Locality-based Electromagnetic Leakage Assessment Using CNN

Ian Heffron and James Dean
Air Force Institute of Technology, Wright-Patterson AFB (Air Force Base), Ohio, USA

ian.heffron@afit.edu
james.dean@afit.edu

Abstract: Deep learning side-channel analysis (SCA) attacks have recently gained in popularity. The ability of deep learning models to retrieve a key byte while minimizing the need for pre-processing steps in both cases of misaligned traces and when the leakage model is multivariate has contributed to the popularity gain. Near-field electromagnetic (EM) probes have enabled leakage capture with high spatial resolution, and the field of deep learning side-channel research looks to find models which reduce the required number of leakage traces to retrieve a key byte successfully. However, despite the many papers researching techniques to reduce the number of traces required for a successful attack, location-based research for EM SCA remains untouched. Due to the nature of EM probes and the architecture of different boards and chips, the location of the collection probe becomes important when attempting to extract the secret key within a reasonable timeframe and with a level of certainty in the result. Our contribution is a framework to determine the best location to assess localized EM leakage against a given chip platform. We use a raster scan to collect localized leakage results at 25 points in a 5x5 grid pattern on a ChipWhisperer Lite XMEGA board running an open-source implementation of the AES-128 encryption algorithm. We demonstrate the use of our framework to locate the best points for an attacker to execute a profiling attack by identifying the grid point with the most detectable leakage for a chip platform. We further execute a smaller localized 25-point raster scan in a grid over the identified location to further refine our estimate of the optimal collection location. We then demonstrate that this location can be used to execute a side-channel profiling attack against the same chip architecture and will result in a lower number of traces required to retrieve the key byte.

Keywords: side-channel analysis, localized leakage, neural networks, electromagnetic leakage, leakage detection

1. Introduction

Side-channel analysis attacks are a class of attacks that seek to exploit the physical phenomena related to cryptographic primitives. During any encryption operation, sensitive variables are processed using a combination of a known public variable, called plaintext, and a secret variable, a key. Due to processor architecture and the manufacturing process, information leakage occurs during encryption operations. This information leakage comes in the form of EM, power, timing, and other physical phenomena. These phenomena can be observed and recorded to perform SCA work and SCA attacks. Encryption algorithms such as the Advanced Encryption Standard (AES) can be defeated by SCA profiling attacks. Profiling attacks involve collecting profiling traces on a prototype device, similar to the intended target device. The profile traces are then used to train a neural network to create a model that will be used in the attack. Attack traces are then gathered on the intended target device and predictions are generated based upon the attack traces and the generated model. A Bayesian process is then used by the attacker to determine the likelihood for each potential key byte value. The candidate byte value with the highest probability is predicted to be the correct key (Masure, Dumas and Prouff, 2019). These profiling attacks are often performed one key byte at a time in a divide-and-conquer strategy. If each byte can be retrieved, the whole key can be reassembled. Much of current SCA research focuses on retrieving the correct key bytes while minimizing number of required attack traces. This paper seeks to further this focus and presents a framework with which to determine, for a given chip, a point with the highest perceived leakage from which to collect leakage and execute an attack. Simply put our hypothesis is that there are points on a device that exhibit leakage that is more susceptible to an electromagnetic SCA attack than other points on the device.

1.1 Our Contribution

We develop and demonstrate the effectiveness of a framework designed to determine the highest perceived leakage point for an attacker to collect EM leakage information for a given chip and board combination. We do this by training a neural network to detect leakage in traces collected at several points above the target. At each point we collect training and testing traces where both the plaintext and key values are randomized. The attack traces are collected using randomized plaintext, and a randomly generated but fixed key. We train the network using a combined dataset of all training traces collected at each of our points. Then we evaluate the accuracy of our model by feeding the fixed number of testing traces from each point into our model and generate a heatmap based upon the returned accuracies. The points on the heatmap with the highest scores indicate areas with the strongest leakage. We then perform attacks using the traces collected at the points with the strongest and weakest leakages to demonstrate that the key byte can be retrieved more reliably and more quickly at the point with the strongest leakage. Using this framework, we can demonstrate that a single trained network given

information from all collection points can correctly ascertain the best point on a given architecture to retrieve key byte information when fed into a neural network. We do not contend that this deep learning architecture is the optimal architecture for this problem, however we are able to demonstrate the effectiveness of the technique to find points with higher discernable leakage.

2. Background

This section will briefly discuss the terminology and background material required to understand our paper. We will discuss previous research, deep learning, and SCA attacks.

2.1 Previous Research

The work of Schlösser et al, (2013) successfully demonstrates identification of the exact SRAM location accessed during the activation of the substitution box (SBox) lookup table during AES encryption operations. Andrikos et al., (2019) uses neural networks and an information-based approach to retrieve information about the AES encryption key after finding the SRAM location of the lookup table containing the key by using the framework given in (Schlösser et al., 2013). They demonstrate a successful attack using memory table exploitation. Moos, Wegener, and Moradi (2021) demonstrate the effectiveness of deep learning models when used during side-channel leakage assessment compared to more traditional statistical analysis-based side-channel leakage tests such as the Welch's t-test and the Pearson's χ^2-test. Moos et al. demonstrate the effectiveness of deep learning over traditional tests when the input trace data is misaligned, or the leakage model has a multivariate distribution. Wang et al. (2020) demonstrates the effectiveness of training a neural network on power trace side-channel data pulled from multiple homogenous and non-homogenous devices. They increased the success rate of attacks executed with fewer traces using a network trained on the homogenous and non-homogenous devices. Their research is based on the idea that signal and noise diversity can often help to solve problems more efficiently. Danial et al. (2022) also uses this principle, training a Deep Neural Network on EM traces captured from multiple devices to demonstrate the effectiveness of diversity when confronted with leakage that has a low signal-to-noise ratio (SNR). Das and Sen, (2020) uses test-vector leakage assessment and SNR as inputs into a greedy gradient-search heuristic to find a predicted optimal point to conduct leakage collection to minimize the time required to perform successful key retrieval.

2.2 Deep Learning

Deep learning is a form of machine learning that emphasizes learning successive layers of increasingly meaningful representations of data. Historical machine learning models relied on complex mathematical models and highly sanitized input to turn inputs and outputs into learned rules. The addition of many layers of neurons that add those meaningful representations of data allows deep learning to eliminate the need for those extra preprocessing steps. The ability of deep learning to forgo this excess work has contributed to the gain in popularity in recent years. Deep learning has been used in many fields and for many purposes, including image classification, speech recognition, natural language processing, autonomous driving, and strategy gaming. More recently, the field of side-channel research has found deep learning to be quite useful in performing SCA profiling attacks (Moos, Wegener, and Moradi, 2021)(Wang et al., 2020).

For our experiments, the input data $d_i = \mathbb{R}^e$ where $i \in \{1,2, \dots, N\}$ with N denoting the number of traces in our dataset, and e denoting the number of input features. Our input data is mapped to a set of labels L where $l(d_i) \in L$. And our labels are presented as a one-hot encoded vector $v = \{0,1\}^{|L|}$ where $m \in L$ and $v_m = \begin{cases} 1 \; if \; m = l(d_i) \\ 0 \quad otherwise \end{cases}$.

This research uses a convolutional neural network (CNN) which is a feed-forward deep learning algorithm. A convolutional neural network is made up of one input, several convolutional and hidden layers, dense layers, and one output layer. These layers often include pooling, batch normalization, and dropout layers. Pooling layers slide a two-dimensional filter over the feature map of the input to the layer. This filter summarizes the features within the region covered by the filter (Gholamalinezhad and Khosravi, 2020). Batch normalization serves to prevent overfitting of a model. It normalizes the output of each layer to enable the layers to perform learning independently. Dense layers are used as they are fully connected layers and facilitate the classification for the output layer. Each layer is made up of a matrix of neurons with initially randomized and later learned weights w that get applied to its inputs x linearly. The convolutions in a CNN generate features. Features are the measurable property found in the input data that allows the CNN to guess the correct output more accurately. Then a non-linear activation function is applied to each point in the resulting matrix. The output of a given layer is the input for the following layer. The final layer of a neural network is the output layer. In our case, we use a one-hot encoded vector to represent the classified output.

At the beginning of training all the weights are randomized. The input samples are separated into batches of a fixed size and all samples in a batch are evaluated simultaneously. Each time the network completes a training on a batch, the output layer presents a prediction y' for each of the inputs. These predictions are then compared to their corresponding label values and based upon the distance from the labels to the predictions, loss values are calculated using a loss function. Once all the losses are calculated, backpropagation is used to update the weights of all the neurons based on the gradient of the loss function. Each time covering the entire input dataset during training is called an epoch.

2.3 Side-Channel Analysis

Side-channel analysis is the analysis and exploitation of the weaknesses in the physical implementation of electronic devices. Often, SCA work is used against algorithms to recover the algorithm's secret information. This can be accomplished via observing power draw, EM emissions, and execution time of operations on a system. The purpose of SCA is to determine if there is discernable leakage from a given device, and if that leakage can lead to an attacker learning secret information from the device such as an encryption key. In the most effective SCA attacks, two phases are used. The first is a profiling phase in which the attacker uses a similar version of the device under attack to profile the device's leakage and create a model with which to execute an attack. The attack phase consists of gathering traces from the target device and running those traces through the profiling model to determine the key. During the attack phase, Bayesian analysis is used to execute a Maximum Likelihood strategy, which reduces to estimating the likelihood of each key byte candidate being the correct key byte and selecting the candidate with the highest probability. As each new trace is ran through the Bayesian process, the likelihood of each key candidate gets updated, and each candidate gets ranked in order of overall likelihood. The attack is considered a success if the correct key byte is ranked first in the ranked list (Benadjila et al., 2019).

2.4 Advanced Encryption Standard

The Advanced Encryption Standard is a widely used encryption algorithm. It uses three different key lengths, 128, 192, and 256 bits. For our experimental setup we use the 128-bit key length. The AES-128 algorithm uses 10 rounds of encryption, and each round is comprised of four steps: a non-linear substitution of bytes using the SBox lookup table, a row shift step, a column mixing step, and adding a round key. During the rounds of AES, the raw plaintext and raw key are only exposed in the first of the 10 rounds (Wang et al., 2020). Therefore, our experiment will focus on the trace data pulled from the first round of encryption. In the ChipWhisperer XMEGA's software implementation of AES, the rounds are completed in sequence. However, in many other AES implementation, including on FPGA devices, the rounds are completed simultaneously.

3. Experimental Design

This section describes the precise EM setup that we used to detect leakage during our collection. It further describes the neural network setup that we used to assess the leakage and carry out our attacks.

3.1 Collection

Our experiment sought to define a framework for determining the best EM collection point to perform EM SCA attacks on a given chip. Our setup consisted of the ChipWhisperer Lite XMEGA platform running a software implementation of the AES-128 encryption standard. Figure 1 displays the scanning setup. The ChipWhisperer device was scanned using a LeCroy WaveRunner 10xXi-A oscilloscope and a Riscure PR137LS probe with a diameter of 1.5mm. As the ChipWhisperer platform uses a software implementation of AES, side-channel leakage is not so highly localized that we would need a probe with a higher spatial sensitivity (Danial et al., 2022). As depicted in Figure 2, the scan was performed in a raster scan 5x5 grid. The XMEGA chip is 10mm by 10mm and we collected measurements every 2.5mm.

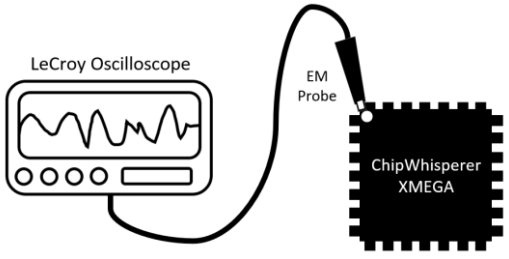

Figure 1: The ChipWhisperer is scanned using a near-field EM probe with data sent back to the LeCroy oscilloscope.

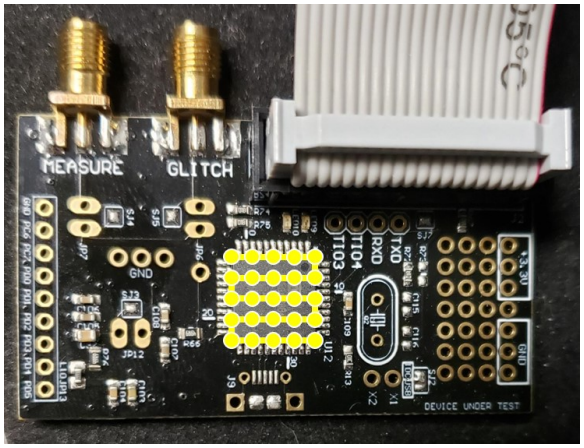

Figure 2: Depiction of the raster scan above the ChipWhisperer XMEGA.

The probe was moved across the board using an XYZ translation table. At each point, 24,000 profiling traces, 3,000 testing traces, and 3,000 attack traces were collected with a sampling rate of 100MHz and a sampling time of 5ms for a total of 500,000 samples per 500ms trace. Figure 3 shows the area we isolated which includes the first round of AES as this is the only time the real key is exposed in the SBox operation. This translates to the samples between 150,000 and 180,000. To collect our profiling traces, we generated randomized plaintext and randomized keys, and to collect our attack traces we generated randomized plaintexts and used a single consistent key across all 25 collection points.

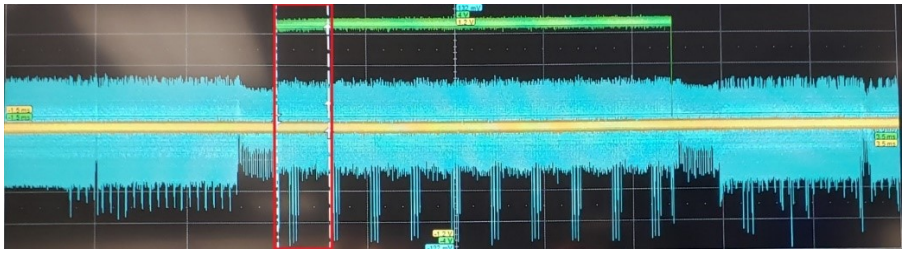

Figure 3: Oscilloscope output showing location of isolated samples

Prior to entering our data into our neural network, we create and attach the labels to our datasets. These labels are generated to match the output of the SBox operations. This simply means we assign the output of $p[1] \oplus k[1]$ to each trace (Benadjila et al., 2019). Then the collected profiling traces are separated out into the first 24,000 traces, used for training, and the final 3,000 traces, used for individual point testing. The training traces for each point are combined into one large training dataset consisting of 600,000 traces.

3.2 Neural Network

Our neural network, shown in Figure 4, consists of five convolutional blocks of increasing size. We used the following convolutional network with batch normalization, Rectified Linear Unit (ReLU) as our activation function, and a single pooling layer in each of our convolutional blocks.

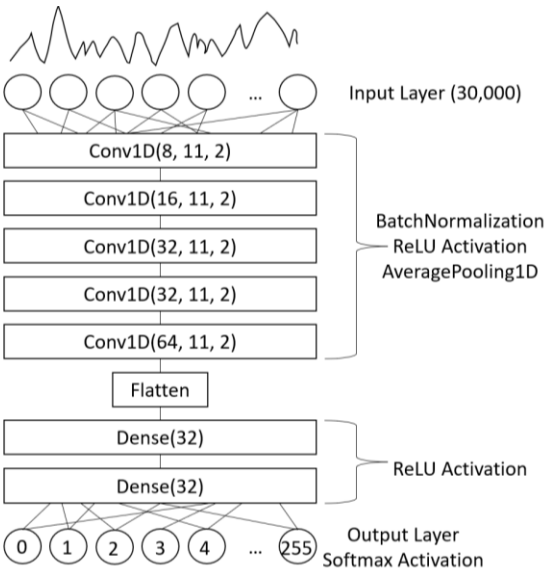

Figure 4: Architecture of the CNN. The network contains five convolutional blocks. Following each convolutional layer is a batch normalization layer, a ReLU activation function, and an average pooling layer. Two dense layers are used as the classification block, and the final layer uses softmax as the activation function and consists of a one-hot encoded vector that represents the predicted key byte output.

We used *CategoricalCrossentropy* as our loss function, *Adam* as the model optimizer, and *softmax* as the activation function of the final layer. Our network was built, tested an evaluated in the Python library Keras (keras-gpu version 2.9.0) using TensorFlow (tensowflow-gpu 2.10.0) as the backend. *Adam* is first introduced by Kingma and Ba, (2015) and sought to combine the benefits of *RMSPropagation* which performs well with on-line and non-stationary settings, and *AdaGrad* which performs well with sparse gradients (Tieleman, and Hinton, 2012)(Duchi, Hazan and Singer, 2011). We used a learning rate of 0.00002, a batch size of 1,000, and train over 50 epochs.

4. Experimental Results

This section describes the results provided by our framework using the above experimental setup. We trained 10 models using the architecture defined in Section 3.2. Then we tested the accuracy when running the model against the 3,000 attack traces collected at each of the 25 collection points.

4.1 Point Test Results

The average final training accuracy of the 10 trained models was 0.86%. Despite this, the overall average testing accuracy of all 10 models across all 25 points was 0.38%. This is almost equal to the likelihood of randomly guessing the correct key byte, or 1 in 256. Individual points did better than average chance and several points did worse. This means that the model was able to extract useful information from certain points that contributed to correctly guessing the key byte value. However, the lower accuracy points were less helpful in determining the key byte value and in some cases did worse than random guessing. Using the accuracies determined from testing all 10 models, we were able to define which points consistently resulted in higher test accuracy. Figure 5 displays the average accuracies of each collection point in a heat map format, and Figure 6 displays the 95% confidence intervals associated with each collection point. While we cannot conclude that point 6 is statistically better than all other points, points 6 and 10 are statistically significant when compared to one another. This means we can reject the null hypothesis that the location of the collection does not matter. Point 6 returned an average accuracy of .45% and point 10 returned an average accuracy of .33%.

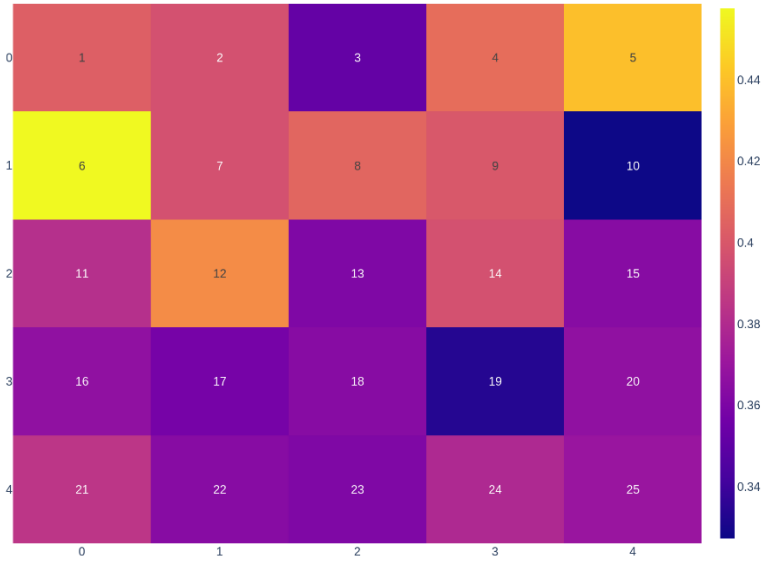

Figure 5: Heat map of average accuracies at each of the 25 points for all 10 trained models

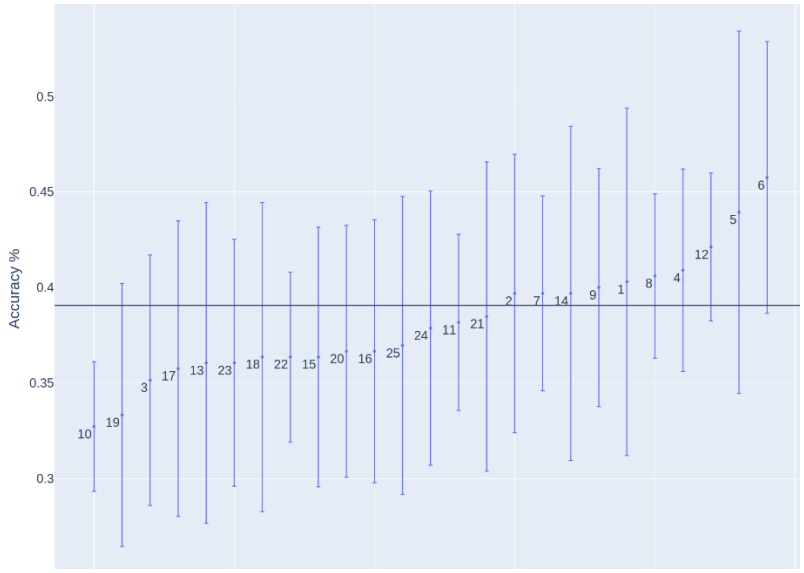

Figure 6: Confidence intervals for each of the 25 points. The horizontal line displays the likelihood of success with random guessing.

To further test the null hypothesis and determine if the accuracies returned by points 6 and 10 were statistically different, a right-tailed t-analysis is performed with an α of 0.05, 9 degrees of freedom, and means of 0.45% for point 6 and 0.33% for point 10. Using the t distribution table along with this alpha and degrees of freedom, results in a critical value of 1.833 (Montgomery, 2020). Using the right-tailed t-test results in a t value for this test of 2.828 which is higher than the critical value. Therefore, the null hypothesis, that position does not matter when collecting EM SCA leakage data, is rejected in favor of the alternate.

4.2 Attack Results:

To demonstrate the effectiveness of our framework, we must execute attacks at each of the identified points to show that the point with the higher test accuracy returns the key byte more effectively. We trained two models using the same CNN architecture defined in Section 3.2, one using the data at point 6 and one using the data at point 10. We then executed attacks using each model. We will refer to these models from here on as model 6 and model 10 respectively. Each attack randomized the order in which the attack traces were received by the Bayesian process to represent real attack scenarios more accurately. Each trace fed into the Bayesian process updates the maximum likelihood array of possible key byte values based upon the previous maximum likelihood array and the prediction returned by the model based upon the new trace.

The final training accuracy of the model from point 6 was 2.98%, and the accuracy at point 10 was 2.28%. The test accuracy at point 6 when tested using model 6 was 3.56%, and the test accuracy at point 10 when using model 10 was 2.4%. We then ran attacks against each model until we had 30 successful attacks. Model 6 was able to return 30 successes in 194 attacks, and model 10 was able to return 30 successes in 549 attacks. This translates to a success rate of 15.46% for model 6 and 5.46% for model 10. Model 6 was able to retrieve the key byte 2.83 times more often than model 10. This demonstrates the effectiveness of our framework in enabling more successful key recovery.

5. Conclusion

Our research demonstrates the success of our leakage search framework on the ChipWhisperer LITE XMEGA platform when tested against a software implementation of AES-128. We trained a CNN on EM data collected from several points over the device of interest and identified points that were statistically significantly different in their accuracy. We demonstrated that training a CNN on the leakage data collected at the identified points with statistically higher accuracy and lower accuracy in a distinguishable difference in the likelihood for success when conducting a side-channel attack. However, the ChipWhisperer Lite platform was developed as a teaching mechanism for those new to the world of SCA, and as such this same research and framework must be demonstrated as successful on other more secure platforms and against common countermeasures such as introducing random noise and lower-level metal routing. This framework could also be tested for effectiveness against a device implementing a hardware-based AES scheme. Other researchers have also used other algorithms to determine the point on a chip with the highest perceived leakage. Further research should be done to understand the effectiveness of both approaches compared to one another to determine the better approach.

References

Andrikos, C., Batina, L., Chmielewski, L., Lerman, L., Mavroudis, V., Papagiannopoulos, K., Perin, G., Rassias, G. and Sonnino, A. (2019). Location, Location, Location: Revisiting Modeling and Exploitation for Location-Based Side Channel Leakages. Lecture Notes in Computer Science, pp.285–314. doi:10.1007/978-3-030-34618-8_10.

Benadjila, R., Prouff, E., Strullu, R., Cagli, E. and Dumas, C. (2019). Deep learning for side-channel analysis and introduction to ASCAD database. Journal of Cryptographic Engineering, 10(2), pp.163–188. doi:10.1007/s13389-019-00220-8.

Danial, J., Das, D., Golder, A., Ghosh, S., Raychowdhury, A. and Sen, S. (2022). EM-X-DL: Efficient Cross-device Deep Learning Side-channel Attack With Noisy EM Signatures. ACM Journal on Emerging Technologies in Computing Systems, 18(1), pp.1–17. doi:10.1145/3465380.

Das, D. and Sen, S. (2020). Electromagnetic and Power Side-Channel Analysis: Advanced Attacks and Low-Overhead Generic Countermeasures through White-Box Approach. Cryptography, 4(4), p.30. doi:10.3390/cryptography4040030.

Duchi, J., Hazan, E. and Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. The Journal of Machine Learning Research, 12.

Gholamalinezhad, H. and Khosravi, H. (2020). Pooling Methods in Deep Neural Networks, a Review. Computer Vision and Pattern Recognition.

Kingma, D. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. International Conference for Learning Representations.

Masure, L., Dumas, C. and Prouff, E. (2019). A Comprehensive Study of Deep Learning for Side-Channel Analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems, pp.348–375. doi:10.46586/tches.v2020.i1.348-375.

Montgomery, D.C. (2020). Design and analysis of experiments. Hoboken, Nj: Wiley.

Moos, T., Wegener, F. and Moradi, A. (2021). DL-LA: Deep Learning Leakage Assessment. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp.552–598. doi:10.46586/tches.v2021.i3.552-598.

Schlösser, A., Nedospasov, D., Krämer, J., Orlic, S. and Seifert, J.-P. (2013). Simple photonic emission analysis of AES. Journal of Cryptographic Engineering, 3(1), pp.3–15. doi:10.1007/s13389-013-0053-7.

Tieleman, T. and Hinton, G. (2012) Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude. COURSERA: Neural Networks for Machine Learning, 4, 26-31.

Wang, H., Forsmark, S., Brisfors, M. and Dubrova, E. (2020). Multi-Source Training Deep-Learning Side-Channel Attacks. 2020 IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL). doi:10.1109/ismvl49045.2020.00-29.