# Little or Large?

## The effects of network size on AI explainability in Side-Channel Attacks

Author: Djoshua Moonen[1], {D.D.M.Moonen@student.tudelft.nl }
Supervisors: Stjepan Picek[1], {S.Picek@tudelft.nl}
Marina Krcek[1], {M.Krcek@tudelft.nl}

[1]Faculty of Electrical Engineering, Mathematics and Computer Science. Delft University of Technology. The Netherlands

June 21, 2020

**Abstract**

For a system to be able to interpret data, learn from it, and use those learnings to reach goals and perform tasks is what it means to be intelligent [1]. Since systems are not a product of nature, but rather made by humans they are called Artificial Intelligence (AI). The field of Side-Channel Attacks (SCA) has benefited from applying AI systems to their problems. Operations previously to resource-intensive to perform can now be executed using AI. Currently, the focus lies on exploring which parameters result in the optimum performance when classifying side-channel traces. But since this application has only recently been applied, there is much more research to be done. As of now, the literature claims that a reduction in the size of the architecture would result in an improvement of the explainability of the models used. However, this change in explainability has not been explicitly proven to hold for SCA models. This created a gap in knowledge. This paper aims to close this gap by exploring these assumptions. The goal is to explore if a reduction in complexity of SCA models leads to improved explainability. An experiment was conducted using two existing SCA architectures with a small and large complexity respectively. Using heatmaps, the explainability of these models were assessed to investigate the existence of patterns. The results show a difference in the consistency of the classification process, where the model with the lowest complexity could more consistently state why a certain classification was made. The results indicate that the explainability of a given SCA model can be improved by decreasing its complexity.

***Keywords:*** Heatmap, Visualization, Side-Channel Analysis, Explainability, Artificial Intelligence

# 1 Introduction

The options available in Side-Channel Attacks (SCA) have increased during the 20 years that have passed since the field was first invented. By studying the power consumption, sound or electromagnetic output, a person or group harboring malicious intent, the so-called

threat actors have various methods to obtain information about systems. These actors can obtain this information either with or without invading the systems at hand. An actor could perform a profiled attack, in which case they would have access to a device that acted similar to the one they wanted to attack. After obtaining such a device they would analyze it to reveal the relation between observation and internal behaviour. After this process has been completed they would use this relation to extract the internal secret from the device they wanted to attack [2].

Dhem et al. demonstrates how it is possible to recover encryption keys using a timing attack [3]. Simple Power Analysis (SPA) allows actors to gain information on key material and operations by observing the power consumption of a certain device [4]. Differential Power Analysis (DPA) attacks allows one to undo the process of almost any encryption algorithm by correlating power consumption with computation values [4]. The possibility for an actor to use one of the aforementioned techniques to obtain information about a computer system makes it vulnerable. Therefore it is crucial to analyze and resolve these threats and protect the vulnerable systems before the information is gathered, or other harm is done.

A common way to protecting such a leaking system is to apply countermeasures. These countermeasures can be grouped into masking and hiding countermeasures [2]. When masking countermeasures are applied the aim is to divide the information gained from sensitive variables, by randomly splitting variables. If the split is done in a way that does not reveal information unless all parts of the split are in possession, the sensitive variable is successfully masked. Hiding countermeasures is when the physical implementation is made constant or random, thus not revealing the potential change that might be happening as a result of computational operations. An example of this would be constant ventilator speed. This would make it harder to detect patterns from it.

Recent developments in Artificial Intelligence (AI) and in turn in Machine Learning (ML) allow these technologies to be useful as tools in the field of SCA. Deep Neural Networks (DNN's) can be used to classify observed patterns gained from system data with very high accuracy. The most notable example where this accurate matching can be applied is in cryptography, where it makes it possible to recover an encryption key by monitoring power consumption. As first proposed in Lerman et al. [5].

But the advantages AI brings are not without its drawbacks. Where models are now able to map data to create an effective classification process, these models suffer from the problem that its processes are difficult to assess due to its black-box nature. This nature stems from the inability to distinguish whether the result is followed from a process not previously thought of, or if a mistake was made. Therefore it is crucial for AI to have the ability to provide insight to their users of its inner workings, in turn explaining its decision-making process. Therefore it is a good thing that such insights can be obtained by using visualization techniques. These techniques allow for insight to be gained by highlighting determining factors in the decision-making process or input. This is under the assumption that the information can be displayed in a way that allows the user to interpret them.

This paper will analyze the benefits obtainable by using visualization techniques currently available in side-channel attacks. Specifically, what are the effects are that network size has on the ability of AI to explain its decision-making process, when classifying side-channel traces? This question follows from a literature study performed and is followed by an experiment, both described in this paper.

This paper is structured as follows. Section 2 describes the steps taken to conduct the literature study, this section also provides the results from which the experiment conducted

stems. Section 3 states the motivation behind the experiment. Section 4 discusses the theoretical idea behind the experiment. In section 5 the steps taken to perform the experiment are discussed. This includes the setup, execution, and results of the experiment. Section 6 contains a reflection on the ethical aspects of the experiment and discusses its reproducibility. In section 7 the results are put into context and reflected upon. Section 8 concludes the paper by answering the research question, after which the future work is presented.

# 2 Literature analysis

Section 2.1 states the methodology of the literature study. Section 2.2 describes the process used to analyze the existing literature on visualization techniques used in SCA, after which 2.3 shows and analyses the contributions made by the individual papers. Section 2.4 highlights common themes found throughout the literature, which forms the basis on which the rest of the paper is build.

## 2.1 Methodology

The application of AI in the field of SCA is a relatively new thing and because AI and thus AI visualization techniques are rather new to this field, there are many unexplored areas. Therefore it was decided to conduct a literature study, to inspect the work conducted on visualization techniques and assess their contributions. This was done to better understand this emerging technology. The literature study should highlight commonly discussed topics that will serve as an indication as to what themes are worth exploring in the short term.

## 2.2 Setup of the analysis

Based on the need for knowledge, a literature analysis was conducted. Figure 1 visually describes the steps taken. The first step was to create search queries and execute them. These queries were performed on paper databases Google Scholar and DBLP, resulting in a combined total of 49 papers.

The second step was to screen the scientific papers found, and exclude papers deemed irrelevant. 6 duplicate papers were removed after which the collection consisting of unique papers had their titles, abstracts and, keywords analyzed for relevance. To remain included in this step the papers had to meet three criteria. Firstly, the language used in the paper needed to be English. Secondly, the papers needed to be related to the field of SCA. Thirdly the papers needed to have a visualization technique as part of the content, and not merely present in a citation or the introduction. These criteria would ensure that the papers left were relevant to the topic and that they were readable and comprehensible. By applying these basic, but crucial criteria 37 papers were excluded, which left 6 papers included for the eligibility check.

The third step taken was to scan the rest of the papers and further exclude papers that would not contribute to this research. The remaining papers were fully read, and excluded if the scope of the papers was not similar or relatable to our research question. This excluded 3 papers, thus resulting in 3 papers remaining included.

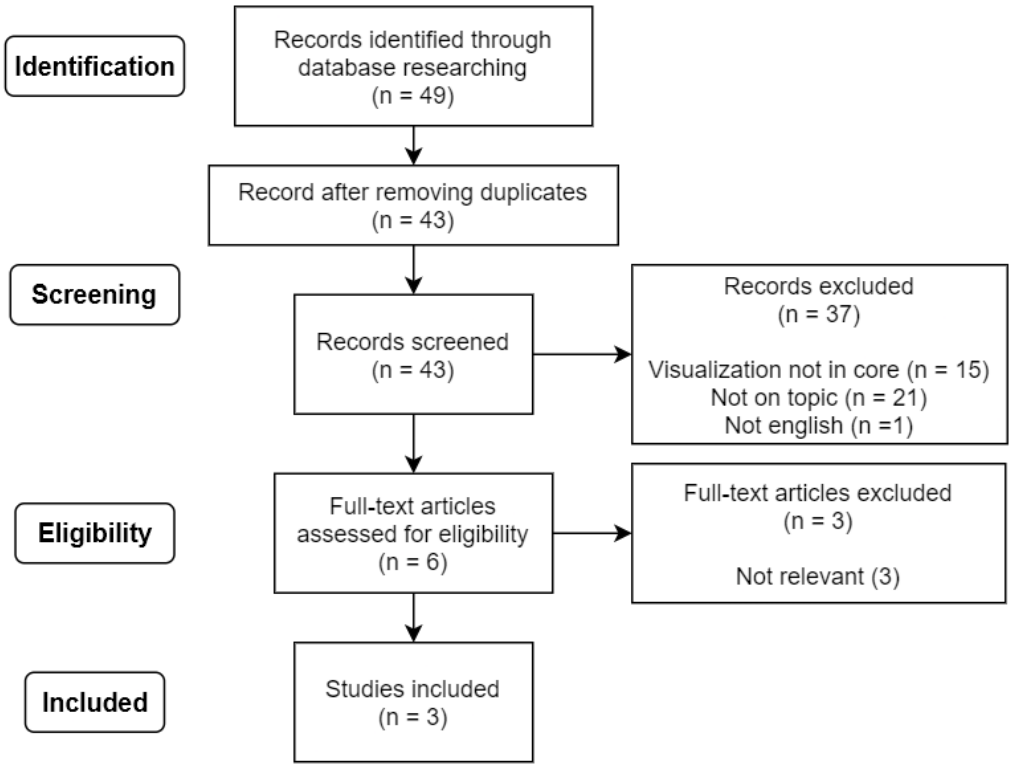


**Figure 1:** PRISMA flow diagram describing the process used to conduct the literature study.

## 2.3 The included papers

### 2.3.1 Methodology for Efficient CNN Architectures in Profiling Attacks

The paper on efficient CNN Architectures in profiling attacks by Zaid at al. [6] addresses the problem of complex inner workings that neural networks have. By applying weight visualization, gradient visualization and heatmaps, the aim is to provide insight into the role that hyperparameters have in the feature selection phase.

After the different visualization techniques are explored the paper explores how to create efficient CNN architectures. It does so by analysing the length of filters and the number of convolutional blocks. The settings are explored by looking at different optimizers, weight initialization techniques, activation functions, learning rates, batch sizes, amount of epochs, and network size. It is theoretically shown that an increase in filter length decreases the network confidence. Since network confidence is defined as the ability to detect POI's in the convolutional part, this impacts the performance of the model. Furthermore, the way pooling is used between convolutional was shown to have impact on the performance of a model. The paper states that when MaxPooling is used, the training time should increase and performance should increase. This is result from relevant information being spread over pooling samples, and thus more iterations are needed during training to find this relevant information. It is recommended to use AveragePooling since no relevant points are discarded

this way, and training time can be reduced.

It is proven that changes proposed to filter length and number of convolutional blocks result in an performance improvement of the architecture. The architecture proposed is proven to be less complex and faster than the current state of the art. It is proposed to test on other masking schemes then the ones used.

### 2.3.2 Deep Neural Network Attribution Methods for Leakage Analysis and Symmetric Key Recovery

The paper from Hettwer et al. [7], aims to investigate the ability for DNN's to be used as tools assessing countermeasures applied to the field of SCA. This investigation is done by examining three visualization techniques proposed in image classification, and see how well they perform on side-channel traces. This should more closely highlight the decisions made by DNN's, in turn allowing the decision to be interpreted. When these decisions are able to be highlighted information leakage of model can be analysed. This in turn provides the data from which it is assessed if a countermeasure is effective, or not.

The paper introduces the reader attribution or heatmapping, a visualization technique that visualizes the level of contribution to the classification process for every datapoint in the input. Furthermore the authors state the techniques they intent to investigate, namely saliency maps, occlusion and Layer-wise Relevance Propagation (LRP).

The paper continues by stating the datasets used as well as the process used to create heatmaps from trained CNN's, after which they evaluate the results. It is stated that this evaluation process is normally done qualitatively by human experts, but is assumed to not hold for side-channel traces since 'it is not possible to judge whether a 1D heatmap indicates the "important" sample points by visual inspection' as found on page 9. After which, the two techniques proposed are used to evaluate a heatmap. The paper concludes that there is no clear evidence to suggest that any of the techniques outperforms the others in detecting sensitive sample points.

### 2.3.3 Gradient Visualization for General Characterization in Profiling Attacks

The paper provided by Masure et al. [8] aims to analyse succesfully trained models that extract sensitive data. The paper makes use of the fact that the loss gradient can be used to locate the POI's. This is then further tested by conducting a study on the ASCAD database. Using this database an experiment was conducted in which the different parts of the dataset are tested. The paper shows that gradient visualization can be used identify POI's, an the technique would remain effective even when countermeasures were applied.

## 2.4 Common themes

The first common theme is the discussion of smaller sized neural networks. Multiple times the benefits of small sized networks are mentioned. Masure at al. shows that a network reduced in filter amount, performs well. Zaid et al. concluded that smaller sized models would require less time to train compared to their larger sized counterparts. Furthermore it is claimed that smaller sized networks would have improved explainability due to the POI's not being spread over neurons but rather grouped together. This would result in said neuron obtaining a relatively higher weight in the decision making process.

Even in the limited amount of papers that fit the selection criteria, little work was done on the explainability of AI models in SCA. The focus remained mostly on locating leakage

in datasets or picking the right parameters to optimize the performance of AI models. Even when visualization techniques were used, their use was as a tool to investigate the effects of said parameters.

The papers show that there is no standard visualization technique to use as different techniques were used throughout the literature. The techniques discussed or used were heatmaps, Layer-wise Relevant Propagation (LRP), occlusion, and weight visualization. Hettwer et al. concluded that there was no observed difference between 3 of the 5 mentioned techniques. This further confirms the statement that there is no standard.

Furthermore the papers shared two datasets. The ASCAD dataset occurred the most, but the DPA contest v4 datasets also occured more than once throughout the literature. The amount of datasets seemed to be limited and the datasets the papers shared was even smaller.

# 3 Methodology

The literature study revealed a gap in knowledge regarding the explainability of models that differ in complexity. To fill this gap, we propose to conduct an experiment. This section describes the proposed experiment and justifies its decisions. Section 3.1 contains the motivation of the experiment. Section 3.2 argues why the ASCAD dataset is the preferred choice. Section 3.3 states why heatmaps were chosen as a visualization technique.

## 3.1 Motivation

The experiment would compare the decision-making processes of two preexisting architectures and determine the existence of any notable differences. The reasoning behind using existing models is that creating models high in performance is the current focus of the field. With this being a time consuming task it would not fit in the scope of this project. More information on the models chosen and the training process can be found in section 5.2.

An alternative option would be to up- or downscale one of the current models and compare the two. We are of the opinion that not would not be feasible. Either the difference would become trivial making the comparison not realistic, or the performance would be impacted in a negatively harming the results. This has to do with the parameters. The papers in the literature use different parameters for the small and large architectures. This different parameters are expected to correlate with with network size, impacting the performance.

## 3.2 Dataset selection

It was decided to use the ASCAD dataset. Firstly, because it makes use of the masking countermeasure, which makes it closely resemble the data a threat actor would need to use to perform an attack. This leads to the results of the experiment also being more applicable to a realistic setting. Secondly, because it has already served as the basis for many experiments. This proved to be the case for the models used in the experiment, which would ensure that the performance required for the experiment could be met. Furthermore, the ASCAD dataset came with a checksum, which could be used to verify the integrity of the data. The only other dataset that appeared in more than one paper from the literature study was the DPA-contest v4 dataset, however this dataset can be considered as unprotected since it has a leaking mask [6]. The lack of effective masking makes it a less realistic option compared to the ASCAD dataset.

### 3.3 Visualization technique selection

The visualization technique used in the experiment was the heatmap. This technique was chosen because of its visual relation between input and output. It makes use of the weights stored in the model to compute the effective weight of each of the neurons, allowing the user to gain insight into the decision-making process. We perceive the process of the model indicating which neurons are high in weight and thus high in importance, as simple and effective. Furthermore, a heatmap can be created by classifying an image once, this ensures no computational overhead, something that other techniques require. The other options considered were Weight visualization, Occlusion, and LRP. Weight visualization misses the relation to the input which was important for our analysis. Recall that occlusion and LRP were concluded to not outperform heatmaps in Hettwer et al. further justifying the decision.

Furthermore, Hettwer et al. makes the claim that heatmaps are not able to indicate sensitive samples. Ans while this might be true for some SCA model the aim of this experiment is to gain insight into the decision-making process of the model, which can be explored using heatmaps.

## 4 The placement of information

The general idea rests on the fact that neural networks have a decision-making process. This process resulting from the training is used to classify new traces. By visualizing the importance of the input it highlights influential areas. By studying these areas, a better understanding can be obtained on why a certain classification was made. This general idea is then applied to analyze the decision-making process of classes, which allows for the comparison of multiple models using the results from the same class. The process of the model large in complexity is expected to be more difficult to interpret due to the Points of Information (POI's) being spread over multiple neurons. This would result in a decrease in importance for the individual neurons. The same reasoning applies to the model low in complexity. Here the POI's are expected to be spread less resulting in a higher weight difference between the neurons.

## 5 Experimental setup and results

This section shows the steps taken to set up and perform the experiment. Section 5.1 shows how the environment for the experiment was created. The models used are described in section 5.2, whereas the corresponding performance is shown in section 5.3. Section 5.4 describes the steps taken to conduct the experiment, whereas section 5.5 shows its results.

### 5.1 Environment

The aim was for the experiment to not be influenced by existing dependencies present on the machine. Therefore Anaconda 3 [] was used to create an isolated environment to test on. In this clean environment the following dependencies were installed Python version 3.7.7 to act as the programming language [9], Tensorflow version 2.1 as the machine learning library of choice [10], Keras-GPU version 2.3.1 to serve as the hardware-accelerated backend for Tensorflow [11], Numpy version 1.18.1 for scientific calculations [12], ScikitLearn version 0.22.1 [13] and H5py version 2.10 as a dependency from previous work [14], used to prepossess and manipulate data respectively. Lastly, CV2 version 4.2.0 [15] and MatplotLib version

3.2.1 [16] were used to alter the size of and to plot the results. These packages were installed using Anaconda's package manager where possible and Pythons pip package manager when this was not the case. The experiment was conducted on a machine equipped with an Intel i7 CPU, a GeForce GTX 1650 graphics card and 8GB of RAM.

## 5.2 The architectures

This section describes the two models used in the experiment. Both models were trained based on the code found at [17]. Firstly, we describe the model large in complexity in section 5.2.1. Secondly, section 5.2.2 describes the model small in complexity. It was decided to use CNN's for this experiment because of its inclusion of feature extraction [18]. This was required to calculate the heatmap. By CNN's having the ease of calculation a benefit was gained over the MLP, since it is more difficult to extract this information. Furthermore, CNN's perform well at classifying traces [6], ensuring the two requirements on the model type would be met.

### 5.2.1 ASCAD model

The first model was implemented as it was described in Benadjila et al. [19] and is referred to as the ASCAD model in this paper. This model contains 5 blocks, each containing a convolutional and an average pooling layer represents the model large in complexity. The 5 convolutional layers all have a kernel size 11 but a different amount of filters; 64, 128, 256, 512, 512 respectively. All of the 5 layers used the standard ReLu activation function, the standard initialization Glorot uniform and, padding of type same. The average pooling layer all had a pooling window of 2, and thus down-scaled the input by a factor of two. After the 5 blocks that perform feature extraction, the model contains two dense layers for classification containing 4096 neurons each. The classification was finalized by a Softmax activation.

The model was trained using a batch size of 200, RMSprop as its optimizer, categorical cross-entropy as loss function and a learning rate of $10^{-5}$. The number of epochs used to train the model was 100 as opposed to the 75 epochs that was stated to be sufficient in the original paper.

### 5.2.2 ZAID model

The second model implemented was the one described in Zaid et al. [6] and is referred to in this paper as the ZAID model. This model was shown to be more efficient and less complex than the previous state of the art and it consists of 1 block. This block contains a convolutional and an average pooling layer. The convolutional layer has a kernel size of 1 and uses 4 filters. The ZAID model makes use of He uniform initialization, SeLu activation and, padding of type same. The average pooling layer is set up the same as in the ASCAD model with an average window of 2 and down-scaling the input by two as well. Note that this model makes use of batch normalization to speed up the training process, but the input data was not preprocessed. The classification is done using two dense layers of 10 neurons each, after which the Softmax function is applied.

The model was trained with a batch size of 50, Adam as its optimizer and used the categorical cross-entropy as the loss function. The model makes use of a one-cycle policy that can determine learning rate boundaries by running the model for a small number of epochs and where the learning rate varies within this interval [20]. This would result in a

learning rate within the interval [ $5*10^{-4}$ , $5*10^{-3}$ ]. The training process consisted of 50 epochs.

## 5.3 Performance evaluation

After the models were trained, their performance were assessed. This was done to see how the performance of the replicated models relate to the models discussed in their respective papers.

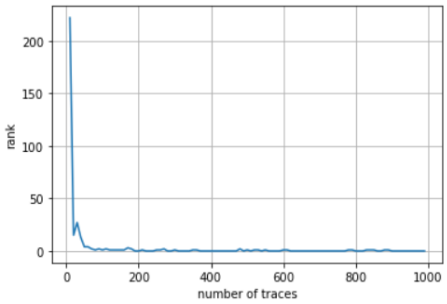


**Figure 2:** Key rank (batch size 10) of the ASCAD model

Figure 2 shows the key rank of the ASCAD model described in 5.2.1. The line converges to rank 0 as more traces are evaluated. This is proof that the replicated model performs well, and is representative of the model described by Benadjila et al.

Assessing the performance of the replicated ZAID model proved to be difficult. The code used to assess the performance would crash during the assessment of the ZAID model which would leave us unable to prove that the performance of the replicated model matched the performance of the original. As a workaround it was explored if the trained model provided Zaid et al. could be used for the experiment, but this proved to not be possible. The last convolutional layer would output an empty response, which prevented the heatmap from being calculated. Therefore the replicated model was used in the experiment.

## 5.4 Conducting the experiment

Now that the environment, trained models, the dataset and, the visualization technique are discussed, a better understanding can be obtained on how the experiment was conducted. Section 5.4.1 shows the steps taken to compute the heatmaps, where section 5.4.2 discusses the techniques used to represent the heatmaps in order to allow the comparison.

### 5.4.1 Heatmap calculation

To calculate the heatmap the code Keras Class Activation Map [21] was used as a base, from which the code was altered to be usable on 1D input. The process of creating a heatmap is as follows. A model is created that, given an input trace, outputs the values present in the last convolutional layer, as well as the computed means. The output present during the classification process is then multiplied with the standard influence to create the total impact each of the neurons had. This data is then is normalized, which gives an array with the same length as the last convolutional layer containing a score between 0 and 1 depending

on the relative importance the corresponding neuron had in the classification process of the trace. This heatmap is then displayed using a color gradient to create the colorized image.

### 5.4.2 Visualization

The visualization techniques were to be used to inspect the decision-making processes of the two models and comparing them. However working with changes in color proved to be a difficult way to compare heatmaps, especially heatmaps that differ in size. Therefore other ways to represent the heatmap were used to make the comparison possible. We determined it as important that the methods used stayed visual, as opposed to numeric. What was to be avoided was to turn the visual insight gain from the black box AI back into a black box metric, because then the benefits gained by visualizing would be lost. Two ways were used to aid the analysis and comparison.

The first method was to overlay the heatmap with the input data. This way the trace would be colorized, indicating where the influential data is located. The problem encountered with this method was that the size of the trace and the size of the heatmap were different from each other. This was resolved using by interpolating values and creating a larger sized heatmap. This solution shown in [21] ensured that input size matched with the size of the last convolutional layer, which allowed them to be combined in one figure.

The second method in which the data visualization was improved was by treating the heatmap as a function. When this is done, the data indicating the influence in the decision-making process will not be represented by color, but as a data point in a function. The heatmap made it difficult to detect some peaks and this technique would make it more evident. For convenience, the data points of the function were linearly interpolated. Note that this interpolation is different from the one used to create the overlay.

Using the models described in section 5.1, trained on the ASCAD dataset described in section 3.2 heatmaps were created using the test data. From these heatmaps were created according to the process described in section 5.4.1, after which the two techniques were applied. This resulted in graphs that could be used to analyze and compare the models.

## 5.5 Experiment Results

This section is dedicated to the results obtained from the experiment. Section 5.5.1 shows the results of the heatmap along with the corresponding overlay that was created. Section 5.5.2 discusses the impact misclassification has on the heatmap, whereas 5.5.3 compares the data from a whole label to compare larger patterns between the models. Note that the results shown represent interesting or common behavior found and is not an exhaustive analysis of all the different graphs found.

### 5.5.1 Heatmap

Figure 3 depicts an example of the heatmaps encountered as described in section 5.4.2. On the left, we see the heatmap corresponding to the ASCAD model with 43 bars corresponding to the 43 neurons in the last convolutional layer of the model. Depicted on the right is the heatmap from the ZAID model containing 700 bars, from its corresponding last convolutional layer. The color depicts the influence the neuron has on the decision-making process, where blue represents low influence and yellow high influence.

The depicted trace, #17 was chosen because it was classified correctly by both models. This allows the optimal scenario to be represented and analyzed. As can be seen from the

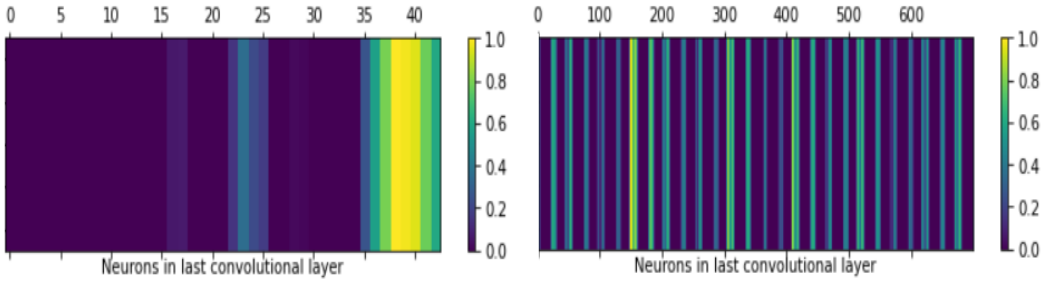


**Figure 3:** A Heatmap visualization of trace #17 from the ASCAD model (left) and the ZAID model (right). Neurons weight in the decision-making process is shown where a high value in the colorbar indicated high influence.

figure, both models show a yellow bar indicating the neuron most influential. Note that due to the large amount on neurons in the ZAID model, the yellow bar is more difficult to spot.

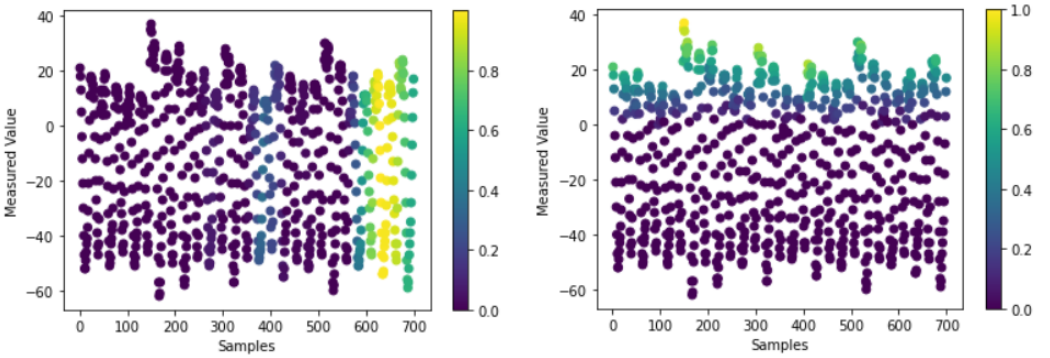


**Figure 4:** A heatmap overlay of trace #17 from ASCAD model (left) and the ZAID model (right). Trace data is colorized depending on the influence of the corresponding neuron, indicating influential areas in the input data.

Using the heatmap shown in figure 3 and the data from trace #17, an overlay was created. This overlay allows us to observe which parts of the input data were the most influential. Figure 4 depicts the overlay created, by depicting the sample data from the trace with the intensity found in the heatmap. The sample data is plotted using the sample ID against the value measures.

What can be observed from the overlay is the different behavior in areas of interest. The ASCAD model has its area of interest shown vertically, whereas the area of interest of the ZAID model is shown horizontally in the figure.

### 5.5.2 Misclassification heatmaps

When observing the results it was discovered that the difference between the predicted label and the true label varied. The difference could be roughly categorized into being small or large. Therefore these two cases were investigated to analyze if they impacted the explainability of the models.

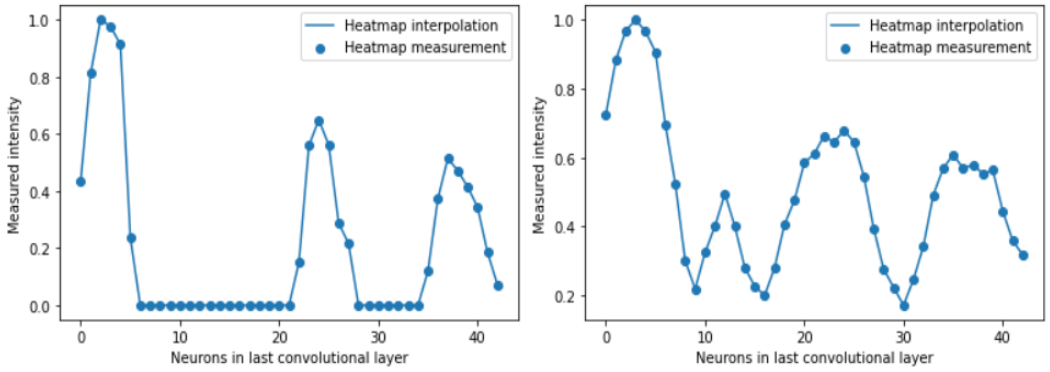


**Figure 5:** A heatmap from the ASCAD model of trace #131, visualized as a function. This figure represents a misclassification where the difference in response between the true label (left plot) and predicted label (right plot) is small.

Figure 5 depicts a heatmap of trace #131 visualized as a function according to the steps in 5.4.2. The trace represents the misclassified traces in the dataset that have a response similar to the true label. As can be seen in the figure, it is often the case that the predicted label has peaks that cover more neurons than the peaks in the true label. Furthermore, the predicted label often contains peaks not present in the true label.

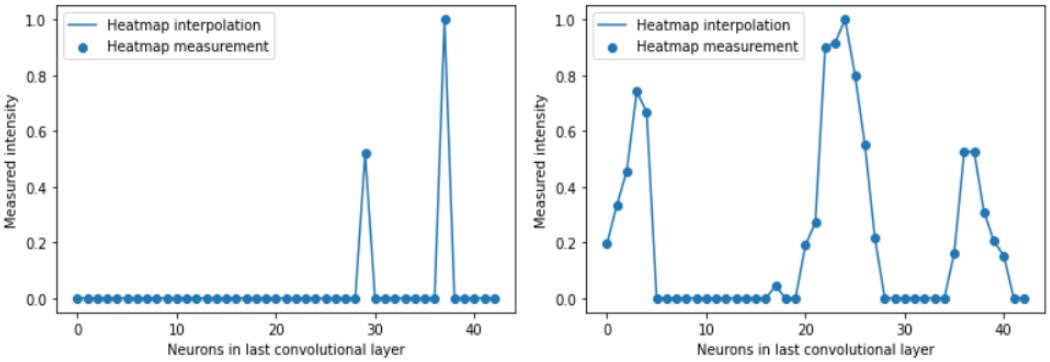


**Figure 6:** A heatmap from the ASCAD model of trace #194, visualized as a function. This figure represents a misclassification where the difference in response between the true label (left plot) and predicted label (right plot) is large.

Figure 6 depicts a heatmap of trace #194, visualized as a function according to the steps in 5.4.2. This trace represents the misclassified traces in the dataset that have a response that is not similar to its true label. When the response differs greatly it is often the case that the most influential neurons of the true label have a less influential peak in the predicted label if any peak at all. A second noteworthy observation is the additional peaks that occur frequently at the predicted label.

Figure 7 depicts a heatmap of trace #125 from the ZAID model. The figure represents the cases in which the difference in response is small. When such a case occurs in the ZAID model, the difference between individual neurons were smaller than $1e^{-2}$. However, the peaks

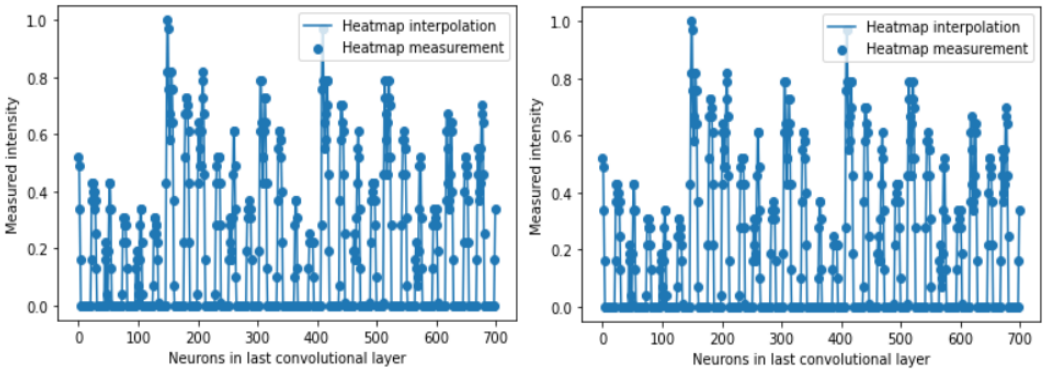


**Figure 7:** A heatmap from the ASCAD model of trace #125, visualized as a function. This figure represents a misclassification where the difference in response between the true label (left plot) and predicted label (right plot) is small.

lie in the same location and no peaks are added.

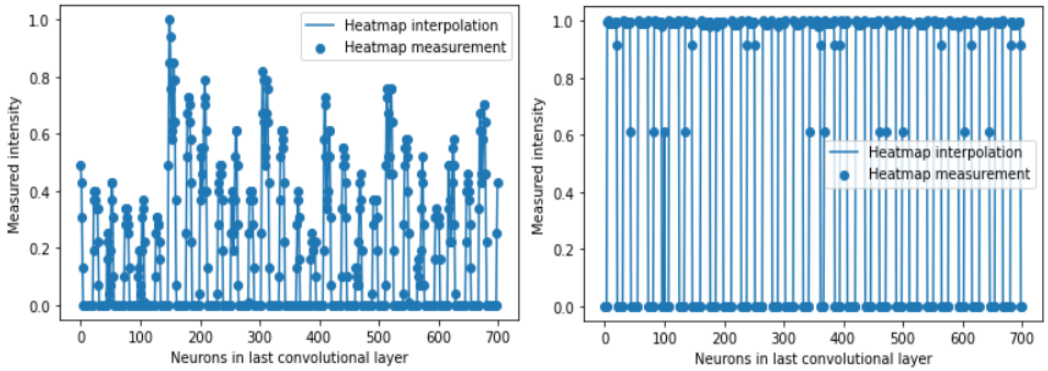


**Figure 8:** A heatmap from the ZAID model of trace #131, visualized as a function. This figure represents a misclassification where the difference in response between the true label (left plot) and predicted label (right plot) is large.

Figure 8 depicts a heatmap of trace #131 of the ZAID model. This trace represents the cases in which the difference in response between the true and predicted label is large. In this case, the neuron weights seem to be uniformly distributed and no clear no pattern can be found.

### 5.5.3 Class level patterns

At this stage we would like to discuss the patterns visible at class level. Figure 9 depicts such a pattern in the decision-making process of the models. It was created by combining the function heatmaps of 5 traces according to 5.4.2. The black lines represent the influence of a certain neuron for individual traces, whereas the red lines represents the average response of the traces. Class #8 represents the reoccurring trend that is found by comparing classes between the models. As can be seen in the figure, the peaks in explanation from the ASCAD

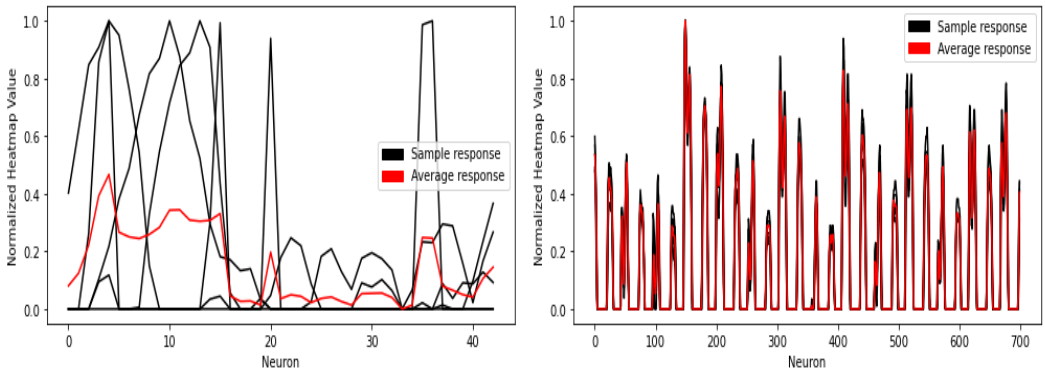


**Figure 9:** Class level pattern of class #8 from the ASCAD model (left) and the ZAID model (right). This pattern is created by taking the average of multiple function represented heatmaps.

model can be found at different neurons. In the case of the ZAID model, the response is present at the same neurons. There seems to be more consistency in the decision-making process of the ZAID model.

# 6 Responsible Research

In this paper, the appropriate time has been taken to cite the work, code, and ideas presented by others. This ensures that credit is placed where it is due. Since there exists no relation between the field of SCA and the team that conducted the research, this paper is freed from the conflicts of interest that might have occurred otherwise. Furthermore, the experiment does not involve human subjects, thus there exists not the possibility for unethical human research to occur. However, we acknowledge that the information provided in this paper can be used for malicious purposes. Therefore it is important to note that this work is meant to help prevent malicious acts, not to enable them.

An effort was made to improve the reproducibility of this work. It was performed using an environment, models, visualization technique, and a dataset that is open to be used by the public. This ensures that any party that wishes to reproduce the results found in this work, has the opportunity to do so. As mentioned earlier the scientific literature used is available to the public, which allows the literature to be accessed and reviewed. This further improves the reproducibility of this work.

# 7 Discussion

Section 5.5.1 shows that the models have different area of interest. This difference is likely due to the ASCAD model performing a large number of convolutions. This likely causes the most informative samples to become mixed with their surroundings. When the area of interest is upscaled to fit the input size it results in a vertical area of interest. It is likely that in the earlier layers the same horizontal area of interest would be found.

Section 5.5.2 analyzed the responses of samples that were misclassified. Here it shows the existence of additional peaks and noted that the peaks are spread over more neurons in the heatmap of the predicted label. It seems to be the case that these types of traces

have features that the model addresses to other classes. This results in a high response on the incorrect label, which leads to an incorrect classification. We also notice peaks that are spread over multiple neurons. This indicates that the most informative datasamples are spread over multiple neurons by the convolutions. However in some cases the peaks consist of only a single neuron indicating that this spread does not always occur. The ZAID model was shown to have difference in heatmap response of misclassified traces that were either very small or very large. In the case of the small difference the responses were nearly the same, implying that the model thought of them as the same. In the case of a large difference the model was not able to differentiate between neuron relevance resulting in a uniform distribution. Both cases seem to indicate that the model was not able to determine what the distinctive features between classes are, implying the model was underfitted.

In section 5.5.3 the decision-making process was compared between the models. Here the results show that the ASCAD model was inconsistent when classifying traces of the same class, whereas the ZAID model was shown to be consistent. These results seems to indicate that the ASCAD model is not able to explain why it makes its decisions.

Now that the results have been discussed, the limitations of the work will be mentioned. The first limitation is that the models make use of a averagepooling layer instead of a maxpooling layer. While this does not impact our comparison, it has to be noted that our work cannot directly be compared to models that make use of a maxpooling as the results might differ.

The second limitation is that we were not able to determine the performance of one of the models used. As discussed in 5.3 our replication of the ZAID architecture cuold not be proven to be of high performance.

The third limitation is the lack of a numerical analysis between the models. One could argue that the use of numerical evaluation would improve the quality of the comparison. However this was shown to not be possible. The size of the last convolutional layer differs between models therefore not allowing for a direct comparison to be made.

# 8 Conclusions and Future Work

We have experimentally shown that the previously assumed claim that a reduction in network size has a positive impact on the explainability of AI models in SCA. This was done by showing the existence of observable patterns in the classification process between the two models on the trace and class level. The patterns found indicate different levels of explainability between the two models, where the smaller model was shown to be more consistent in its classification process. These finding imply that the explanation of SCA systems can be improved by reducing the model size and in turn model complexity.

Our work also raised some questions. Firstly what the effect of using different datasets would be on the observed patterns. Datasets that include different masking or hiding techniques could potentially alter the patterns found, and it would be interesting to observe these potential changes. Secondly it could be analyzed what these patterns are like in different SCA models. This could lead to a better understanding of the SCA models and potentially allow them to be grouped based on explainability besides performance. Thirdly it could be explored if other visualization techniques could be used to find these patterns. Specifically occlusion shows promise due to its similarity to the technique used in this work.

# 9 Acknowledgements

The author would like to thanks Doreen Mulder, Jim Kok, and Johannes IJpma for fruitful discussions about this work. Furthermore the author would like to thank Stjepan Picek and Marina Krcek for their guidance throughout this project.

# References

[1] Andreas Kaplan and Michael Haenlein. Siri, siri, in my hand: Whos the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 62(1):15–25, 2019.

[2] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

[3] Jean-Francois Dhem, Francois Koeune, Philippe-Alexandre Leroux, Patrick Mestré, Jean-Jacques Quisquater, and Jean-Louis Willems. A practical implementation of the timing attack. In *International Conference on Smart Card Research and Advanced Applications*, pages 167–182. Springer, 1998.

[4] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual International Cryptology Conference*, pages 388–397. Springer, 1999.

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

[6] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 1–36, 2020.

[7] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. In *International Conference on Selected Areas in Cryptography*, pages 645–666. Springer, 2019.

[8] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 145–167. Springer, 2019.

[9] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[10] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin

Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[11] Francois Chollet et al. Keras, 2015.

[12] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[13] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[14] Andrew Collette. *Python and HDF5*. O'Reilly, 2013.

[15] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[16] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[17] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, Cécile Dumas, and Adrian Thillard. *ASCAD Github*, 2018. https://github.com/ANSSI-FR/ASCAD (accessed May 11, 2020).

[18] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 45–68. Springer, 2017.

[19] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ascad database. *ANSSI, France & CEA, LETI, MINATEC Campus, France. Online verfügbar unter https://eprint. iacr. org/2018/053. pdf, zuletzt geprüft am*, 22:2018, 2018.

[20] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[21] Nickbiso. *Keras-Class-Activation-Map*, 2018. https://github.com/nickbiso/Keras-Class-Activation-Map/blob/master/Class%20Activation%20Map(CAM).ipynb (accessed May 21, 2020).