:108405, 2021.

[43] M. Rivain, E. Prouff, and J. Doget. Higher-order masking and shuffling for software implementations of block ciphers. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 171–188. Springer, 2009.

[44] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[45] D. B. Roy, S. Bhasin, S. Guilley, A. Heuser, S. Patranabis, and D. Mukhopadhyay. Cc meets fips: A hybrid test methodology for first order side channel analysis. *IEEE Transactions on Computers*, 68(3):347–361, 2018.

[46] W. Schindler, K. Lemke, and C. Paar. A stochastic model for differential side channel cryptanalysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 30–46. Springer, 2005.

[47] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[48] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[49] C. R. Shelton. Importance sampling for reinforcement learning with multiple objectives. 2001.

[50] F.-X. Standaert, T. G. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 443–461. Springer, 2009.

[51] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[52] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[53] B. Tao and H. Wu. Improving the biclique cryptanalysis of aes. In *Australasian Conference on Information Security and Privacy*, pages 39–56. Springer, 2015.

[54] C. J. C. H. Watkins. Learning from delayed rewards. 1989.

[55] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[56] L. Wu, G. Perin, and S. Picek. The best of two worlds: Deep learning-assisted template attack. *Cryptology ePrint Archive*, 2021.