Interpreting Information of Deep Neural Networks for Profiled Side Channel Analysis

Interpreting Information of Deep Neural Networks for Profiled Side Channel Analysis

Thesis

To obtain the degree of Master of Science in Computer Science
at Delft Technical University,
under the supervision of Dr. S. Picek,
to defend publicly on Monday, 30 September, 2019 at 10 a.m.

by

Marius Pop

Delft Technical Univeristy, Delft, The Netherlands,

Supervisor:	Dr. S. Picek	TU Delft
Thesis Committee:	Dr. Z. Erkin	TU Delft
	Dr. E. Isufi	TU Delft

CONTENTS

ABSTRACT

Security has become ever more important in today's quickly growing digital world as the number of digital assets has quickly grown. Our thesis focuses on devices that compute a secure cryptographic operation such that information can be communicated or authenticated. The attack vector utilized is known as Profiled Side-Channel Analysis (SCA) which aims at extracting a cryptographic key from a device through unintended behavior expressed through power monitoring or electromagnetic radiation. Profiled SCA attacks assume the most powerful adversary and therefore allows us to make a sound security assessment of a device in this setting. Our utilized profiling technique includes deep neural networks such as the multi-layer perceptron and the convolutional neural network. As this adds a layer of complexity to our assessment, we must understand how the properties of the network consolidate our security assessment. Previous research has shown that classical neural network metrics such as accuracy does not correlate to how successful or efficient a side-channel analysis is, therefore, we have proposed a mutual information metric. We measure mutual information across each layer in the neural network such that the behavior of each layer in interpreting how each layer is benefiting our classification. We investigate if the mutual information measure can be used to make a beneficial architectural distinction of the neural network for our side-channel analysis problem. Finally, we show there is a relationship between the mutual information and the guessing entropy for our side-channel attack and that it can be used to confirm that the chosen model is fully optimized for the side-channel problem.

PREFACE

This project took place, for the most part, at the premise of Riscure BV. in Delft, The Netherlands. I would like to take the opportunity to convey my gratitude to everyone who I have ever worked with. Firstly, I would like to thank my daily supervisor from Riscure, Dr. I. Buhan for the continued support, research expertise, beneficial insights, guidance and inspirations throughout the research thesis. I would like to thank Guilherme Perin for his instrumental guidance and technical direction during this project. Your expertise was essential to my learning experience at Riscure. Thank you to the company's lunch ladies for providing a great Dutch "second breakfast".

I would like to thank my parents for their continued support and upbringing. They are the ones who taught me to never leave home without your sword - your intellect. I would like to now show them my new sharper and wiser sword. Thank you to all the support from my friends I have made while abroad. Thanks to Jeff, Daan, Valentina and Rico for proofreading my work.

Lastly, the biggest thanks to supervisor Stjepan, for the guidance, inspiration and setting the highest expectations for this work. Thank you for pushing me higher than I could ever think of during this work and believing in me throughout.

Marius Pop
Delft, September 2019

1

INTRODUCTION

The use of deep neural networks in the field of side channel analysis has recently gained traction among researchers for many applications. Although these methods have proved successful in their research, the choice of the architecture for the attack has yet to be explained. We propose to utilize a measure of mutual information to gauge how much each layer is aiding to the classification and train-ability of the deep network. The measure can give us an insight to how the information contributes to the classification accuracy propagating through the network, as well as how the network only regards samples that contribute to classification in a side channel analysis setting.

Secure processing has become more and more apparent in the devices we use in our day to day lives. These include devices such as bank smartcards, networking servers, point-of-sale devices and even some personal computers (PCs). Since devices such as these contain highly confidential or secret information, we want to ensure the security of their computations and communications such as payment, access control, identification and many more. These devices, usually make up many Internet of Things (IoT) hardware involving secure processes utilizing cryptographic algorithms such as the widely adopted, advanced encryption standard (AES) used in our thesis. Due to the nature of these devices, the mechanism of these devices are required to be both secure and efficiently implemented to save on cost or increase speed (depending on function).

Attacks on the AES, recovering the cryptographic key, have yet to be computationally feasible on a text-book [36] implementation. However, side channel analysis enables us to utilize an unintended side effect of circuit design, such as power consumption or electromagnetic radiation, and the communication sequence, such as the plaintext or ciphertext, to disclose some information about the secret cryptographic key. This thesis outline how these type of devices are assessed against side channel attacks and how these assessments can be improved and understood in a profiled setting utilizing deep neural networks. We outline how we determine if the chosen deep neural network is being utilized to its full potential during the side channel assessment.

Side channel analysis against cryptographic algorithms fall under two possible categories, non-profiled and profiled attacks, the difference being the capabilities of the

attacker. In a profiled setting the attacker is given the privilege to have labeled data to be able to train a model. In this thesis, we focus on the profiled attacks such that a deep neural network can be trained from the labeled data. Create a probabilistic model which can further be applied to a target device for an attack on the same cryptographic algorithm's secret key.

1.1. MOTIVATION

Deep Neural Networks are known to produce some of the best results among machine learning frameworks. However this method is notorious for being regarded as a black box, as the inner workings of artificial neural networks are abstract and hard to understand. Security assessment utilizing deep neural networks therefore becomes inexplicable and hard to comprehend how the predictions are made. Without the ability to explain such assessments the methods to improve such devices also become bewildering. As the final goal of such protections is clear, to stop protected information from leaking, the method does not disclose how such information is gained. With better security assessments of these devices, it will enable future devices and algorithms to be improved, such that they become more resilient to side channel attacks.

1.2. OBJECTIVE

The goal of this study is to understand how the layers of the neural network contribute to the attack vector during the side channel analysis. We want to produce a metric that helps the security researcher fully employ the strengths of neural networks to the security assessment of the device under attack. We look into the field on information theory to provide an insight into the neural network's inner workings to make better decisions when choosing network parameters. We propose the following research questions:

1. Can a mutual information measure tell us how much information is needed to correctly guess the correct byte of the cryptographic secret key?

2. Can a mutual information metric tell us how well generalized a neural network model is for the profiled side channel analysis problem?

3. Can a mutual information metric tell us how to choose an effective neural network architecture for the profiled side channel analysis problem? Or why do some architectures work better than others?

4. Can a mutual information metric prevent us from undesired effects of machine learning such as overfitting?

1.3. SCIENTIFIC CONTRIBUTION

Through this thesis, we explore the evolution of neural networks through the learning phase when applied to a side channel analysis scenario. We aim to make such attacks more effective and efficient by understanding how each layer of the network contributes to the final prediction. We apply measures such as mutual information from information theory to explain how the relevant data is processed in the network and when the

network has reached its maximum potential when applying its classification predictions for the security evaluation. At the moment of writing this thesis, the mutual information measure of the neural network has never been applied to side channel analysis setting to the best of our knowledge. Utilizing this new perspective of the deep neural network we plan to:

1. Assess how well a model's metrics translate to the side channel analysis attack success;
2. Assess the information metrics to improve the deep neural network's architecture choices;
3. Determine if the amount of information available in the training traces is adequate for building an adept model;
4. Determine how different models define the input data that is necessary to create an accurate classification.

1.4. OUTLINE

We first introduce some background information about cryptography and machine learning. We further introduce how mutual information is measured in our network across the layers.

Furthermore, we apply our methodology to a synthetic dataset based on the AES intermediate values. We present our initial results for this dataset. We apply the same methodology to a realistic measured dataset and how that our finding still holds for this dataset. We then again apply our methodology to a standardized side channel analysis dataset which contains a masking countermeasures and show that our findings still hold for a more difficult implementation.

2

BACKGROUND

This section will outline the necessary tools and information needed in undertaking the carried out research. This section will explain current techniques used in profiled side channel analysis such as neural networks, cryptographic algorithms such as AES, and the information theory measure of mutual information.

2.1. SIDE CHANNEL ANALYSIS - POWER CONSUMPTION

Power consumption of cryptographic devices such as microcontrollers, smart cards, and FPGAs can be subject to side-channel attacks. Power consumption of these devices can be classified in two ways: static and dynamic consumption. Static power consumption refers to the power needed to keep the device operational and therefore is influenced by circuit design and power required by transistors to preserve their states. Moreover, dynamic power consumption refers to the increase in consumption when circuit logic toggles from 0 to 1 or inversely. In the side channel analysis problems that we look at in this research, we only consider the dynamic power consumption of the circuit, as it depends on the operations and data flowing through the circuits. The static power consumption is regarded as noise.

To successfully mount such attacks the adversary exploits the behavior described above in the circuit design but therefore requires physical access to the target device. Such attacks were initially outlined by Kocher[13]. The attack that will be outlined in this paper will be mainly the profiled attack which makes some initial assumptions about the adversary. Merely that the adversary can profile the characteristics of the current consumption using an identical device, and the control of communication data channel.

2.2. MACHINE LEARNING - DEEP LEARNING

Although deep learning is not a new discovery in the field of machine learning, it has only recently been utilized more by researchers in many different types of fields with promising results and potential. As announced in mainstream media, deep learning has been utilized in the research of autonomous driving cars and image recognition. Through-

out our thesis, we utilized similar techniques to evaluate security through profiled side channel analysis. We start by describing specific deep learning methodologies used in our study. Mainly, we focused our work on two different neural network families the Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN) for our classification tasks. We begin by giving a full description of these types of deep networks and describe the perceived benefit of each.

2.2.1. ARTIFICIAL NEURAL NETWORKS

The deep neural network is based on a chain of artificial neurons. The name suggests a brain-inspired system which abstracts a natural neuron to a computational model. Neurons in artificial networks receive signals from formerly connected neurons which are then multiplied by the weight depending on their importance. These neurons are only activated when a strong enough signal is received and therefore an output signal is sent when the threshold is surpassed by another function. A single neuron is depicted in 2.1

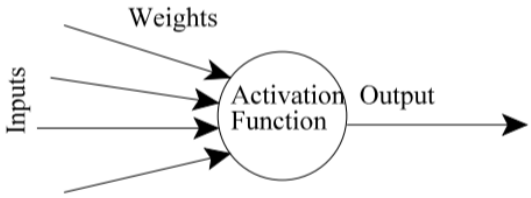


Figure 2.1: Artificial Neuron

Multiple artificial neurons are then organized in layers and feed their signals forward to adjacent layers of artificial neurons, into what we call a artificial neural network (ANNs) that feeds forward their signals without loops or cycles. For simplicity we refer to a neuron $\eta_i \in \eta$ by just the output it generates through an activation function α over the input $x_i \in X$ multiplied by the connection weight $\omega_i \in \omega$ and added to a bias b such that each neuron in layer $\lambda_i \in \lambda$ such that:

$$\eta_i^\lambda = \alpha(\eta_i^\lambda(x_1 \cdot \omega_1) + \eta_i^\lambda(x_2 \cdot \omega_2) + \ldots + \eta_i^\lambda(x_{n-1} \cdot \omega_{n-1}) + \eta_i^\lambda(x_n \cdot \omega_n) + b)$$

Furthermore a layer can be defined as a collection of neurons such that they are connected only to adjacent layers. ANNs only became widespread when a learning method such as the backpropagation algorithm gave the ability of correcting errors over iterations called epochs in a supervised setting.

A supervised setting defines examples of inputs and outputs we expect the network to compute, over which the difference in results is called the network error. Backpropagation, therefore, reduces this error and learns the given training data by adjusting the connection weights to all neurons in the network.

The first layer in the neural network is defined as the input layer. It only propagates each feature individually from the input data and is represented by a neuron for each

feature. Moreover, the final layer is referred to as the output layer where each layer represents a classification class and therefore performs the classification task. Any layer between the input and output layers is referred to as hidden layers, which can take many forms and will be discussed in the next subsections.

Such networks can become large with a high number of tune-able parameters that can be used to induce the behavior that can contribute to increased accuracy. These adjustable parameters are referred to as hyper-parameters which include properties such as network structure, training parameters such as learning rate, batch size, activation functions. The hyper-parameters are chosen before the network begins to optimize the bias and weight variables during the optimization phase.

The activation functions used in this thesis are Rectified Linear Unit (ReLu), Hyperbolic Tan (TanH) and Softmax. In general we utilized ReLu activation function across the convolutional layers, TanH across the fully connected Layers and Softmax at the last output layer in our DNN architecture. These activation functions are chosen based on early experimentation which showed the highest accuracy metrics. These activation functions α are non-linear functions defined by the following formulas:

$$\alpha(x) = max(0, x) \tag{2.1}$$

ReLu is defined in formula 2.1 which outputs 0 or the input if positive.

$$\alpha(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \tag{2.2}$$

TanH is defined in formula 2.2 which outputs on the continuous scale between $[-1, 1]$ similar to the Sigmoid function.

$$\alpha(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \text{for } i = 1, \ldots, k \tag{2.3}$$

Softmax defined in formula 2.3 which normalizes the outputs between (0,1) such that all neuron outputs in the layer sum to 1.

These hyper-parameters are chosen before training and influence the learning and structure of the neural network. Moreover, the network parameters such as the weights and bias are randomly initialized for each neuron in the network and further updated throughout the training phase. During this thesis, we control the hyper-parameters, but allow the optimization algorithm to fully automate the parameter values.

TRAINING AND OPTIMIZATION

Some of the tune-able parameters of a neural network can be updated literately during the training of the neural network through the use of the backpropagation algorithm. Connection weights, bias and convolution filters can be adjusted at the end of each epoch to optimize a chosen metric. This is usually done by minimizing a loss function through the stochastic gradient descent.

Stochastic gradient descent (SGD) performs the optimization task by minimizing the value of the gradients observed from the network based on our loss function. The loss function utilized in our thesis is categorical cross-entropy (CE) which is often used in

multi-class classification [19]. The labels are created such that only one element in the labels is not 0 and therefore the loss is given by formula 2.4, where s_p defines the score of the correct class. Loss, therefore, optimizes the accuracy of the model.

$$CE = -\log(\frac{e^{s_p}}{\sum_j^C e^{s_j}}) \tag{2.4}$$

Stochastic gradient descent can then be computed by computing the gradient at the output neurons, backpropagating through the network by computing the derivative with respect to s_p and updating the weights according to the learning rate (LR). Choosing an appropriate learning rate is required such that the optimization does not get stuck in local minima or take to long to learn due to the LR being too small or does not converge to a minimum due to the LR being too large [39].

A variant of SGD called Adam [12], which promises to be more efficient since it only utilizes first-order derivatives of the gradients and adaptive learning rates, has been utilized in this thesis. Adam is well suited for a side channel analysis problem due to its robustness, ability to deal with high-dimensional parameter spaces, and the ability to perform non-convex optimizations[12].

2.2.2. MULTI-LAYER PERCEPTRON

The feed-forward neural network is utilized in this thesis which entails all neurons only pass their outputs to deeper neurons without loops or backward connections. The multi-layer perceptron (MLP) entails a neural network that is comprised of fully-connected dense layers that feeds-forward all outputs. Each neuron in every layer is connected to all preceding and succeeding layers' neurons. An MLP typically has at least 1 hidden layer in addition to the input and output layers. This network utilizes the backpropagation algorithm to train for the classification problem in a supervised setting.

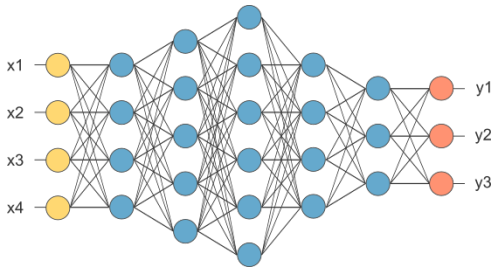


Figure 2.2: Multi-layer perceptron neural network

2.2.3. CONVOLUTIONAL NEURAL NETWORK

Besides, the dense fully connected layers outlined above the use of convolutional layers was also explored. A neural network that makes use of a convolution layer is referred to as a convolutional neural network (CNN). Such a layer consists of filters which are applied to a section of input data much like a sliding window. The input data is operated

on by chunks, equal to the filter size. The filter is moved across the entire input by a specified step size until all data has undergone the transformation through the filter.

The inputs under the filter go through dot multiplication by the filter values and added together to create one value. The filter is then moved and the calculation repeated for the entire input. A bias is then added to the value and the activation function is applied to the resulting value. Convolution typically increases the dimensionality of the input data. The filter values are updated through the training algorithm so that the convolution encompasses the suitable features for the classification task depending on the optimization task.

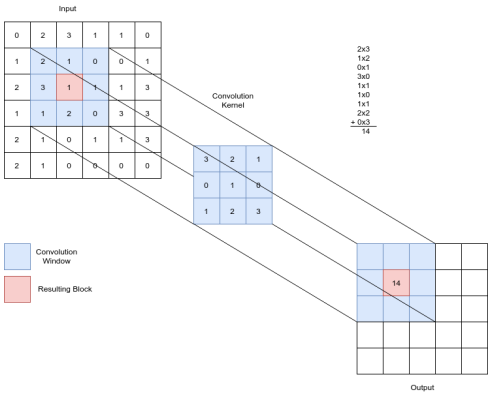


Figure 2.3: Convolution Layer, filter being applied to subsection of input data

Additional to the convolution layer the pooling layer is utilized to both reduce the dimensionality of our data and generalize the input data further. The pooling layer has no trainable parameters but just applies a function to a section of the input data utilizing a sliding window. The function we utilized in this thesis is the max pooling and average pooling such that the largest value in the window is outputted and the average value in the window is outputted respectively. The size and stride of the windows are chosen before training begins. With a pooling size of 2 and stride of 2, the dimensionality of our input data is reduced by half.

It is typical in a CNN that preceding some or a convolution layer(s) that a pooling layer is applied. Since the convolution will increase or preserve the dimensionality, it is reduced again utilizing the pooling layer. This is a way to encompass multiple features or samples in a smaller dimension but preserve the information. These types of layers have been made popular by image classification networks such as AlexNet[14] or GoogLeNet[30].

2.3. CRYPTOGRAPHY - AES

Advanced Encryption Standard, better known as AES is a symmetric-key block cipher algorithm that was approved by NIST in 2001 and has since become an industry-standard [29]. AES operates on data block of 128 bits using cipher keys of 128, 192, and 256 bits. Although it has become an industry standard, its implementations still possesses weak-

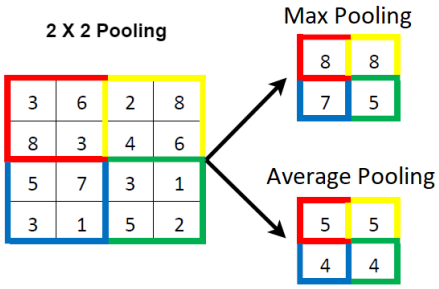


Figure 2.4: Pooling Layer, Outputs of average and max pooling respectively. Dimensionality was reduced from 4x4 to 2x2 utilizing a 2x2 pooling with stride 2

nesses such as the side channel attack studied in this paper.

In this thesis, we have focused on AES-128 which consists of 10 rounds of operation on one input block of 128bits. Each rounds consists of 2 linear permutations, *MixColumns*(except last round), and *ShiftRows* and two non-linear substitution step called *SubBytes*, and *AddRoundKey*. As a result of the non-linearity and dependencies on the input data and the key, we focus on these non-linear operation which often leaks information in the power measurements that could be utilized for a side channel analysis.

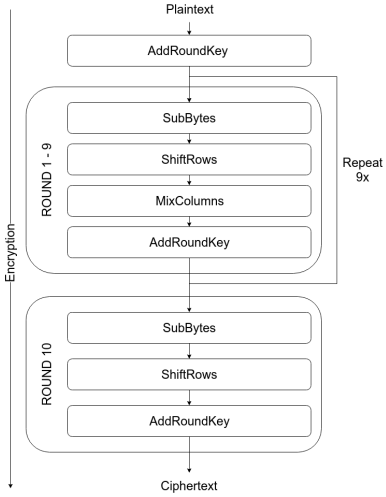


Figure 2.5: AES-128 algorithm

The substitution box output (Sbox Out) is often referred to as an intermediate value in the AES algorithm. Additionally, in a side channel analysis, it is also frequently used as the leakage value of the performed encryption process. The choice of leakage function is however dependent on the target device and implementation of the encryption algorithm.

Side channel analysis methods such as DPA[13] and CPA[2] have been used to examine AES implementations across diverse devices. DPA entails a visual inspection of the measured trace, attempting to suggest specific operations or data dependencies that can be seen graphically utilizing intermediate values expected in the AES algorithm. A more powerful attack according to Briber et al.[2] the CPA utilizes a statistical relationship between the measured traces and hypothetical intermediate values. It makes use of the Pearson correlation 2.5 creating peaks in the correlation coefficient when a correct intermediate is correlated to the measured trace. Utilizing this technique over multiple traces results in an average higher correlation for the correct key guesses. Therefore, this in itself can perform the side channel analysis but also depict wherein the measured traces the points of interest lie.

$$Pearson(x,y) = \frac{\sum_{i=1}^{N}((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{2.5}$$

We made use the Pearson correlation to measure the known key correlation (KKC) by utilizing the trace set and the correct AES intermediate value of the s-box output. By using only the correct intermediate, the correlation coefficient will be highest (or lowest) at the parts of the trace that express the first-order leakage. These points can be referred to as the points of interest (POI) during the AES execution.

2.3.1. PROTECTED IMPLEMENTATIONS

The operations that occur throughout the AES encryption and decryption algorithm are mostly linear operations with the exception of the S-Box. Therefore this operation becomes a perfect candidate for side channel analysis as wrong key guesses will result in correlation coefficients which be distinguishable from the correct key guess. Therefore, the S-Box can be protected by masking the operation so that the all key guesses again become uncorrelated from the real key, including the correct key guess. This protection is better known as a countermeasure and is often used as a hardening technique against side channel attacks. There are many proposed implementations of masking countermeasures such as [1, 6], however, they are all designed to ultimately mask the S-box operation.

Masked implementations are considered to be more secure against side channel analysis however they are still vulnerable to higher-order attacks which correlate multiple power signals samples with the selected computation utilizing the sub-key byte or intermediate value. These types of attacks are often referred to as high order attacks and deep networks are known to be capable of engaging in such attacks [7].

2.4. SIDE CHANNEL ANALYSIS UTILIZING DEEP LEARNING

As the number of embedded devices has quickly grown, the necessity to protect the data on these devices has gained attention, resulting in the necessity of equipping them with cryptographic modules. Side channel analysis (SCA) has emerged as un unexpected behaviour of these devices attackers can exploit. Deep learning as only recently emerged as a powerful machine learning tool attackers can utilize to carry out profiled deep learning

SCA. In addition to other machine learning techniques such as Random Forest and Support Vector Machines [9] the neural networks ability to model complex functions and perform classification tasks it has become widely used in the SCA community [11, 16, 22, 24, 27].

The novelty that came along with the use of neural network came from the reduction in the pre-processing steps required before the attack is carried out. Due to the robustness of the neural networks the need to align the power traces due to clock jitter, random delay countermeasures or desynchronization [3] is eliminated as the neural network can handle these differences in the power traces internally. Usually, networks make use of convolutions to overcome these differences in the input data. Additionally, DNNs are also known to be able to circumvent other countermeasures such as masking [15], due to the DNNs ability to aggregate multiple features.

DNNs typically consists of three parts in a side channel setting. Firstly a convolution part which helps the network identify the most relevant features, a fully connected part which aggregates these features and builds a probability density function (PDF) so that the last part can perform a classification of the data [39]. The first part, however, may be committed for attacks utilizing an MLP which may impact the networks ability to perform the implicit feature selection. A common profile attacks against cryptographic devices are the Template attack. They rely on the probability density function of the data is a multivariate Gaussian distribution. However, the DNN is known to be at least as powerful as other classical profiled attacks [39]. According to Picek et al [11] the VGG network architecture introduced in [33] performs well in the classification task of side channel analysis. The VGG architecture is characterized by many small filter convolutional-pooling layers followed by a small amount of large fully connected (FC) layers. We will continue to use these types of networks in our thesis.

In order to attack an implementation of AES utilizing a DNN, some assumptions are made about the device. A leakage function needs to be chosen for the device. In this thesis, we will be looking at the power leakage of the device being assessed. Due to the physical nature of the device, the leakage function problem is often reduced to a hamming weight (HW) or hamming distance (HD) as it encompasses the number of bit flips which correlates to the power consumption and allows us to translate the problem to a machine learning classification problem. We can, therefore, utilize the DNN utilizing the leakage function as the labels for our network, to perform the profiling task on one AES key byte at a time.

We can utilize the HW of an AES intermediate state as our data labels (9 classes) and perform the profiled SCA utilizing the DNN. This, however, creates imbalanced classes which can affect our classification process [23] but this behavior encompasses the true distribution of the physical behavior. The DNN attempts to maximize the accuracy of the classification. However, this only tells us in which HW class the data belongs to and is, therefore, not the exact data we need to uncover the AES key. This requires another metric called guessing entropy (GE), which aggregates the HW class probabilities by enumerating through all possible AES keys and ranking each key's $k_i \in K$ probability over a large set of inputs $x_i \in X$. This will finally uncover the best key guess we can make based on the classification model. GE has become a standard metric [11] in SCA since it better explains the success of our attack than accuracy. It encompass what is the best key guess

on average across the entire traceset based on our trained model.

$$Accuracy = \frac{\text{Number of True Preddictions}}{\text{Total Number of Classifications}}$$

$$GE = \text{for } k_i \in K \sum_{x_i}^{X} \hat{p}(x_i)$$

Furthermore since a profiled setting utilizing neural networks does not require much pre-processing, it also does not require selecting points of interest (PoIs), since the neural network can do that itself. According to Masure et al. [17] "[PoI] localization can be deduced by simply looking at the gradient of the loss function with respect to the input traces for a trained model". Therefore we can simply look at the input gradients of our network model and identify which points form our input data has the highest impact on our classification choices. In a side channel analysis setting, this essentially identifies the points that the network found to express the most relevant leakage.

2.5. INFORMATION THEORY - MUTUAL INFORMATION

Information theory has been used in the field of security before, more specifically to asses if a cryptosystem can be considered unbreakable due to an adversary not having enough information. Information-theoretic security has been used to show if a cryptosystem achieves perfect secrecy, which is determined by the amount of information a ciphertext conveys about the content of the plaintext. In the case of perfect secrecy, it has been proven by Shannon [31] that the key material needs to be as long as the plaintext to achieve this property. Furthermore, this implies that the probability distribution of any plaintext is completely independent of the ciphertext. The only cipher that is known to achieve this is the one time pad [18]. As AES is not perfectly secret, it is still highly difficult to base an attack of this sort without unlimited computing power.

Entropy was introduced by Shannon [31] as a measure of information. It measures the rate at which information is produced when conducting a process such as encryption or a communication channel. Using this measurement we can determine the amount of uncertainty and choice in the cryptographic system. The formula for entropy is given by 2.6 2.7:

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i) \tag{2.6}$$

$$H(X|Y) = -\sum_{i=1}^{n} p(X_i) \left[\sum_{j=1}^{n} p(Y_j|X_i) \log_2 p(Y_j|X_i) \right] \tag{2.7}$$

Given these formulas we have two special properties expressed by the information measure.

1. H = 0 when x is a constant. This is when we are completely certain of the outcome, entropy is also void.

2. H is maximized when the probability associated with variable x is uniformly distributed and thus the information observed is equal to $log_2(n)$. This is also the context where we are most uncertain about the output.

In the setting of cryptography, we are more interested in the second property since we expect the plaintext and key to be uniformly distributed; we expect entropy to be maximized in the plaintext and key choices. This, therefore, elucidates that we have the highest average uncertainty when guessing the key or plaintext as they are unknowns. Entropy allows us to make a quantitative interpretation of the uncertainty.

With side channel analysis we can gain information from the power consumption measurements about the key and therefore reduce this uncertainty. This relationship can be interpreted as the mutual information between the key and the power consumption measurements.

Mutual Information is a measurement of the shared information between two random variables. This is a excellent measurement for side channel analysis since we are trying to correlate the power consumption measurements to the cryptographic key. To utilize this measure we need to further explore the join and conditional entropies of the two random distributions. The conditional entropy $H(X|Y)$ depicts the average uncertainty about X after observing the second random variable Y. In our context, the measured traces and the intermediate HW values are our X and Y variables in this case, such that we expect that the mutual information to be greater than 0. Figure 2.6 shows exactly how the formula for Mutual Information $I(X;Y)$ can be derived 2.8 and 2.9.

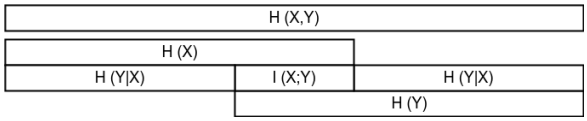


Figure 2.6: Mutual Information relationship to Entropy

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{2.8}$$

Or alternatively

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ I(X;Y) &= -\sum_{i=1}^{n} p(x_i) log_2 p(x_i) + \sum_{i=1}^{n} p(y_i) \left[\sum_{j=1}^{n} p(x_j|y_i) log_2 p(x_j|y_i) \right] \end{aligned} \tag{2.9}$$

Given the formula mutual information also follows two special properties that will further be utilized in the experimental section.

1. Mutual Information is symmetric.

$$I(X;Y) = I(Y;X)$$

2. Mutual Information is additive for independent variables and therefore follows the Data Processing Inequality (DPI)[4], which states for any 3 variables that form a Markov Chain X → Y → Z

$$I(X;Y) \geq I(X;Z)$$

We want to compute this measure for DNN's therefore we utilize the technique from Tishby et al.[32], where we depict each layer T_i of the neural network as a single multivariate variable defined by the encoder, $P(T_i|X)$ and decoder, $P(\hat{Y}|T_i)$ and calculate the mutual information between T_i and the output label Y, and the input data X and T_i.

3

Mutual Information Framework

This section outlines how the mutual information measure is applied to deep neural networks and side channel analysis in our research.

3.1. Side Channel Analysis

In this thesis, we focused entirely on profiled side channel attacks since they estimate the worst-case security risk. Such a setup considers the most powerful attacker and therefore estimates a worst-case scenario attack of the proposed cryptographic implementation.

We are not only interested in predicting the labels but also the guessing entropy of the AES key byte. Although accuracy is being optimized in our deep neural network, it is not exactly what we are interested in, since it only reveals the HW class and does not necessarily give us the information about the secret key we are attacking. Therefore a more representative metric of a side channel attack is guessing entropy or success rate. By utilizing the HW model this is a trade-off we make to reduce the complexity of our model. However, this reduction of the byte representation comes at a cost, subsequent to model training, of key enumeration. The idea that optimization metrics such as accuracy from machine learning techniques do not represent a side channel analysis when using the HW model has been explored by Picek et al. [24]. They argue that guessing entropy is more representative. We, therefore, explore another metric for machine learning techniques, mutual information, to see if it also representative or help explain the guessing entropy metric.

3.1.1. Model Selection

A combination of the fields of cryptography and machine learning is utilized to perform the SCA in this thesis. We have chosen to utilize a supervised machine learning technique using deep learning neural networks. Since side channel analysis can be described as a general classification problem in the profiled setting, therefore deep neural networks

are a feasible tool to enable accurate approximation.

Experiments were carried out utilizing both MLP and CNN deep neural networks since they have shown to perform well in previous studies [27]. Since we have used different datasets, each one needed distinct DNN architecture and hyperparameters to achieve the best classification accuracy. The DNN utilized in this study can be seen in table 4.2. However, in the field of machine learning and optimization algorithms, the "No free lunch theorem" [38] states that "without special consideration of the algorithm at hand, simply observing how well that algorithm has done so far tells us nothing a priori about how well it would do if we continue to use it on the same cost function" [38]. As deep learning has done well so far in this field, the choice of the DNN architecture and hyperparameters has rarely been motivated, rather chosen without explanation, based on prior experience or based on testing accuracy in a black-box setting.

In a side channel analysis setting, we have some assumptions, outlined in section 2, where the model being used to represent the leakage function is the optimal model. However, there has not been any way to determine the validity of this assumption in a profile setting utilizing deep neural networks. The best metric we could currently use in practice has been to compare it to other types of analysis such as template attack or other machine learning techniques utilizing guessing entropy or success rate [25, 35]. As we can't make any correlation between how well a chosen model has performed and will perform, we are forced to train and optimize the model thoroughly to find the model with the best guessing entropy or success rate.

For this study, we propose a method introduced in the field of machine learning by Tishby et al.[32] and apply it to the field of side channel analysis. The study showed that DNNs with enough information follow a specific learning procedure consisting of two phases, stochastic compression, and diffusion. To observe if this behavior is present in the SCA setting, we experimented with different DNN architectures and hyperparameters across different in domain datasets.

All DNN were trained utilizing hardware acceleration on an Nvidia GeForce GTX 1050, additionally, the mutual information calculation is an overhead calculation computed at the end of any chosen epoch which was computed as a multi-threaded process on 8 threads.

3.1.2. EXPLAINABILITY

Utilizing the technique from Tishby et al. [32] will allow us to have insight into the layers of the DNN and allow us to better understand the propagation of information across the network and final decision layer. We will be able to tell which layers preserve information and which layers aid in the learning of the network in a side channel setting. We want to determine how the number of neurons per layer and the number of layers benefits the ability to precisely model a leakage function for datasets with varying levels of noise and difficulties deriving from countermeasure techniques.

With a better understanding of which layers are beneficial to the leakage modeling process, or the profiling phase, we can make better DNNs architectures for the given side channel analysis task. The question about how important the tuning of the neural network in side channel analysis has yet to be answered. Therefore this could potentially help us create models which take advantage of the full potential of DNNs ability to model

complex functions, more efficiently employ computational resources and deliver more thorough security evaluations of cryptographic devices.

3.2. INFORMATION THEORY AND DEEP LEARNING

Deep neural networks generate representations given an input pattern which enable them to make good predictions about the output. They enable the precise estimation of the unknown joint distributions of input and output data. Deep neural networks generate a Markov Chain between the hidden layers of the network, resulting from the hidden layers creating a chain of transitions from one state to another. Since each layer can be regarded as a state in the Markov Chain we expect that the information from one layer is propagated to the next layer. To achieve this characterization using deep neural networks we represent each layer T in the network as a single variable characterized by its encoder P(T|X) and decoder P(Y|T) distributions as in [32]. The encoder is part of the network which takes an input sequence and maps it to a different representation, while the decoder is another part of the network which takes an arbitrary sequence and maps it to the output labels.

To do this estimation of the mutual information at each layer, we bin each layer's output activation function, in our case *relu* or *tanh*, into 30 equal intervals between -1 and 1. This allows us to estimate the join distribution over the each equally likely input trace x ∈ X, $P(T_i|X)$, since each measured trace is unique, and $P(Y|T_i) = \sum_x P(x|Y)P(T_i|x)$ using the Markov chain for each hidden layer[32].

3.2.1. MUTUAL INFORMATION

To understand how networks evolve from random initialization to optimized representations of leakage functions we measured the mutual information between each layer and the input and with the labels at logarithmic scale epoch intervals. This measure how much each layers contributes to the final classification or shows the incremental transformation in information, affects adjacent layers in the network. More specifically mutual information w.r.t the input $I(X;T_i)$ and output $I(T_i;Y)$ is utilized to investigate each layer of the DNN as outlined in section 3.4.

$$I(Y;T_i) = H(Y) - H(Y|T_i)$$
$$I(Y;T_i) = -\sum_{j=1}^{n} p(Y_j)\log_2 p(Y_j) + \sum_{k=1}^{bins} p(T_{i_k})\left[\sum_{l=1}^{n} p(Y_l|T_{i_k})\log_2 p(Y_l|T_{i_k})\right] \tag{3.1}$$

$$I(T_i;X) = H(T_i) - H(T_i|X)$$
$$I(T_i;X) = -\sum_{k=1}^{bins} p(T_{i_k})\log_2 p(T_{i_k}) + \sum_{j=1}^{n} p(X_j)\left[\sum_{l=1}^{bins} p(T_{i_l}|X_j)\log_2 p(T_{i_l}|X_j)\right] \tag{3.2}$$

As we can see in the two formulas the encoder and decoder distributions are necessary to calculate the mutual information at each layer. Since we are binning the layer activation outputs, this is an estimation of a continuous distribution as we do not know the true distribution.

According to the DPI property, information can not increase as it propagates through the layers of the network. All information is available in the first layer. Due to random

initialization of the neural networks, the layers are expected to quickly lose information the deeper they are in the architecture as the network does not yet have the ability to process this information. However, the back-propagation algorithm which is essential in training the neural networks through stochastic gradient descent (SGD) optimization, allows the network to more effectively propagate the information that is required by the classification process. The measure of mutual information will be used to see how effectively the DNN's back-propagation algorithm can identify the relevant information at each layer of the NN and how this information aids in making the classification decisions.

In a side channel analysis setting little information regarding the leakage exist in each trace initially. Therefore, even when maximizing mutual information, may not result in a high accuracy across the network, but shows that the network is maximizing all available information. The probabilities utilized in the mutual information measure, mainly the decoder, depicts which samples in each measured trace are correlated to leakages, and are therefore useful in calculating the leakage function; and the encoder, depicts the smallest number of binary questions which can be used to classify the inputs.

Considering DNNs are initialized randomly, it is highly probable that multiple optimized generalizations can exist for the same problem. This, however, does not affect our mutual information measures since we are representing a layer by only the encode and decoder distributions which are not affected by an individual neuron but rather the aggregation of an entire layer [32]. Additionally, we measure the mutual information across 5 recurrences of training for each experimental architecture and hyper-parameter choice to create a generalization of the measure. An aggregation was utilized for the sole purpose of making a good compromise between run-time and precise representation of the mutual information metric.

3.2.2. THE INFORMATION PLANE

The mutual information was calculated at each layer according to the DPI property as follows:

$$I_Y \cong I(X;Y) \geq I(T_1;Y) \geq I(T_2;Y) \geq \ldots \geq I(T_k;Y) \geq I(\hat{Y};Y) \tag{3.3}$$

and

$$I_X \cong H(X) \geq I(X;T_1) \geq I(X;T_2) \geq \ldots \geq I(X;T_k) \geq I(X;\hat{Y}) \tag{3.4}$$

Each term in the inequality above is plotted in what is called the DNN's information plane [32], such that the terms of inequality 3.3 is plotted on the Y-axis and the terms of inequality 3.4 are plotted on the X-axis. Additionally, we depict the function at logarithmic scale epoch intervals and represent it as different colors. This, therefore, represents where each layer of the chosen DNN architecture sits in the information plane.

In a side channel analysis setting, where information in the traces about the key byte we attempt to attack is not initially high in most cases, as this is the main motivator to why many traces are needed for such an analysis; The chosen model must optimize any of the available information for both a successful attack and an effective security evaluation. We will be able to evaluate architectures and hyper-parameters that make this optimization possible due to the ability to inspect how individual layers favor the analysis. The information plain effectively shows how each layer in the DNN aids to the

classification of the data and optimization of the leakage function. According to Tishby et al. [32] an optimized network maximizes I_Y and minimizes I_X such that, I_Y grows due to stochastic optimization and the empirical error decreasing and I_X decreases due to layers losing irrelevant information and the compression of data until convergence.

Furthermore, we want to investigate how the amount of information from the traces affects the DNN's ability to effectively model a leakage function. We look to answer how noise affects the choice of architecture and hyper-parameters of the neural network, or whether noise can only be overcome by measuring more traces to make them available for training the DNN. Under added noise or masking we can observe what benefits of adding hidden layers or increasing layer widths, contribute in a side channel analysis setting. If layers still propagate information such that I_Y is increasing there are still benefits to adding additional layers in a noisy environment.

The information plane allows us to see how early in the training phase a network becomes optimized or under/over-trained by seeing which layers reach their maximum information bottleneck and which don't. As DNNs are known to utilize hardware extensively, this method allows us to visualize if there are any benefits to be gained from consuming these resources for longer to optimize the chosen DNNs.

To investigate how the mutual information measures translate to a side channel analysis measure we use guessing entropy in parallel to I_Y and correlation of input gradient to known key correlation in parallel to I_X to observe if these measures are consistent. We expect as the output layer's mutual information about the labels to increase as the guessing entropy and empirical error reduces. Additionally, as the mutual information about the input decreases in the output layer, the correlation between the input gradients and the known key correlation is expected to increase. Both of these metrics relate the mutual information measure to quantities more representative of a side channel analysis problem.

4

METHODOLOGY

This section outlines how datasets were acquired to carry out the side channel study, as well as what experiments were carried out to test our hypothesis.

4.1. EXPERIMENTAL SETUP

4.1.1. DATASETS

The experiments in this study utilized power consumption measurements referred to as traces, measured by an oscilloscope, of the cryptographic engines or general-purpose processor depending on the device architecture and cryptographic implementation. The exact measurement setup for each implementation will be discussed below independently. Generally, the setup included an oscilloscope connected to the VCC or VDD terminals of the computational core across a resistor. The power consumption is measured in different versions of AES-128. The traces were exported to a computer where they were pre-processed utilizing ©Riscure's proprietary software called Inspector. The number of traces collected varied per implementation, generally, the more protected implementations required more traces to mount the proposed attack. Acquisition of the traces is started by utilizing a trigger, which sets a general-purpose pin high at the exact moment the encryption operation begins in the target device 4.1.

TRAINING AND VALIDATION SETS

One of the most important stages in a side channel analysis is the acquisition phase. Having traces that accurately represent the device's security computation is essential for an effective security assessment. Additionally, machine learning algorithms such as DNN can be highly affected by the type and amount of data they receive as input. Some problems have been outlined by researchers such as overfitting [8] and class imbalance [24] that are also apparent in a side channel analysis problem. Since class imbalance is naturally occurring in the device we should not remove this attribute of the dataset. However, overfitting has been addressed by many methods in DNNs, such as dropout and regularizes [34]. Overfitting in a side channel analysis setting causes the illusion of

Figure 4.1: Acquisition Setup

a good model but does not offer any practical quality estimation of the security of the system or the attack's success. To address this issue at a higher level and reduce the possibility of overfitting, we built the training set such that each trace was collected using a random key and random plaintext (RKRP) and only used a unique static key (SKRP) in the validation set.

This setup allows us to more closely imitate a multiple device model where the issue of portability is addressed in a side channel setting. Using random keys in training ensures we don't allow our model to overfit the correct key. This dataset model does not, however, explore the different behavior observed in power measurements across multiple devices and remains to be explored in further studies.

4.1.2. DATASET ACQUISITION SIMULATED AES

To firstly explore a simple dataset where HW values of AES-128 are feature samples part of the traces. The simulated traces contain the hamming weight of the S-box output state of each byte in the first round, preceded and succeed by 100 discrete uniform random samples between 0 and 8 4.1. The samples of interest lie at indexes 100-115 and are identical to the labels which will be used later in training. There are 100,000 traces available in both training and validation sets which have been generated through Riscure's Inspector.

$$\begin{aligned} & r \in \mathbb{W}_9\{0 \dots 8\} \\ & b \in \text{AES state after S-box Round 1} \\ & t = r_0 \dots r_{99}\ HW(b_0) \dots HW(b_{15})\ r_{100} \dots r_{199} \end{aligned} \tag{4.1}$$

The proof of concept aimed at investigating if the information plain findings outlined by Tishby et al. [32] held valid in a side channel analysis attack and if the finding were

reproducible with our dataset. This would allow us to investigate if DNN can learn leakage functions in the same way they DNNs are used in image recognition or other general applications of DNNs.

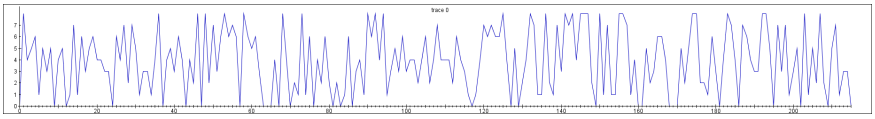


Figure 4.2: AES Simulated Trace - All S-Box Outs

MODEL OUTLINE

The analysis performed on this dataset was conducted in a profiled setting utilizing both MLP and CNN deep neural network architectures. This is the simplest dataset in our study, as the label resides in one of the columns of the input data, therefore the smallest network base architecture was chosen to carry out our analysis. Additionally, the hyperparameter search was minimal for the same reasons. Moreover, we also correlated the known intermediates with each trace to create a known key correlation figure 4.3, which reinforced our assumption of the leakage function. The results of this plot proved that only one sample was highly correlated to our labels.

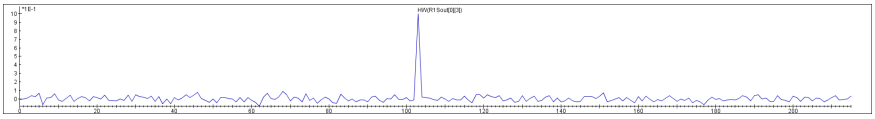
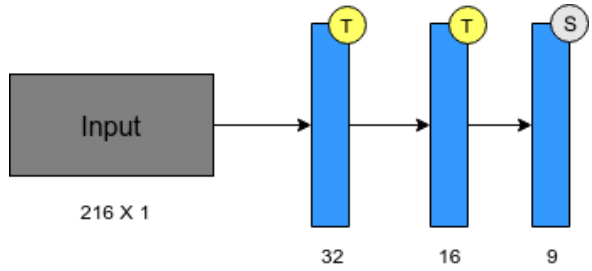


Figure 4.3: AES Simulated Trace - Known Key Correlation S-Box Out Byte 3

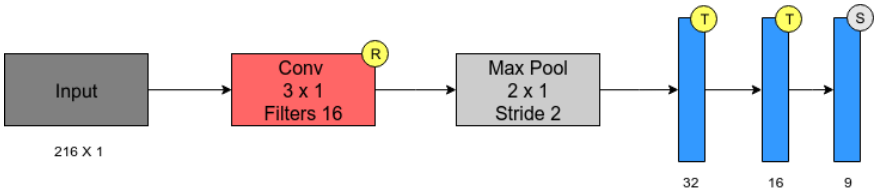
A base training model was chosen for each DNN type. As stated earlier, this dataset did not require any complex functions to be modeled as the labels exist as a feature in the input. Therefore, the base MLP included an input layer (216), 2 fully connected dense layers with 32 and 16 neurons wide respectively, and 1 fully connected softmax layer with 9 neurons (representing each hamming weight class). The CNN model was similar with the addition of a convolutional layer with 16 filters and size 3 and a max-pooling layer with size 2 and stride 2 between the input and dense layers. Summary of the networks used in the experiments can be seen in table 4.2.

4.1.3. DATASET ACQUISITION SMART CARD

The smart card dataset was acquired from a Riscure training academy smart-card based on a ATMega 163 IC in a smartcard form factor implementing an unmasked software AES-128 FIPS 197 standard; 8-bit implementation. The measurements were taken using a PicoScope 5000 at 500 mega samples per second (MS/s) and 100mV of range, with each trace containing 31,000 samples. The smart card included no data countermeasures and only incorporated a software random delay also known as a jitter that was corrected through a post-acquisition alignment. Since the algorithm is software-based we were able to analyze the traces collected and determine the operations by visual inspec-



(a) Simulated AES MLP Base Network



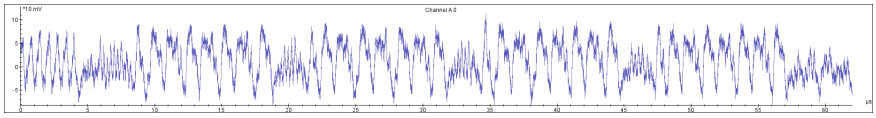
(b) Simulated AES CNN Base Network

Figure 4.4: Simulated Networks

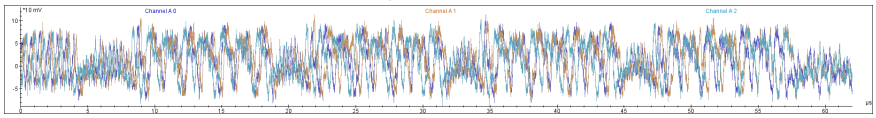
Activations: R:ReLu T:TanH S:Softmax

tion and calculating intermediate value correlation. Therefore the final traceset used for the attack included only the first round until the first s-box output.

As this was a measured power consumption from a physical device, this dataset contained a more natural sample pattern which added a higher difficulty of analysis. The measurements contain the power consumption of the AES operations, along with measurement error from the tooling and static consumption of the smart card itself. There are 100,000 and 50,000 traces available for training and testing sets respectively. Preprocessing included a window re-sample to reduce the number of features and alignment to eliminate the jitter miss-alignment issues. Utilizing a known key correlation measure 4.7 we can see that multiple samples correlate to the AES intermediate we are attacking. Therefore, a much more complex leakage function needed to be modeled us-

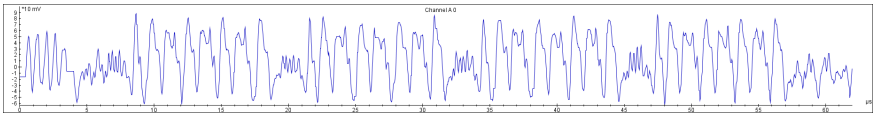


(a) Single SmartCard Trace

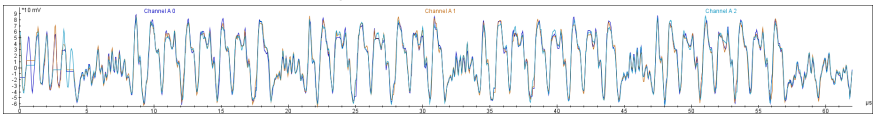


(b) 3 SmartCard Traces Overlapped

Figure 4.5: Raw AES SmartCard Traces



(a) Single SmartCard Trace Re-sampled



(b) 3 SmartCard Traces After Re-sample and Alignment

Figure 4.6: Pre-Processed AES SmartCard Traces

ing an appropriate neural network. The points of interest highly depend on the target byte being attacked. For byte 3 the points of interest lie between samples 530 - 830 seen in 4.3.

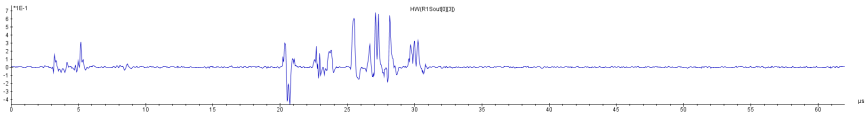


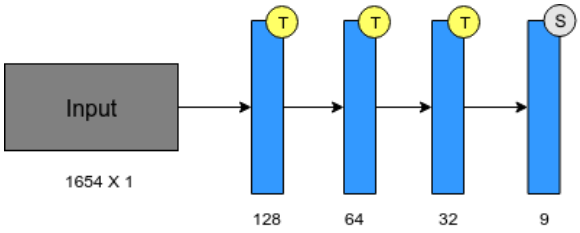
Figure 4.7: SmartCard AES Trace - Known Key Correlation S-Box Out Byte 3

MODEL OUTLINE

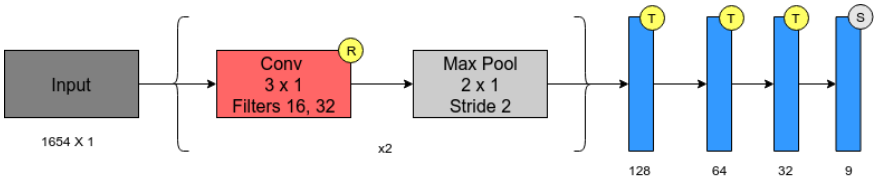
The base DNN model for this dataset included more layers as this was a measured dataset and a statistical model needed to be built to identify the leaking samples. As shown in figure 4.7, this implementation has a considerably large amount of leakage in the S-Box out HW model, therefore, we started experimenting with a shallow and narrow network initially. The architecture's chosen complexity addressed a trade-off between runtime and size of the network. The base MLP architecture included 1654 node input layer, 3 fully connected dense layers of 128, 64, and 32 neurons respectively, and a fully connected softmax layer with 9 neurons (representing each hamming weight class). The CNN base model added 2 convolutional layers of size 3 with 32 and 16 filters respectively followed by 1 pooling layer with stride 2 and size 2, between the input layer and the dense layers. Summary of the networks used in the experiments can be seen in table 4.2.

4.1.4. ASCAD AES

This dataset is acquired by ANSII and CEA under the REASSURE project and has become a standardized dataset in the side channel analysis community. This is a masked implementation of AES-128 with varying levels of desynchronization. However, it should be noted at the time of carrying out our experiments, the secret key is used for the encryption process was identical in the profiling and attacking data tracesets. This dataset does not add to our portability and generalization characteristics for our tests but contributes to the explainability of the deep network utilized for the experiment. We utilize this dataset to compare to the originally reported results and use the proposed metric



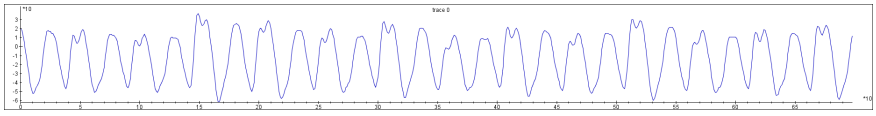
(a) SmartCardAES MLP Base Network



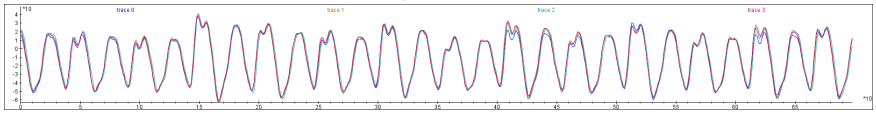
(b) SmartCardAES CNN Base Network

Figure 4.8: SmartCardAES Networks

Activations R:ReLu T:TanH S:Softmax



(a) Single ASCAD Trace



(b) Overlapped ASCAD Traces

Figure 4.9: ASCAD 0 De-synchronization Traces

to understand the inner workings of the utilized deep network architecture, including a proposed improvement of the architecture.

This dataset did not undergo any pre-processing steps so that the experiments performed could be closely compared to existing side channel analysis results. Therefore the raw traces were used as input to our neural network. This dataset included the least amount of traces which contributed as a large challenge with only 60,000 traces in total. Granted it did not incorporate the RKRP characteristic for the training set, due to the traceset being a standardized dataset from ANSII. Therefore the entire dataset utilized a uniform secret key, however, masked, for both the trainset and validation set. For our experiments, we only utilized the 0 de-synchronization dataset. However, it prevails as more difficult than the previous datasets used due to the masking of the key and the small amount of traces available.

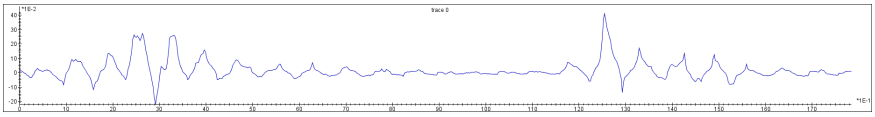
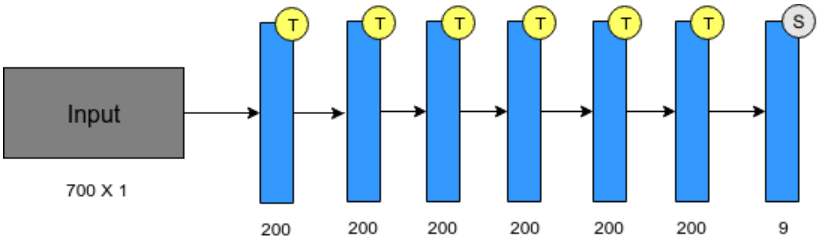


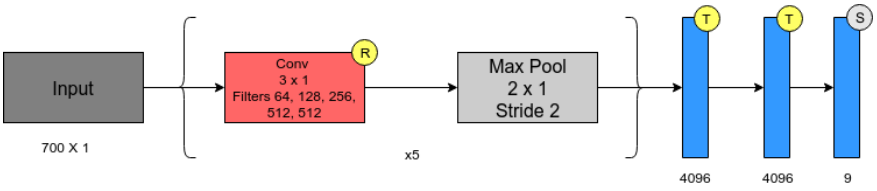
Figure 4.10: ASCAD Trace - High Order Known Key Correlation Masked S-Box Out Byte 3 and Mask

MODEL OUTLINE

The utilized base neural network architectures were taken from the original research [27] to first investigate how well we can reproduce their reported results. We made a small modification to the dataset. The labels were converted to use the HW leakage model instead of the byte ID. Therefore the output softmax layers of the mlp_{best} and cnn_{best} architecture were modified so that it only contains 9 neurons to represent the nine possible HW values. This modification was done so that our results are consistent in using the same leakage model across all datasets. The MLP model included 6 dense fully connected layers of 200 neurons. The CNN included 5 convolutional layers with 64, 128, 256, 512, and 512 filters respectively, each followed by their own max-pooling of size 2, followed by two fully connected dense layers of 4096 neurons. Summary of the networks used in the experiments can be seen in table 4.2.



(a) ASCAD MLP Best Network



(b) ASCAD CNN Best Network

Figure 4.11: ASCAD Best Networks

Activations: R:ReLu T:TanH S:Softmax

4.2. EXPERIMENTAL PROCESS

Experiments were carried out by first training the base model, performing the mutual information calculation and guessing entropy metric at logarithmic scale epoch intervals. All calculations reported in this thesis were performed using only the attack set, while

the training set was used solely for training the neural network.

We performed 5 different types of tests utilizing each dataset:

1. Increasing or reducing the number of layers in the neural network
2. Increasing or reducing the width of the layers in the neural network
3. Reducing the size of the training set size (number of traces)
4. Adjusting Hyper-parameters such as batch size, convolution kernel size, or pooling window size, etc.
5. Adding random Gaussian noise to the training data synthetically

After each test, we investigate the information plane to consider what influence, if any, it had on the neural network's ability to produce favorable side channel analysis results. We also considered the benefits against the runtime of the training procedure. Increasing the width and number of layers in the DNN will allow the network more flexibility to the type of complex functions, such as the leakage function, it can model [28, 37]. Reducing the size of the training set and adding random Gaussian noise investigates how flexible the neural network is to perturbations [11]. Previous research [11], outlined that noise allows the network to generalize beter as it acts as a natural regularizer, we want to reproduce the results and investigate how the layers behave in regards to the information plane. Adjusting the hyper-parameters allows us to investigate how important tuning is in a side channel analysis setting. The resulting differences can be observed in the information plane as to which layers are propagating in the plain in the favorable direction outline by Tishby et al.[32] and reinforced with metrics such as accuracy and guessing entropy.

To investigate how the information plane metrics relate to side channel analysis metrics we utilized the metrics of guessing entropy and input gradient to KKC correlation. As the final decision in the neural network is taken in the final layer, the guessing entropy should be highly related to the amount of information about the output of I_Y. Additionally, the information about the input I_X is expected to reduce as the points of interest are identified and compressed, which results in increasing the input gradient to the KKC correlation. If these metrics do not show the expected behavior it becomes evident that the neural network did not perform a successful side channel attack, but may also hold information as to how we could modify the neural network so that more favorable results could be obtained. Modifying the neural network in one of the 5 ways listed above may be enough to produce a large enough difference in the model's ability to preserve the correct information and transform the data to the appropriate classification.

Dataset Summary									
Dataset	Cipher	Measurement			Countermeasure	Pre - Processing	Trace Properties		
		Oscilloscope	Sample Rate (MS/s)	Range (mV)			Number of Features	Number of Profiling Traces	Number of Attack Traces
Simulated	AES - 128	N/A			None	N/A	216	100,000	100,000
SmartCard AES	AES - 128	PicoScope 5000	500	100	None	Resample + Allignment	1654	100,000	50,000
ASCAD AES	AES - 128	EM Measurement	2000	N/A	First Order Masking	N/A	700	50,000	10,000

Table 4.1: Dataset Characteristics

Base Model Summary									
	Convolution			Pooling		Fully-Connected		Training	
Datasets	Size	Padding	Channels	Type	Size	Units	Activation	Batch Size	Learning Rate
Simulated						32, 16	TanH	521	0.0004
	3	Same	16	Average	2	32, 16	TanH	521	0.0004
SmartCard AES						128, 63, 32	TanH	521	0.0004
	3	Same	16, 32	Average	2	128, 63, 32	TanH	521	0.0004
ASCAD						200,200,200,200,200,200	TanH	521	0.0004
	3	Same	64, 128, 256, 512	Average	2	4096, 4096	TanH	521	0.0004

Table 4.2: Base Models

5

Experimental Validation

Results will be presented iterative through each dataset. We outline our findings and present how mutual information contributes to improving profiled deep learning side channel security analysis.

5.1. Simulated AES Results

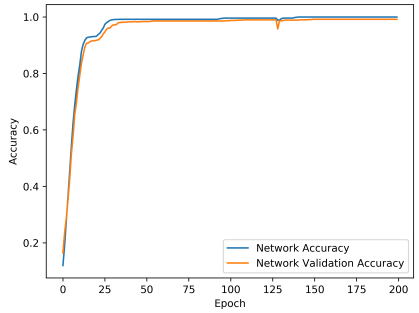
5.1.1. Deep Neural Network Architecture

MLP

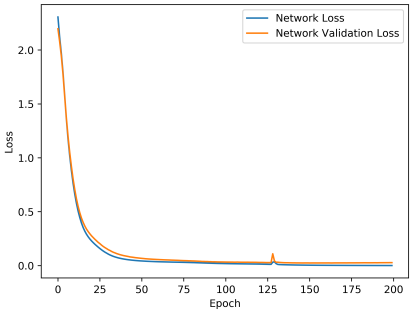
The chosen base models already show positive results in terms of both classical machine learning metrics and side channel metrics. The small and shallow architecture performs well in this side channel scenario as both training and validation accuracy increase together with minimal overfitting. Additionally, the mutual information measure starts quite high in both measures, I_X and I_Y across all layers of the DNN. From the two hidden layers of width 32 and 16, only the latter can be seen moving along the information plane and aiding the output layer in the later phase of training.

We can see from the accuracy and loss metric 5.1 that this DNN is highly successful at making precise and consistent classification eventually reaching close to 100% accuracy and 0 loss on both training and validation data sets. In side channel analysis such high accuracy is not necessarily required to have a successful attack but will significantly reduce the number of attack traces needed. We therefore achieve a successful attack on this dataset with as little as 1 trace 5.2.

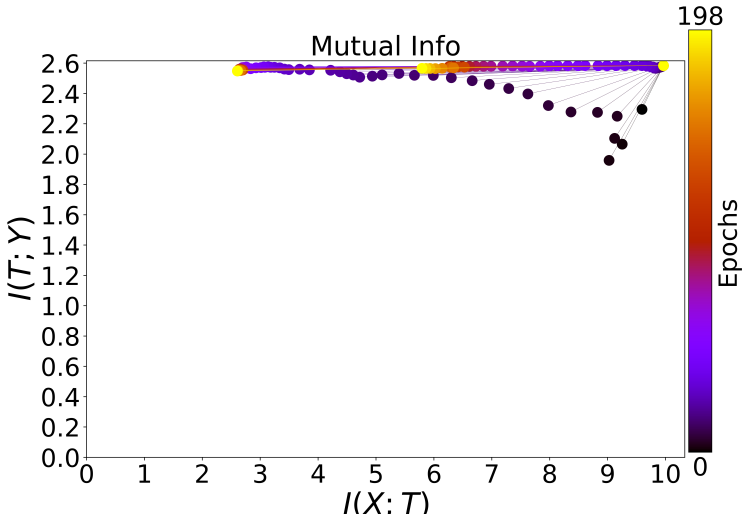
We reported guessing entropy by utilizing the validation traces as the attack set and computing the guessing entropy throughout the training. This shows us exactly how the model evolves with respect to a side channel metric rather than a classical machine learning metric. This metric can also show how early in training we have successful attacks and at what point the most efficient attack lies. We classify a successful attack when the guessing entropy of the correct key is 0, meaning that the best key guess we can make is the correct key.



(a) Simulated Dataset Base MLP Model Accuracy



(b) Simulated Dataset Base MLP Model Loss



(c) Simulated Dataset Base MLP Model Mutual Information Measure

Figure 5.1: Simulated AES dataset Base Model Results

We can see that in this dataset we performed a successful attack after just 2 epochs utilizing about 150 traces out of our 1000 attack trace set 5.2. However, the most successful attacks, which utilized the least amount of traces, occurred after epoch 10. To give us the best chance at a successful attack with the current architecture and hyperparameters we should, therefore, stop training after 10 epochs.

CNN

The base CNN network also shows high performance in terms of the metrics however it did not perform as well as the MLP in terms of classification accuracy or GE. The model, however, is still good enough to perform a side channel attack on key byte 3 with a low amount of attack traces. Utilizing the base architecture we reach a validation accuracy of about 50% however the training accuracy is above 80% 5.3. This metric tells us that the model is overfitting the training data, this behavior is considered to be undesired in

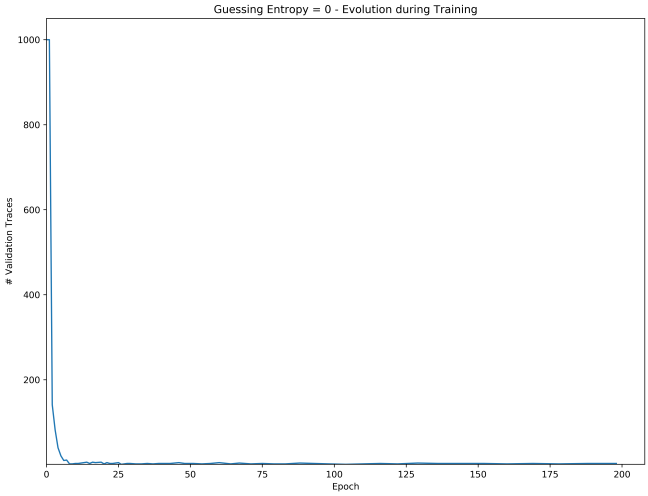
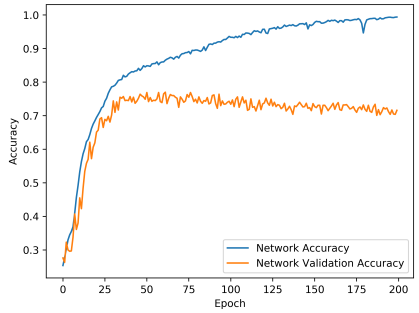


Figure 5.2: Simulated Dataset Base MLP Model - Number of traces needed to reach a guessing entropy of 0 computed throughout training utilizing the validation set

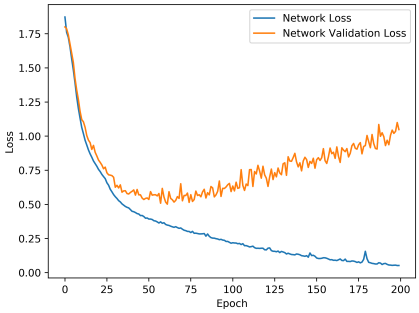
machine learning models. Typically this can be avoided by utilizing regularization deep learning techniques such as batch normalization [10], dropout or regularization (L1/L2) [20, 34], or early stopping[26]. Since we wanted to explore the effect of overfitting in the side channel analysis setting we did not employ these techniques and are recommended for further works.

We believe the CNN model has lower accuracy than the MLP since the convolution combines multiple samples however the points of interest in this dataset reside in only one feature. Due to the setup of our dataset, an MLP is expected to perform better. This can be seen in the Input gradient correlation to the known key correlation plot 5.4. We can see that the MLP model achieves perfect correlation between the input gradients and the KCC while the CNN get very close but varies throughout training as the convolution filters and neuron weights are adjusted.

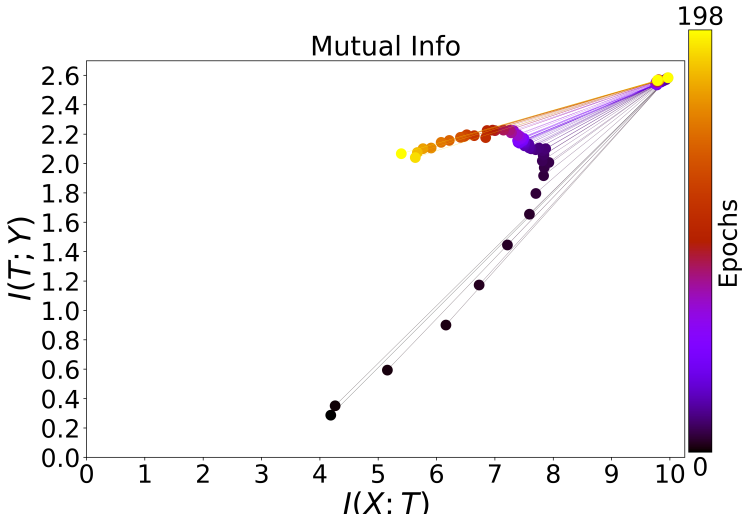
The side channel attack on byte 3 was also successful utilizing the base CNN architecture with GE reaching 0 after about 20 epochs however the most effective attacks where the least amount of attack traces are needed is reached after epoch 50. We then label epochs 0 to 20 as the learning phase of the network, epochs 20 to the end of training the generalization phase. We do not consider this network to be overfitting in terms of side channel analysis since the number of traces needed to reach a guessing entropy of 0 does not degrade as seen in figure 5.5. We can, therefore, stop training around epoch 50 as the model no longer develops after this point.



(a) Simulated Dataset Base CNN Model Accuracy



(b) Simulated Dataset Base CNN Model Loss



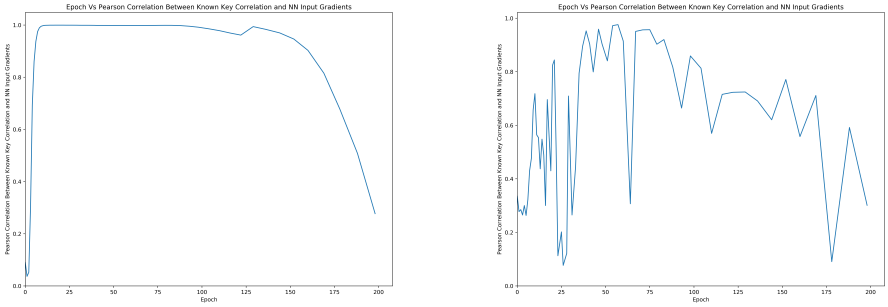
(c) Simulated Dataset Base CNN Model Mutual Information Measure

Figure 5.3: Simulated AES dataset Base CNN Model Results

5.1.2. INFORMATION PLANE

Calculating mutual information across the layers of a DNN has not been done in a side channel analysis setting to the best of our knowledge. We, therefore, wanted to compare initial results in this thesis to the findings of Tishbi et al.[32]. We, however, do not witness the deeper layers losing relevant information until the last layer 5.1c. The output layer then starts by gaining information about the output very quickly, furthermore, the layer loses information in the later phases of training about the input as outlined in the original study. We can, therefore, say that the two-phases witnessed in the original study, mainly stochastic drift and relaxation are also apparent in a side channel analysis setting.

In the case of the base MLP architecture mutual information regarding the output labels I_Y remains high at the beginning of training, quickly rises to nearly the maximum



(a) Simulated Dataset Base MLP Model Pearson Correlation between the KCC and the Input Gradients

(b) Simulated Dataset Base CNN Model Pearson Correlation between the KCC and the Input Gradients

Figure 5.4: Simulated AES dataset Base

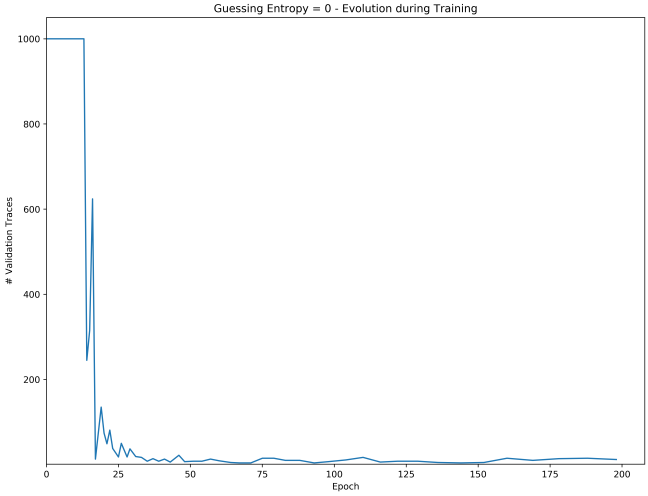
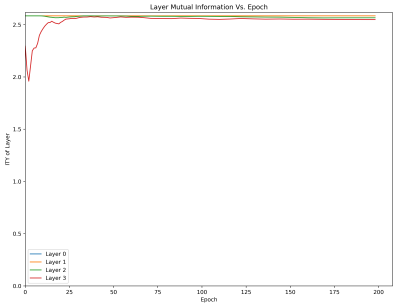


Figure 5.5: Simulated Dataset Base CNN Model - Number of traces needed to reach a guessing entropy of 0 computed throughout training utilizing the validation set

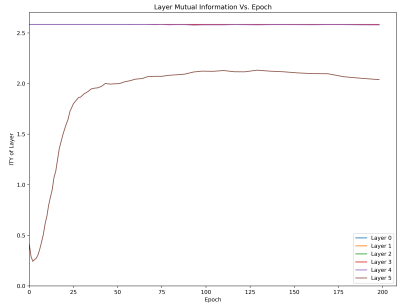
in the first 20 epochs after which the network is optimized such that input information is understood by the network and therefore the network starts decreasing mutual information about the input I_X so that only the necessary information is utilized. The last hidden layer also starts decreasing the mutual information regarding the input I_X beginning epoch 30 which accelerates the information gain in the output layer. It is possible that the shallow layers only generalize or prepare the data while the deeper layers are specialized and participate in the classification process. This hypothesis can be tested

by changing the architecture of our network, which is discussed in a later section.

Furthermore, the base CNN architecture followed the same pattern. Only the output layer of the network lost information at the random initialization of the network parameters. The CNN network quickly gained information in both I_X and I_Y in the early epochs, where I_Y peaks around epoch 45. The networks diffusion appears to occur at this point while I_X decreases which also causes a slight decrease in I_Y as a side effect.



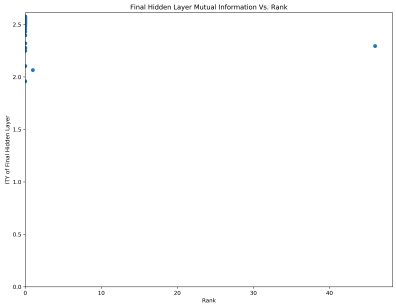
(a) Simulated Dataset Base MLP Model - I_Y mutual information in the output layer



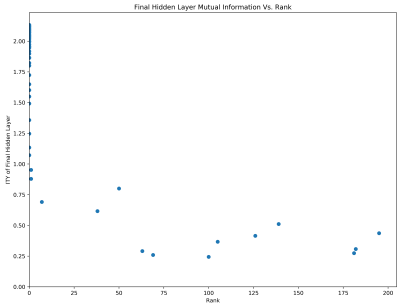
(b) Simulated Dataset Base CNN Model - I_Y mutual information in the output layer

Figure 5.6

We investigate how the information plane translates to side channel analysis by plotting the rank of the correct key, assessed after utilizing 1000 attack traces, to the amount of mutual information I_Y in the output layer 5.7. This tells us how much information is needed to have an optimal result, or how much information is needed for a successful attack. We can see in the case of MLP any I_Y above 2.2 will result in rank 0 for the correct key, while in the CNN any I_Y above 1.0 will also result in rank 0 for the correct key. It is important to see that the rank consistently stays at 0 through the rest of training.



(a) Simulated AES MLP base model



(b) Simulated AES CNN base model

Figure 5.7: Relationship between I_Y and Rank for the SmartCard Dataset

5.1.3. SENSITIVITY TO MODEL ARCHITECTURE

We investigated how important it was to choose a model that can be optimized for the side channel analysis problem. We carried out experiments by changing the number of layers, by either removing or adding a layer to the base model. Additionally, we experimented with increasing or decreasing the width of layers. We have made these changes without changing other parameters so that we can be sure that the changes in the behavior of the model are a result of the architectural changes.

Adding a larger 64 neuron wide layer at the beginning the network does not seem to make any differences in the results of our experiments using this dataset. We still witness only the last two layers of our network traversing through the information plane. This, however, adds tune-able parameters to our network, we can, therefore, expect that the network may need more epochs or a higher learning rate to achieve the same measures. The slight differences we see in both accuracy and the information plane do not make much of a difference when considering the side channel analysis metrics.

For the MLP slight overfitting can be seen in the accuracy metric 5.8a, however, the accuracy is already high enough for a successful side channel attack such that the rank of the correct key is unaffected5.8c. Mutual information in I_Y is unaffected by the extra layer of the DNN utilizing this dataset.

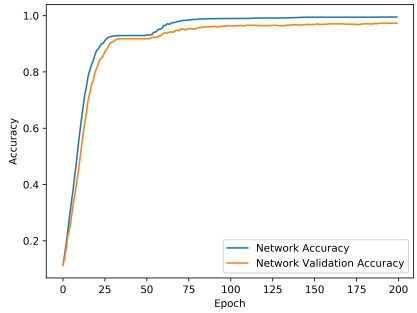
Changing the number of dense layers in a CNN had a large impact on the accuracy of the model. By removing a dense layer from our model we were able to increase accuracy and reduce overfitting according to accuracy 5.9a. However, the amount of information available at the output layer stayed the same, while the number of validation traces to reach a GE of 0 was relatively unaffected at the end of the training. It did, however, take less training epochs to reach this point 5.9c.

Since we only witness two of the layers moving through the information plane in the base models, meaning only two of the layers are specialized in transforming the input data to the classification decision, we expect that a network with only one dense hidden layer and output layer should be able to perform with similar performance as our base model. Our results show that even a one hidden layer network is able to extract the correct key byte with very similar results to the base dataset for both MLP and CNN models with only one dense hidden layer with width 16 seen in 5.8 and 5.9 respectively.

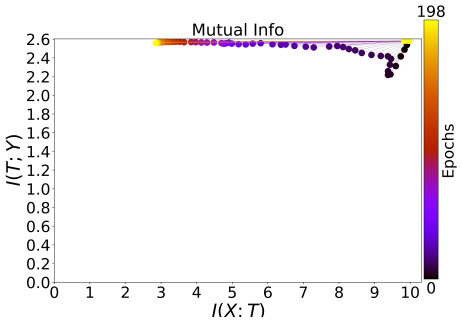
A sufficiently wide neural network with just a single hidden layer can approximate any (reasonable) function has been proposed in previous research [5]. Therefore, smaller networks should be able to model the same leakage function with a similar performance given our large training set. This hypothesis is further addressed in later datasets. We wish to utilize the data plane to determine how many layers are beneficial to the network's ability to gain information in I_Y quicker and more accurately or if a side channel leakage function is better model utilizing a wide hidden layer.

5.1.4. SENSITIVITY TO TRAINING SET SIZE

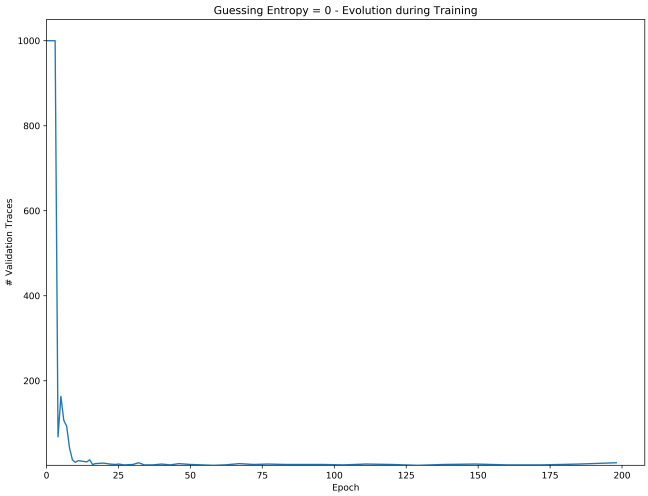
Using less training data has a large effect on our networks ability to learn and model the leakage function. In this situation, we halved the training dataset such that only 5000 traces were using in the training phase. Limiting the training data intensifies the necessity of an optimal network to model the leakage function. Utilizing less training information causes our mutual information to suffer as a sound generalization can not



(a) Model accuracy



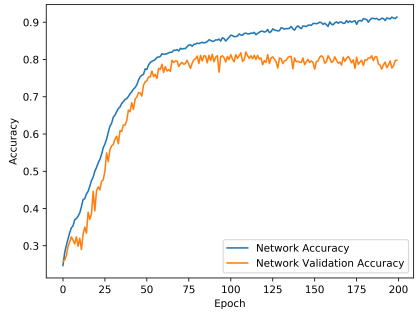
(b) Information plane



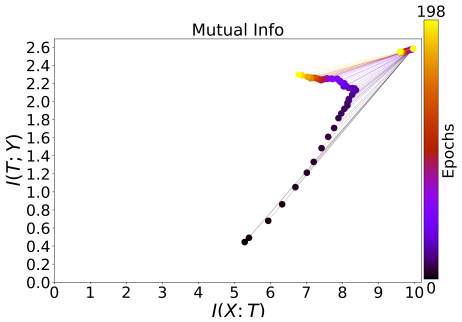
(c) Attack efficiency

Figure 5.8: MLP Deep Neural Network Utilizing a Single Hidden Dense Layer - 16

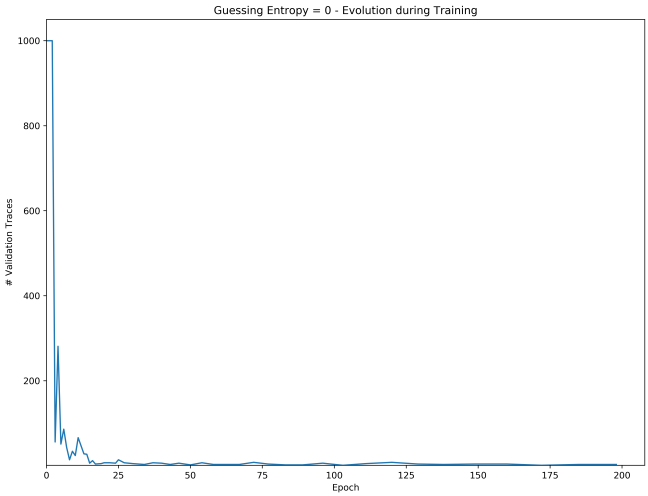
be made with little data about the target. This can be seen when mutual information is decreasing instead of increasing in the generalization phase5.10b. Additionally, mutual information I_Y never reaches as high values in the output layers as seen in the previous cases in both MLP and CNN architectures 5.10b 5.1c 5.11b 5.3c. We can see that comparable side channel attack metrics are reached latter in the training phase of the DNN. In the case of our base MLP architecture, the most successful attack is only reached after about 20 epochs compared to 10 utilizing the full 10000 training traces5.10c. Moreover, the base CNN architecture saw little differences in the side channel analysis metrics or even improved with the first successful attack after just 5 epochs. Although validation accuracy was not as high as in the initial experiments the side channel metrics did not



(a) Model accuracy



(b) Information plane



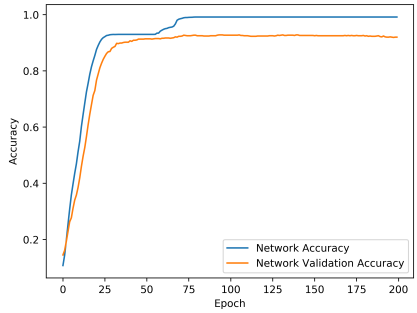
(c) Attack efficiency

Figure 5.9: CNN Deep Neural Network Utilizing a Single Hidden Dense Layer - 16

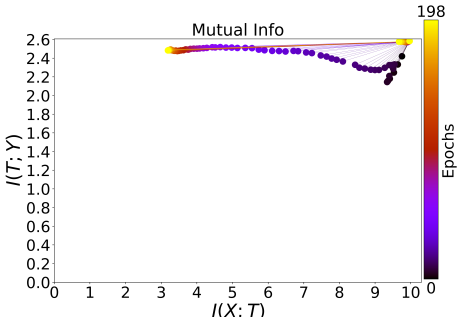
suffer. In the CNN model, the information plane was mostly unchanged and therefore the side channel metrics was also consistent.

5.1.5. SENSITIVITY TO NOISE

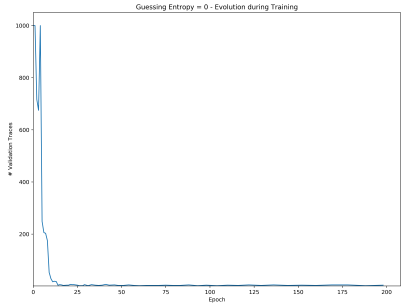
Adding noise to the training data had a large impact on our DNNs ability to generalize effectively 5.12a. Results with the same base architecture showed similar results to limited training data experiments where I_Y decreases in the later epochs due to overfitting. Since mutual information is calculated utilizing the validation set, the measure can show us where overfitting could be occurring. We can link this decrease in I_Y with overfitting when investigating the GE. Our results show when utilizing noise data, our models



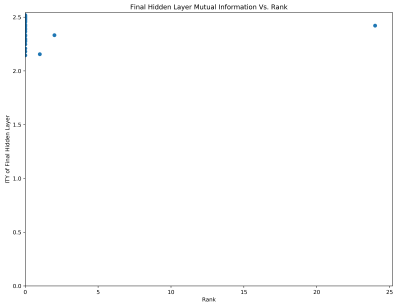
(a) Model Accuracy



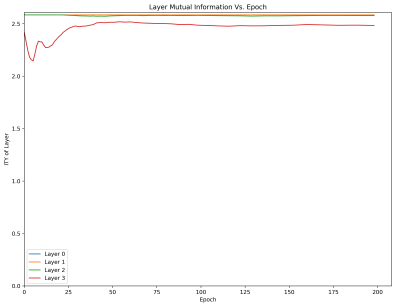
(b) Model Information Plane



(c) Attack Efficiency



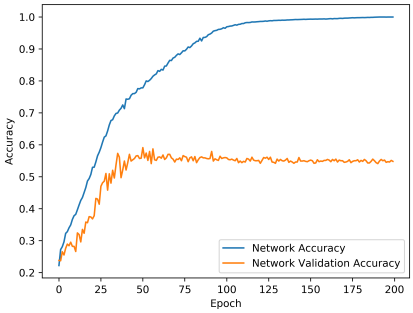
(d) Attack Success Progression



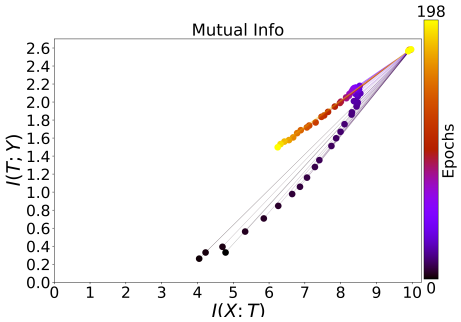
(e) Output mutual information across layers

Figure 5.10: Simulated Dataset Base MLP - trained with half the training data 5000 traces

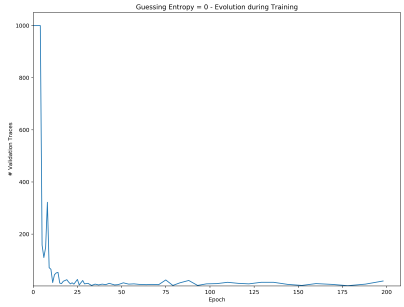
overfit much sooner in the training phase and accuracy suffers immensely. In classical machine learning, we utilize the difference between training error/accuracy and validation error/accuracy to be a good indicator of overfitting. Some DNN model training tools have been developed to stop training when overfitting is detected such as early stopping [26]. However, in a side channel analysis setting the moment overfitting is occurring



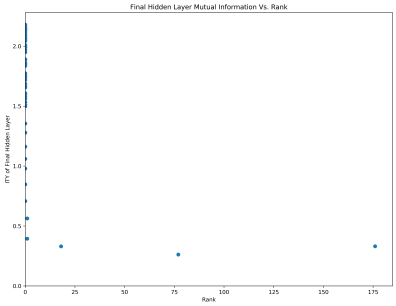
(a) Model Accuracy



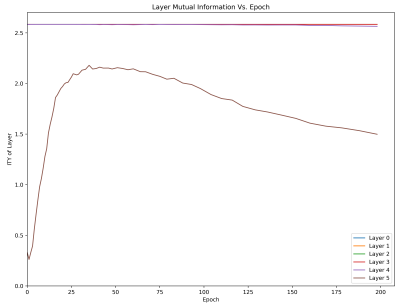
(b) Model Information Plane



(c) Attack Efficiency



(d) Attack Success Progression



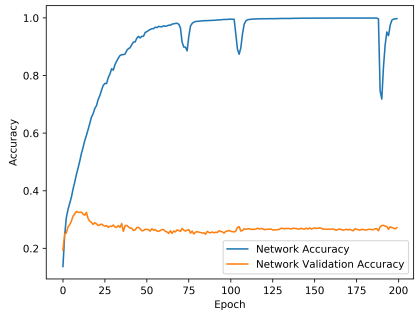
(e) Output mutual information across layers

Figure 5.11: Simulated Dataset Base CNN - trained with half the training data 5000 traces

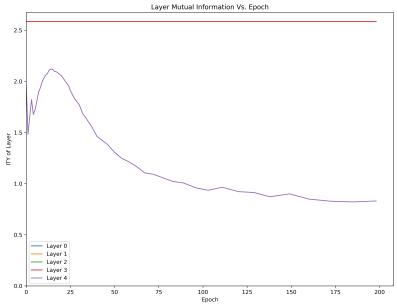
in the accuracy plot is too early to stop learning. We propose a better indicator is the information plane where we can decide to stop training when I_Y has been maximized. According to the guessing entropy (GE), the least amount of traces are needed to achieve a GE of 0 is just before I_Y peaks.

Presented results for the MLP and CNN with 3 hidden dense layers of 64, 32, 16 neu-

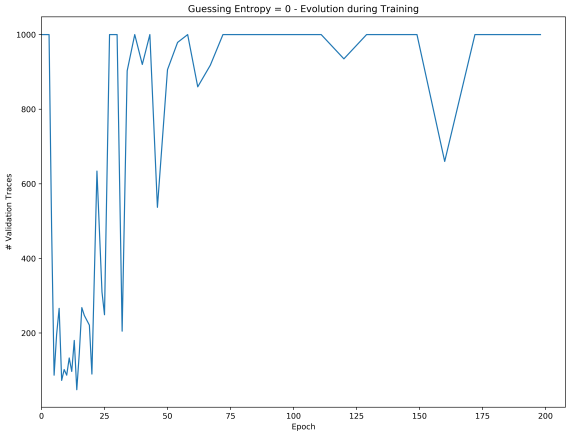
ron show how overfitting can occur in the later epochs of training. According to the accuracy plot 5.12a 5.13a, training, and validation accuracy begin to divert from each other around epoch 5 which implies that we are overfitting the training data from here on out, however mutual information in the final layer I_Y peaks around epoch 20 and 25 for the MLP and CNN respectively. GE, therefore, depicts the best attacks utilizing the least amount of traces happens just before this point.



(a) Noisy Model Accuracy



(b) Output mutual information across network layers

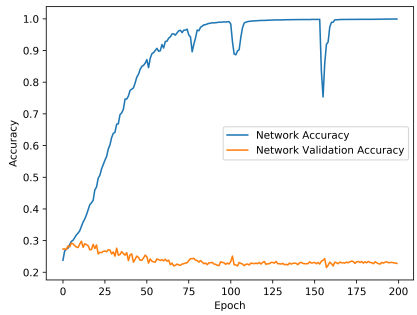


(c) Noisy data attack efficiency

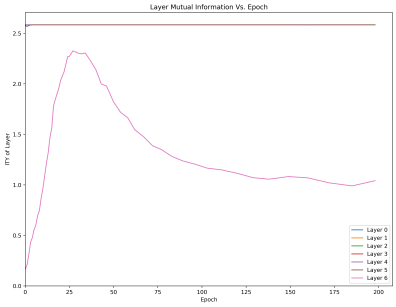
Figure 5.12: MLP Deep Neural Network Synthetic Noise added to training data - resulting in overfitting

5.1.6. GENERALIZATION VS. OVERFITTING

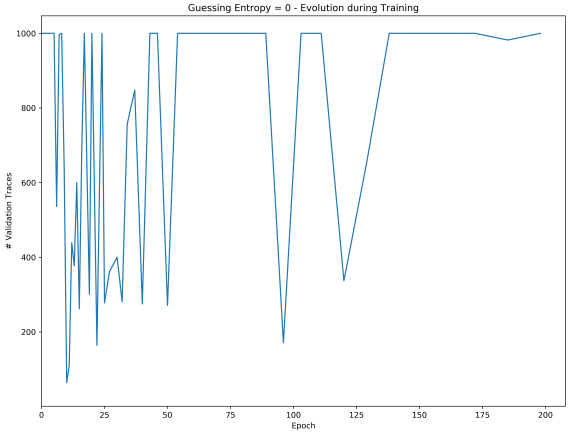
The models all generalized well with this dataset, overfitting was only seen when noise was added. More will be discussed in later sections with regards to models that do overfit in regards to side channel metrics. Since the leakage function in this dataset is not difficult, as the labels exist in the training data as a feature, this dataset is easily classified by a



(a) Noisy Model Accuracy



(b) Output mutual information across network layers



(c) Noisy data attack efficiency

Figure 5.13: CNN Deep Neural Network Synthetic Noise added to training data - resulting in overfitting

small neural network with high performance. This can be seen from our results and will be further explored with a more intricate dataset that can offer a higher difficulty to the neural network and resemble real measurements from physical devices. Furthermore, we can conclude that the deeper layers in the neural network are the layers that specialize in the side channel classification task whereas the shallow layers are more abstract. We do not yet understand if the shallow layers are necessary or beneficial in this dataset.

5.2. SMARTCARD AES RESULTS

We carry out further experiments utilizing a physical device to show that our findings hold in a measure trace set.

5.2.1. DEEP NEURAL NETWORK ARCHITECTURE

Utilizing this dataset we show that DNNs are able to effectively perform side channel attacks on secret keys from a physically measured device and that the mutual information framework is able to give some insights on the performance of the DNN's. We show that there is a connection between the output DNN layer's I_Y and the guessing entropy of the key byte under attack, and I_X and the leaking samples of our dataset.

MLP

This is a measured power trace set, the amount of leakage seems to be quite high with the MLP having high accuracy very quickly with many small DNN architectures. This dataset behaved similarly to the simulated dataset producing very similar results in terms of our side channel analysis metrics. The chosen base architecture of the MLP performed well from the onset given a large amount of training data, due to the number of parameter updates done per batch. Limiting the number of training traces required more epochs for the network to become optimized, achieving a validation accuracy above 90%. For a side channel analysis, this accuracy is more than enough to attack a secret key byte in a handful of traces.

The 3 hidden layers were easily able to model a leakage function precisely. Although our experiments show that even a shallow network is able to model the leakage function given enough training data. The architecture utilized was insignificant when enough data was provided to the network for training. However, the information plane usually showed that the output layer is the most significant when making predictions for this dataset.

CNN

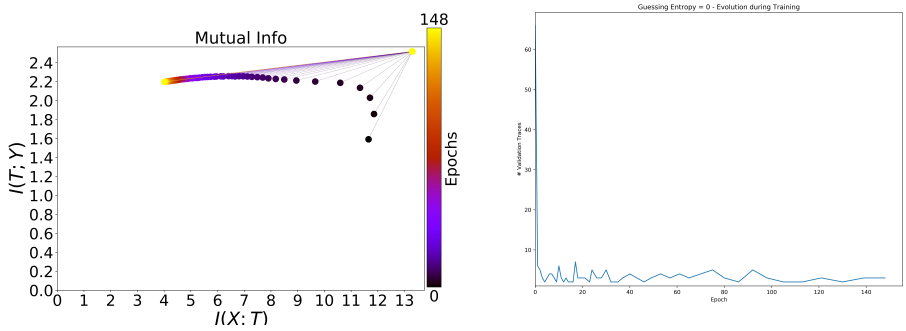
Since the leakage function was not correlated to a single sample in our power traces a CNN network functioned much better in the smartcard AES dataset. The use of a convolutional layer combined samples allowing the network to more easily identify the leaking samples. This can be seen from the input gradient correlation to the input gradients of the network. In general, the correlation is much higher for CNN networks compared the MLP networks using an identical dataset.

For this dataset out CNN quickly gained information and was able to successfully achieve a GE of 0 very early in training. Although the architecture utilized is not large it was very efficient at the side channel analysis task, even achieving classification accuracy high enough in the first epoch to correctly identify the correct key given the size of our validation set.

5.2.2. INFORMATION PLANE

The base MLP performed extremely well from the onset. Since the DNN parameters are updated every batch, even after the first epoch the classification accuracy was so high that the key was easily attacked utilizing less than 400 validation traces. The information started very high, nearly equal to the input layer information and did not decrease much throughout the training. We expect that the layers lose information due to random initialization however only the output layer is affected in our case. The DNN is so effective in this dataset that almost all the relevant information is preserved by the output layer

when training with 10,000 traces. A small amount of information is lost during the optimization of the network or diffusion phase, however, this does not affect our attack as more information than is needed is still available for an effective attack.



(a) Mutual Information utilizing base architecture

(b) Number of Validation Traces needed to Reach GE - 0

Figure 5.14: The MLP deep learning architecture utilized is [128,64,32] FC layers SmartCard AES MLP_{base}

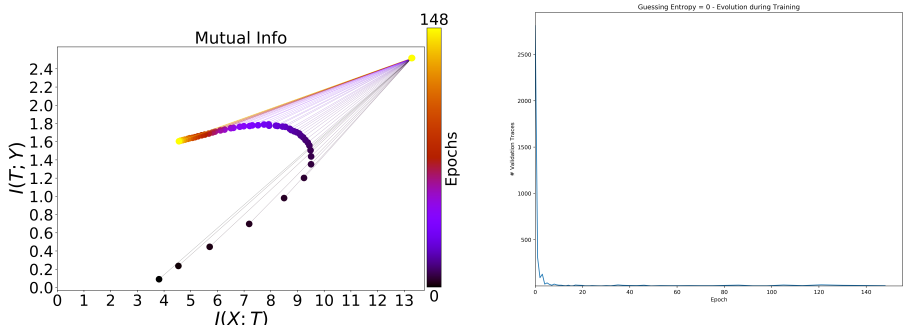
5

The base CNN model performed if not better than the MLP however it did take more epochs of training to reach the most efficient attack but was again successful from the onset. Again, the classification accuracy was high enough in the fist epoch to successfully attack the secret key byte after only the first epoch utilizing no more than 2500 traces. Information I_Y at this point is much lower than in the MLP. We can see that the output layer of the CNN is more heavily affected by the random initialization of the network is requires more epoch to more precisely achieve similar classification accuracy as the optimized MLP. The CNN architecture starts at a much lower level of I_Y than the MLP however quickly increases and peaks around epoch 30. The most efficient attack, utilizing the least amount of validation traces also occurs around this point. Fewer validation traces were utilized to carry out the attack on the secret key byte utilizing a CNN network rather than the proposed MLP.

5.2.3. IMPLICIT FEATURE SELECTION

Since the traces are very highly reprocessed the network is able to easily model a precise leakage function and identify the samples that express leakage. Since this implementation has no counter measures the first ordered leakage was utilized to compute the KKC and correlated to the input gradients of the network throughout training to identify how well the network is able to identify the same leaking samples. This in a way tells us how well the network was able to identify which input features can be used to make a precise prediction when modeling a first order leakage. We supply the DNN with traces that incorporate the entire first s-box operation for all secret key bytes but only a section of this is needed when attacking only one of the bytes.

Both models ability to identify the first order leaking samples were highly dependent on the amount of input data available for training. Generally, the more training data that was available the better the KKC to Input Gradient correlation was observed 5.16. During

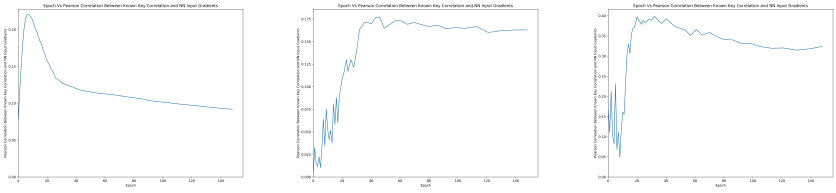


(a) Mutual Information utilizing base architecture

(b) Number of Validation Traces needed to Reach GE - 0

Figure 5.15: The CNN deep learning architecture utilized [16,32] Convolution layer filter size [128,64,32] FC layers SmartCard AES CNN_{base}

the diffusion phase, the networks generally decrease I_X as the network implicitly selects the points that are needed for the classification task and therefore reduces the amount on the information we are receiving from the trace set. For the base MLP, we notice that the KKC and input gradient correlation starts growing rapidly while I_X decreases in the output layer during the diffusion phase 5.17.



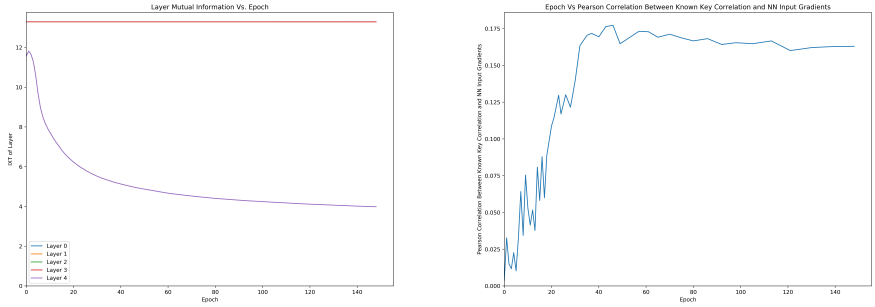
(a) Training Set Size 2000

(b) Training Set Size 10,000

(c) Training Set Size 50,000

Figure 5.16: Correlation Between Input Gradients and KKC utilizing the MLP base DNN

Similar to the MLP, the correlation for a CNN network was observed to be very inconsistent in the beginning as the network is still learning, however, did not necessarily increase during the diffusion phase. The correlation, however, was quite low for CNN networks as the leakage function is likely using different points for the classification. Although good results are achieved utilizing CNNs, the samples from the trace that are being utilized are highly different from the KKC. Therefore the correlation does not seem to be a good measure for identifying points of interest. The network may be modeling a higher order leakage model. Figure 5.18 shows that the correlation at the end of training is similar to the beginning of training after random initialization.



(a) Layer-wise Input Mutual Information I_X

(b) Correlation between Input Gradients and First order KKC

Figure 5.17: Base MLP implicit feature selection and Mutual Information

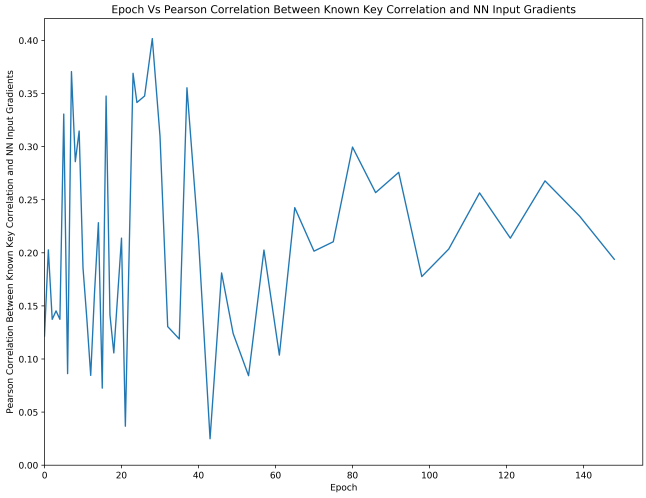
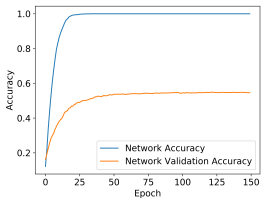


Figure 5.18: Correlation between Input Gradients and KKC of CNN base architecture utilizing 50,000 training traces

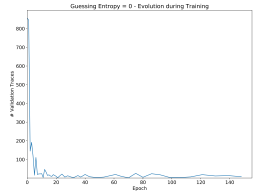
5.2.4. SENSITIVITY TO MODEL ARCHITECTURE

The base model has already been performing efficiently but the influence of the different layers are investigated in this section. The experimented MLP models are modified versions of the base models such that it contained less or more layers or modified layer widths. However, this dataset is easily attacked utilizing many simple architectures. This dataset was even successfully attacked using a shallow network. The information plane only shows the output layer having any change in the quantity of information available through training regardless of the number of fully connected layers 5.19,5.20,5.21. This,

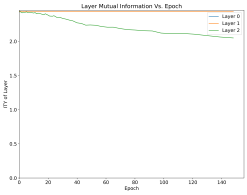
therefore, tells us that the network does not need to be deep to have an effective network that is able to quickly gain I_Y information as even a single layer is able to fulfill this task. Further experiments will be conducted utilizing the ASCAD dataset, as its countermeasures require deeper NNs for a successful evaluation.



(a) Model Accuracy - Overfitting occurring after the initial epochs

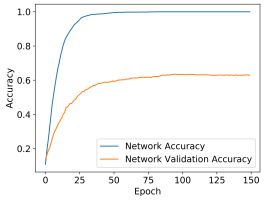


(b) Number of Validation traces needed to reach a GE = 0

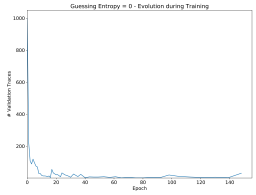


(c) Mutual Information I_Y by Layer

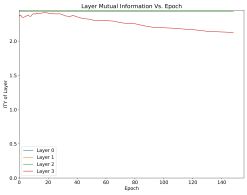
Figure 5.19: SmartCard Shallow Network [128] MLP model - 2,000 Training Traces



(a) Model Accuracy - Overfitting occurring after the initial epochs

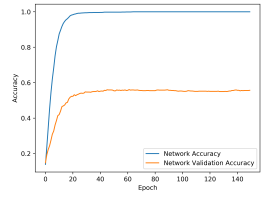


(b) Number of Validation traces needed to reach a GE = 0

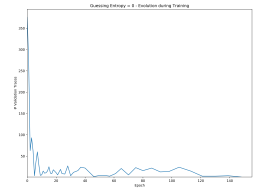


(c) Mutual Information I_Y by Layer

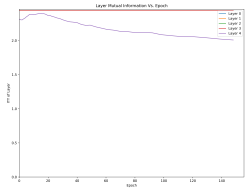
Figure 5.20: SmartCard 2 layer [64,32] MLP model - 2,000 Training Traces



(a) Model Accuracy - Overfitting occurring after the initial epochs



(b) Number of Validation traces needed to reach a GE = 0

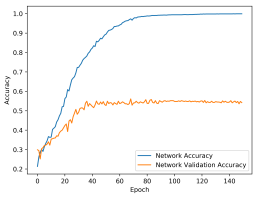


(c) Mutual Information I_Y by Layer

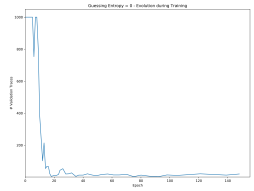
Figure 5.21: SmartCard base [128,64,32] MLP model - 2,000 Training Traces

Regarding the CNN networks, modifying the convolutional layers by either increasing the convolutional blocks or changing the number of pooling layers. This again had minimal effect on both the side channel metrics and the information plane; Accuracy was however affected. This shows that although accuracy is affected the side channel attack was independent to the accuracy of the classification model. Since I_Y remained

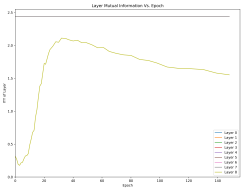
high, the GE also quickly reduced. A clear link still exists between the mutual information I_Y observed in the output layer and the optimization of the guessing entropy. The number of traces needed to achieve a GE of 0 is minimized just before the point in which I_Y is maximized.



(a) Model Accuracy - Overfitting occurring after epoch 10

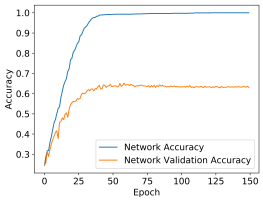


(b) Number of Validation traces needed to reach a GE = 0

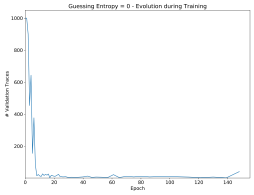


(c) Mutual Information I_Y by Layer

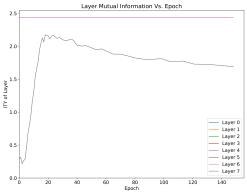
Figure 5.22: SmartCard base CNN model - 2,000 Training Traces



(a) Model Accuracy - Overfitting occurring after epoch 10



(b) Number of Validation traces needed to reach a GE = 0



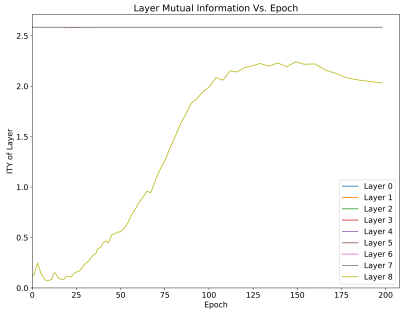
(c) Mutual Information I_Y by Layer

Figure 5.23: SmartCard base CNN model utilizing 1 pooling Layer - 2,000 Training Traces

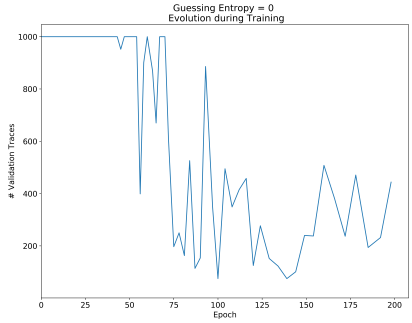
We experimented with convolutional networks that contained three convolutional layers, organized in different blocks depending on the pooling location in the architecture. We found that two convolutional blocks produced the best results when the first block contained two small (number of filters) convolutional layers and the second block containing one single large convolutional layer. The number of filters in each convolution in the block did make a difference when comparing the side channel attack as guessing entropy, reaching 0 with fewer validation traces when the first convolutional layers had only a few number of filters and the second layer had a large number of filters. The behavior of the network concerning information at the output layer or accuracy was not changed by the different number of filters. We think this is due to the input data was only transformed by the convolutional layers and different functions are modeled by the different networks that both produce the same amount of information at the output layer.

5.2.5. SENSITIVITY TO TRAINING SET SIZE

The amount of training data had the biggest influence on the success of our side channel attack on the secret key byte. The amount of label information the network was able to

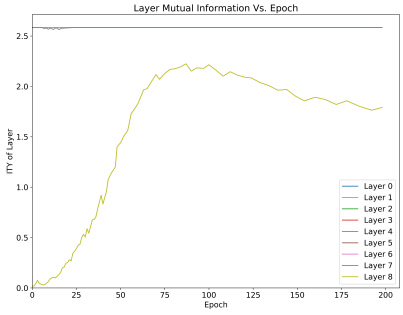


(a) Model Mutual Information with the output

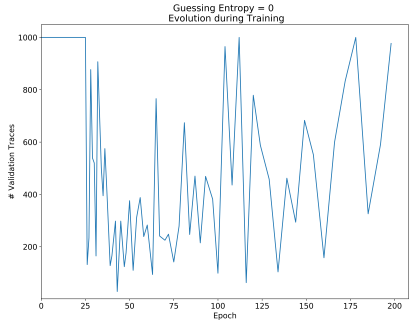


(b) Number of Validation traces needed to reach a GE = 0, Best - 75

Figure 5.24: SmartCard base CNN model utilizing small final convolutional layer [16,32][64]- 2,000 Training Traces



(a) Model Mutual Information with the output



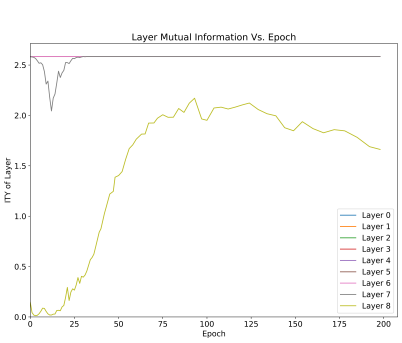
(b) Number of Validation traces needed to reach a GE = 0, Best - 29

Figure 5.25: SmartCard base CNN model utilizing medium final convolutional layer [16,32][128]- 2,000 Training Traces

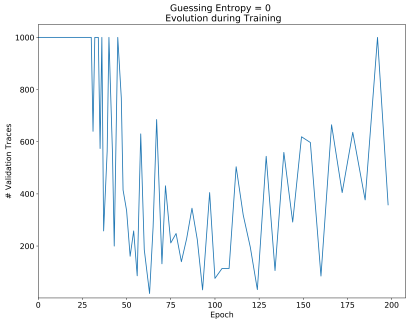
preserve was higher with additional training traces. The network was also able to identify the relevant data from the traces when providing more training data as stated in previously 5.16. This also allowed our network to more efficiently achieve 0 GE. The hidden layers also aided in speeding up the classification layer when a larger training dataset was utilized. The amount of mutual information the network as able to achieve in the output layer was higher in I_Y and lower in I_X 5.27 for both MLP and CNN base networks. This is a desirable behavior which allowed us to also achieve better side channel attacks 5.28.

5.2.6. SENSITIVITY TO NOISE

The noise had a large impact on both of our base models. Surprisingly our MLP model suffered from noise such that it decreased information I_Y as training undergone. Since

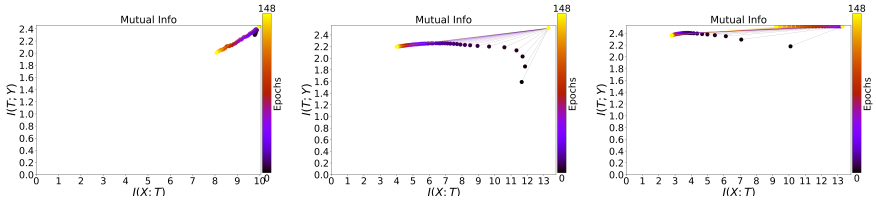


(a) Model Mutual Information with the output



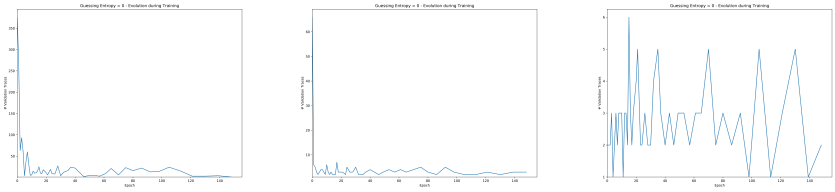
(b) Number of Validation traces needed to reach a GE = 0, Best - 18

Figure 5.26: SmartCard base CNN model utilizing large final convolutional layer [16,32][256]- 2,000 Training Traces



(a) 2,000 Training Traces (b) 10,000 Training Traces (c) 50,000 Training Traces

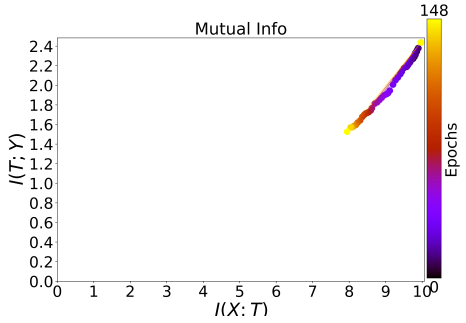
Figure 5.27: SmartCard base MLP model information plane utilizing varying amounts of training traces



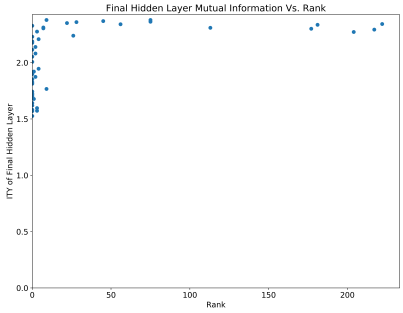
(a) 2,000 Training Traces (b) 10,000 Training Traces (c) 50,000 Training Traces

Figure 5.28: SmartCard base MLP model GE utilizing varying amounts of training traces

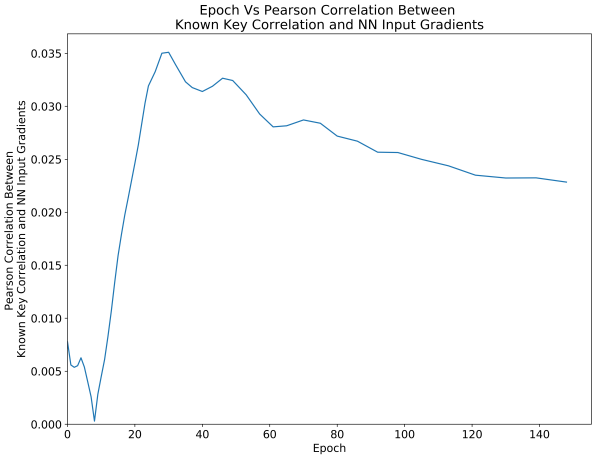
noise was only added to our training set our network learns from inaccurate data, therefore when measuring the information utilizing the clean validation dataset our model does not produce desirable outputs. The model spent the entire training decreasing I_X which resulted in the model seeking the leaking samples from our traces and therefore drastically increased our chances at a successful attack. We see that I_X decreased throughout training causing an increase in the Pearson correlation between the input gradients and the KKC 5.29. Our MLP network was able to still have consistent successful attacks on the key byte up to a maximum of 6 standard deviations of artificial noise.



(a) Mutual Information when dataset is affected by noise



(b) relationship between I_Y and rank is affected when noise is apparent

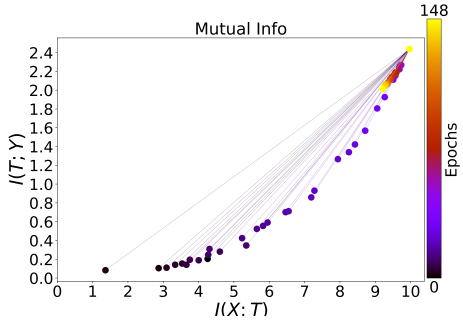


(c) Leakage sample detection

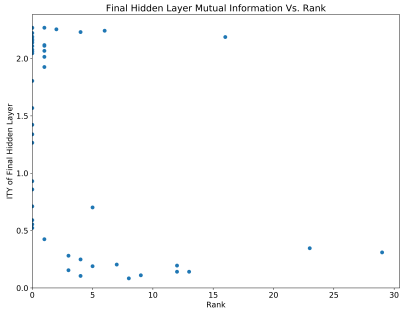
Figure 5.29: SmartCard base MLP model trained utilizing noisy 5,000 training traces - 8 STD

The information plane was a good indicator of an effective model however the MLP was still able to have successful results although the I_Y started decreasing. This seems to be contrary to previous findings, however, we contribute this to a different learning pattern which focused on feature selection rather than classification optimization.

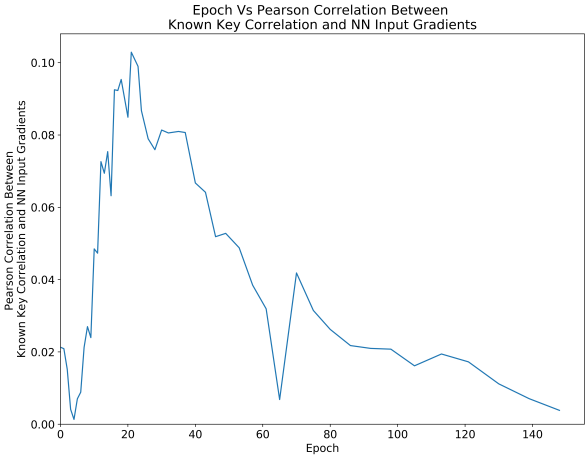
CNN networks were able to more effectively deal with the noise, while still being able to gain I_Y in the output layer. The convolutional layers were able to deal with the noise more effectively 5.30 which enabled our network to have desirable behavior when utilizing our validation data. The CNN network was resilient in overcoming up to a maximum of 10 standard deviations of artificial noise.



(a) Mutual Information when dataset is affected by noise



(b) relationship between I_Y and rank is affected when noise is apparent



(c) Leakage sample detection

Figure 5.30: SmartCard base CNN model trained utilizing noisy 5,000 training traces - 8 STD

5.2.7. GENERALIZATION VS. OVERFITTING

This dataset was able to achieve very good generalization and the mutual information I_Y in the final layer proved to be a good identifier as to when the best generalization was achieved by our trained model. The models achieved GE of 0 with the least amount of traces as I_Y was maximized. Generally the models then slightly over fit as I_Y started decreasing and the number of traces to reach GE of 0 also slightly increased. Overfitting was most apparent when noise is added causing the model to diverge from GE=0 after a maximized I_Y.

We show that this technique is beneficial to explain a connection between mutual information and GE. We now want to apply our technique to a publicly available dataset where side channel results have been published by other researchers. We show that this technique can be utilized with datasets that contain countermeasures and that our re-

sults are consistent regardless of the dataset.

5.3. ASCAD RESULTS

5.3.1. DEEP NEURAL NETWORK ARCHITECTURE

MLP

Due to the changes we have made to the leakage model, by using the HW model instead of the ID model, the base architecture behaved very differently than reported in the original ASCAD reported results [27]. The base model reported as MLP_{best} in their results was not able to achieve a high enough classification accuracy to extract the correct key byte with GE never below 50 after utilizing the entire validation set. We, therefore, attempted the side channel analysis with a much smaller DNN utilizing smaller and smaller networks until a better generalization was achieved with shallower and narrower architectures. Results never showed success in extracting the key in as little validation traces as reported in the ASCAD paper 5.31, but this may be closely derived as a result of changing the leakage function.

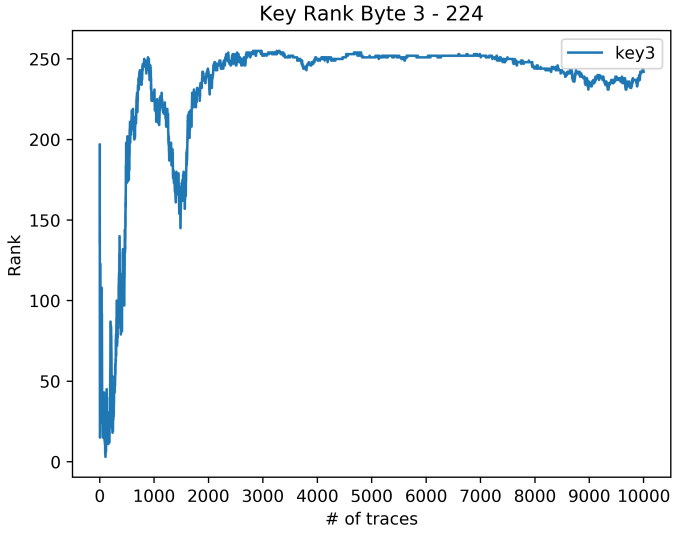


Figure 5.31: Guessing Entropy Computed at the end of Training - 500 epochs - Utilizing MLP_{best} DNN architecture [6 * 200]

The masked implementation required still a larger network than our previous two datasets to have a successful attack on the 3rd key byte. The best generalization was achieved utilizing 3 FC dense hidden layers of 32 neurons each. The decisions to continuously remove FC dense layers were based on the information plane. Results can be seen in figure 5.41. And further explained in the next section.

CNN

The CNN_{best} architecture model proposed in the original ASCAD results was also modified due to dimensionality issues caused by convolutional layer hyper parameters (which are held constant across this thesis). Additionally, the used leakage model was also modified to use the HW model as the labels for our neural network, therefore the softmax output layer only contained 9 neurons. A similar neural network with less convolutional layers was utilized in our study, again with successful attacks on the key byte but utilizing more attack traces then outlined in the original ASCAD study [27].

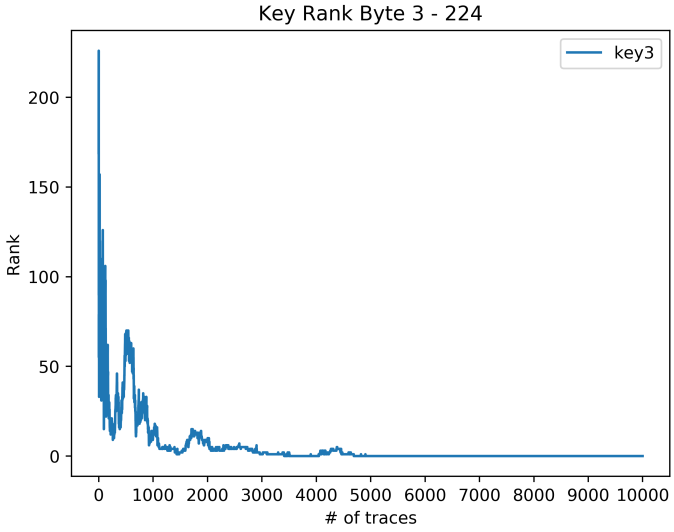


Figure 5.32: Guessing Entropy Computed at the end of Training - 500 epochs - Utilizing CNN DNN architecture

We utilized a VGG-like architecture [33] such that 1 or more convolutional layers followed by a single pooling were repeated before the FC dense layers of the CNN network. We modified the CNN_{best} architecture by increasing/reducing the number of filters and adding/removing a convolutional layer resulting in a feasible dimension for the neural network. Further experimentation showed very similar results between varying sizes of architectures, but vary similar layer behavior according to mutual information measures.

Smaller architectures like our base model showed much more inline results with the reported ASCAD results [27]. We further experimented with the effects of the convolutional block on the information plane, by modifying the number of blocks as well as the number of pooling layers.

5.3.2. INFORMATION PLANE

We firstly began experiments utilizing the MLP_{best} architecture however not all hidden layers were observed traversing the information plan to aid the acceleration of the output layer gaining useful information. Therefore we reduced the number of layers and

additionally experimented with layer widths. We noticed that as we reduce the size of the layers, and hereby reducing the number of tune-able parameters, the network was more effective and quicker at increasing I_Y. Incrementally reducing the number of layers, observing if all the hidden layers in the chosen architecture aided the classification process. Tishby et al.[32] state that additional layers help speed up the learning occurring at the deeper layers however we witnessed some side effects to this finding. Generally, in this side channel analysis cases, over fitting was frequent and quickly occurring in our training for both MLP and CNN models. Large MLP models were very slow at increasing relevant information I_Y in the output layer with smaller and shallower models having favorable side channel results more quickly and consistently.

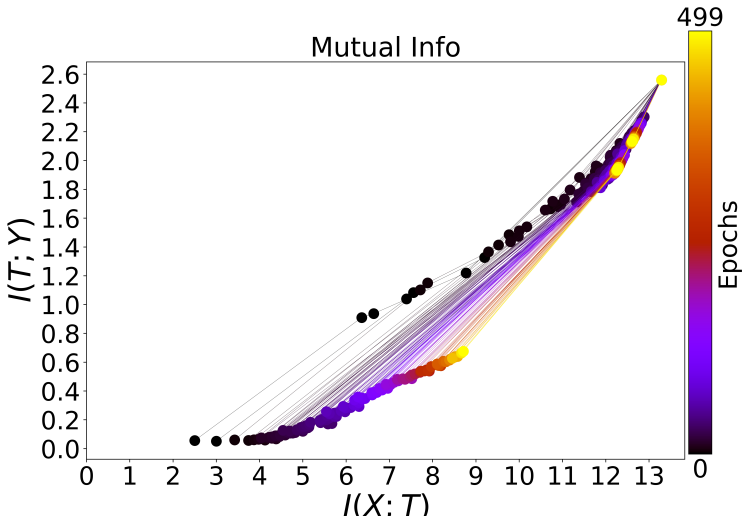


Figure 5.33: Information Plane - ASCAD MLP [32, 32, 32] - Mutual Information per Layer

We noticed that shallow layers regardless of the architecture did not lose any information or show any changes in the amount of mutual information in either of I_X or I_Y. This makes us believe these layers are non-specialized layers that only do some transformations of the input data that are necessary for the deeper layers to compute the modeled leakage function. Utilizing the mutual information measure we can not tell exactly what operations the shallow layers are performing however we know that the information available does not change.

For CNN architectures it is apparent that the number of convolutional layers has a large influence on the success of the DNN however these layers do not appear in the information plane. The dense layers have minimal influence and in almost all of our experiments, only the output layer is seen in the information plane. This enforces the claim that in CNN the important operations on the input data occur in the convolutional layers and only a small dense FC layer is needed in the end. This behavior is seen in figure 5.46 as the varying number of convolutional layers made only a scalar change in the amount of information at the output layer with little impact on the rest of the layers.

The architecture incorporating 1,2 or 3 convolutional layers with a constant number

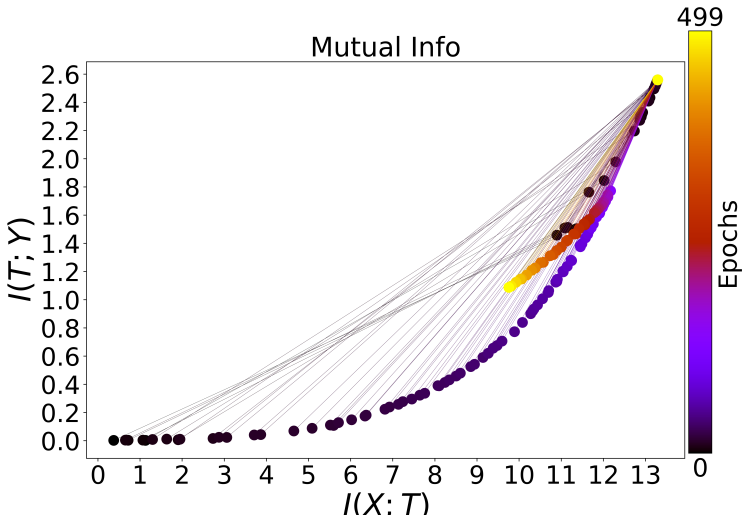
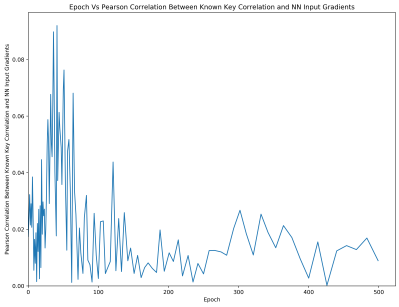


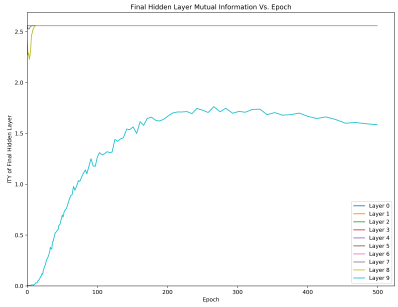
Figure 5.34: Information Plane - ASCAD CNN [16,32,64],[32, 32, 32] - Mutual Information per Layer

of fully connected layers behaved similarly in terms of the information plane measures however showed varying side channel performance metrics. I_Y was not by itself a good enough indicator as to at what part in our training the best guessing entropy would be achieved. We, therefore, incorporated a combination between I_Y and the correlation between the input layer gradients and the KKC. We interpret the KKC as the optimal leakage from the traces the more correlated the input gradients the better the network is utilizing the leaking points from our trace set. The KKC was computed with regards to the specific masked implementation, by Pearson correlation between the masked sbox-out and the round out mask. We, therefore, achieved the best attacked where I_Y is maximized and the correlation is also maximized. As these were not simultaneously maximized a best case must be chosen. Additionally, for this dataset, the KKC to Input Gradient correlation was not very strong but it can still be utilized as an indicator when comparing to other epochs in the same training sequence.

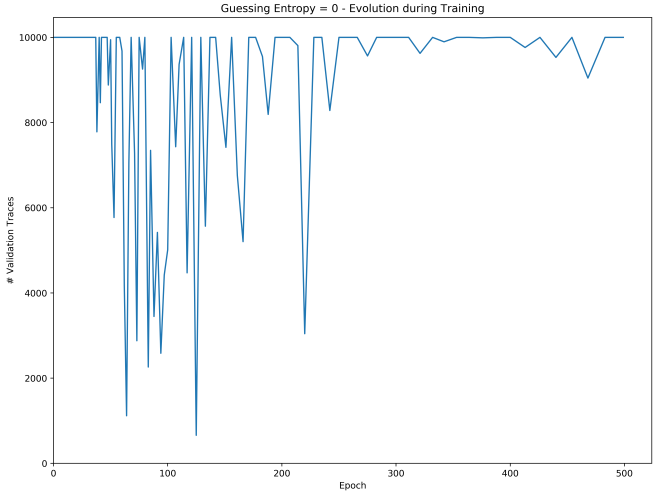
We can see from the KKC correlation 5.35b that the correlation is best in the first 20 to 40 epochs however the information regarding the output 5.35a is not high enough in our network to have a good classification and therefore at this point GE is not at its best. As the network increase I_Y in the output layer, the correlation I_X is increased 5.34. We interpret I_X in the output layer as how much of the input information is propagated through the entire network. The higher I_X the more samples the network is utilizing from the input data to make the classification. In side channel analysis we only need the leaking samples from the input to make an accurate classification and therefore we expect that the network implicitly chooses the sample it identifies as leaking. Once the network identifies these samples we expect that I_X is low. In the case of ASCAD, noise and the deficient training set causes the network to not only remove non-leaking samples but also the necessary samples. This is the reason we see the network in the later epochs reducing I_X but consequently also results in a reduction of I_Y. We can see this



(a) Neural Network's Ability to identify Leaking Samples



(b) Mutual Information between each layer and the labels - I_Y



(c) Guessing Entropy throughout Training

Figure 5.35: ASCAD CNN relationship between KKC and I_Y maximize GE

by looking at figure 5.36 which shows that even if I_Y stays high the GE starts to diverge from 0. This pattern was seen across multiple architectures utilizing this dataset.

The final layer's mutual information I_Y gave a very good indication of optimal guessing entropy. Generally, I_Y peaked early in training however KKC correlation seems to peak earlier in the training process. Guessing entropy reaches 0 utilizing the least amount of traces between the epochs that the KK correlation is maximized and the I_Y is maximized in the output layer.

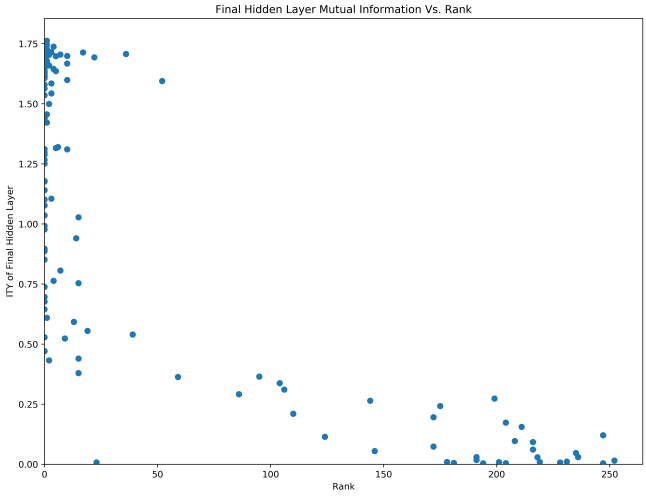


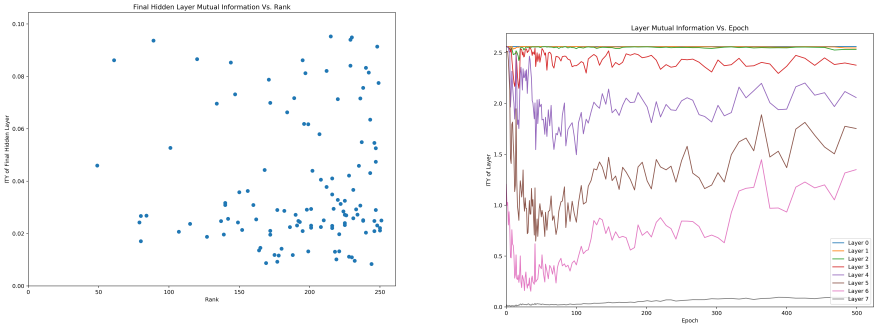
Figure 5.36: GE Rank with relation to ITY of output layer

5.3.3. SENSITIVITY TO MODEL ARCHITECTURE

MLP architecture choice seemed to highly influence side channel metrics. We started by training a large architecture that was able to increase I_Y only slightly in the 500 epochs. The MLP_{best} architecture, however, was not very successful at the classification of the HW values and therefore unable to correctly distinguish the correct key. Figure 5.37 shows only 4 of the hidden layers and the output layer in the architecture gaining any information after the initial random initialization and the output remaining very low. The other layers having little contribution as they just seem to forward the same amount of information to the more specialized deeper layers. There however exists an upward trend in figure 5.37b which can show that the neural network may have still been learning and required more epochs due to the large number of trainable parameters in such a large network.

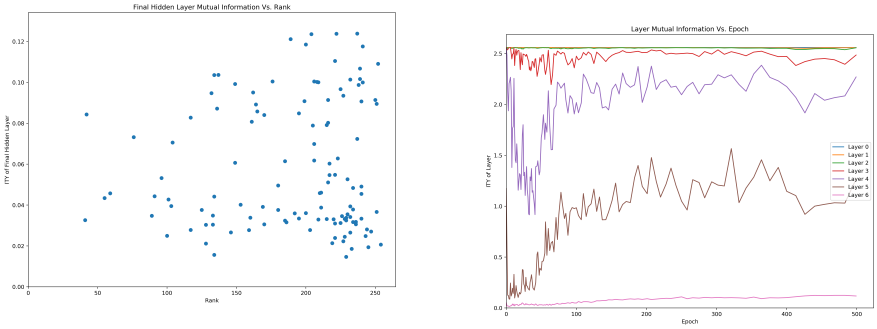
We then experimented with an architecture with less hidden layers. The architecture utilizing 5 FC dense layers seen in figure 5.38 showed much the same pattern as above however managed to increase I_Y in the output layer slightly more than the previous architecture. The number of layers becoming specialized for the classification task decreased again but still show a general upward trend, with potentially better results in a longer training period. Guessing entropy still showed unsuccessful side channel attacks on the key bytes. The increase in I_Y of the output layer, within the same number of epochs contradict finding by Tishby et al. [32]. We contribute this behavior to the increased number of tune-able parameters in the additional hidden layers. We, therefore, decided to keep reducing the number of layers hoping for a quicker increase in I_Y.

By reducing the number of hidden layers by one 5.39 we started to witness a lower GE near the peak of I_Y in the output layer, however the attack on the key byte was still



(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process (b) Mutual Information between each layer and the labels - I_Y

Figure 5.37: The MLP deep learning architecture utilized is [200,200,200,200,200,200] ASCAD MLP_{best}



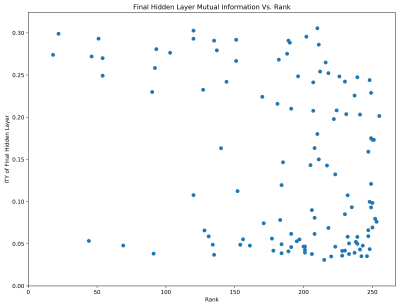
(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process (b) Mutual Information between each layer and the labels - I_Y

Figure 5.38: The MLP deep learning architecture utilized is [200,200,200,200,100]

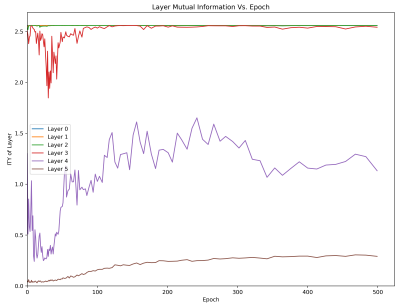
unsuccessful as GE never reached 0. We again see the same trend as the two previous architectures and therefore decide to again reduce the number of layers.

At three hidden layers, we finally begin to succeed in the side channel attack on the key byte 5.40a. The information I_Y in the output layer much greater than in the previous architectures 5.40b. However, information seems to stabilize in the later epochs, showing the network is not learning any additional information by further training. At this point, the reduced number of hidden layers contributing to the output layer's evolution is somewhat vague 5.40b as we have an increase in performance.

In an attempt to explain the reason why we only see two of the hidden layers in 5.40b contributing to the output layer's evolution in increasing I_Y, we reduce the number of tune-able parameters in the layers by reducing the width of the shallower layers. We can see from 5.41b that all layers are contributing to the output layer in the initial epochs. This network evolves much quicker due to the decreased number of parameters and

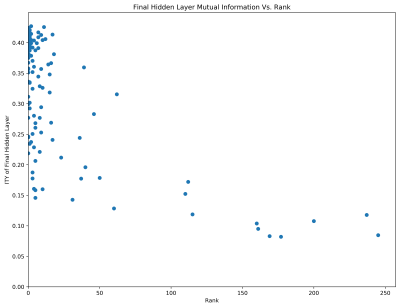


(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process

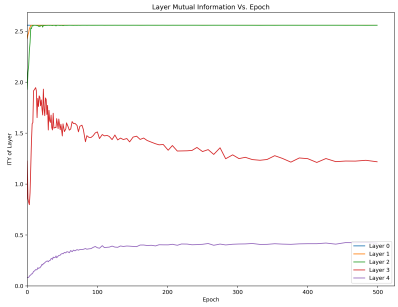


(b) Mutual Information between each layer and the labels - I_Y

Figure 5.39: The MLP deep learning architecture utilized is [200,200,200,100]



(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process



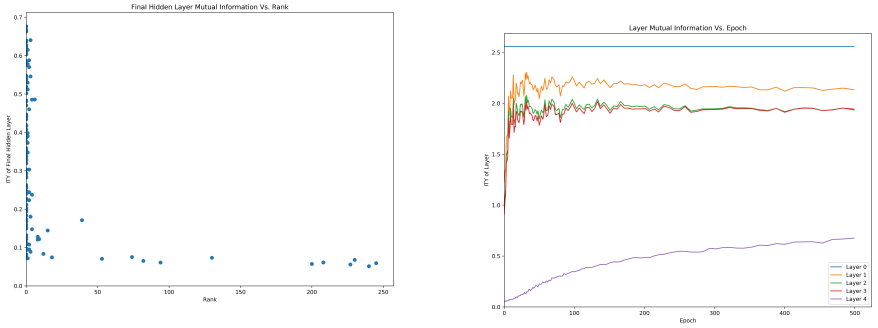
(b) Mutual Information between each layer and the labels - I_Y

Figure 5.40: The MLP deep learning architecture utilized is [200,100,32]

therefore we suspect the previous architecture would have behaved very similarly with a larger number of epochs. The output layer in 5.41 was able to gain the most information and thus also showed the best results in terms of side channel analysis. With GE typically reaching 0 at most epochs where the output layer's I_Y was above 0.2.

Generally, for MLPs we noticed that a higher number of hidden layers did not speed up the output layer's ability to gain relevant information about the output but rater increased the number of parameters which took much longer to update. Therefore, favorable side channel analysis results were achieved more quickly and successfully with smaller and shallower networks. Additionally, the smaller DNN was able to achieve a higher I_Y with fewer layers which contributed to a more efficient side channel attack.

For CNN we tested two different scenarios, changing the number of fully connected layers, and changing the number of convolutional blocks. We start by showing our results for the model with a constant number of filters for the convolutional blocks and a



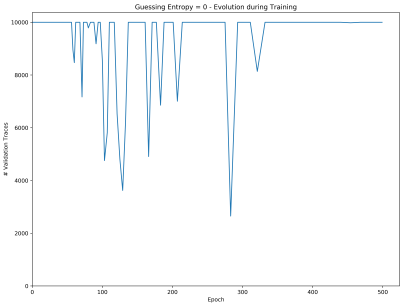
(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process (b) Mutual Information between each layer and the labels - I_Y

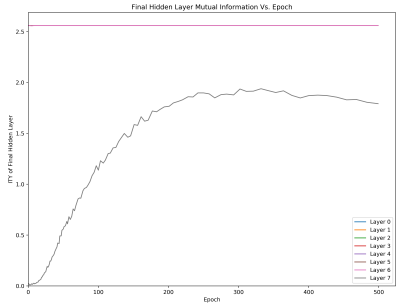
Figure 5.41: The MLP deep learning architecture utilized is [32,32,32]

varying number of dense layers. We show that the number of dense layers did not make a considerable difference to the attack success but decreased the number of epochs needed to perform an efficient attack. The networks utilized from 1 to 3 fully connected layers of 32 neurons. We see that the network with 1 or 2 layers performed considerably similar, GE reaching 0 around the same epoch in both cases with the number of validation traces that are needed to perform a successful attack slightly reduced in the larger network. We can see with an additional fully connected layer, the number of epochs needed to reach the most efficient attack is reduced and a further reduction in the number of traces is also seen. These results, however, can not be determined from the information plane as only the output plane is gaining information throughout the training. We found that the addition of fully connected layers increases the number of epochs where we have successful attacks, such that GE is 0 5.46. The network additionally reaches a successful attack earlier in training with more additional fully connected layers however it also start to overfit earlier in training 5.42 5.43 5.44 5.45.

Regarding the convolutional block, we experimented with two parameters, the number of blocks (convolution and pooling) and the number of pooling layers for 3 convolutions. We utilized a CNN where the fully connected blocks were kept constant at 3 layers of 32 neurons each at the end of the network, before the output layer. We found that the network with less pooling layers propagates much faster through the information plane but also over fits more in the later epochs.

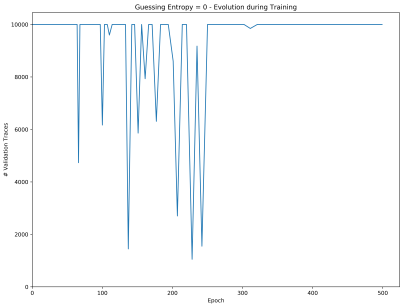
Varying the number of pooling for 3 convolutions resulted in varying levels of attack success concerning GE. The more pooling layers that were applied the slower the output layer propagated through the information plane. As the convolutional or pooling layers do not provide new information in the network they can not be seen in the information plane. We, therefore, can not explain how these layers work from an information measure. However, they do affect the output of the network since they do transform the input information. The most efficient side channel attack was achieved with 2 pooling layers; we can also see that the attack was successful in the largest number of epochs compared the other two scenarios. We see that the patters that I_Y in the output layer increases

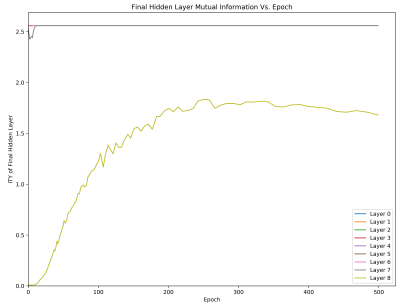




(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process (b) Mutual Information between each layer and the labels - I_Y

Figure 5.42: The CNN deep learning architecture utilized is single FC layer - 32





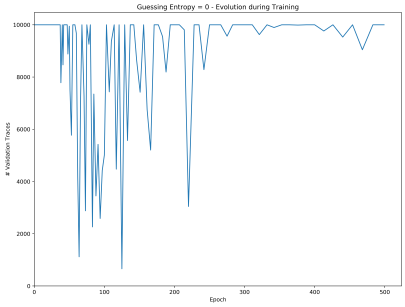
(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process (b) Mutual Information between each layer and the labels - I_Y

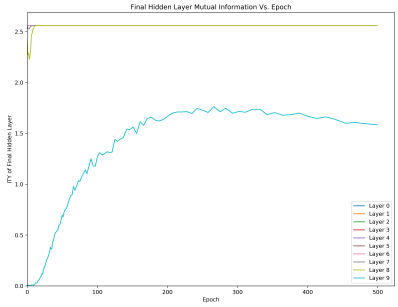
Figure 5.43: The CNN deep learning architecture utilized is two FC layers - 32

while the DNN is learning, reaching near 0 through learning, optimizing and diverging later in the training process around the top of the information measure. The less pooling layers we use the lower the amount of I_Y is needed to reach a GE close to 0. We can see in figure 5.47 that a good GE is reached with 1 pooling layer around an I_Y of 0.25, 2 pooling layers around an I_Y of 0.5 and 3 pooling layer around an I_Y of 1.0 as seen in figure 5.47. In all cases we see overfitting is occurring after the highest I_Y is achieved and causing the GE rank to increase and I_Y to start decreasing. We can see that pooling, although it reduces dimensionality, causes us to lose information and therefore requiring higher mutual information in the output layer to achieve a good guessing entropy.

Discuss that point where GE rank starts to grow again can be defined as the second stage of learning. Diffusion

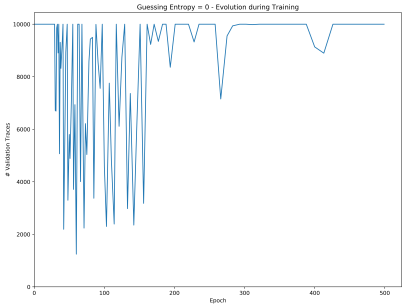
Altering the number of convolution blocks has an influence on the networks and the amount of overfitting we can see regarding the side channel analysis measures. In

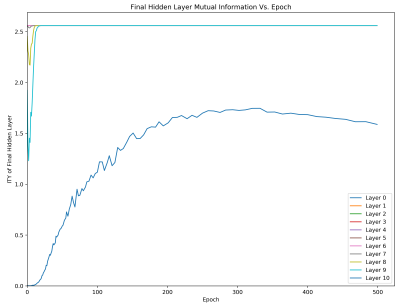




(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process (b) Mutual Information between each layer and the labels - I_Y

Figure 5.44: The CNN deep learning architecture utilized is three FC layers - 32

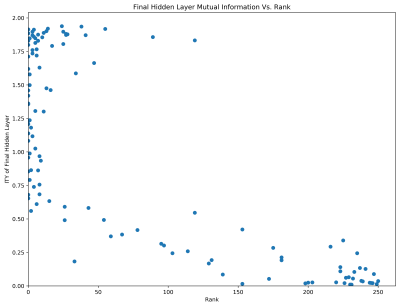




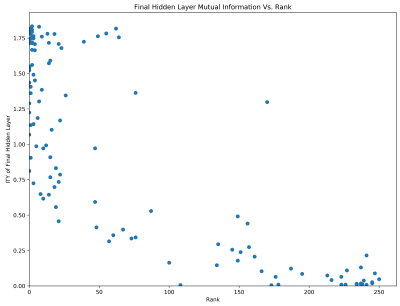
(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process (b) Mutual Information between each layer and the labels - I_Y

Figure 5.45: The CNN deep learning architecture utilized is four FC layers - 32

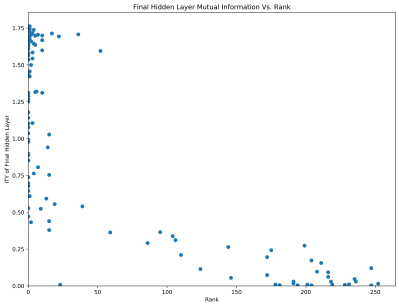
general, having less convolutional layers speeds up how fast the output layer is able to increase I_Y. This also puts our network in diffusion earlier which can cause overfitting to occur earlier in training. We found that the most efficient attack lies before the peak of I_Y in the output layer as stated in previous sections. However, we don't know how early this will occur in CNN networks for this dataset. Mutual information does not seem to give us much additional information as to how well the GE is performing or when we should expect the best GE when changing the number of convolutional blocks. We do however see that the CNN with more convolutions are achieving a GE of 0 during training more frequently and the best attack needs fewer validation traces. GE starts to rise when I_Y of the output layer starts to decrease in the diffusion phase, therefore this point can still be labeled as the start of overfitting however no considerable differences can be seen between the different architectures in regards to mutual information.



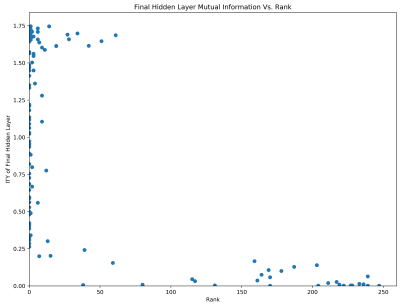
(a) ASCAD CNN Architecture utilizing 1 FC layers



(b) ASCAD CNN Architecture utilizing 2 FC layers



(c) ASCAD CNN Architecture utilizing 3 FC layers



(d) ASCAD CNN Architecture utilizing 4 FC layers

Figure 5.46: The CNN deep learning architecture mutual information against rank

5.3.4. OMITTED EXPERIMENTS

As ASCAD is a public dataset, and therefore the measurement parameters have been external. The training set is already limited in size. Experiments utilizing fewer data yielded little useful insight into the side channel analysis problem as attacks on the key bytes are unsuccessful.

Moreover, the noise was not synthetically added to this dataset as there already exists noise from original measurement errors. Additionally, the masked implementation acts as noise since more than one location in the trace must be identified to create an ideal leakage function given these input traces. Due to the apparent difficulties residing with this dataset, the noise was not considered.

Dataset utilizing desynchronization was not experimented with but could be considered for future work.

5.3.5. GENERALIZATION VS. OVERFITTING

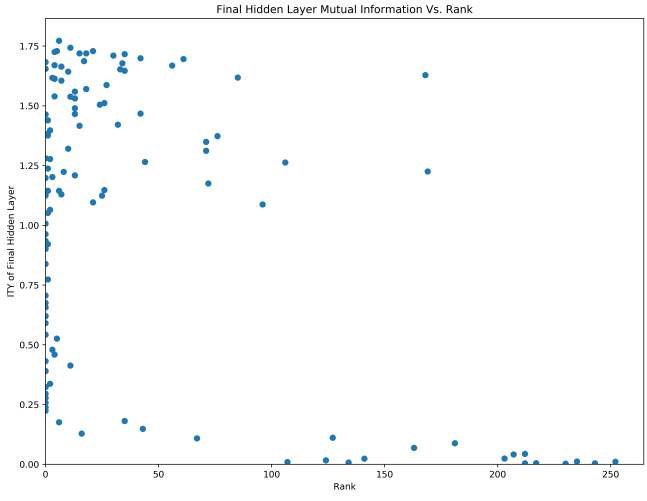
MLP achieved a good generalization in the optimized network architectures. According to traditional machine learning metrics such as accuracy seen in figure 5.48, we generally identify the difference between training and validation accuracy as overfitting. However,

according to side channel metrics such as guessing entropy overfitting is very minimal. Our results show that as long as mutual information I_Y in the final layer is increasing or is maximized, the guessing entropy stays very low. The best attacks on the key bytes utilizing the least amount of attack traces occurred just before I_Y in the output layer was maximized.

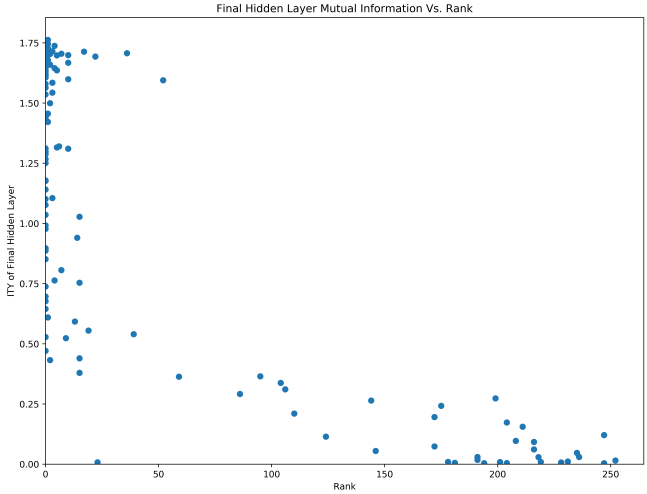
CNN models were often overfitting the data in terms of both classical ML metrics and side channel metrics. Information plane still shows to be a good indicator when the overfitting starts to influence the side channel task, however, is not an indicator of how well the architecture suits the classification task. A good stopping point for training is just the peak of mutual information I_Y in the output layer as the best GE is achieved before this point.

5.4. VALIDATION SUMMARY

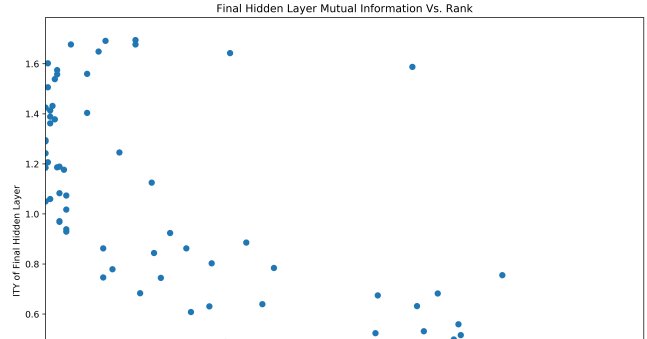
Utilizing the mutual information metric we were able to closely identify the epoch in which the neural network model began to over-fit out dataset in a side channel setting. We showed that the classical accuracy metric does not identify the most efficient attack since the model is not necessarily trained to its fullest potential. We showed a better identifier of this maximally trained model to be just before the point in which I_Y is maximized in the output layer. We also showed that deeper layer are more specialized layers since they affect the mutual information metric and therefore influence the decision, whereas, shallower layers are more general and only transform the data. Convolutional layers did not have an visible effect in the information plane however, it affected the deeper fully connected layers which showed variant levels of attack success at different mutual information metric levels. Architectural decisions can be made based on information plane findings for MLP networks, however, they do not offer a guaranteed beneficial result. In CNN networks the majority of the classification decision is made in the convolutional blocks where the number of dense layers do not have a large effect on the network's efficiency for a side channel setting.

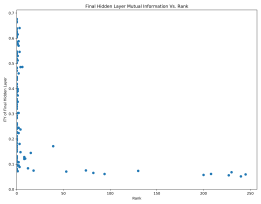


(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process for 1 Pooling Layer

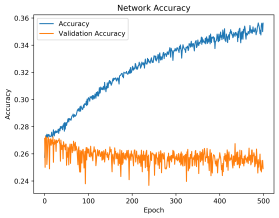


(b) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process for 2 Pooling Layer

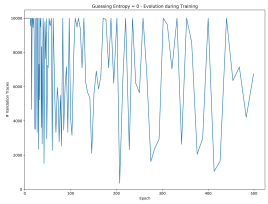




(a) Guessing Entropy Utilizing 10,000 validation traces as a function of I_Y - Computed Throughout the Training Process. Overfitting is not apparent given side channel analysis measures

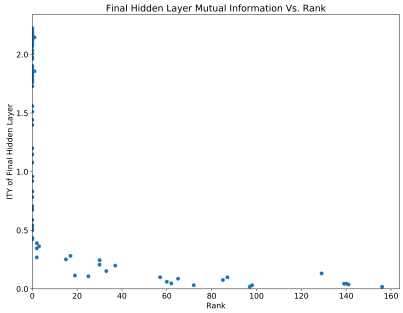


(b) Accuracy of ASCAD MLP model. Showing large Overfitting in terms of classical ML measures

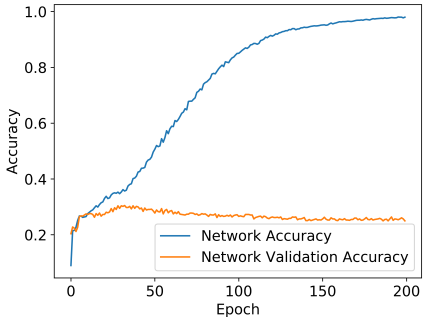


(c) Guessing Entropy computed throughout training using validation set

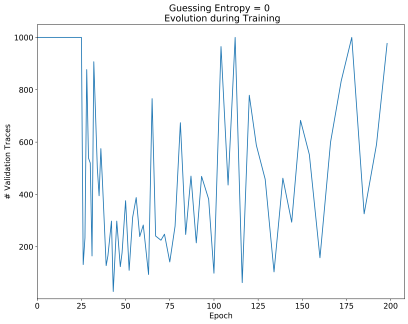
Figure 5.48: The MLP deep learning architecture utilized is [32,32,32]



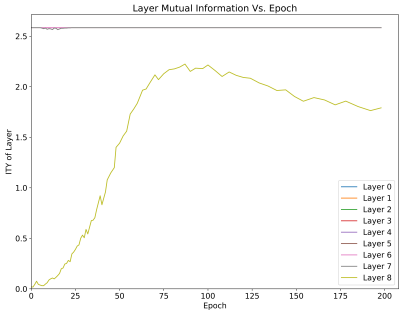
(a) Guessing Entropy Utilizing 1,000 validation traces as a function of I_Y - Computed Throughout the Training Process. Overfitting is not apparent given side channel analysis measures



(b) Accuracy of ASCAD CNN model. Showing large Overfitting in terms of classical ML measures starting epoch 10



(c) Guessing Entropy computed throughout training using validation set showing most successful attack before epoch 50



(d) Mutual information measure of DNN layers through training showing a maximization of ITY at epoch 75

Figure 5.49: The CNN deep learning architecture utilized is filters:[32,64],[128] fully connected layers:[32,16]

6

CONCLUSION

In this chapter, we outline the proposed findings concerning our research question in Section 1.2. We will propose further research based on our outcomes and summarize our contributions.

We show a information measure, mutual information, across each layer in a neural network as a way to represent each layer individually as proposed by Tishby [32] for a profiled side channel analysis task. As proposed in previous literature, classical machine learning metrics such as accuracy [24] have been proven to have a short coming when assessing deep learning models for profiled side channel analysis. As accuracy does not necessary correlate to the success of the attack on the secret key byte in a cryptographic algorithm, we explored how well mutual information correlates to the proposed success metric, guessing entropy (GE). Our finding show that mutual information between the output labels and the final layer's activations give a better distinguisher when a chosen neural network architecture has reached its most generalized state for a given dataset such that GE is minimized just before this point. The mutual information measure allows us to utilize the neural network model to its highest potential such that the security assessment for the device under attack is ameliorated for the chosen architecture.

We show that deep learning tools used to detect overfitting such as early stopping should not be utilized when performing a profiled side channel analysis utilizing deep neural networks, as they would generally stop the learning process earlier than desired. It would prevent the neural network's learning algorithm from developing the network parameters to perform the most optimized side channel attack. We proposed that the best time to halt training is when the mutual information I_Y in the output layer starts decreasing. Our experiments show that the most efficient (least number of traces) side channel results are always reached before this point.

For further research regarding overfitting in neural networks, the use of regularization such as L_1 or L_2 on the cost function can be utilized to see what effects if would have on the mutual information measured in our network. Additionally, batch normalization layers can also be utilized in our network to reduce overfitting. The effects of these layers have on the mutual information metric would also show if these layers are beneficial

to the side channel task and how certain layers are affected. Normalizing should not change the amount of information we have at a specific layer but it may change how a layer behaves and what function it models such that mutual information can be further maximized.

We explored if the mutual information plane can tell us how well suited our neural network architecture is for the side channel analysis task. The mutual information plane could tell us which layers were contributing to that output layer's predictions. We found that deeper layers are specialized for the classification task and shallow layers were more generalized layers. Given this we could not tell how an architecture could be changed to guarantee a benefit for the classification process. Furthermore, as we outline a clear relationship between I_Y and GE, the quantities of this two metrics where not comparable between two different architectures. We notices that different architectures reach a GE of 0 at different quantities of mutual information in the output layer.

Further research could entail a theoretical explanation between the GE and the mutual information between the output layer and the classification labels. A mathematical relationship may exist such that our findings are further validated and an explanation between that quantity of I_Y needed to reach GE of 0 is evident for a given architecture.

Since our finding generally look at the amount of mutual information in the output layer, we still do not know what happens when changing the DNN architecture such that the output layer in the network is a sigmoid layer instead of a softmax. Classification therefore yield the byte value, however the aggregation of many classification is still needed (GE). As we did not experiment with changing the output layer in our DNN architectures, we do not know how the mutual information correlates to GE when utilizing a different output layer type.

Our findings show that training on a large training trace set is essential for the deep neural network to maximize I_Y and minimize I_X. The larger the training set the higher I_Y was achieved during training of our DNNs. Our DNN was also able to better implicitly identify the leaking samples from our traces when the training set was large enough. Utilizing the correlation between the input gradients and the KKC we learned that neural networks become able to identify leaking samples as I_X decreases. To further validate our results experiments utilizing a random delay countermeasure could be utilized, however the correlation measure would not be of any use.

Looking back at our research objectives:

Q1: Can a mutual information measure tell us how much information is needed to correctly guess the correct byte of a cryptographic secret key?

A1: The metric that requires optimization in a side channel analysis study is the Guessing Entropy. Our thesis outlined that the mutual information between the output layer and the HW value labels varied in quantity between different architectures and prediction generalizations. The amount of mutual information in the output layer needed to on average predict the correct key across the entire trace set (aka. GE) varied from architecture to architecture. Therefore, no quantitative value can be declared. Generally, CNN networks required less mutual information to perform a successful attack than MLP network architectures.

Q2: Can a mutual information metric tell us how well generalized a neural network model is for the profiled side channel analysis problem?

A2: The most important finding in our thesis outlines that the most efficient attack occurs just before the mutual information in the output layer regarding the output is maximized, regardless of machine learning metrics such as accuracy. The most efficient attack is defined as the attack which utilizes the least amount of traces to reach a constant GE rank of 0. Additionally, we can better determine the point in which our model is generalized to its fullest and overfitting starts to occur. If the mutual information metric start to decrease during the training phase, we can determine that the network is overfitting.
Q3: Can a mutual information metric tell us how to choose an effective neural network architecture for the profiled side channel analysis problem? Or why do some architectures work better than others?
A3: We can not determine what architectures will perform better or worse by just observing the mutual information metrics of a trained network. We can, however, observe which layers are specialized to the classification task and which layers perform a generalization task on the input data. We can make some predictions as to how to modify some architectures, but no guarantees can be made about the success of the network's evaluation.
Q4: Can a mutual information metric prevent us from undesired effects of machine learning such as overfitting?
A4: Yes, we can identify when a network starts to over-fit the training data by observing the mutual information between the output layers and the dataset labels. We generally identify overfitting by observing the difference between training and validation accuracy. Side channel attacks require the optimization of the guessing entropy instead of the accuracy, and the two metrics are not consistently aligned. Therefore, for side channel analysis the point in which a model is overfitting begins can not be defined by the difference in the training and validation accuracy. We proposed this point be defined when a network's output layer mutual information metric starts decreasing.

6.0.1. CONTRIBUTIONS

- Overfitting is better detected by the mutual information measure in the output layer compared to classical measures such as the difference between training and validation accuracy.
- We can use the mutual information measure to predict when our DNN model is more likely to have the best attack according to GE for a given architecture. There is a clear connection between the GE and the mutual information I_Y measure in the output layer, which contribute to the most efficient attack.
- The mutual information plane can tell us which layers are specialized for the side channel classification task. This can give us an idea if the chosen architecture could be modified, but does not guarantee improved results.
- Training with a larger training set allows our model to have a larger increase in mutual information in the output layer, resulting in better a side channel analysis regardless of noise.

REFERENCES

[1] Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably secure masking of aes. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography*, pages 69–83, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[2] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 16–29, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[3] 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.

[4] Thomas M Cover and Joy A Thomas. Entropy, relative entropy and mutual information. *Elements of information theory*, 2:1–55, 1991.

[5] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[6] Laurie Genelle, Emmanuel Prouff, and Michaël Quisquater. Montgomery's trick and fast implementation of masked aes. In Abderrahmane Nitaj and David Pointcheval, editors, *Progress in Cryptology – AFRICACRYPT 2011*, pages 153–169, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[7] Richard Gilmore, Neil Hanley, and Maire 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 106–11. Institute of Electrical and Electronics Engineers (IEEE), 5 2015.

[8] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[9] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293, Oct 2011.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[11] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 148–179, 2019.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[15] 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.

[16] Zdenek Martinasek, Jan Hajny, and Lukas Malina. Optimization of power analysis using neural network. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications*, pages 94–107, Cham, 2014. Springer International Publishing.

[17] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. Cryptology ePrint Archive, Report 2018/1196, 2018. https://eprint.iacr.org/2018/1196.

[18] Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, Jan 1992.

[19] J. W. Miller, R. Goodman, and P. Smyth. On loss functions which minimize to conditional expected values and posterior probabilities. *IEEE Transactions on Information Theory*, 39(4):1404–1408, July 1993.

[20] Mee Young Park and Trevor Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):659–677, 2007.

[21] Christophe Pfeifer and Patrick Haddad. Spread: a new layer for profiled deep-learning side-channel attacks. *IACR Cryptology ePrint Archive*, 2018:880, 2018.

[22] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. Profiling side-channel analysis in the restricted attacker framework. *IACR Cryptology ePrint Archive*, 2019:168, 2019.

[23] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):209–237, Nov. 2018.

[24] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):1–29, November 2018.

[25] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In Anupam Chattopadhyay, Chester Rebeiro, and Yuval Yarom, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 157–176, Cham, 2018. Springer International Publishing.

[26] Lutz Prechelt. *Early Stopping - But When?*, pages 55–69. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[27] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cecile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ascad database. Cryptology ePrint Archive, Report 2018/053, 2018. https://eprint.iacr.org/2018/053.

[28] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2847–2854. JMLR. org, 2017.

[29] Vincent Rijmen and Joan Daemen. Advanced encryption standard. *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pages 19–22, 2001.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[31] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[32] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *CoRR*, abs/1703.00810, 2017.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[35] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, pages 443–461, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[36] NIST-FIPS Standard. Announcing the advanced encryption standard (aes). *Federal Information Processing Standards Publication*, 197(1-51):3–3, 2001.

[37] Matus Telgarsky. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*, 2016.

[38] David H Wolpert, William G Macready, et al. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[39] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. Cryptology ePrint Archive, Report 2019/803, 2019. https://eprint.iacr.org/2019/803.