**UNIVERSIDADE FEDERAL DE PELOTAS**
**Centro de Desenvolvimento Tecnológico**
**Programa de Pós-Graduação em Computação**


Tese

# UM FLUXO DE PODA PARA REDES NEURAIS DEDICADAS A ATAQUES A CANAIS LATERAIS BASEADOS EM APRENDIZADO PROFUNDO

**Rodrigo Nuevo Lellis**

Pelotas, 2023

**Rodrigo Nuevo Lellis**

# UM FLUXO DE PODA PARA REDES NEURAIS DEDICADAS A ATAQUES A CANAIS LATERAIS BASEADOS EM APRENDIZADO PROFUNDO

Tese apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Rafael Iankowski Soares
Coorientador: Prof. Dr. Guilherme Perin

Pelotas, 2023

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

L542f Lellis, Rodrigo Nuevo

Um fluxo de poda para redes neurais dedicadas a ataques a canais laterais baseados em aprendizado profundo / Rodrigo Nuevo Lellis ; Rafael Iankowski Soares, orientador ; Guilherme Perin, coorientador. — Pelotas, 2023.

153 f.

Tese (Doutorado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2023.

1. Ataques a canais laterais. 2. Aprendizado de Máquina. 3. Aprendizado profundo. 4. Poda. I. Soares, Rafael Iankowski, orient. II. Perin, Guilherme, coorient. III. Título.

CDD : 005

Elaborada por Simone Godinho Maisonave CRB: 10/1733

# RESUMO

LELLIS, Rodrigo Nuevo. **UM FLUXO DE PODA PARA REDES NEURAIS DEDICADAS A ATAQUES A CANAIS LATERAIS BASEADOS EM APRENDIZADO PROFUNDO**. Orientador: Rafael Iankowski Soares. Coorientador: Guilherme Perin. 2023. 153 f. Tese (Doutorado em Ciência da Computação) - Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2023.

Devido ao crescente número de serviços disponíveis por meio da Internet nas últimas décadas, é cada vez mais importante oferecer segurança às informações de usuário que trafegam por diferentes meios de comunicação. Para isso, sistemas computacionais se apoiam no uso de criptografia como modo de proteger as informações dos usuários. Atualmente, o uso de algoritmos de criptografia encontra-se embarcado em muitos dispositivos e aplicações, o que potencializa a ameaça de ataques que exploram características físicas do hardware que executa tais algoritmos. Esta classe de ataques é chamada de Ataques de Canal Lateral (do inglês Side Channel Attacks ou SCAs). Muitas proteções contra SCAs, chamadas contramedidas, são encontradas na literatura. Entretanto, são encontradas vulnerabilidades nessas contramedidas. Neste contexto, técnicas de Aprendizado profundo (do inglês Deep Learning ou DL) têm atraído interesse crescente por constituírem-se como recursos eficientes e em constante evolução. No entanto, o custo computacional de aplicar DL no cenário de SCA é alto. Estudos relatam experimentos com duração de semanas de uso de infraestrutura computacional. Este trabalho propõe reduzir o esforço computacional de SCAs baseados no uso de redes neurais pela redução do tamanho destas via técnicas de poda. Além disso, otimiza-se aqui o esforço computacional do processo de redução de redes. Resultados experimentais demonstram reduções entre 40 a 50% na quantidade de parâmetros de redes, bem como reduções de até 57.17% no tempo de treinamento. Redes reduzidas conseguem realizar ataques utilizando menos traços que as respectivas redes originais, em todos os casos. Adicionalmente, as redes reduzidas são treináveis por menos épocas que as respectivas redes originais. Com isto reduz-se ainda mais o tempo para realizar ataques. Esta Tese demonstra o potencial de aumento do nível de ameaça representado por SCAs baseados em DL.

Palavras-chave: Ataques a Canais Laterais. Aprendizado de Máquina. Aprendizado Profundo. Poda.

# ABSTRACT

LELLIS, Rodrigo Nuevo. **A PRUNING FLOW FOR NEURAL NETWORKS DEDICATED TO SIDE-CHANNELS ATTACKS BASED ON DEEP LEARNING**. Advisor: Rafael Iankowski Soares. Coadvisor: Guilherme Perin. 2023. 153 f. Thesis (Doctorate in Computer Science) - Technology Development Center, Federal University of Pelotas, Federal University of Pelotas, Pelotas, 2023.

Due to the increasing number of services available over the Internet in recent decades, it is becoming increasingly important to provide security for user information transmitted through various communication channels. To achieve this, computational systems rely on the use of encryption as a means to protect user information. Currently, encryption algorithms are embedded in many devices and applications, which enhances the threat of attacks that exploit the physical characteristics of the hardware executing these algorithms. This class of attack is called Side Channel Attacks (SCAs). Many protections against SCAs, referred to as countermeasures, are found in the literature. However, vulnerabilities are discovered in these countermeasures. In this context, Deep Learning (DL) techniques have attracted increasing interest as they are efficient and continuously evolving resources. Nevertheless, the computational cost of applying DL in the SCA scenario is high. Studies report experiments lasting weeks using computational infrastructure. This work aims to reduce the computational effort of SCA based on neural networks by reducing their size through pruning techniques. Additionally, the computational effort of the network reduction process is optimized. Experimental results demonstrate reductions of 40 to 50% in the number of network parameters, as well as reductions of up to 57.17% in training time. Reduced networks can perform attacks using fewer traces than their respective original networks in all cases. Furthermore, reduced networks require fewer training epochs than their original networks, reducing the time needed to carry out attacks. This thesis demonstrates the potential for an increased threat level posed by DL-based SCAs.

Keywords: Side Channel Attacks. Machine Leaning. Deep Learning. Pruning.

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AES | *Advanced Encryption Standard* |
| CMOS | *Complementary Metal–Oxide–Semiconductor* |
| CNN | *Convolutional Neural Network* |
| CPA | *Correlation Power Analysis* |
| DA | *Data Augmentation* |
| DDLA | *Differential Attack Deep Learning Analysis* |
| DES | *Data Encryption Standard* |
| DK | *Domain Knowledge* |
| DL | *Deep Learning* |
| DL-PA | *Deep Learning based Power Analysis* |
| DL-SCA | *Deep Learning based Side Channel Attacks* |
| DNN | *Deep Neural Network* |
| DPA | *Differential Power Analysis* |
| DT | *Decidion Tree* |
| DVFS | *Dynamic Voltage Frequency Scaling States* |
| EM | *Electromagnectic* |
| FDF | *Frequency Domain Features* |
| FFT | *Fast Fourier Transform* |
| FL-PA | *Frequency and Learning based Power Analysis* |
| GE | *Guess Entropy* |
| HD | *Hamming Distance* |
| HW | *Hamming Weight* |
| ICA | *Independent Component Analysis* |
| KGE | *Key Guess Entropy* |
| kNN | *k-Nearest Neighbor* |
| LDA | *Linear Discriminant Analysis* |

LPN *Learnin Parity with Noise*
LSTM *Long Short Term Memory*
LTH *Lotery Ticket hypotesis*
ML *Machine Learning*
MLP *Multi-Layer Perceptron*
NB *Naive Bayes*
NN *Neural Networks*
OTA *Online Template Attacks*
PCA *Principal Component Analysis*
POI *Points of Interest*
RC *Random Clock*
RDI *Random Delay Insertion*
RNN *Recurrent Neural Network*
RSM *Rotating SBoxes Masking*
SA *Sensitivity Analysis*
SCA *Side Channel Attacks*
SGD *Stochastic Gradient Descent*
SPA *Simple Power Analysis*
SSL *Semi-Supervised Learning*
SVM *Support Vector Machine*
TA *Template Attack*

# SUMÁRIO

# 1 INTRODUÇÃO

Desde as últimas décadas tem-se intensificado o uso de sistemas computacionais interconectados pela rede mundial de comunicação, a Internet, onde cresce também o número de usuários mal-intencionados dedicados a espionagem eletrônica, fraudes e diversas outras práticas para obter vantagens ou acesso a serviços e dados sigilosos, de modo que a segurança se torna imprescindível (FRIEDEL; HOLZER; SARKANI, 2020). Como exemplos de sistemas computacionais do nosso cotidiano é possível destacar o *e-commerce*, serviços bancários, reservas de passagens e mais recentemente, com a possibilidade de objetos trocarem informações entre si com o surgimento da Internet das Coisas, novos serviços tem surgido como casas, edifícios e até mesmo cidades inteligentes (KIRIMTAT et al., 2020), de modo que a informação deve ser protegida por meio de protocolos especiais e, sem dúvida, o uso obrigatório da criptografia para ocultar as informações. Os algoritmos criptográficos se apoiam em recursos matemáticos e lógicos para alterar uma mensagem a ser transmitida, também conhecida como texto claro, que se transforma em uma mensagem cifrada ou texto cifrado após computada pelo algoritmo criptográfico. Esta computação fica condicionada ao uso de uma chave criptográfica de modo que somente possa ser interpretada pelo transmissor e receptor que tiverem conhecimento da chave criptográfica (QADIR; VAROL, 2019).

Os avanços da microeletrônica permitiram implementações em *hardware* dos algoritmos criptográficos (CHOI et al., 2020). Por outro lado, Kocher (1996) mostra ser possível relacionar dados computados em sistemas computacionais mesmo fazendo uso de criptografia com propriedades físicas tais como por exemplo, o tempo de execução (KOCHER, 1996), consumo de energia (KOCHER; JAFFE; JUN, 1999), emissão eletromagnética (AGRAWAL et al., 2002), entre outros (GENKIN; SHAMIR; TROMER, 2017). Os ataques desse tipo são chamados de ataques a canais laterais ou ocultos (do inglês, *Side Channel Attacks* - SCAs). Neste contexto, existem na literatura diferentes tipos de ataques: ataques de indução de falhas (FUHR et al., 2013), Ataques Simples (do inglês, *Simple Power Analysis* – SPA) e Diferencial por Análise do Consumo (do inglês, *Differential Power Analysis* – DPA) (KOCHER; JAFFE; JUN, 1999)

e Análise da radiação Eletromagnética (em inglês *Simple ElectroMagnetic Analysis* – SEMA ou Differential Electromagnetic Analysis - DEMA) (AGRAWAL et al., 2002).

DPA/DEMA são bastante populares por serem não-invasivos, não deixando vestígios e eliminando a possibilidade de comprometimento do funcionamento do sistema criptográfico após a realização do ataque. Além disso, o *setup* para realização dos ataques possui um custo relativamente baixo, e o atacante não precisa ter um conhecimento detalhado sobre o algoritmo criptográfico, tampouco sobre sua implementação (MANGARD; OSWALD; POPP, 2007). Entretanto, esses ataques exigem uma grande quantidade de informação, na forma de traços, para que seja possível decifrar a chave criptográfica. Mais ainda, os traços devem estar alinhados no domínio do tempo para obter sucesso. Essa fragilidade temporal, tornou possível a criação de proteções, chamadas contramedidas que podem atuar de diversas formas, como por exemplo, introduzindo ruído nas medições do consumo como em (BOEY et al., 2010), (CHOU; LU, 2019), (DAS et al., 2020), (KIZHVATOV, 2009), (LIU; CHANG; LEE, 2010) e (LIU; CHANG; LEE, 2012), impedindo análises de correlação mascarando os dados processados (TRICHINA; DE SETA; GERMANI, 2003), (GOLIC, 2007) e (CORON; GOUBIN, 2000), ou então buscando obter um consumo uniforme para qualquer sequência de dados de entrada (LIM et al., 2017). Também foram propostas combinações de contramedidas, como em (WANG et al., 2016) e (SOARES, 2010). Entretanto, etapas de pré-processamento podem ser incorporadas ao fluxo de ataques com o intuito de neutralizar tais contramedidas realinhando os traços, como pode ser observado em (LELLIS; SOARES, 2017), (LODER, 2014), (LE et al., 2007) e (NAGASHIMA et al., 2007).

Os ataques DPA/DEMA são realizados a partir de um modelo de consumo, geralmente baseado na Distância *Hamming* (*Hamming Distance* – HD) ou no Peso *Hamming* (*Hamming Weight* – HW), com o qual as amostras dos traços do consumo são analisadas estatisticamente. Esta análise ocorre por meio da diferença das médias. De maneira análoga, o ataque conhecido como *Correlation Power Analysis* – CPA (BRIER; CLAVIER; OLIVIER, 2004), do mesmo modo pode usar ambas as métricas matemáticas HD e HW para definir modelos hipotéticos de consumo e calcular a correlação existente com os traços de consumo medidos usando a Correlação de Pearson. Já os chamados *Template Attacks* - TA propostos por (CHARI; RAO; ROHATGI, 2002), utilizam um modelo Gaussiano de ruído para definir e registrar os modelos de traços DPAs relativos a um conjunto pré-definido de operações. Assim, esse tipo de ataque compara os traços de consumo medidos ao modelo através de parâmetros como a média e covariância.

Em função da evolução da área de inteligência artificial, técnicas de Aprendizado de Máquina (do inglês, *Machine Learning* – ML) e de Aprendizagem Profunda (do inglês, *Deep Learning* – DL) foram recentemente empregadas para criar modelos a partir

de um dispositivo criptográfico específico, para que este possa ser utilizado em dispositivos com características semelhantes (*Profiling Attacks*), abordados em diversos trabalhos encontrados na literatura (HETTWER; GEHRER; GÜNEYSU, 2020), (YANG et al., 2012) e (LERMAN et al., 2013). Entretanto, estes trabalhos têm como estudo de caso dispositivos desprotegidos. Contudo, atualmente existem várias propostas de contramedidas disponíveis na literatura, das quais muitas estão baseadas no desalinhamento temporal dos traços do consumo (HETTWER et al., 2020), (SINGH et al., 2020) e (CHONG et al., 2021). É, portanto, interessante verificar o desempenho de tais técnicas aplicadas a dispositivos protegidos, com o intuito de avaliar vulnerabilidades dos mesmos.

Neste contexto, são encontrados na literatura trabalhos que realizam ataques SCA baseados em ML e DL em dispositivos dotados de desalinhamento temporal como contramedidas (LERMAN; MARTINASEK; MARKOWITCH, 2017), (**?**) e (PROUFF et al., 2018). Além disso, ataques *Non-Profiled* baseados em técnicas de ML/DL foram apresentados em Timon (2018) e Timon (2019). Entretanto, os métodos apresentados exigem um grande esforço computacional, o que dificulta bastante o ataque na maioria dos casos. Como exemplo, Zhou; Standaert (2020) reportaram um tempo de uma semana para revelar um único *byte* da chave criptográfica.

Métodos para reduzir o tamanho de redes neurais, reduzindo assim o esforço computacional exigido, são encontrados na literatura dentro das mais diversas áreas. Knight; Lee (2021), Guan; Zhang (2020), Gkalelis; Mezaris (2020), Kim; Kim (2020), Ghosh et al. (2019) e Li; Zhu; Sun (2019) usam a técnica de poda como uma forma de reduzir o tamanho de redes neurais. Nestes trabalhos, os autores apresentam significativas reduções na complexidade de suas redes, geralmente com o propósito de aplicá-las em sistemas embarcados (que como se sabe, estas possuem restrições de memória e processamento). Apesar do treinamento de redes neurais não ser realizado em sistemas embarcados na área de SCA, os modelos necessários para essa finalidade podem ser constituídos de até centenas de milhões de parâmetros (para modelos avançados, utilizados para ataques em dispositivos protegidos) (VAN DER VALK et al., 2020), elevando exponencialmente o tempo total do ataque. Outro exemplo interessante é o trabalho proposto por Timon (2018), onde a mesma rede neural precisa ser retreinada 256 vezes para atacar um *byte* de chave do AES em um cenário de aprendizado profundo sem perfil.

As técnicas de poda podem ser aplicadas em diferentes granularidades: pesos, canais (neurônios) ou camadas. Han et al. (2015) e Guo; Yao; Chen (2016) propõem métodos que removem iterativamente os pesos (conexões entre os neurônios) considerados não-importantes. Embora resulte em uma redução na quantidade de parâmetros das redes neurais, e consequentemente no tempo de treinamento das mesmas, essa abordagem é bastante custosa computacionalmente, pois, trata-se de métodos

iterativos, necessitando de muitos treinamentos da rede (um a cada iteração) para verificação da acurácia da rede após cada remoção. Além disso, a poda de pesos é a opção menos eficiente dentre as diferentes granularidades de poda, pois visa a remoção de pesos individuais, ao passo que conjuntos de pesos (neurônios ou camadas) poderiam ser considerados no processo. A remoção de canais, também referida como cirurgia ou processo de ablação por Hu et al. (2016), fornece um modelo reduzido que é mais rápido de treinar, levando em conta um método mais eficiente. Em particular, Hu et al. (2016). apresentaram uma técnica que remove neurônios de acordo com sua taxa de ativação. Em Chen et al. (2021) e Fan; Tang; Ma (2022), os autores propuseram métodos para remover canais (filtros ou mesmo a camada inteira) de camadas de convolução em uma rede neural convolucional (CNN). Entretanto, é mais interessante obter métodos para remoção de canais mais abrangentes que possam ser aplicados à filtros e a neurônios de camadas densas (aplicável a CNNs e *MultiLayer-Perceptrons* - MLPs). Ainda, poda considerando camadas são viáveis somente em redes maiores do que as encontradas na literatura para aplicações em casos de SCA (com muito mais camadas), pois caso contrário a degradação causada pelo método tende a afetar seu funcionamento. A cirurgia ou ablação já foi considerada no contexto de criação de perfil SCA em (WU et al., 2021) como um método para explicar a neutralização de contramedidas ocultas. Aqui, adotamos cirurgia para reduzir o tempo total de treinamento e o número de parâmetros treináveis de redes neurais profundas, o que é altamente atraente em avaliações de segurança. Encontrando uma rede neural menor e mais rápida para treinar, com o intuito de revelar um único *byte* de chave criptográfica (comumente feito na literatura), o mesmo modelo reduzido pode revelar os *bytes* restantes da chave, uma vez que atacar outros *bytes* de chave requer retreinamento da rede.

Diante do exposto, é notória a necessidade de métodos de treinamento mais rápidos. Embora apresente benefícios em relação à eficiência dos ataques, a aplicação desta técnica a SCAs foi muito pouco explorada por pesquisadores. Perin; Wu; Picek (2021) mencionam que existem muitas possibilidades de implementar a técnica de poda em redes neurais aplicadas à SCA, reduzindo sua limitação em termos de complexidade, o que motiva o problema de pesquisa considerado nesta Tese.

## 1.1 Problema de Pesquisa

Considerando o objetivo de contribuir na área de SCA através do uso de algoritmos de inteligência artificial, o problema central desta Tese consiste em soluções para o seguinte questionamento:

> É possível desenvolver um fluxo de ataque para SCA, baseado em métodos de aprendizado profundo que seja eficiente contra dispositivos dotados de

contramedidas temporais, com esforço computacional menor do que os trabalhos propostos na literatura até o presente momento?

Este problema se desdobra nos seguintes aspectos:

- Quais algoritmos de aprendizado profundo se adaptam melhor ao problema de SCA?
- Quais etapas devem compor o fluxo de ataques capaz de neutralizar contramedidas temporais?
- Existem métodos para reduzir o tamanho (quantidade de parâmetros) da rede neural, a fim de mitigar o excesso de esforço computacional empregado para realizar o ataque, mantendo a eficiência necessária?

Os desafios relacionados acima, constituíram a origem dos objetivos perseguidos nesta Tese.

## 1.2 Objetivos

O objetivo principal desta Tese, considerando o problema de pesquisa apresentado, é desenvolver um fluxo para ataques a canais laterais baseado em técnicas de aprendizado profundo que apresente um esforço computacional reduzido comparado à literatura.

Esta Tese considera o uso de redes neurais, além de técnicas que possibilitem a redução destas redes, mantendo a sua eficiência no ataque.

A fim de alcançar esse objetivo principal, são destacados a seguir, os objetivos específicos que foram contemplados:

- Buscar na literatura, trabalhos que aplicam algoritmos de aprendizado profundo no contexto de SCA sob diferentes cenários e contramedidas, assim como estudos que comparam algoritmos de aprendizado profundo no contexto de SCA;
- Analisar os trabalhos que empregam inteligência artificial em SCAs sob diferentes métricas;
- Explorar métodos para reduzir as redes empregadas, a fim de mitigar o excesso de esforço computacional empregado para realizar o ataque, mantendo a eficiência necessária;
- Contribuir para a área de SCAs, propondo um fluxo de ataques a canais laterais baseado em DL, com esforço computacional e eficiência aceitáveis;
- Avaliar o fluxo de ataque desenvolvido, através de métricas de eficiência do ataque e esforço computacional adequadas;

## 1.3 Contribuições da Tese

Uma vez alcançados os objetivos descritos na Seção 1.2, o trabalho desenvolvido nesta Tese traz as seguintes contribuições no campo dos SCAs:

1. Verificação da viabilidade de aplicação de técnicas de redução, através de poda, de redes neurais no contexto de SCA. Com isto, mostrou-se que é possível realizar esse tipo de ataques com redes de tamanhos reduzidos em relação às encontradas na literatura;

2. Obtenção de redes neurais mais eficientes do que as encontradas na literatura. As redes reduzidas através do método proposto necessitam de menos traços do consumo para realizar ataques bem sucedidos. Além disso, por se tratarem de redes menores, o tempo de treinamento também é menor em relação às redes apresentadas na literatura, tornando assim, os ataques mais rápidos e com a necessidade de menos recursos computacionais;

3. Processo de redução de redes neurais *One-Shot*, evitando custosos ciclos de remoção e treinamento comuns na maioria das técnicas de poda encontradas na literatura;

Através das contribuições citadas acima, pudemos apontar o potencial aumento da ameaça causada por essa classe de ataques, uma vez que atacantes são capazes de realizar ataques mais rápido e com menos recursos computacionais.

## 1.4 Estrutura da Tese

Esta Tese está organizada em sete Capítulos, descritos a seguir. No Capítulo 2 é apresentada uma revisão sobre o algoritmo alvo dos SCAs considerados nesta Tese e os principais Ataques *Non-Profiled* encontrados na literatura. Este Capítulo traz uma revisão sobre os principais conceitos e o cenário do problema aqui abordado. Ainda com o objetivo de fornecer embasamento ao leitor, são apresentados no Capítulo 3 conceitos sobre DL, assim como seus principais algoritmos. No Capítulo 4 são discutidos trabalhos que empregam algoritmos de aprendizado de DL no contexto de ataques por canais laterais, sendo apresentada uma comparação entre eles. O Capítulo 6 o método proposto nesta Tese, assim como os experimentos realizados neste trabalho. Finalmente, o Capítulo 7 traz as considerações finais desta Tese, bem como propostas para trabalhos futuros.

# 2 ALGORITMO AES, ATAQUES POR CANAIS LATERAIS E CONJUNTOS DE DADOS EXPERIMENTAIS

Este Capítulo inicia com uma descrição do algoritmo criptográfico *Advanced Encryption Standard* – AES, o qual é atualmente o objeto de estudo da maioria dos trabalhos encontrados na literatura. Seguindo esta abordagem, esta Tese apenas considera o algoritmo AES como objeto de estudo. Aborda as principais categorias de ataques por canais laterais: ataques *non-profiled* e *profiled*. Também, descreve-se os conjuntos de dados abertos utilizados nos experimentos.

## 2.1 Algoritmo Criptográfico AES

O algoritmo de criptografia simétrico AES surgiu a partir da necessidade de substituir seu antecessor, o *Data encryption Standard* – DES (DES, 1977), principalmente devido ao tamanho de chave de 56 bits. Portanto, em 1997 o NIST (do inglês, *National Institute of Standards and Technology*) lançou um concurso para a criação de um novo algoritmo simétrico. No ano 2000, após análises de especialistas, *Rijndael* foi dado como o vencedor do concurso. Assim, o algoritmo proposto por Vicent Rijmen e Joan Daemen (DAEMEN; RIJMEN, 1999) cumpriu todos os requisitos com bom desempenho de *hardware* e *software*.

### 2.1.1 Estrutura

A estrutura do AES suporta chaves e blocos de dados de entradas de 128, 192 e 256 bits. O número de etapas do algoritmo, chamadas de rodadas, varia de acordo com o tamanho da chave. Assim, podemos ter 10, 12 ou 14 rodadas no algoritmo para respectivas chaves de 128, 192 e 256 bits. Para manter a abordagem simplificada, este Capítulo irá considerar apenas a versão AES128.

No processo de cifração, o AES executa as seguintes operações em cada rodada: $\mathbf{AddRoundKey}$, $\mathbf{SubBytes}$, $\mathbf{ShiftRows}$ e $\mathbf{MixColumns}$. O processo de descifração executa em cada rodada a operação $\mathbf{AddRoundKey}$ e a operações inversas $\mathbf{InvSubBytes}$, $\mathbf{InvShiftRows}$ e $\mathbf{InvMixColumns}$. Em ambos os processos, uma

operação inicial de $\mathbf{AddRoundKey}$ ocorre antes da primeira rodada (e também na última rodada, como vemos na Figura 1). Na última rodada a operação $\mathbf{MixColumns}$ (respectivamente $\mathbf{InvMixColumns}$) é suprimida (para a decifração, esta operação é suprimida na primeira rodada). Os diagramas em blocos das operações de cifração e decifração do AES são representados na Figura 1.

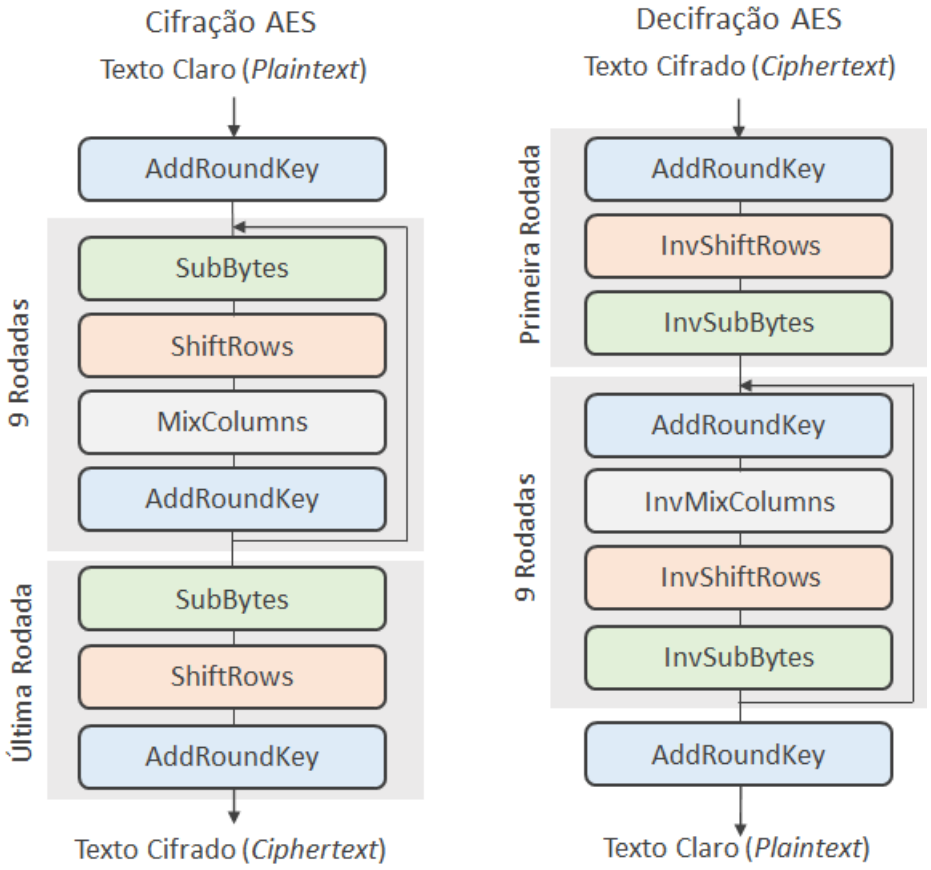


Figura 1 – Diagrama em blocos dos processos de cifração e decifração do AES128.

Entre cada uma das operações do algoritmo AES, há um *estado* (gerado como saída das operações) formado por 16 *bytes*, organizados em formato matricial, com 4 linhas e 4 colunas. A chave de cifração ou decifração é expandida em $N + 1$ (onde $N$ refere-se ao número de rodadas) chaves de rodadas em uma operação de $\mathbf{KeySchedule}$. A seguir, descreve-se em detalhes cada umas das etapas do AES.

#### *2.1.1.1* $\mathbf{AddRoundKey}$

Na etapa de $\mathbf{AddRoundKey}$ é realizada uma operação XOR entre o estado e a chave da rodada. Sendo assim, essa transformação opera cada *byte* individualmente do estado e da chave correspondente. Dessa forma, a operação executada é: $b_{i,j} = a_{i,j} \oplus k_{i,j}$. Onde, $b_{i,j}$ representam os *bits* do estado gerado, $a_{i,j}$ são os *bits* do estado de entrada e $k_{i,j}$ são os *bits* da chave. A Figura 2 ilustra a operação $\mathbf{AddRoundKey}$ no AES128.

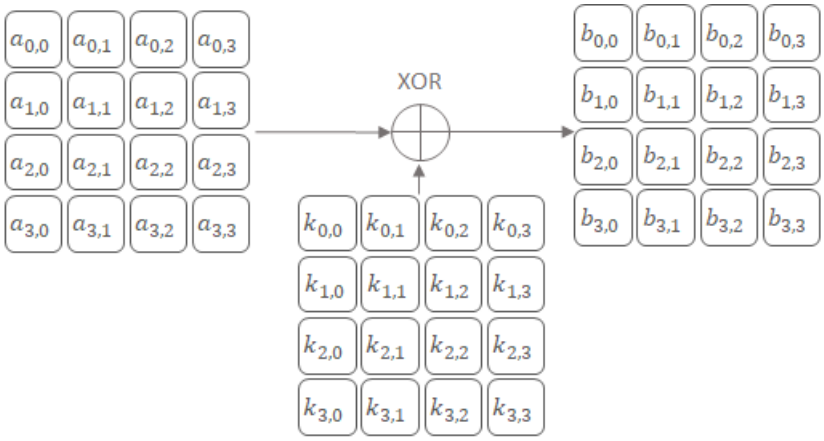


Figura 2 – Operação **AddRoundKey**.

### *2.1.1.2* SubBytes

A camada seguinte consiste na parte não-linear do algoritmo, a operação **SubBytes**. Nesta camada, cada byte da matriz de estado é substituído por outro em uma caixa de substituição chamada SBOX (do inglês, *Substitution Box*). Todos os valores da SBOX estão em hexadecimal. A substituição é realizada da seguinte forma: os quatro primeiros e os quatro últimos *bits* do *byte* representam em hexadecimal, respectivamente, a linha e a coluna em que se encontra o novo *nibble*.

| | | *y* | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
| *x* | **0** | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 01 | 67 | 2B | FE | D7 | AB | 76 |
| | **1** | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| | **2** | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| | **3** | 04 | C7 | 23 | C3 | 18 | 96 | 05 | 9A | 07 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| | **4** | 09 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| | **5** | 53 | D1 | 00 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| | **6** | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 02 | 7F | 50 | 3C | 9F | A8 |
| | **7** | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| | **8** | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| | **9** | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| | **A** | E0 | 32 | 3A | 0A | 49 | 06 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| | **B** | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 08 |
| | **C** | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| | **D** | 70 | 3E | B5 | 66 | 48 | 03 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| | **E** | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| | **F** | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

Tabela 1 – Tabela de Substituição para operação **SubBytes**. Fonte: (HERON, 2009)

Com base na Tabela 1, podemos perceber que, se por exemplo, temos o *byte* c2 a ser substituído, isso significa que devemos buscar o novo *byte* na linha c, coluna 2. Olhando na Tabela, podemos identificar o *byte* 25.

#### *2.1.1.3* $\mathbf{ShiftRows}$

Na operação de $\mathbf{ShiftRows}$, as linhas do Estado são rotacionadas ciclicamente, conforme ilustrado na Figura 3. A operação inversa, $\mathbf{InvShiftRows}$, realiza os deslocamentos na direção oposta. Esta fase do algoritmo AES garante com que as colunas da matriz de estado interajam entre si durante as rodadas da cifração ou decifração.

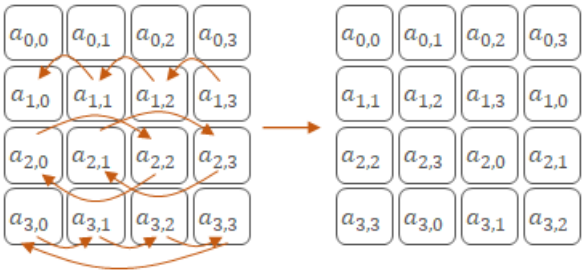


Figura 3 – Operação de $\mathbf{ShiftRows}$.

#### *2.1.1.4* $\mathbf{MixColumns}$

A operação de $\mathbf{MixColumns}$ garante que as linhas da matriz de estado do algoritmo AES interajam entre si durante as rodadas de cifração e decifração. Combinada com a operação de $\mathbf{ShiftRows}$, a operação $\mathbf{MixColumns}$ garante que cada *byte* da saída da matriz de estado dependa de cada *byte* da matriz de estado de entrada. Considerando-se cada coluna da matriz de estado $[a_0, a_1, a_2, a_3]$, constrói-se com os elementos desta coluna um polinômio de grau menor que 3 com coeficiente em $GF(2^8)$ (isto é, Campo de *Galois*). Uma nova coluna é produzida utilizando-se o polinômio 1:

$$a(X) = a_0 + a_1.X + a_2.X^2 + a_3.X^3 \tag{1}$$

e multiplicando-se pelo polinômio:

$$c(X) = \mathbf{0x02} + \mathbf{0x01}.X + \mathbf{0x01}.X^2 + \mathbf{0x03}.X^3 \tag{2}$$

e com o resultado da multiplicação modulo $M(X) = X^4 + 1$. Esta operação é representada pela seguinte operação matricial em $GF(2^8)$:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} \mathbf{0x02} & \mathbf{0x03} & \mathbf{0x01} & \mathbf{0x01} \\ \mathbf{0x01} & \mathbf{0x02} & \mathbf{0x03} & \mathbf{0x01} \\ \mathbf{0x01} & \mathbf{0x01} & \mathbf{0x02} & \mathbf{0x03} \\ \mathbf{0x03} & \mathbf{0x01} & \mathbf{0x01} & \mathbf{0x02} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} \tag{3}$$

Pelo fato da operação matricial da Equação (3) ser executada em $GF(2^8)$, ela se torna invertível. Portanto, a inversa da operação de $\mathbf{MixColumns}$, $\mathbf{InvMixColumns}$, pode ser implementada com a mesma operação matricial da Equação (3).

Os algoritmos de cifração e decifração para o AES128 são descritos nos Algoritmos das Figuras 4 e 5, respectivamente.

1: $\mathbf{AddRoundKey}(S, K_0)$
2: **for** $i = 1$ to $9$ **do**
3: $\quad \mathbf{SubBytes}(S)$
4: $\quad \mathbf{ShiftRows}(S)$
5: $\quad \mathbf{MixColumns}(S)$
6: $\quad \mathbf{AddRoundKey}(S, K_i)$
7: $\mathbf{SubBytes}(S)$
8: $\mathbf{ShiftRows}(S)$
9: $\mathbf{AddRoundKey}(S, K_{10})$

Figura 4 – Algoritmo AES128 - Cifração.

1: $\mathbf{AddRoundKey}(S, K_{10})$
2: $\mathbf{InvShiftRows}(S)$
3: $\mathbf{InvSubBytes}(S)$
4: **for** $i = 9$ downto $1$ **do**
5: $\quad \mathbf{AddRoundKey}(S, K_i)$
6: $\quad \mathbf{InvMixColumns}(S)$
7: $\quad \mathbf{InvShiftRows}(S)$
8: $\quad \mathbf{InvSubBytes}(S)$
9: $\mathbf{AddRoundKey}(S, K_0)$

Figura 5 – Algoritmo AES128 - Decifração.

#### *2.1.1.5* $\mathbf{KeySchedule}$

Conforme pode ser observado nos Algoritmos mostrados nas Figuras 4 e 5, chaves de rodada $K_i$, para $i = 0 \ldots 10$, são empregadas em cada operação de $\mathbf{AddRoundKey}$. A obtenção dessas chaves de rodada resulta da operação de $\mathbf{KeySchedule}$. Cada uma das 11 chaves de rodada para o algoritmo AES128 consiste de 4 palavras de 32 *bits*, sendo que cada uma dessas palavras corresponde a uma coluna da matriz de estado. A operação $\mathbf{KeySchedule}$ faz uso de uma constante de rodada ($RC_i$) que pode ser descrita como:

$$RC_i \leftarrow x^i \ (mod\ x^8 + x^4 + x^3 + x + 1) \tag{4}$$

Assim, as chaves de rodada são rotuladas como ($W_{4i}$, $W_{4i+1}$, $W_{4i+2}$, $W_{4i+3}$) onde $i$ indica a rodada. A primeira chave (chave de cifração) é dividida em quatro palavras de 32 *bits*, ($k_0$, $k_1$, $k_2$, $k_3$). As chaves de rodada são então calculadas através do Algoritmo mostrado na Figura 6, onde a operação $\mathbf{RotBytes}$ é a função que rotaciona uma

palavra em um *byte* para a esquerda e $\mathbf{SubBytes}$ é a mesma operação de substituição utilizada na cifração e aplicada em cada *byte* da palavra.

1: $W_0 \leftarrow k_0, W_1 \leftarrow k_1, W_2 \leftarrow k_2, W_3 \leftarrow k_3$
2: **for** $i = 1$ to $10$ **do**
3: $\quad T \leftarrow \mathbf{RotBytes}(W_{4i-1})$
4: $\quad T \leftarrow \mathbf{SubBytes}(T)$
5: $\quad T \leftarrow T \oplus RC_i$
6: $\quad W_{4i} \leftarrow W_{4i-4} \oplus T$
7: $\quad W_{4i+1} \leftarrow W_{4i-3} \oplus W_{4i}$
8: $\quad W_{4i+2} \leftarrow W_{4i-2} \oplus W_{4i+1}$
9: $\quad W_{4i+3} \leftarrow W_{4i-1} \oplus W_{4i+2}$

Figura 6 – Operação *KeySchedule* para AES128.

O algoritmo AES possui diferentes modos de aplicação. O mais simples de todos é o modo ECB (*Eletronic Code Book*) e será considerado durante todo este projeto de pesquisa.

## 2.2 Principais Ataques por Canais Laterais

Esta Seção revisa o fundamento dos principais ataques por canais laterais. Ataques por canais laterais utilizam diferentes vazamentos de informação não intencionais a partir de um dispositivo eletrônico. O consumo, inicialmente referenciado por ataque por análise de potência (em inglês, *Power Analysis Attack*), é apenas uma delas. Outros exemplos são emissão eletromagnética, tempo de execução, temperatura ou canais acústicos. No entanto, o termo potência é utilizado na nomenclatura dos ataques sem necessariamente indicar que os sinais coletados representem medições de potência.

Uma forma comum de dividir ataques por canais laterais é com relação ao modelo de ameaça. Dessa forma, a divisão de mais alto nível é entre ataques *non-profiled* e *profiled*, conforme descrito a seguir.

### 2.2.1 Ataques Non-Profiled

Esta Seção apresenta os ataques *non-profiled* que usam modelos de consumo de energia genéricos para circuitos CMOS, tais como os baseados em Peso *Hamming* e Distância *Hamming*, e através de análises estatísticas ou modelos de correlação realizam a comparação entre o consumo hipotético modelado e o consumo medido.

#### *2.2.1.1 Ataques DPA e DEMA*

Os ataques DPA apresentados por Kocher; Jaffe; Jun (1999), exploram a relação de dependência entre o consumo de um dado circuito digital dedicado à execução

de um algoritmo criptográfico com os dados processados pelo mesmo. Para realizar o ataque, não é necessário que o atacante tenha um conhecimento detalhado do algoritmo criptográfico nem de sua implementação. Porém, DPA exige uma grande quantidade de traços para a análise estatística. Outra característica interessante de ataques DPA é que mesmo diante de perturbações elétricas durante o monitoramento e aquisição dos traços, é possível realizar ataques bem-sucedidos. Seu custo de execução é relativamente baixo, e por tratar-se de um ataque não-invasivo, não deixa vestígios no dispositivo atacado. Ainda, este tipo de ataque possui um bom índice de sucesso. Essas características tornaram os ataques DPA, os mais populares dentro da área de criptoanálise.

O ataque DPA é composto de 5 etapas: *(i)* escolher um resultado intermediário alvo, *(ii)* medir e coletar traços, *(iii)* calcular valores intermediários hipotéticos, *(iv)* aplicar um modelo de consumo ao dispositivo atacado e *(v)* avaliar hipóteses de subchaves.

A primeira etapa do ataque consiste em escolher um resultado intermediário do algoritmo criptográfico alvo. Esse resultado precisa ser uma função $f(d, k)$, onde *k* é uma porção da chave secreta e *d* é parte da mensagem de entrada ou saída conhecida. Se o atacante obtiver uma função que satisfaça essa condição, esta pode ser utilizada como alvo do ataque para encontrar *k*. A mensagem conhecida *d* pode ser tanto uma mensagem de entrada ou um criptograma de saída, ou até mesmo outro dado intermediário que seja conhecido.

Na segunda etapa do ataque, a potência dissipada é medida enquanto várias encriptações ou decriptações são executadas sobre um conjunto de *D* dados distintos, usando a mesma chave criptográfica. Assim, *D* é um conjunto contendo diferentes dados aleatórios *di*, $D = \{d_1, d_2, \ldots, d_D\}$.

Para cada encriptação ou decriptação é armazenado um traço $t_i$ de potência correspondente, formando um conjunto *T* de traços. Como temos um traço a cada encriptação ou decriptação de um dado, *D* e *T* possuem o mesmo tamanho. São armazenadas *J* amostras de potência em cada traço. Portanto, cada traço $t_i$ pode ser descrito como $t_{i,j} = \{t_{0,j}, \ldots, t_{0,J}\}$. Finalmente, esses traços são armazenados em uma matriz $M_T$ de tamanho D×J (ou T×J), conforme Equação (5).

$$M_T = \begin{bmatrix} t_{0,0} & t_{0,1} & t_{0,2} & \ldots & t_{0,J} \\ t_{1,0} & t_{1,1} & t_{1,2} & \ldots & t_{1,J} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{D,0} & v_{D,1} & t_{D,2} & \ldots & t_{D,J} \end{bmatrix} \tag{5}$$

Nesta etapa, pode-se perceber que os traços de potência devem estar bem alinhados para que o ataque DPA obtenha sucesso, pois cada coluna $t_j$ da matriz $M_T$ deve corresponder às mesmas operações durante a encriptação ou decriptação para que

possam ser comparadas e analisadas. Assim, para que o ataque tenha eficiência, os traços de potência analisados, devem estar alinhados no domínio do tempo.

A terceira etapa consiste no cálculo de valores hipotéticos intermediários para todas as possibilidades de valores de *k*, relembrando que *k* são hipóteses da chave secreta. Portanto, faz-se o cálculo de valores hipotéticos intermediários, de acordo com $f(d, k)$, para todos os valores possíveis de chave.

As hipóteses de chave são denotadas por um conjunto $Hk = \{hk_0, hk_1, \ldots, hk_K\}$, onde *K* é o número total de possibilidades de chave *k*. Assim, através do conjunto de dados *D* e do conjunto de chaves hipotéticas *Hk*, o atacante pode calcular todos os valores intermediários hipotéticos possíveis para $f(d, k)$.

$$v_{i,j} = f(d_i, k_j)\ i = 1, \ldots, D\ e\ j = 1, \ldots, K \tag{6}$$

Esses valores formam uma matriz denominada $M_V$, que como visto na Equação (6), tem tamanho *D* × *K*. A matriz $M_V$ é mostrada na Equação (7).

$$M_V = \begin{bmatrix} v_{0,0} & v_{0,1} & v_{0,2} & \ldots & v_{0,K} \\ v_{1,0} & v_{1,1} & v_{1,2} & \ldots & v_{1,K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{D,0} & v_{D,1} & v_{D,2} & \ldots & v_{D,K} \end{bmatrix} \tag{7}$$

Podemos observar, através da Equação (7), que cada coluna $j$ da matriz $M_V$ contém os resultados calculados para a hipótese de chave $hk_j$, através de $f(d_i, k_j)$. Obviamente que, se $M_V$ possui os resultados intermediários para todas as possibilidades de chave *k*, então uma de suas colunas possui os valores intermediários reais calculados pelo sistema criptográfico durante a encriptação ou decriptação dos dados, realizados na segunda etapa.

Também é óbvio que a chave secreta é um dos elementos do vetor *Hk* descrito anteriormente. Esse elemento é denominado de $hk_{ck}$. Assim, o ataque DPA procura descobrir, em qual coluna da matriz $M_V$ encontram-se os mesmos valores produzidos por $f(d, k)$ durante a encriptação ou decriptação do vetor *D*.

A quarta etapa do ataque DPA, é a etapa em que o modelo de consumo é aplicado ao dispositivo que está sendo atacado. São relacionados os traços de potência com os Pesos *Hamming* (quantidade de bits diferentes de zero) dos resultados intermediários $v_{i,j}$ anteriormente calculados, pois o Peso *Hamming* de um circuito CMOS é proporcional à potência do mesmo. Uma alternativa a este modelo é o uso da Distância *Hamming* para modelar a potência. Esta alternativa aproxima-se mais do consumo real de um circuito digital e, portanto, é mais usado na prática. A Distância *Hamming* entre dados com representação binária é igual a diferença de seus Pesos *Hamming*, o que pode ser obtido logicamente pela execução de uma operação XOR. Nos ataques

DPA a Distância *Hamming* é aplicada sobre os valores intermediários calculados pela função escolhida na primeira etapa, tal como mostrado na Equação (8).

$$HD = r_0 \oplus r_1 \tag{8}$$

onde $r_0$ e $r_1$ são pesos *Hamming* e *HD* é a distância *Hamming*.

Com isso, (KOCHER; JAFFE; JUN, 1999) associaram a cada valor intermediário hipotético calculado $v_{i,j}$ um único valor binário $h_{i,j}$ relacionado ao consumo deste valor intermediário da seguinte forma: se o consumo relacionado à este valor for alto, ou seja, $v_{i,j} = 1$, então $h_{i,j} = 1$ , caso contrário, $h_{i,j} = 0$ . Estes valores $h_{i,j}$ formam uma matriz chamada de $M_H$.

Finalmente, a última e quinta etapa do ataque DPA, tem como objetivo avaliar as hipóteses de subchaves, atividade realizada a partir das matrizes $M_T$ dos traços de potência e $M_H$ a matriz dos valores de consumo hipotéticos calculados a partir do modelo de potência utilizado.

Cada coluna $h_j$ da matriz $M_H$ é comparada com a coluna correspondente $t_j$ da matriz $M_T$, ou seja, nesta etapa o atacante compara os valores de consumo hipotéticos de cada hipótese de chave com os traços de potência coletados do dispositivo atacado. Os resultados são armazenados em uma matriz $M_R$ de tamanho *D x K*. Cada elemento dessa matriz, denominado de $r_{i,j}$ é a comparação com base no método da diferença das médias, entre as colunas $h_j$ e $t_j$, proposto por Kocher et al. em 1999.

Pode-se concluir do exposto até agora que temos os traços de potência $t_{i,j}$ do sistema criptográfico ao executar encriptação ou decriptação para diferentes dados de entrada e os resultados intermediários $v_{i,k}$ calculados com base nos dados de entrada, dentre os quais está o resultado obtido com a chave secreta do sistema de criptografia $v_{ck}$, pois todas as possibilidades de chave são utilizadas nos resultados intermediários. Assim, pode-se notar que em algum momento, ou instante de tempo denominado $ct$, os traços de potência estão relacionados ao resultado intermediário $v_{ck}$ executado sobre a chave secreta verdadeira do circuito criptográfico atacado.

Como os valores hipotéticos de potência $h_{i,j}$ são calculados sobre os resultados intermediários $v_{i,j}$, podemos perceber que $h_{ck}$, que é a potência hipotética calculada com o resultado intermediário da chave secreta verdadeira $v_{ck}$, está fortemente relacionado à $t_{ct}$, que é o traço de potência medido, no instante que está executando a encriptação ou decriptação do resultado intermediário para a chave secreta verdadeira $v_{ck}$.

Devido a esta forte relação entre $h_{ck}$ e $t_{ct}$, o atacante pode descobrir o índice $ck$ da chave secreta, e por sua vez a própria chave $hk_{ck}$, através da observação dos valores da matriz $M_R$.

Como mencionado anteriormente, Kocher et al. apresentaram como método para

avaliar as chaves hipotéticas $hk_i$, o método da diferença das médias, que no caso dos ataques DPA é utilizado para relacionar as matrizes $M_H$ e $M_T$. Esse método dá-se da seguinte forma: primeiramente o atacante divide a matriz dos traços de potência medidos $M_T$ em duas outras matrizes$M_{T0}$ e $M_{T1}$, sendo $M_{T0}$ composta pelas linhas da matriz $M_T$ cujos coeficientes $h_{i,j}$ da matriz $M_H$ são iguais a zero. E a matriz $M_{T1}$, é montada com as linhas restantes da matriz $M_T$. Depois disso, a média das linhas de cada uma das matrizes $M_{T0}$ e $M_{T1}$ é calculada, sendo $avg0$ o vetor que contém a média das linhas da matriz $M_{T0}$ e $avg1$ o vetor que contém a média das linhas da matriz $M_{T1}$. A hipótese de chave $k_i$ está correta se a diferença absoluta entre os vetores $avg0$ e $avg1$ for maior do que para as outras hipóteses de chave. Cabe aqui destacar que existem outras formas de relacionar as matrizes $M_H$ e $M_T$, dependendo do ataque realizado. Como exemplo, podemos citar o coeficiente de correlação utilizado no ataque CPA descrito na Seção 2.2.1.2. A Figura 7 mostra um fluxo de execução do ataque DPA descrito acima.

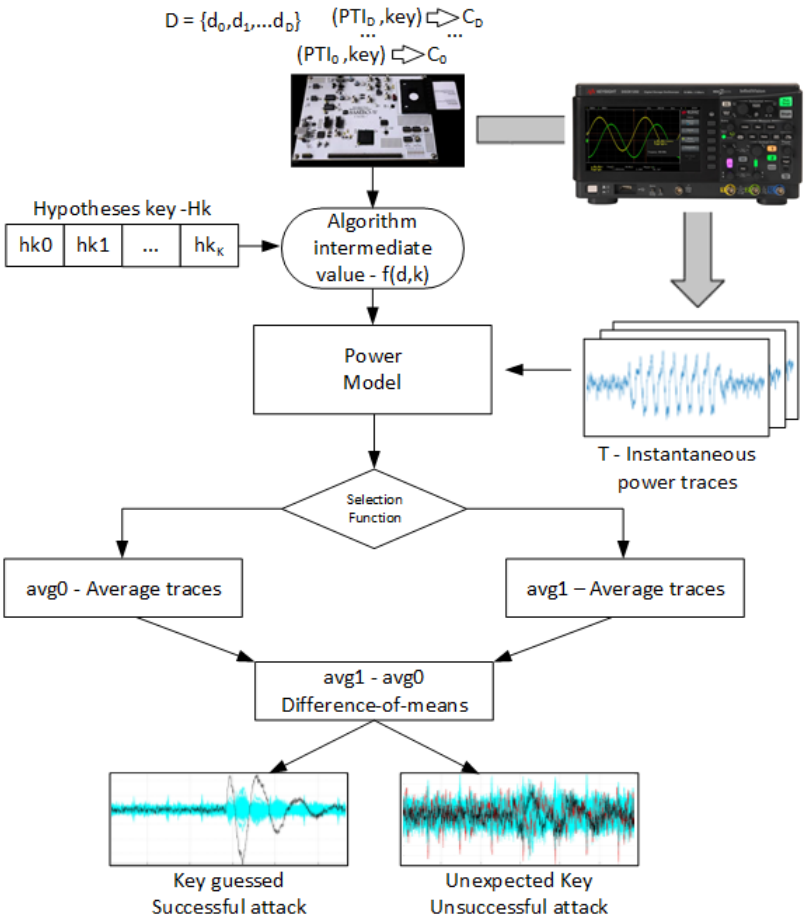


Figura 7 – Fluxo de execução do ataque DPA. Fonte: (SOARES, 2010).

Da mesma forma que o ataque DPA, o ataque DEMA (AGRAWAL et al., 2002) avalia e monitora a emissão de ondas eletromagnéticas do dispositivo atacado. Os ataques DEMA capturam os traços gerados pelos campos eletromagnéticos emitidos

pelos circuitos durante a execução de encriptação ou decriptação, através de sondas especiais utilizadas em conjunto com estágios amplificadores devido à baixa intensidade do sinal produzido. Para esse tipo de ataque, o atacante deve levar em conta os problemas causados por ruídos e interferências eletromagnéticas do ambiente onde é realizado o ataque, ocasionando erros nas leituras dos traços.

#### *2.2.1.2 Ataques CPA*

Os ataques por Análise de Correlação de Potência (em inglês, *Correlation Power Analysis*) operam basicamente do mesmo modo que os ataques DPA/DEMA descritos na Seção 2.2.1.1. Na verdade, CPA é uma especialização de DPA proposta por (BRIER; CLAVIER; OLIVIER, 2004). Este tipo de ataque foi proposto com a finalidade de reduzir o problema de picos fantasmas que eventualmente ocorrem nos ataques DPA/DEMA.

Neste ataque, o coeficiente de correlação é usado para avaliar a relação entre cada coluna $h_i$ da matriz $M_H$ com cada coluna $t_j$ da matriz $M_T$. Isto resulta em uma comparação entre os valores de consumo hipotético e os traços adquiridos em cada posição de tempo. O resultado é armazenado em uma matriz $M_R$, onde cada elemento representa o coeficiente de correlação estimado. Assim, pode-se descrever cada valor $r_{i,j}$ pela Equação (9), onde $\overline{h_i}$ e $\overline{t_j}$ representam os valores médios das colunas $h_i$ e $t_j$:

$$r_{i,j} = \frac{\sum_{d=0}^{D}(h_{d,i} - \bar{h_i}) \cdot (t_{d,j} - \bar{t_j})}{\sqrt{\sum_{d=0}^{D}(h_{d,i} - \bar{h_i})^2 \cdot \sum_{d=0}^{D}(t_{d,j} - \bar{t_j})^2}} \tag{9}$$

Segundo (BRIER; CLAVIER; OLIVIER, 2004) estes coeficientes, ou fatores de correlação, são capazes de rejeitar falsas hipóteses de chave aceitas pela aplicação dos métodos DPA/DEMA.

### 2.2.2 Ataques Profiled

Os ataques *profiled* são aqueles que constroem um modelo estatístico a partir de medições obtidas de um dispositivo alvo. O atacante tem o controle total sobre o circuito atacado e pode usar o conjunto de dados a sua escolha para construir o modelo de consumo para o circuito alvo. A partir de então, por meio de análises estatísticas pode comparar os traços de consumo medidos com o modelo construído para inferir hipóteses sobre os dados usados na computação.

#### *2.2.2.1 Ataques por Templates (Template Attacks, TA)*

Em TAs, o adversário usa um dispositivo experimental, idêntico ao dispositivo em teste, para identificar uma pequena seção da amostra *S*, ou seja, apenas alguns *bits* da chave desconhecidos. Empiricamente, ele constrói modelos correspondentes a cada valor possível dos *bits* de chave desconhecidos. O modelo consiste nas distri-

buições de probabilidade média de sinal e ruído. Ele então usa esses modelos para classificar aquela porção de *S* e limitar as escolhas para os *bits-chave* a um pequeno conjunto. Isso é então repetido com um prefixo mais longo de *S* envolvendo mais *bits* da chave. Assim, os ataques TA usam essencialmente uma estratégia de estender e remover dirigida pela amostra única *S* a ser atacada: são usados prefixos cada vez mais longos de *S* e os modelos correspondentes para diminuir o espaço de possíveis prefixos de chave. O sucesso depende criticamente de quão eficazmente a estratégia de remoção reduz a explosão combinatória no processo de extensão.

Em Chari; Rao; Rohatgi (2002) os autores destacam que os ataques DPA/DEMA assumem o ruído como um obstáculo que precisa ser reduzido ou eliminado. Neste sentido, a abordagem proposta concentra-se na modelagem precisa do ruído, a fim de extrair as informações presentes em um único traço. Portanto, é possível obter-se sucesso em TA, com apenas um, ou um número limitado de traços. Por outro lado, em um ataque DPA/DEMA muitos traços são necessários para que o ruído seja eliminado pela média. Diante disso, a proposta dos autores potencializa ataques a canais laterais quando o atacante tem acesso a um ou poucos traços de consumo disponíveis. Entretanto, os autores destacam que é fundamental para o TA que o adversário tenha um dispositivo experimental idêntico que pode ser programado, o que claramente se traduz em uma desvantagem do método.

É destacado que os experimentos mostram que uma implementação de um algoritmo criptográfico chamado RC4 (do inglês, *Rivest Cipher* 4), não passível de técnicas como SPA e DPA, pode ser facilmente quebrada usando TA com um único traço.

É explicado que o conceito de TA é baseado na Teoria de Detecção e Estimação de Sinais e, em particular, no uso de técnicas de Teoria da Informação, como razões de probabilidade para testes de hipóteses. E que embora outras técnicas, como DPA, também possam ser vistas como aproximações grosseiras de razões de probabilidade, o uso de modelos de mistura Gaussiana é a chave para extrair o máximo de informações de uma única amostra (traço). Empiricamente, observa-se que em várias situações as estatísticas univariáveis não são suficientes e apresentam resultados ruins.

Chari; Rao; Rohatgi (2002) colocam que se tivermos um dispositivo executando uma das possíveis *K* sequências de operações $\{O_1, \ldots, O_K\}$, um adversário pode amostrar o consumo durante a operação que deseja identificar ou reduzir significativamente o conjunto de hipóteses possíveis. Isto é uma prática comumente empregada em processamento de sinais, onde modela-se o sinal, referenciado aqui como um traço, observado como uma combinação de um sinal intrínseco gerado pela operação e ruído que é gerado pelo ambiente. Portanto, enquanto o componente do sinal que representa a operação é o mesmo para diferentes invocações da mesma, o ruído é melhor modelado como uma amostra aleatória retirada de uma distribuição de proba-

bilidade de ruído que depende da operação e outras do ambiente de medição. Assim, a abordagem ótima para o adversário que está tentando encontrar a hipótese certa dada uma única amostra *S* é usar a abordagem de máxima probabilidade: a melhor suposição é escolher a operação de modo que a probabilidade do ruído observado em *S* é maximizada. O cálculo dessa probabilidade exige que o adversário modele com precisão o sinal intrínseco e a distribuição de probabilidade de ruído para cada operação.

Pode-se perceber, que para o sucesso dos ataques TA, é estritamente necessário que o atacante consiga obter caracterizações extremamente boas e precisas do ruído. Contudo, muito embora o adversário se considere altamente competente com tais caracterizações, na prática aproximações como um modelo Gaussiano multivariável para as distribuições de ruído produzem resultados satisfatórios.

O trabalho apresentado por Chari; Rao; Rohatgi (2002) enumera quatro passos para a elaboração de um modelo Gaussiano:

*i*. Coletar um grande número (L) de amostras (tipicamente mil) no dispositivo experimental para cada uma das *K* operações, $O_1, \ldots, O_K$.

*ii*. Calcular o sinal médio para cada uma das operações $M_1, \ldots, M_K$.

*iii*. Calcular a diferença entre os pares de sinais médios $M_1, \ldots, M_K$ para identificar e selecionar somente os pontos $P_1, \ldots, P_N$ em que grandes diferenças aparecem. O modelo Gaussiano se aplica a esses *N* pontos. Esta etapa opcional reduz significativamente a sobrecarga de processamento com apenas uma pequena perda de precisão.

*iv*. Para cada operação $O_i$, o vetor de ruído *N*-dimensional para a amostra *T* é $N_i(T) = (T[P_1] - M_i[P_1], \ldots, T[P_N] - M_i[P_N])$. Calcular a matriz de covariância entre todos os pares de componentes dos vetores de ruído para a operação $O_i$ usando os vetores de ruído $N_i$ *s* para todas as *L* amostras. As entradas da matriz de covariância $\sum N_i$ são definidas como:

$$\sum N_i[u, v] = cov(N_i(P_u), N_i(P_v)) \tag{10}$$

Usando a Equação (10), calcula-se os modelos $(M_i, \sum N_i)$ para cada uma das *K* operações. O sinal para a operação $O_i$ é $M_i$ e a distribuição da probabilidade de ruído é dada pela distribuição Gaussiana multivariável $pN_i(.)$ onde a probabilidade de um vetor de ruído *n* é:

$$pN_i(n) = \frac{1}{\sqrt{2\pi^N |\sum N_i|}} exp\left(\frac{-1}{2} n^T \sum_{N_i}^{-1} n\right), n \in R^N \tag{11}$$

Onde $|\sum N_i|$ representa o determinante de $\sum N_i$ e $\sum_{N_i}^{-1}$ é o seu inverso.

Os autores observam que neste modelo, a técnica ideal para classificar uma amos-

tra *S* é a seguinte: para cada operação hipotética $O_i$, calcule a probabilidade de *S* ter de fato se originado de $O_i$. Esta probabilidade é dada calculando primeiro o ruído *n* em *S* usando o sinal médio $M_i$ no modelo e depois calculando a probabilidade de *n* usando a expressão para a distribuição de probabilidade de ruído e o $\sum N_i$ calculado do modelo. Se o ruído for realmente Gaussiano, então a abordagem de selecionar o $O_i$ com a maior probabilidade é ótima. A probabilidade de cometer erros em tal classificação também é computável. Portanto, se o uso dessa abordagem para distinguir duas operações $O_1$ e $O_2$ com a mesma caracterização de ruído $\sum N_i$ , o erro de probabilidade é dado por:

$$P_\epsilon = \frac{1}{2} erfc\left(\frac{\Delta}{2\sqrt{2}}\right) \tag{12}$$

Onde $\Delta^2 = (M_1 - M_2)^T \overset{-1}{\underset{N}{\sum}} (M_1 - M_2)$ e $erfc = 1 - erf(x)$.

A seguir, os autores destacam que é necessário que o processo de remoção reduza o conjunto de possíveis hipóteses de operações para um número muito pequeno, garantindo com alta probabilidade que a hipótese correta não seja descartada.

De acordo com Chari; Rao; Rohatgi (2002), uma abordagem que funciona bem é dimensionar as probabilidades de modo que as probabilidades de ruído em todas as hipóteses somem. Em seguida, descarta-se as hipóteses com as probabilidades escalonadas mais baixas até que a probabilidade cumulativa de erro devido às hipóteses descartadas alcance o limite de erro desejado. Ao longo do texto, os autores apresentam outras abordagens para a etapa de remoção das hipóteses de operações.

## 2.3 Considerações sobre o Capítulo

Este Capítulo apresentou a revisão do algoritmo criptográfico AES utilizado como estudo de caso para os experimentos aqui propostos. Além disso, foi realizada uma revisão das principais categorias de ataques por canais laterais, entre eles, ataques *non-profiled* e ataques *profiled*, que são o foco deste trabalho.

O Capítulo busca revisar os modelos de consumo empregados em ataques clássicos, como os apresentados na Seção 2.2. Estes modelos são baseados em análises estatísticas do comportamento do consumo em circuitos implementados com a tecnologia CMOS e, além de serem estáticos, muitas vezes não se adaptam da melhor forma ao real comportamento dos traços. Desta forma, percebe-se ser interessante o uso de algoritmos de aprendizado profundo capazes de produzir modelos mais ajustados aos SCA, estratégia usada nos ataques *profiled*.

# 3 APRENDIZADO PROFUNDO

Este Capítulo revisa conceitos de Aprendizagem Profunda (do inglês, *Deep Learning* – DL). Com isso pretende-se fornecer um *background* para embasar o entendimento do objeto de estudo desta Tese: redes neurais que são capazes de realizar ataques por canais laterais. Embora possam existir outros tipos de redes neurais que realizem essa tarefa, aqui nos deteremos nas duas arquiteturas mais recorrentes na literatura para este fim e que são usadas no estudo de caso deste trabalho: redes Perceptron de Múltiplas Camadas (do inglês, *Multi-Layer Perceptron* - MLP) e as Redes Neurais Convolucionais (do inglês, *Convolutional Neural Network* - CNN).

## 3.1 Redes Neurais

A criação e o desenvolvimento de redes neurais artificiais teve como motivação o reconhecimento de que o cérebro humano processa informações de uma forma totalmente diferente do computador digital convencional. O cérebro é uma espécie de computador altamente complexo, não-linear e paralelo. Ele tem capacidade de organizar suas células (neurônios) de forma a realizar processamentos (reconhecimento de padrões, percepção e controle motor, por exemplo) muito mais rápido que os computadores digitais convencionais (HAYKIN, 2001).

Desde o nascimento, um cérebro humano tem uma grande estrutura formada por diversas áreas, que por sua vez são formadas por neurônios e habilidade de desenvolver suas próprias regras através do que chamamos de "experiência", que vai sendo acumulada com o tempo.

De uma forma mais geral, pode-se ver o sistema nervoso humano como um sistema de três estágios. Os receptores convertem estímulos do corpo humano ou do ambiente externo em impulsos elétricos que transmitem informação para a rede neural (cérebro). E os atuadores convertem impulsos elétricos gerados pela rede neural em respostas adequadas para cada saída do sistema (Figura 8).

As conexões entre as células da rede neural humana (o cérebro) são chamadas de sinapses. As sinapses responsáveis pelas zonas receptivas são os dendritos, en-

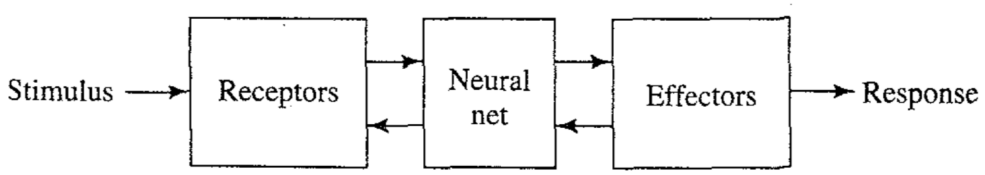


Figura 8 – Representação em diagrama de blocos do sistema nervoso. Fonte: (HAYKIN, 2001)

quanto as sinapses transmissoras denominam-se axônios.

Um neurônio pode receber 10.000 ou mais conjuntos sinápticos e pode se projetar sobre milhares de células alvo. Através das sinapses, as informações são transmitidas de um neurônio a outro através de pulsos breves de tensão (ou impulsos - *spikes*).

A forma como aprendemos a realizar tarefas, e a estrutura organizacional do cérebro, com neurônios conectados entre si, inspiraram o desenvolvimento de equivalentes eletrônicos ou computacionais do cérebro humano (mesmo que ainda de uma forma primitiva).

### 3.1.1 Modelo de um Neurônio Artificial

No que concerne as redes neurais artificiais, um *neurônio* consiste na sua unidade fundamental de operação, fazendo uma analogia com os *neurônios* do cérebro humano. *Neuronios*, no campo das redes neurais artificiais, são também denominados *perceptrons*. Usaremos os termos *perceptron* e *neurônio* nesta Tese.

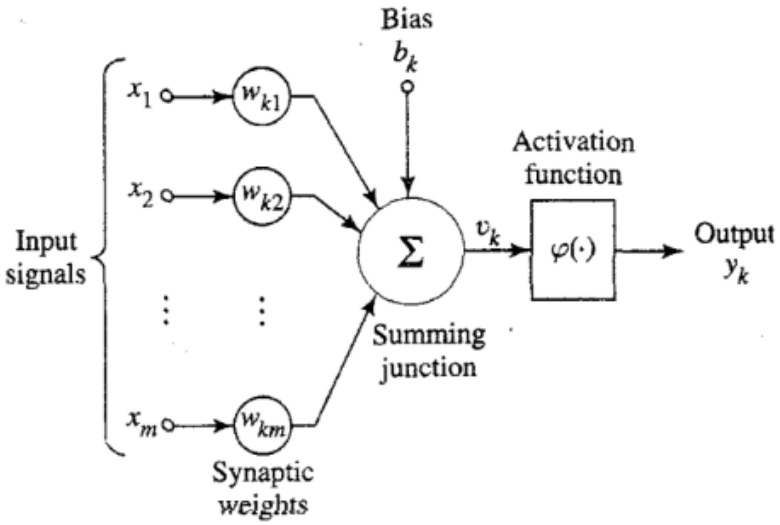


Figura 9 – Simbologia de um *Perceptron*. Fonte: (HAYKIN, 2001)

A Figura 9 ilustra o modelo de um *perceptron*. Aqui, pode-se identificar três elementos básicos deste modelo:

1. Um conjunto de *sinapses* ou elos de conexão caracterizados por um *peso*. Como

podemos na Figura 9, um sinal de entrada $x_j$ da conexão (sinapse) $j$ conectada ao neurônio $k$ é multiplicado pelo seu peso sináptico correspondente $w_{kj}$. Ao contrário da sinapse do cérebro, o peso sináptico de um *neurônio* artificial pode estar em um intervalo que inclui valores negativos bem como positivos;

2. Um somador que realiza uma soma ponderada dos sinais de entrada. Assim, são somados todos os valores resultantes da multiplicação de $x_j$ por $w_{kj}$. Essas operações constituem um *combinador linear*;

3. Uma *função de ativação* para restringir a saída do *neurônio*. Geralmente o intervalo normalizado da amplitude da saída de um *neurônio* é dado por um intervalo unitário fechado [0,1], ou ainda [-1,1];

Portanto, para um conjunto de valores de entrada $x = (x_1, \ldots, x_m)$ (comumente referidos como *features* na literatura), o *perceptron* efetua um somatório da combinação linear com pesos $w_{k1}, \ldots, w_{km}$ (ou conexões) relacionados a cada elemento de entrada $x_j$, isto é, $\sum_{j=1}^{m} w_{kj}x_j$. O resultado do somatório é adicionado a um valor de viés (ou *bias*), $b_k$.

O resultado da Equação do perceptron, $b_k + \sum_{j=1}^{m} w_{kj}x_j$, passa por uma *função de ativação* $\phi$, normalmente não linear. Exemplos de funções de ativação muito utilizadas na literatura (e como veremos adiante, em ataques por canais laterais) são as funções tangente hiperbólica, sigmóide, unidade linear retificada, etc. Dessa forma, o valor de saída $y_k$ de um *perceptron* é apresentado pela Equação abaixo:

$$y_k = \phi(b_k + \sum_{j=1}^{m} w_{kj}x_j) \tag{13}$$

#### *3.1.1.1 Tipos de Funções de Ativação*

A forma mais básica de definir o estado de um neurônio é aplicando à saída de $(b_k + \sum_{j=1}^{m} w_{kj}x_j)$ uma *Função de Limiar* ou como comumente chamada nas áreas de engenharia, *Função Degrau Unitário*. Para simplificar as notações subsequentes, vamos considerar:

$$v = b_k + \sum_{j=1}^{m} w_{kj}x_j \tag{14}$$

A partir disso, podemos definir matematicamente a *Função de Limiar*, através da Equação 15:

$$\phi(v) = \begin{cases} 1 & se \quad v \geq 0 \\ 0 & se \quad v < 0 \end{cases} \tag{15}$$

Neste ponto, é interessante ressalta-se que a *Função de Limiar* não é diferenciável, pois esta função apresenta uma descontinuidade na origem. Isso resulta em um problema para o algoritmo de retropropagação, como veremos mais adiante.

Uma *função de ativação* que resolve o problema da diferenciabilidade é a função chamada de *Rectified Linear Unit* (ReLU). A Função ReLU comporta-se melhor do que outras funções diferenciáveis, como a *tanh* por exemplo, pois a derivada desta função diminui para valores maiores de $v$. Isso faz com que o gradiente seja próximo de zero nesses casos, tornando o treinamento da rede bastante lento. Por outro lado, a derivada da ReLU com relação a sua entrada é sempre 1 para valores positivos. A *função de Ativação* ReLU é bastante utilizada em redes neurais (como as MLPs e CNNs, os estudos de caso presentes nesta Tese).

A Equação 16 descreve o comportamento da *Função de Ativação* ReLU:

$$\phi(v) = \begin{cases} v & se \quad v \geq 0 \\ 0 & se \quad v < 0 \end{cases} \tag{16}$$

As camadas de saída das redes neurais utilizadas como estudo de caso neste trabalho, aplicam a *Função de Ativação Softmax*. Ela fornece a probabilidade de cada classe (ou saída) da rede neural. Isso permite que obtenham probabilidades úteis para problemas multiclasses (como os SCAs).

Em *Softmax*, a probabilidade de uma amostra particular com a entrada da rede $v$ pertencendo à $i - sima$ classe pode ser calculada com um termo de normalização no denominador, ou seja, a soma de todas as $M$ (número de classes) funções lineares:

$$p(y = i|v) = f(v) = \frac{e^{v_i}}{\sum_{j=1}^{M} e^{v_j}} \tag{17}$$

### 3.1.2 Perceptrons de Múltiplas Camadas

Nesta Seção, falaremos sobre redes neurais de múltiplas camadas alimentadas adiante (do inglês, *feedforward neural networks*). Neste tipo de redes, o sinal de entrada se propaga para a frente através da rede, camada por camada. Assim, essas redes são comumente chamadas de *perceptrons de múltiplas camadas* (do inglês, *Multi Layer Perceptron* -MLP).

As MLPs têm sido aplicadas com sucesso para resolver muitos problemas complexos, através de seu treinamento de forma supervisionada com um algoritmo conhecido como *algoritmo de retropropagação de erro* (do inglês, *error back-propagation algorithm*) (HAYKIN, 2001). Como podemos perceber, esse algoritmo baseia-se na *regra de aprendizagem por correção de erro* (revisada na Seção 3.2.2).

A aprendizagem por retropropagação de erro consiste em dois passos através das

camadas da rede: um passo para frente (*propagação*), e um passo para trás (*retropropagação*). No passo para a frente, os dados de entrada são aplicados às camadas de entrada da rede, e então propagados camada a camada até produzirem uma saída da rede. Durante a *propagação*, os pesos da rede são fixos. No passo para trás, os pesos são todos ajustados de acordo com uma regra de correção de erro. Neste caso, a resposta da rede obtida durante o passo de *propagação* é subtraída de uma resposta desejada (alvo) para produzir um sinal de erro. Este sinal de erro é então propagado para trás através da rede, contra a direção dos pesos (daí o nome de retropropagação de erro - *back-propagation error*. Os pesos são então ajustados para que a resposta da rede se aproxime da resposta desejada. O algoritmo de retropropagação de erro é também chamado na literatura simplesmente de algoritmo de retropropagação (do inglês, *back-propagation*). E o processo de aprendizagem realizado com o algoritmo, por sua vez, é chamado de aprendizagem por retropropagação.

MLPs, consistem na associação em camadas de diversos *percepetrons* (Figura 9). A Figura 10 ilustra uma estrutura de rede neural com três camadas:

- Camada de entrada: esta primeira camada da rede neural é definida com o número de neurônios equivalente aos dados de entrada (dados de treinamento).
- Camada oculta: esta camada localiza-se entre as camadas de entrada e saída e pode possuir qualquer número de neurônios.
- Camada de Saída: esta última camada possui o número de neurônios correspondente à tarefa definida para rede neural. Para problemas de classificação, o número de neurônios na camada de saída é equivalente ao número de possíveis classes associadas aos dados de treinamento. Para problemas de regressão linear, o número de neurônios da camada de saída é normalmente equivalente ao número de neurônios na camada de entrada. Em problemas de classificação, cada neurônio da camada de saída indica a probabilidade na qual um dado de entrada processado pertence a uma determinada classe.

Como mostra a Figura 10, todos os neurônios da camada de entrada são totalmente conectados a cada neurônio da camada seguinte, que no caso é a camada oculta (do inglês, *hidden layer*). Da mesma forma, a camada oculta é totalmente conectada aos neurônios da camada de saída. Como consequência, é comum encontrar na literatura a denominação de camadas em redes neurais como camadas totalmente conectadas (do inglês, *fully-connected layers*).

Quando define-se uma rede neural com apenas uma camada oculta, esta rede neural é considerada uma rede neural *rasa* (ou do inglês, *shallow neural networks*). Este tipo de modelo atende ao Teorema de Aproximação Universal o qual define que

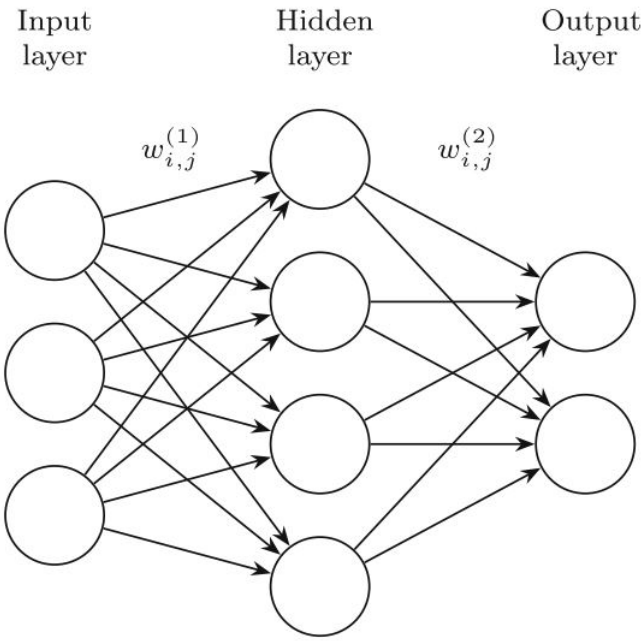


Figura 10 – MLP com três camadas. Fonte: (HETTWER; GEHRER; GÜNEYSU, 2020)

“uma rede neural com apenas uma camada oculta é capaz de aproximar qualquer função contínua” (HORNIK; STINCHCOMBE; WHITE, 1989). Como exemplo, podemos afirmar que redes neurais rasas são capazes de implementar funções *Booleanas*, o que é referido como Aproximação *Booleana*.

Nos últimas anos, passou-se a utilizar largamente redes neurais com múltiplas camadas ocultas. Estes modelos mais complexos apresentaram resultados superioroes aos que vinham sendo reportados com redes neurais rasas ou mesmo outros algoritmos de aprendizado supervisionado. Para estes novos e mais complexos modelos associou-se o termo *aprendizado profundo* ou, *deep learning*, DL. DL é um tipo particular de técnicas de ML bastante poderosas, que são capazes de representar a tarefa de aprendizado como uma hierarquia aninhada de conceitos, onde representações mais abstratas de conceitos são construídas a partir das mais simples. Recentemente, técnicas de DL têm ganho crescente interesse, motivado pelo fato de terem sucesso em resolver problemas centrais de inteligência artificial tais como reconhecimento de fala e classificação de imagens. Como se pode intuir, essas tarefas lidam com dados de alta dimensionalidade o que torna exponencialmente mais difícil fazer um classificador aprender a generalizar bem exemplos não vistos. Isto é um desafio conhecido como maldição da dimensionalidade (do inglês, *Curse of Dimensionality*) (GOODFELLOW; BENGIO; COURVILLE, 2016).

## 3.2 Aprendizado de uma Rede Neural

Esta Seção aborda alguns detalhes sobre a forma como as redes neurais aprendem a realizar uma determinada tarefa. Nos nossos estudos de caso, as redes são utilizadas para encontrar as relações entre os traços do consumo medidos e os dados processados, assim como as operações realizadas por um determinado dispositivo criptográfico, ou seja, a realização de um ataque por canais laterais.

### 3.2.1 Aprendizado Supervisionado

Os algoritmos de inteligência artificial podem realizar o aprendizado de uma tarefa sob dois paradigmas: o aprendizado não supervisionado, onde o algoritmo aprende somente a partir dos dados fornecidos, e o aprendizado supervisionado. No caso das redes neurais mais específicamente, o aprendizado é realizado de forma supervisionada (aprendizado supervisionado). Nesta categoria de Aprendizado Profundo o seu processamento é dividido em duas etapas. Em uma etapa inicial é realizado um treinamento do sistema, onde são fornecidos ao algoritmo um conjunto de dados com entradas do problema propriamente ditas e suas respectivas saídas esperadas. A partir destes dados, a rede neural ao computá-los busca aprender uma relação, ou função, existente entre entradas e saídas. Após isto, na segunda etapa, o objetivo é que para novas entradas, a rede neural seja capaz de predizer as saídas com base em seu aprendizado.

Algoritmos com estas características são chamados de supervisionados, pois seu processo de aprendizagem se assemelha a de um instrutor, ou supervisor, fornecendo as respostas corretas para as entradas dadas. Para a avaliação destes algoritmos, diversas métricas e técnicas são utilizadas. Como exemplo, podemos citar a validação cruzada em que medidas apropriadas são calculadas para os conjuntos de treinamento e teste dos dados e, posteriormente, comparadas e analisadas.

Uma tarefa na qual esses algoritmos são bastante requisitados, inclusive no contexto de SCAs é a classificação. Até mesmo problemas que podem parecer se enquadrar em problemas com solução por predição podem na verdade, ser problemas de classificação. Na classificação, os dados são divididos em determinadas (através de rótulos) classes e o algoritmo deve gerar um modelo que vincula novas entradas a uma dessas classes.

Geralmente no campo de SCAs, um exemplo de entrada de treino são vetores de números $(x_j)$ que representam os valores de medições de uma característica física do dispositivo criptográfico durante uma operação de encriptação ou decriptação, ou seja, os traços do consumo ou radiação eletromagnética. Em um processo usual de ML, como mostrado na Figura 11, esses traços são preprocessados em uma etapa inicial.

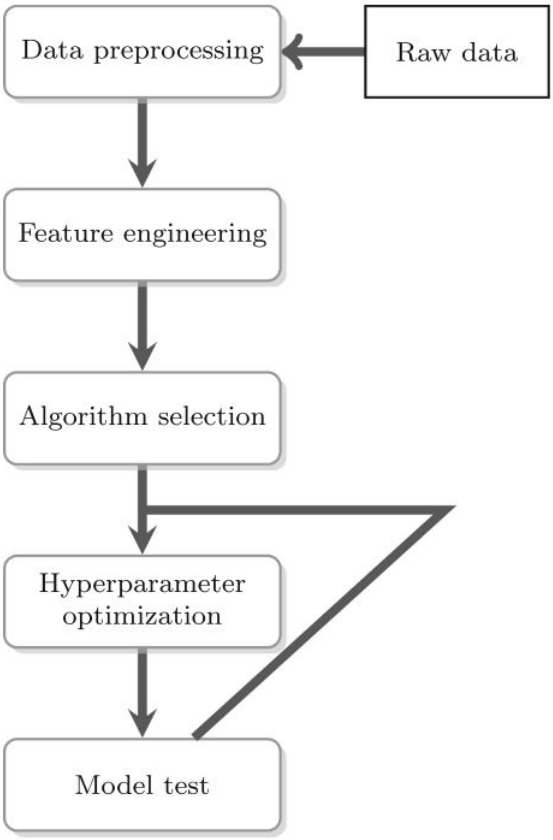


Figura 11 – Processo padrão de DL. Fonte: (HETTWER; GEHRER; GÜNEYSU, 2020)

Dessa forma, muitos algoritmos de DL exigem que os dados de entrada sejam normalizados (ou seja, reescalado para valores entre 0 e 1) ou padronizados (tendo média zero e variância unitária). Então, pontos de dados com maior conteúdo de informação são extraídos ou construídos, combinando ou criando dados adicionais. Depois, um algoritmo precisa ser selecionado para um dado problema de ML e seus hiperparâmetros têm que ser adaptados. Os hiperparâmetros controlam o comportamento do algoritmo e são previamente definidos. A partir disto, a performance do modelo otimizado é verificada com dados que não foram utilizados na etapa de treino. O utilizador do algoritmo divide o conjunto de dados definidos entre dados para treino e para teste. Geralmente, é adotada uma divisão de 60% dos dados para treino e 20% para teste. Sendo os 20% de dados restantes utilizados para otimização dos hiperparâmetros. Obviamente, essa divisão pode ser diferente da apresentada, e diferentes configurações podem ser testadas e experimentos realizados. Contudo, deve-se ter o cuidado de não ter um conjunto de teste pequeno demais, pois isso acarretaria em incerteza estatística em torno do erro médio de teste e pode dificultar a comparabilidade entre diferentes algoritmos de ML. Portanto, pode ser empregado um procedimento chamado *k-fold cross-validation*. Este procedimento consiste em dividir o conjunto de dados inteiro randomicamente em *k folds* (subconjuntos disjuntos), e usar iterativamente um deles como conjunto de teste, enquanto o resto dos dados são usados como conjunto

de treinamento. O erro médio em todas as trilhas é o erro de generalização estimado (ou seja, o erro de teste esperado ao aplicar novos dados ao modelo treinado). A validação cruzada também é frequentemente usada para estimar hiperparâmetros adequados. Sob esse paradigma, dois termos relativos à performance dos algoritmos de DL são *underfitting* e *overfitting*.

Quando o modelo não é capaz de obter um erro suficientemente baixo nos conjuntos de treino e teste dizemos que ocorreu *underfitting*. Entretanto, se o modelo teve uma boa performance sobre o conjunto de dados de treino, mas não no conjunto de teste, verifica-se uma situação de *overfitting*. Nos casos de *overfitting* é como se o algoritmo de DL ficasse "viciado" nos dados de treino, não "aprendendo" a regra que gera as saídas, mas sim "memorizando" os resultados para dadas entradas (GOODFELLOW; BENGIO; COURVILLE, 2016). Uma maneira de controlar o *underfitting* e o *overfitting* de um dado modelo, é alterando sua capacidade (isto é, a habilidade de se adequar a uma ampla gama de funções) incrementando ou decrementando o número de parâmetros. A seguir, veremos uma revisão de alguns dos principais algoritmos supervisionados e não-supervisionados aplicados a SCAs.

### 3.2.2 Aprendizagem por Correção de Erro

A principal propriedade de uma rede neural é sua capacidade de aprender a partir de seu ambiente e de melhorar seu desempenho através da aprendizagem. Uma rede neural aprende acerca de seu ambiente através de um processo iterativo de ajustes aplicados a seus pesos (HAYKIN, 2001).

Dessa forma, Haykin (2001) define o processo de aprendizagem de uma rede neural através da seguinte sequência de eventos:

1. A rede neural é estimulada pelo ambiente;
2. A rede neural sofre modificações nos seus pesos como resultado dessa estimulação;
3. A rede responde de uma maneira nova ao ambiente, devido às modificações ocorridas na sua estrutura interna.

Existem diferentes *algoritmos de aprendizagem*, que determinam as regras para a solução de um problema de aprendizagem. Nesta Tese, aborda-se apenas a *Aprendizagem por Correção de Erro*, pois este é o algoritmo utilizado nas redes neurais que compõem os estudos de caso desta Tese.

Por simplicidade, consideremos uma rede neural com uma camada de saída constituída de um único neurônio $k$. Suponhamos, que o neurônio $k$ receba um sinal $x(n)$ advindo das camadas ocultas anteriores a esta camada de saída. O sinal de saída do neurônio $k$ é representado por $y_k(n)$. Este sinal de saída é então comparado com a

saída desejada (ou saída-alvo) $d_k(n)$. Por consequência, isso gerará um sinal de erro $e_k(n)$. Esta situação hipotética é representada na Figura 12.

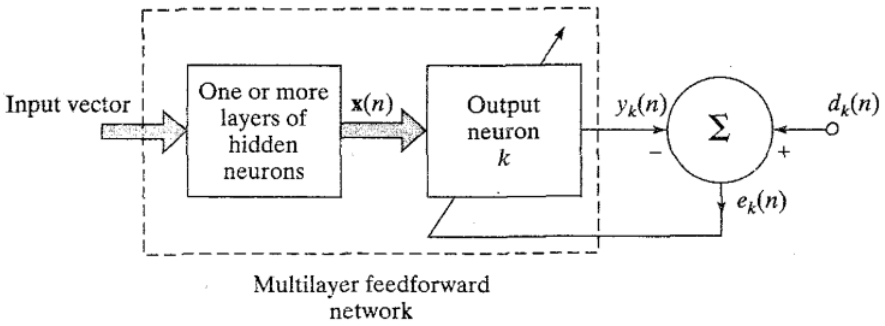


Figura 12 – Diagrama de blocos de uma rede neural, ressaltando o único neurônio da camada de saída. Fonte: (HAYKIN, 2001)

Disso, temos que o erro $e_k(n)$ é dado pela Equação 18:

$$e_k(n) = d_k(n) - y_k(n) \tag{18}$$

Nesta abordagem de aprendizagem, o erro $e_k(n)$ é utilizado como um mecanismo de controle, que aplica uma sequência de ajustes corretivos nos pesos do neurônio $k$, aproximando passo a passo, o sinal de saída $y_k(n)$ da saída desejada $d_k(n)$. Esse objetivo é alcançado minimizando-se a *função de custo* $E(n)$, calculada através da Equação 19:

$$E(n) = \frac{1}{2}e_k^2(n) \tag{19}$$

Como podemos ver pela Equação 19, $E(n)$ é o valor instantâneo da energia do erro $e_k(n)$. Os ajustes passo a passo dos pesos, mencionados anteriormente, continuam até o sistema alcançar um estado em que os pesos estejam estabilizados.

Agora, supondo-se que $w_{kj}(n)$ seja o valor do peso do neurônio $k$ excitado por um elemento de entrada $x_j(n)$, no tempo $n$. Sob este cenário, o ajuste $\Delta w_{kj}(n)$ aplicado ao peso $w_{kj}$ é calculado através da Equação 20:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \tag{20}$$

Onde $\eta$ é uma constante positiva que determina a *Taxa de Aprendizado* quando avança-se no processo de aprendizagem. A *Taxa de Aprendizagem* ($\eta$) é um dos parâmetros configuráveis no âmbito de redes neurais.

Por fim, o valor atualizado do peso $w_{kj}(n+1)$ pode ser computado através da Equação 21:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \tag{21}$$

Aqui, vale ressaltar que, conforme se vê na Figura 12, a *Aprendizagem por Correção de Erro* consiste de um *sistema realimentado em malha fechada*. E, como se sabe da teoria de controle, a estabilidade do sistema depende dos parâmetros que compõem o laço de realimentação. E como se pode ver, o único parâmetro que temos no único laço desta configuração é o que representa a *Taxa de Aprendizagem* ($\eta$). Assim, reforçamos que este parâmetro deve ser cuidadosamente escolhido, pois além de implicar diretamente na estabilidade da rede neural, ele dita a velocidade de aprendizado da rede. Se escolhermos valores muito baixos para $\eta$, a rede apresentará um custo muito alto para realizar o treinamento, em termos de tempo e esforço computacional. Entretanto, se escolhermos valores muito grandes para o referido parâmetro, podemos não alcançar o mínimo global da função de custo $E(n)$.

### 3.2.3 Equacionamento do Algoritmo de Retropropagação

Aqui, iremos avançar na descrição matemática das expressões que envolvem o algoritmo de retropropagação, com o intuito de entender alguns pontos importantes sobre esse algoritmo de correção de erro.

Primeiramente, é preciso saber que o algoritmo de retropropagação é um algoritmo que se baseia na aprendizagem por correção de erro (revisada na Seção 3.2.2). Portanto, o algoritmo de retropropagação aplica um ajuste ou correção $\Delta w_{kj}(n)$ a um peso genérico $w_{kj}$ de um neurônio $k$. Esse ajuste é proporcional à derivada parcial $\partial E(n)/\partial w_{kj}$ (gradiente). Esse gradiente pode ser representado, através da regra da cadeia, pela Equação 22:

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = \frac{\partial E(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial w_{ki}(n)} \tag{22}$$

Retomando-se a expressão que calcula a energia instantânea do erro para um neurônio $k$ ($\frac{1}{2}e_k^2(n)$), apresentada na Equação 19 desenvolvida na Seção 3.2.2, pode-se calcular a primeira derivada parcial da Equação 22, como se segue na Equação 23:

$$\frac{\partial E(n)}{\partial e_k(n)} = \frac{1}{2}\frac{\partial}{\partial e_k(n)}[e_k^2(n)] = \frac{2}{2}e_k(n) = e_k(n) \tag{23}$$

Como vimos, através da Equação 18, o erro $e_k(n)$ é dado pela diferença entre a saída desejada, e a saída real da rede $d_k(n) - y_k(n)$. Isto nos dá informações suficientes para calcular a segunda derivada parcial encontrada na Equação 22. Isto é feito de acordo com a Equação 24;

$$\frac{\partial e_k(n)}{\partial y_k(n)} = \frac{\partial}{\partial y_k(n)}[d_k(n) - y_k(n)] = -1 \tag{24}$$

Recorrendo à Equação 13, temos que a saída real da rede é dada por $y_k = \phi(b_k +$

$\sum_{j=1}^{m} w_{kj}(n)x_j(n))$. E ainda, a Equação 14 diz que $v = b_k + \sum_{j=1}^{m} w_{kj}(n)x_j(n)$. Dessa forma, a terceira derivada parcial da Equação 22 é resolvida, ou deixada de forma implícita, na forma apresentada pela Equação 25:

$$\frac{\partial y_k(n)}{\partial v_k(n)} = \frac{\partial}{\partial v_k(n)}[\phi(v_k(n))] = \phi_k'(v_k(n)) \tag{25}$$

O apóstrofe em $\phi'$ indica diferenciação em relação ao argumento. Finalmente, a quarta e última derivada parcial da Equação 22 é dada através da Equação 26:

$$\frac{\partial v_k(n)}{\partial w_{kj}(n)} = \frac{\partial}{\partial w_{kj}(n)}[b_k + \sum_{j=1}^{m} w_{kj}(n)x_j(n)] = x_k(n) \tag{26}$$

Portanto, o gradiente da Equação 22 é dado pela Equação 27:

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = -e_k(n)\phi'(v_k(n))x_k(n) \tag{27}$$

Assim, chegamos em uma expressão que calcula o valor da atualização dos pesos ($\Delta w_{kj}(n)$) descrita pela Equação 28:

$$\Delta w_{kj}(n) = -\eta e_k(n)\phi'(v_k(n))x_k(n) \tag{28}$$

Onde $\eta$ é a *Taxa de Aprendizagem*, vista anteriormente.

A Equação 28 nos mostra que, dentre outros fatores, a quantidade pela qual os pesos $w_{kj}$ são atualizados ($\Delta w_{kj}(n)$) depende da derivada parcial da *Função de Ativação* ($\phi$). É imprescindível para o funcionamento do algoritmo de retropropagação que a função de ativação dos neurônios constituintes da rede neural seja *derivável*. Isso fez com que o treinamento das redes fosse muito mais eficiente, ao passo que funções de ativação como a função degrau unitário fossem substituídas por funções como a ReLU, por exemplo.

### 3.2.4 Treinamento de uma Rede Neural

Revisados alguns detalhes sobre a forma de aprendizado das redes neurais, aqui busca-se apresentar de uma forma resumida, o processo de teinamento de uma rede neural. Assim, tem-se que o processo de aprendizado supervisionado implementa (ou treina) uma função de aproximação a partir de um conjunto de dados de entrada (ou dados de treinamento). Para cada elemento dentro do conjunto de dados de treinamento é associado um rótulo que define a classe a qual pertence este dado. Dessa forma, a rede neural processa diversas vezes os dados de entrada até satisfatoriamente classificar eles acordo com os rótulos esperados. Como esta Tese trata apenas de problemas de classificação, o processo de treinamento envolve as seguintes fases:

1. Todos os pesos da rede neural, $w_{kj}^{i}$, assim como os valores de viés, $b_k$, são inicializados aleatoriamente de acordo com uma distribuição estatística pre-definida (por exemplo, distribuição normal ou uniforme), sendo $k$ e $j$, a conexão entre o $k$-ésimo nó da camada $i$ e o $j$-ésimo nó da camada $i+1$.

2. Os dados de treinamento são previstos pela rede neural, resultando em rótulos $\hat{y}$ para cada elemento do conjunto de treinamento.

3. Tendo os rótulos verdadeiros, $y$, implementa-se uma função de erro (do inglês, *loss function*), que estabelece um valor para o erro do treinamento. Funções de erro comumente utilizadas em problemas de classificação são entropia cruzada categórica (do inglês, *Categorical Cross Entropy* - CCE) e erro quadrático médio (do inglês, *Mean Square Error* - MSE). Aqui, a função de erro é definida como $E(y, \hat{y})$.

4. A partir da função de erro, os pesos da rede neural são atualizados a partir do cálculo do gradiente da função de erro em relação à cada um dos pesos da rede. O mesmo é aplicado para os valores de viés. A atualização de cada peso considera uma taxa de aprendizado (do inglês, *learning rate*) que é empregada numa equação juntamente ao valor do gradiente de cada peso. Existem diversos algoritmos para atualização de pesos que leva em conta o valor do peso atual, $w_{kj}$, o gradiente da função de erro em relação a $w_{kj}$ e a taxa de aprendizado $\eta$. O mais básico dentre estes algoritmos é o chamado Gradiente Descendente Estocástico (do inglês, *Stochastic Gradient Descent ou SGD*):

$$w_{kj} = w_{kj} - \eta \frac{\partial E(y, \hat{y})}{\partial w_{kj}} \tag{29}$$

   Portanto, o tempo de treinamento, assim como memória necessária pelo sistema será proporcional ao tamanho da rede neural e à complexidade das funções de erro e atualização. Esta útlima é também referida como função otimizadora. É importante ressaltar que esse processo de atualização é comumente referido como retropropagação (do inglês, *backpropagation*).

Após a atualização de todos os pesos da rede neural e uma passada completa dos dados de treinamento, retoma-se a etapa 2. Essa passada completa do conjunto de dados é chamada de época (do inglês, *epoch)*. Aqui é importante ressaltar que o SGD opera através de pequenos lotes, o que torna o aprendizado mais eficiente e evita problemas de *overfitting*.

Além de redes neurais de múltiplas camadas, esta Tese também considera redes neurais por convolução, descritas a seguir.

### 3.2.5 Redes Neurais Convolucionais

Redes Neurais Convolucionais (do inglês, *Convolutional Neural Networks* – CNNs) (LECUN et al., 1999) constituem um modelo de aprendizado capaz de implementar extração de detalhes dos dados de entrada através de filtros em camadas de convolução.

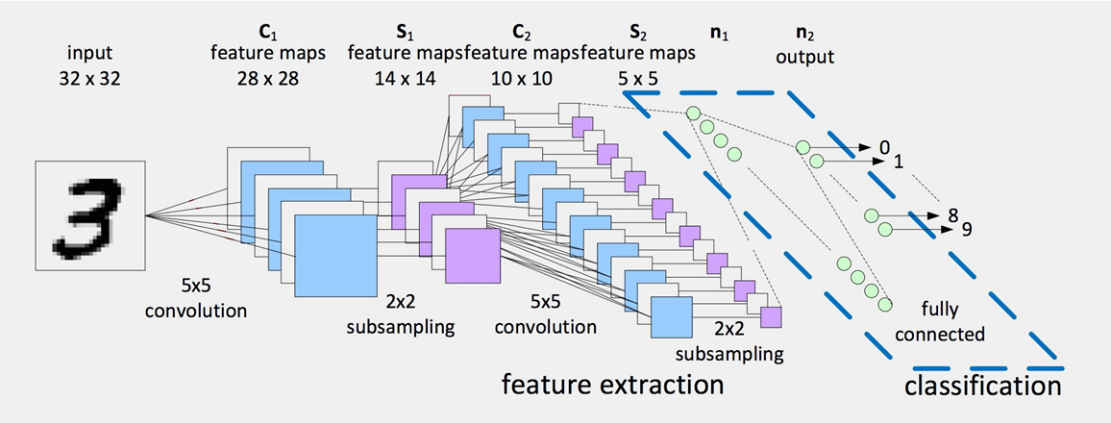


Figura 13 – Estrutura básica de uma rede neural convolucional. Fonte: (DEOTTE, 2018).

#### *3.2.5.1 Estrutura*

A Figura 13 apresenta a estrutura básica de CNN aplicada à classificação de imagens ou dados bidimensionais. Para ser uma CNN, a rede neural deve possuir ao menos as seguintes camadas:

- Camada convolucional (e camada de entrada): esta camada constitui um conjunto de filtros com dimensão equivalente ao dado de entrada (por exemplo, unidimensional, bidimensional ou tridimensional). A camada executa uma operação de convolução entre os filtros e os dados fornecidos à camada. Uma CNN pode possuir diversas camadas por convolução sequenciais. Os dados de entrada são, obviamente, conectados à primeira camada. A saída de cada camada por convolução é um mapa de detalhes (do inglês, *feature map* que representa detalhes extraídos dos dados de entrada a partir dos filtros de convolução. As operações de convolução caracterizam-se por um núcleo (ou *kernel*, que nada mais é que o tamanho do filtro) e passos (*stride*).

- Camadas densas: da mesma forma que redes neurais MLP, CNNs apresentam camadas densas que nada mais são que camadas ocultas totalmente conectadas. O objetivo destas camadas densas é efetuar a classificação a partir dos mapas de detalhes extraídos pelas camadas de convolução.

- Camada de saída: esta camada possui a mesma estrutura e atribuições da camada de saída de uma rede MLP.

Alternativamente, embora altamente recomendado, a saída de uma camada de convolução (ou seja, o mapa de detalhes) é conectada a camadas de agrupamento (ou *pooling)* que efetuam a redução do tamanho do mapa de detalhes. Camadas de agrupamento executam uma operação de *downsampling* nas suas entradas, a fim de reduzir o número de parâmetros e a complexidade computacional da rede. Esse processo é feito similarmente ao processo de convolução. Nesse caso, ao agrupamento são atribuídos um tamanho de núcleo e um passo, que passam por todo o dado de entrada da camada. Na maioria das vezes considerando o valor máximo (operação chamada de *max-pooling*). Similarmente, o agrupamento também pode considerar a média (*avegare-pooling*). A Figura 14 ilustra um exemplo de processo de convolução seguido por uma operação de agrupamento. Conforme ilustrado na Figura, os filtros de convolução, assim como o núcleo do processo de agrupamento deslizam sobre o dado de entrada. Aqui é também importante ressaltar que, caso as dimensões do núcleo não se ajustem perfeitamente às dimensões do dado de entrada, um processo de *padding* é automaticamente efetuado, normalmente inserindo-se zeros aos valores faltantes.

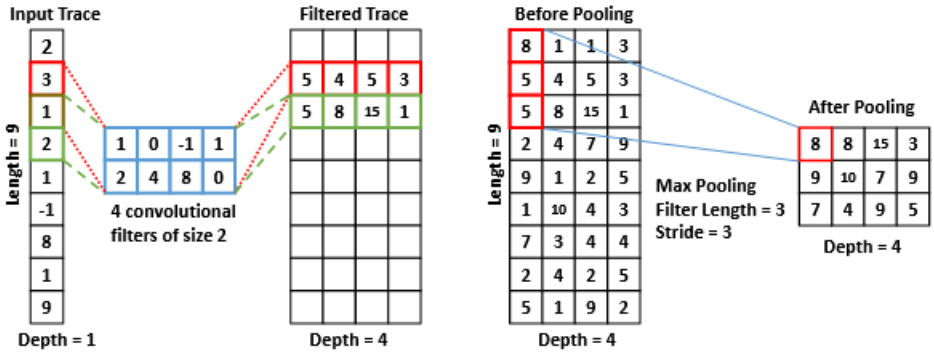


Figura 14 – Convolução e agrupamento em CNNs. Fonte: (CAGLI; DUMAS; PROUFF, 2017)

### *3.2.5.2 Treinamento*

O processo de treinamento de uma CNN segue as mesmas etapas e princípios observados para redes neurais MLPs. O ponto a ressaltar aqui é que cada elemento dentro de um filtro por convolução é considerado como sendo um neurônio da rede neural e portanto atualizado pelo algoritmo de otimização durante o processo de retropropagação.

Uma caracteística das CNNs bastante interessante para SCAs é que esse tipo de rede neural é capaz de extrair as características dos dados independentemente da posição destas características nas amostras do conjunto de dados.

## 3.3 Considerações sobre o Capítulo

Neste Capítulo, foram apresentadas definições sobre aprendizado profundo. Nesta revisão, foram abordadas as principais características das redes neurais artificiais aplicadas a SCAs. A revisão sobre tais assuntos partiu do elemento básico (o neurônio artificial) até a formação das arquiteturas MLPs e CNNs. Neste contexto, foram abordados tópicos de modo a fornecer um embasamento ao leitor sobre os algoritmos de DL utilizados como estudo de caso nesta Tese.

Após revisados estes conceitos é possível ter um conhecimento básico para melhor compreensão da utilização de redes neurais aplicadas a SCAs em diferentes cenários e ainda em arquiteturas com contramedidas. A seguir, o Capítulo 4 apresenta a revisão de trabalhos relacionados ao tema que motiva o desenvolvimento desta Tese.

# 4 TRABALHOS RELACIONADOS

Este Capítulo apresenta uma revisão da literatura sobre trabalhos que unem a aplicação de técnicas de DL para realizar ataques a canais laterais. Para isto, foi realizada uma Revisão Sistemática da Literatura (RSL), detalhada na Seção 4.1. A partir dessa RSL são encontrados diversos trabalhos que relacionam o uso de inteligência artificial com SCA sob diferentes paradigmas.

Na literatura é possível encontrar trabalhos com diferentes propósitos, como por exemplo, Wang et al. (2020), que busca meios de realizar ataques SCA em tempo real em dispositivos criptográficos usando a metodologia SCARF (do inglês, *Detecting Side Channel Attacks at Real-Time using Low-level Hardware Features*). Além disso, existe uma abordagem baseada em aprendizagem supervisionada para inferir aplicativos em execução na plataforma *Android* com base nas características extraídas de traços de radiação eletromagnética (do inglês, Electromagnetic - EM) e de estados de escalonamento de frequência e de tensão dinâmica (do inglês, *Dynamic Voltage Frequency Scaling States* – DVFS) de Chawla et al. (2019). Ainda, o trabalho de Cristiani; Lecomte; Maurine (2020) visa avaliar a quantidade de vazamento de informações em um dispositivo criptográfico através de algoritmos de aprendizado profundo e de um estimador conhecido como MI (do inglês, *Mutual Information*).

Como vemos nas Seções seguintes, pesquisadores aplicam os algoritmos de Aprendizado Profundo em diferentes cenários de ataques SCA. Estes cenários podem representar casos desde uma situação ideal de ataque com um algoritmo sem contramedidas até representar situações reais como ataques a diferentes algoritmos criptográficos, dispositivos diferentes, ambientes com ruído elétrico e dispositivos equipados com contramedidas como Inserção de Atrasos Aleatórios (do inglês, *Random Delay Insertion* - RDI) (YAO; ZHANG, 2012) e Sinal de Relógio Aleatório (do inglês, *Random Clock* - RC) (BOEY et al., 2010), entre outras.

Também foram resgatados da literatura, trabalhos que buscam realizar uma comparação de desempenho entre algoritmos de ML/DL e suas variações. Tais comparações são importantes, pois ressaltam características desses algoritmos, apontando cenários e aplicações mais adequadas para cada algoritmo ou grupos de algoritmos.

Entretando, a pesquisa aqui desenvolvida concentra maior esforço em trabalhos encontrados na RSL que procuram melhorar a eficiência de tais algoritmos quando aplicados ao contexto de SCAs. Por este motivo, tais estudos terão um nível de detalhamento maior dentro da revisão.

## 4.1 Revisão Sistemática da Literatura

Esta Seção apresenta uma Revisão Sistemática da Literatura (RSL), que explora o estado da arte sobre o tema da pesquisa, a fim de identificar, avaliar e interpretar os resultados de estudos relevantes disponíveis na literatura.

Os estudos individuais encontrados através RSL são chamados de estudos primários, enquanto a revisão em si dos trabalhos retornados na busca é chamada de estudo secundário.

Para a condução da seleção e análise dos estudos primários, foi utilizada a metodologia proposta por Kitchenham; Charters (2007). Essa metodologia divide o processo de revisão em três partes principais: Planejamento, Realização (também chamada de execução ou condução da revisão) e Documentação da Revisão.

Hoje em dia, temos algumas ferramentas que auxiliam no desenvolvimento de RSLs. Dentre elas, este trabalho utilizou o *Parsif.al* (PARSIFAL, 2018). Esta ferramenta foi escolhida, por estar disponível *on-line*, mantendo o progresso da pesquisa armazenado na nuvem. Outra característica interessante é a possibilidade de compartilhar revisões com outros autores, como por exemplo, o orientador do trabalho. O *Parsif.al* está baseado nos passos descritos em (KITCHENHAM; CHARTERS, 2007).

Antes de começar a revisão sistemática da literatura, foram buscados alguns artigos através do *Google Scholar* (SCHOLAR, 2021) dentre outros motores de busca, utilizando-se palavras-chave no contexto da pesquisa. Esses artigos, são chamados de artigos de controle e servem para guiar alguns dos parâmetros iniciais do planejamento da revisão, como por exemplo, as bases onde são realizadas as buscas de trabalhos. A partir dos artigos de controle, deu-se inicio a RSL com auxílio da ferramenta *Parsif.al*.

Uma dentre muitas informações estatísticas interessantes levantadas a partir da RSL são os algoritmos de Aprendizado Profundo utilizados nos trabalhos encontrados durante a RSL. Nota-se através da Tabela 2 que os algoritmos de Aprendizado Profundo mais utilizados pelos estudos selecionados são CNNs, que aparecem em 38,46% dos trabalhos, seguido de MLPs com 35,90% de aparições durante as pesquisas.

Também é importante saber quais foram as ferramentas ou *frameworks* de desenvolvimento mais utilizados. Pode-se ver através da Tabela 3, que os *frameworks* mais utilizados nesse escopo são a biblioteca *Keras* (CHOLLET, 2015) baseada na

Tabela 2 – Algoritmos de Aprendizado Profundo mais utilizados.

| Algoritmo | % |
|---|---|
| CNN | 38,46 |
| MLP | 35,90 |
| LSTM | 5,13 |
| AutoEnconder | 2,56 |
| RNN | 2,56 |
| ResNet | 2,56 |

Tabela 3 – Ferramentas mais utilizadas.

| *Framework* | % |
|---|---|
| *Keras* | 33,33 |
| *Matlab* | 16,67 |
| *Sci-Kit Learn* | 13,89 |
| *Weka* | 13,89 |
| *Python* | 11,11 |
| *C* | 2,78 |
| *kit SVM, LIBSVM* | 2,78 |
| *Scipy* | 2,78 |
| *TensorFlow* | 2,78 |

linguagem de programação *Python* com 33,33% do total de ferramentas informadas e o *Matlab* (MATLAB, 2010) com 16,67%.

Sabendo-se que existem *Datasets* públicos disponíveis na internet para uso de experimentos em SCAs, buscou-se identificar quais *Datasets* são mais utilizados dentre os trabalhos selecionados. Com isto, pudemos perceber, olhando para a Tabela 4, que 39,9% dos trabalhos encontrados utilizam traços próprios para seus experimentos. Também, os *Datasets* públicos DPA Contest v.4 (UNIVERSITY, 2015) e ASCAD (PROUFF et al., 2018) também são bastante utilizados com 19,70 e 15,15% respectivamente.

Tabela 4 – *Datasets* mais utilizados.

| *Dataset* | % |
|---|---|
| Traços Próprios | 39,39 |
| *DPA Contest v.4* | 19,70 |
| *ASCAD* | 15,15 |
| *DPA Contest v.2* | 9,09 |
| AES_RD | 7,58 |
| AES_HD | 4,55 |
| *TeSCASE* | 3,03 |
| *Grizzly* | 1,52 |
| *Jit* | 1,52 |

É sabido que implementações em *hardware* dos algoritmos criptográficos são mais difíceis de atacar, visto que as operações realizadas acontecem paralelamente devido à natureza desse tipo de implementação. Talvez por isso, 41 dos trabalhos que mencionam o tipo de implementação atacada são baseadas em *software*, enquanto que apenas 28 são realizadas em *hardware*.

Quanto às métricas de avaliação dos experimentos, a mais encontrada é *Guessing Entropy* (GE), que aparece em 24 trabalhos. A acurácia é a segunda métrica que mais aparece como métrica de avaliação dos experimentos dentre os trabalhos selecionados, com 9 ocorrências. Isso é surpreendente, pois acurácia não é uma métrica adequada ao contexto dos *Side Channel Attacks* (PU et al., 2023). Em seguida, como outra métrica das mais encontradas nos trabalhos revisados, temos a *Success Rate* (SR) que aparece em 6 trabalhos.

Com base nessas informações, foi possível traçar uma tendência comum entre os trabalhos que aplicam técnicas de Aprendizado Profundo a *Side Channel Attacks*. Abaixo, segue um resumo que mostra tal tendência entre os estudos encontrados:

- Algoritmos de Aprendizado Profundo mais utilizados: CNN e MLP;
- *Frameworks* mais utilizados: *Keras* e *Matlab*;
- *Datasets* mais utilizados: Próprio, *DPA Contest v.4* e *ASCAD*;
- A maioria dos trabalhos atacam implementações em *software* do algoritmo criptográfico;
- As métricas de avaliação mais utilizadas são: *Guessing Entropy* (GE), acurácia e *Success Rate* (SR).

## 4.2 Artigos de Controle

Nesta Seção, são apresentados trabalhos buscados numa fase inicial pré-RSL. Mesmo se tratando de artigos base, estes artigos trazem estudos importantes sobre o tema de pesquisa aqui abordado. Portanto, serão analisados a seguir.

Inicialmente, tem-se o trabalho apresentado por Ramezanpour; Ampadu; Diehl (2020), no qual os autores mesclam o uso de algoritmos de DL com algoritmos de ML. Neste trabalho é utilizado *Long Short- Term Memory* (LSTM) auto-enconder para extrair características dos traços do consumo. Essas características são utilizadas para identificar o modelo de consumo, através de MLPs. A partir disso, as características encontradas são clusterizadas e é utilizada uma abordagem não supervisionada para encontrar a chave correta. Com isso, os autores dizem melhorar a eficiência dos ataques em 10x.

Muitos dos trabalhos encontrados na literatura baseiam-se em criar modelos de consumo utilizando técnicas de ML/DL para aplicação em SCA. Os primeiros a fazer isto foram Yang et al. (2012), motivados pelo fato de que Redes Neurais (do inglês, *Neural Networks* – NNs) são capazes de capturar características não-lineares dos traços do consumo sem restrições específicas. A eficácia dos ataques foi relatada com uma série de experimentos, incluindo diferentes níveis de ruído e diferentes métricas de avaliação. Portanto, percebe-se que redes neurais são capazes de realizar ataques a canais laterais sem a necessidade de etapas de pré-processamento dos traços. Dessa forma, essa etapa pode ser suprimida do fluxo de ataques sem maiores problemas.

Em Martinasek; Malina; Trasy (2015) os autores sugerem utilizar *Momentum* ou *Conjugate Gradient Backpropagation* para evitar problemas com o algoritmo de *back-propagation* nas redes MLP. Os resultados obtidos confirmaram que o MLP é muito mais eficaz na criação de *profiling* de ataques de consumo de energia em termos de pequeno número de traços de energia e pontos interessantes.

No trabalho de Carlet et al. (2016), os autores se propõem a dar continuidade à linha de pesquisa baseada na aplicação do aprendizado de máquina ao SCA, aplicando técnicas de *profiling* mais sofisticadas com base no aprendizado profundo. Seus resultados experimentais confirmam as vantagens esmagadoras dos novos ataques resultantes quando têm como alvo implementações criptográficas desprotegidas e protegidas.

Maghrebi; Portigliatti; Prouff (2016) utilizaram Redes Neurais Convolucionais (CNNs) aplicadas à área de SCAs. Além disso, os autores investigaram o uso de auto-encoders empilhados, bem como *Long and Short Term Memory (LSTM)*. O uso de técnicas de DL foi motivado pelo fato destas incorporarem intrinsecamente mecanismos de extração de características. Desse modo, ao contrário da maioria dos classificadores ML padrão, NNs profundas podem aprender com o conjunto de entradas brutas, pois são capazes de identificar os pontos de maior vazamento de informação. Como estudo de caso, os autores realizaram uma série de experimentos com técnicas de DL, classificadores ML clássicos (SVM, *Random Forest* - RF, *Multi-Layer Perceptions* - MLP) e *TA* em implementações em *hardware* e *software* do algoritmo AES protegido e desprotegido. Resumindo os resultados, pode-se dizer que os métodos de DL na maioria das vezes superam outras técnicas de ataque. Interessante o fato de que a combinação de uma etapa de pré-processamento com Análise de Componente Principal (do inglês, *Principal Component Analysis* - PCA) para redução da dimensionalidade dos dados de entrada, com MLP não melhorou a performance do ataque. Este trabalho apresenta uma boa gama de experimentos em diferentes cenários, mostrando que DL se sai melhor do que outras técnicas. Aqui, mais uma vez notamos que etapas de pré-processamento dos traços são desnecessárias quando

aplicamos redes neurais no contexto de SCAs.

Lerman et al. (2018) faz um estudo comparativo entre TA e ataques baseados em ML. Os autores concluem que quando a etapa de *profiling* se aproxima da perfeição, TA é melhor em relação a ML, porém em casos reais, onde erros estão presentes no *profiling* e em casos onde eventualmente poucos traços são disponíveis, ML se sai melhor. Foi observado que RF se sai melhor quando é aumentada a quantidade de informação inútil nos traços, pois o método eventualmente detecta pontos de interesse dos traços.

Prouff et al. (2018) fazem um comparativo entre técnicas de Aprendizado Profundo (NN, CNN) com TAs. Neste trabalho os autores dizem mostrar a hiper-parametrização das redes, diferentemente de outros trabalhos. Para reduzir a dimensionalidade, os autores utilizaram PCA. Para os experimentos, foram utilizados traços do banco de dados ASCAD.

Em Timon (2018), é proposto o uso de poderosas técnicas de Aprendizado Profundo e Aprendizado de Máquina para ataques *non-profiled*. No artigo, é mostrado que é possível explorar a propriedade de *translation-invariance* das CNNs contra traços desalinhados e usar técnicas de *Data Augmentation* também durante os *non-profiled* SCAs. Foi comprovado através de experimentos que *Data Augmentation* melhora os resultados obtidos para ataques de potência baseados em CNN (do inglês, *CNN-based Deep Learning Power Análysis* - CNN-DLPA) e ataques de potência baseados em MLP (do inglês, *MLP-based Deep Learning Power Analysis* - MLP-DLPA). Experimentos com traços desalinhados mostram que CPA e MLP-DLPA sem DA falharam. Enquanto MLP-DLPA com DA e CNN-DLPA com e sem DA obtiveram sucesso. Portanto, através dos resultados os autores mostram que em alguns casos, o método porposto supera alguns *Non-Profiled Attacks* clássicos como CPA. Também estudaram a eficiência deste ataque contra implementações com contramedidas de alta ordem e mostram que este método é capaz de quebrar implementações com proteções de primeira-ordem com um número razoável de traços sem pré-processamento. Entretanto, lacunas quanto a outras métricas, tais como *Guessing Entropy*, ficam abertas para próximos estudos. O próprio autore explora outra métrica chamada de Análise de Sensibilidade (do inglês, *Sensitivity Analysis* - SA) em (TIMON, 2019). Apesar de ser uma abordagem muito intrigante, seu custo computacional seria elevadíssimo, o que não é discutido no trabalho apresentado.

## 4.3 Análise dos Trabalhos Relacionados

Nas Seções seguintes, são descritos os trabalhos encontrados na literatura através da RSL realizada nesta tese (e descrita na Seção 4.1). Estes trabalhos abrangem a aplicação de diferentes métodos baseados em aprendizado profundo, no contexto

dos SCAs. Os trabalhos apresentados, foram divididos em três grupos: Avaliação do ataque baseado em Inteligência Artificial sob diferentes cenários, comparação entre métodos de ataque, e melhoria da eficiência do ataque.

### 4.3.1 Avaliação do Ataque baseado em Inteligência Artificial sob diferentes cenários

É sabido que SCAs podem ser implementados sob condições adversas, como por exemplo, diferentes níveis de ruído ambiente, implementação em *software* ou em *hardware* do algoritmo criptográfico, diferentes tipos de dispositivos atacados, entre outras variações. Dessa forma, atacantes se deparam com diferentes cenários, os quais também são explorados na literatura.

Ataques realizados em laboratórios de pesquisa geralmente são realizados sob condições controladas, como por exemplo, a forma como são capturadas as informações vazadas do dispositivo criptográfico sob ataque. Entretanto, situações ou configurações diferentes podem ocorrer em ataques reais. Por isso, é importante testar diferentes possibilidades como foi feito em (WANG; WANG; DUBROVA, 2020) onde os autores realizam ataques com EM de campo distante, capturados de cinco dispositivos *Bluetooh* diferentes em cinco distâncias diferentes, utilizando aprendizado profundo. Com isto, os autores afirmam que é possível recuperar a chave com menos de 10000 traços capturados em um ambiente de escritório a 15m de distância do alvo mesmo se a medida para cada encriptação é tomada somente uma vez. É mencionado que TAs anteriores precisavam de múltiplas repetições para a mesma encriptação.

Ainda considerando-se a variedade de cenários que podem estar presentes frente aos SCAs, Weissbart; Picek; Batina (2019) abordaram várias técnicas de aprendizado de máquina a fim de montar um ataque de análise de potência em um algoritmo chamado EdDSA (do inglês, *Edwards-curve Digital Signature Algorithm*) usando a curva 25519. Para os experimentos, os autores testaram os algoritmos de RF, SVM e CNN. Para questões de comparação, foi também executado um experimento com TA clássico e covariância combinada. Os resultados mostram que todas as técnicas consideradas são opções viáveis e poderosas. Dentre elas, as redes neurais convolucionais (CNNs) são especialmente eficazes, pois foi possível quebrar a implementação com apenas uma única medição na fase de ataque, e requer menos de 500 traços na fase de treinamento. É interessante observar, que a aplicação de PCA piorou os resultados de um modo geral. Pode-se observar que ao aplicar-se PCA para obter 10 POIs, os resultados são mais estáveis. Porém piores do que não aplicar a redução de dimensionalidade. Isto nos faz perceber que nem sempre técnicas de redução de dimensionalidade ou pré-processamento se traduzem em resultados melhores.

Heuser et al., trazem uma questão interessante quanto a robustez de algoritmos de cifras leves em (HEUSER et al., 2017). Esse estudo busca descobrir se algoritmos

de cifras leves são mais vulneráveis a ataques de canais laterais. Para tanto, foram consideradas várias métricas de avaliação no contexto dos SCAs. Os algoritmos testados nesse trabalho são o *KLEIN, PRESENT, PRIDE, RECTANGLE, Mysterion, AES, Zorro* e *Robin*, implementados em *software*. Sendo esses algoritmos testados com ataques *profiled* e *non-profiled* baseados em *Naive Bayes*, C4.5 (um algoritmo da família DT) e MLP. Os resultados mostram que a diferença entre AES e cifras leves é menor do que o esperado. Curiosamente, para ataques *non-profiled*, as *SBOXs* de 8 *bits* de AES, *Zorro* e *Robin* têm um desempenho semelhante, enquanto que para as *SBOXs* de 4 *bits* tem-se uma classificação clara, com a *SBOX* do *Mysterion* sendo a mais fraca para atacar, e a *SBOX* do *KLEIN*, a mais difícil. Para ataques *profiled*, foram analisadas várias técnicas de aprendizado de máquina para *PRESENT* e *AES*. Neste cenário, os resultados são aplicáveis a todas as *SBOXs* de 4 e 8 *bits*. Os resultados mostram que atacar *PRESENT* é um pouco mais fácil do que atacar AES. Ainda assim, essa diferença não é tão aparente quanto se poderia imaginar.

Sob esse paradigma, Zhang et al. (2020) propõem um novo mecanismo chamado de Análise de Potência baseada em Frequência e Aprendizado (do inglês, *Frequency and Learning based Power Analysis* (FL-PA)) , que é capaz de enfrentar desafios causados por variações de dispositivos (desde dispositivos homogêneos a heterogêneos). De acordo com os autores, pela primeira vez os traços de energia coletados de seus próprios dispositivos PIC podem ser utilizados para atacar com sucesso o conjunto de dados público DPA Contest v4, que é baseado em um microcontrolador AVR totalmente diferente. A ideia básica de Zhang et al. (2020) é combinar aprendizado profundo com análise no domínio da frequência. Os resultados mostram que tanto o TA quanto o DL-PA encontram uma grande dificuldade ao atacar dispositivos heterogêneos. Segundo os autores, a falha dos TA em dispositivos homo/heterogêneos é atribuído à região de seleção dos POIs e ciclos de instrução. Ambas diretamente relacionadas com o relógio no domínio do tempo. Já, aprendizado de máquina tem uma maior capacidade de generalização do que essas análises estatísticas. Portanto, para melhorar DL-PA e mitigar essas questões do domínio do tempo, a transformada de *Fourier* (FFT) é aplicada a todos os traços desses *crossed devices*.

Também com o intuito de buscar uma generalização para ataques sobre dispositivos diferentes, Das et al. (2019) aplicam aprendizado profundo em SCA sobre *cross devices*, e dizem chegar a uma acurácia maior do que 99,9% mesmo na presença de variações *inter-device* significativamente altas. Os autores mencionam que o ataque *X-DeepSCA* quebram a criptografia de diferentes dispositivos alvo em segundos, em comparação com alguns minutos para um ataque de análise de energia correlacional (CPA). Aumentando assim, a ameaça para dispositivos embarcados. De acordo com os autores, mesmo para cenários de SNR baixo, o ataque *X-DeepSCA* necessita de aproximadamente 10 vezes menos traços em comparação com um CPA tradicional

para realizar o ataque.

Em Xu et al. (2019), os autores demonstram um ataque prático contra um módulo de comunicação *LoRa WAN*. Este ataque usa SCA baseado em aprendizado profundo para recuperar a chave usada na encriptação dos dados do *payload*. Segundo os experimentos, com menos de 100 traços, a CNN treinada é capaz de recuperar a chave completa de um AES. Sendo que a maioria dos *bytes* da chave pode ser recuperada com menos de 20 traços. Através dos experimentos, foi observado que no melhor caso, apenas 1 traço é necessário para identificar a estimativa correta como a classificação mais alta, ou seja, a classe que contém a maior probabilidade, e consequentemente o maior *ranking* da subchave; enquanto no pior caso, 73 traços são necessários para reconhecer de forma estável a suposição correta como classificação 0 do *ranking*. Com 20 traços, 14 de todos os 16 *bytes* alcançam a classificação 0 (*byte* 3 classificações 6 e *byte* 7 classificações 2). Para questões de comparação, foi implementado o ataque proposto usando TA. O TA convencional é otimizado com PCA e LDA para extrair-se os POIs. Foi descoberto que o melhor *ranking* é alcançado com 40 POIs para PCA e 8 POIs para LDA. Os resultados do ataque são avaliados calculando a classificação média alcançada ao atacar todos os 16 *bytes* da chave usando um número diferente de traços de alvo.

No estudo apresentado por Brisfors; Forsmark; Dubrova (2021), os autores demonstram um ataque aos USIMs (do inglês, *Universal Subscriber Identity Module*) com base no aprendizado profundo. É mostrado que uma CNN treinada em um USIM pode recuperar a chave de outro USIM usando no máximo 20 traços (quatro traços em média). Ataques CPA, anteriormente aplicados em cartões USIM, exigiam osciloscópios de alta qualidade para aquisição dos traços de consumo, uma quantidade muito maior de traços do cartão da vítima e habilidades de nível de especialista do invasor. Segundo os autores, agora o ataque pode ser montado com um orçamento de $ 1000 e habilidades básicas em análise de canal lateral.

Até mesmo dispositivos que intuitivamente supõe-se serem mais robustos como as GPUs (do inglês, *Graphic Processing Units*) também podem ser vulneráveis a ataques de canais laterais como mostrado em (MUKHERJEE, 2020). Nesta pesquisa, os autores apresentam o chamado GIPSim, que é uma estrutura que permite aos pesquisadores de segurança analisar o vazamento de canal lateral presente no contexto de um simulador orientado para a execução de GPU. Os autores mostram como os pesquisadores podem capturar estimativas de energia detalhadas ao executar programas CUDA em uma GPU da família *Kepler* e usar as informações para ofuscar o consumo de energia, ocultando a dependência do vazamento de energia com os dados processados. É mostrado como as técnicas tradicionais de ocultação e mascaramento podem ser aplicadas no contexto de uma GPU. Estas, por sua vez, reduzem a vulnerabilidade presente neste contexto. Também são apresentadas formas de po-

tencializar as técnicas de aprendizado de máquina usando redes neurais de memória de longo prazo para melhorar ainda mais a ofuscação. O objetivo deste trabalho, é projetar um sistema que possa impedir ataques de canal lateral baseados em energia. Assim, os autores dizem mostrar que é possível modelar a dissipação de energia dependente de dados, capturando a distância de *hamming* dos valores de dados usados durante a execução da criptografia AES. Sendo essa, uma abordagem já usada em ataques de canal lateral baseados em energia. GIPSim é um dos primeiros ambientes de simulação que pode ser usado para avaliar a resiliência do canal lateral de energia e ajudar a construir um acelerador mais seguro.

Uma observação feita por Kubota et al. (2019) é que a maioria das implementações de algoritmos criptográficos atacadas através de DL-SCA é realizada em nível de *software*. Esse estudo destaca que implementações em *hardware* são mais difíceis de atacar devido à característica de execução paralela desse tipo de construção. Portanto, é proposta uma investigação do uso de DL-SCA contra implementações em *hardware* do AES, mostrando que é possível revelar a chave secreta aplicando uma nova técnica chamada *mixed model dataset based on round-round XORed value*. Também foram comparadas a performance e as características de DL-SCA com métodos de análise convencionais tais como CPA e TAs convencionais. Antes de realizar a comparação, os autores lembram que para TA clássico é necessário pré-processamento dos traços, como por exemplo, realinhamento. O que demanda tempo e ajustes manuais por conta do atacante, ao passo que em DL-SCA esse passo pode ser pulado. Os resultados mostram que os valores reais da função alvo sendo utilizadas para gerar o modelo superam a utilização do modelo HD utilizando DL-SCA. Outro experimento consistiu em atacar um AES com a contramedida RSM (do inglês, *Rotate Shift Masking*). Para o dispositivo utilizado, o DL-SCA não conseguiu revelar os *bytes* 6 e 10 da chave. E também, os *bytes* 5, 12 e 14 foram ranqueados dentro do quinto lugar como os principais candidatos corretos. HD-CPA, HW-CPA e TAs não revelaram nenhum *byte* da chave e nenhuma chave foi classificada em quinto lugar.

Um dos trabalhos revisados através da RSL aqui realizada, busca avaliar a eficiência de SCA baseado em aprendizado profundo sobre dispositivos dotados de contramedidas que não são encontradas na literatura (MAGHREBI, 2019). Primeiramente, é testado o esquema de *masking* SSS (do inglês, *Shamir Secret Sharing*). Após, foi conduzida uma avaliação da segurança de duas contramedidas de *side channel attacks*: embaralhamento (do inglês, *shuffling*) e 1-entre-N (do inglês, *1-amongst-N*) contra DL-SCA. Experimentos simulados e práticos provam que, como esperado, estas contramedidas são também vulneráveis a estes *profiling attacks*. Os resultados demonstram que DL-SCA são muito eficientes para quebrar a implementação de SSS. Mais interessante, a arquitetura LSTM supera a CNN e a MLP. Esta observação destaca que a LSTM é uma rede neural interessante a ser considerada em uma avaliação de ca-

nal lateral, especialmente quando o vazamento de dados sensíveis é a combinação de vários compartilhamentos que vazam em diferentes amostras de tempo dos traços (como é o caso típico de SSS). Como esperado, os resultados obtidos para DL-SCA com os traços reais são próximos aos obtidos através de simulação. Além disso, o DL-SCA supera o TA de ordem superior - HOTA (do inglês, *High Order Template Attack*). Os resultados demonstraram por meio de diversas simulações e experimentos práticos com HOTA, que a escolha dos pontos públicos no esquema SSS tem impacto na força da contramedida. Considerando esta propriedade, o LSTM é o melhor modelo seguido pela MLP e pela CNN_2_LAYERS. Do ponto de vista do adversário, a arquitetura LSTM em particular (e as redes neurais de dependência de tempo em geral) é muito adequada para quebrar uma implementação de SSS. O DL-SCA (independentemente da arquitetura DL usada) neutraliza a contramedida de embaralhamento. Sendo, a eficiência dos modelos DL usados bastante semelhante. No entanto, através dos resultados práticos (implementação da *Chipwhisperer*) pode-se ver que é mais interessante considerar as redes CNN quando o embaralhamento está envolvido como proteção.

Também relacionado ao ataque baseado em aprendizado profundo em dispositivos protegidos com contramedidas, os autores em (ALIPOUR et al., 2020) visam aplicar *non-profiled* DL-SCA contra o AES dotado da contramedida baseada em ocultação (do inglês, *hiding*), na qual é utilizada geração de ruído correlacionado. É apontado pelos resultados que contramedidas como mascaramento oferecem alta proteção contra CPA, entretanto DDLA (do inglês, *Differential Attack Deep Learning Analysis*) consegue quebrar essa contramedida. Os experimentos mostram que uma contramedida baseada em ocultação pode fornecer maior proteção contra ataques de canal lateral de aprendizado profundo *non-profiled*. Portanto, para implementar contramedidas que sejam resilientes contra ataques DDLA modernos sem criação de perfil, pode ser necessário usar metodologias que atrapalhem o procedimento de treinamento. Uma perspectiva é investigar mais a fundo se existem perturbações que podem perturbar ainda mais o procedimento de treinamento de ataques DDLA. Além disso, os autores planejam investigar contramedidas que podem proteger igualmente contra ataques DDLA e CPA.

O estudo de **?** aplica Aprendizado de Máquina para SCA em traços desalinhados. Esse trabalho combina CNNs com *Data Augmentation*, que trata-se de uma técnica para gerar novos exemplares de dados de treinamento a fim de aumentar a generalidade do modelo, simulando tanto o efeito de *clock jitter* (chamadas de *add-remove deformations*) quanto inserção de atrasos aleatórios (chamadas de *shifting feformations*). Nesse artigo os autores utilizam uma técnica chamada *Data Augmentation*, que consiste em aumentar o *dataset* inserindo amostras artificialmente modificadas, como por exemplo utilizando amostras distorcidas, deslocadas, etc., para evitar o pro-

blema de *Overfitting*. As contramedidas aplicadas nos exemplos foram RDI em um microprocessador ATMega328P. Em outro experimento os autores utilizam traços resultantes de operações sendo realizadas repetidas vezes. Com os resultados concluem que mesmo que a CNN transforme informações espaciais (ou temporais) em características discriminativas abstratas, ela ainda mantém uma noção de ordenação. Num terceiro experimento os autores simulam *clock jitters*. Com esses experimentos, concluem que CNN é robusta com respeito ao efeito de *jitter*. A seleção de POI e o realinhamento na fase de treino são efetivos. Por último, os autores executam um experimento com um *smart card* (implementado numa tecnologia de 90nm). Nesse *hardware* existe uma contramedida que emprega um forte *clock jitter*. Através dos resultados, os autores verificaram que CNNs são eficazes mesmo em casos onde realinhamento não se aplicaria. Assim, se o realinhamento falhar, os TA se sairão mal, enquanto os CNNs saem bem. Portanto, os autores concluem que CNNs com DA (para evitar o problema de *overfitting*) mostraram-se eficientes aplicados a traços com diferentes tipos de desalinhamentos.

### 4.3.2 Comparação entre Métodos de Ataque

A literatura contém também trabalhos que aplicam algoritmos de aprendizado profundo em SCAs. Muitos desses trabalhos buscam realizar uma comparação de desempenho entre tais algoritmos e suas variações. Tais comparações são importantes, pois ressaltam características desses algoritmos, apontando cenários e aplicações mais adequadas para cada algoritmo ou grupos de algoritmos.

Lerman; Martinasek; Markowitch (2017) fazem uma comparação entre diferentes algoritmos de aprendizado de máquina e aprendizado profundo, com o intuito de verificar quais algoritmos são mais robustos em ambientes ruidosos. Isto é uma característica bem comum em diferentes cenários de ataque quando da aquisição dos traços, sejam eles de radiação eletromagnética ou do consumo. Os resultados destacam que TA representa os melhores modelos quando: (*i*) não há (ou há baixa) variabilidade no conjunto de perfil e no conjunto de ataque, e (*ii*) o nível de ruído varia entre os vazamentos. Os autores observam que no geral, o TA clássico oferece o menor sucesso nos ataques. No entanto, o grande destaque em ataques baseados em aprendizado de máquina encontra-se: (*i*) quando o número de erros (ou seja, o número de vazamentos incorretamente associados a um valor alvo) no conjunto de perfil aumenta, (*ii*) quando os vazamentos estão desalinhados nos conjuntos de perfil e/ou ataque, e (*iii*) quando os vazamentos do conjunto de perfil e do conjunto de ataque diferem de um *offset DC* alto. O estudo apresentado traz comparações bastante interessantes, contudo como os autores mencionaram, experimentos de algoritmos de aprendizado profundo ficaram para trabalhos futuros. Seria interessante ver como esses algoritmos reagiriam frente a traços desalinhados. Além disso, a quantidade de amostras

deslocadas nos desalinhamentos dos traços utilizados nos experimentos não ficaram claros.

Sabe-se que técnicas de aprendizado profundo são ferramentas muito poderosas para resolver inúmeros problemas, dentre os quais estão os SCAs. Contudo, muitos parâmetros precisam ser configurados. Isto se traduz em uma tarefa difícil, além de gerar inúmeras possibilidades para serem testadas. Sob esse aspecto, Carlet et al. (2016) são os primeiros a aplicar diferentes algoritmos de aprendizado profundo ao contexto de SCA, destacando a capacidade de aprendizado profundo de construir um perfil preciso levando a um ataque de recuperação de chave de canal lateral eficiente e bem-sucedido. Os experimentos mostram que os ataques baseados em aprendizado profundo são mais eficientes do que os ataques baseados em aprendizado de máquina e TA quando direcionados a implementações criptográficas desprotegidas ou mascaradas.

Em (ROBISSOUT et al., 2021) os autores propõem uma métrica de avaliação online dedicada ao contexto da análise de canal lateral. Esta métrica pode ser usada para realizar a parada precoce em redes neurais convolucionais existentes encontradas na literatura. Segundo os autores, esta métrica compara o desempenho de uma rede no conjunto de treinamento e no conjunto de validação para detectar *underfitting* e *overfitting*. Consequentemente, pode-se melhorar o desempenho das redes ao encontrar sua melhor época de treinamento e, assim, reduzir o número de traços usados em 30%. O tempo de treinamento também é reduzido para a maioria das redes consideradas. Os experimentos mostram que a parada precoce do treinamento permitiu uma redução de 31% do número de traços necessários para atingir uma taxa de sucesso de 90% e reduziu o tempo de treinamento em 30%.

### 4.3.3 Melhoria da Eficiência do Ataque

Nesta Seção, são apresentados trabalhos que tem como objetivo melhorar a eficiência de SCA baseado em inteligência artificial, seja através de ajuste de hiperparâmetros, da combinação de informações de entrada para os algoritmos ou até mesmo propondo novas abordagens de ataque mais eficientes.

Neste sentido Mukhtar; Kong (2019) busca um melhor ajuste dos hiperparâmetros de algoritmos de aprendizado de máquina, através da análise de características temporais e do domínio da frequência dos traços de EM. Os resultados fornecem a análise comparativa das melhores escolhas e conduz à seleção dos parâmetros. Este trabalho não aplica técnicas de pré-processamento nos traços e o ataque é realizado *bit* a *bit* da chave secreta. Foi observado que o pré-processamento do PCA funcionou apenas na classificação de SVM, levando ao fato de que modelos como RF e MLP lidam bem com os dados de alta dimensão. A MLP mostrou uma precisão de classificação de mais de 90%, levando à constatação de que o uso de algoritmos de aprendizado

profundo mais complexos pode melhorar a precisão da classificação. Também foi concluído que vários parâmetros podem ser alterados para melhorar a acurácia. A discussão trazida pelos autores é bastante interessante, pois independentemente do tipo de algoritmo de inteligência artificial utilizado, o ajuste de parâmetros é uma das etapas mais complexas e importantes do processo do ataque. No entanto, muitos outros parâmetros existem nos algoritmos testados e não foram mencionados. A métrica utilizada é a acurácia, uma métrica que não é adequada para avaliar SCAs em ataques contra dispositivos protegidos ou com elevados níveis de ruídos. Os resultados parecem servir apenas para o estudo de caso em questão. Em outras situações, os parâmetros determinados não se encaixam.

Weissbart (2020) também buscam a melhoria de SCA através da sintonia eficiente de hiperparâmetros. Neste artigo, os autores concentram seus esforços em melhorar MLP, justificando que este tipo de rede neural é menos presente nas pesquisas do que CNN. Também, MLPs são redes mais simples, permitindo o ajuste mais fácil de seus hiperparâmetros, além de serem menos custosas computacionalmente. Foi investigado o comportamento de uma rede MLP no contexto de SCA sobre o AES. Explorando a sensibilidade dos hiperparâmetros da rede MLP sobre o desempenho do ataque, os autores visam fornecer uma melhor compreensão do ajuste de hiperparâmetros bem-sucedido e, em última análise, o desempenho deste algoritmo. Os resultados mostram que a MLP (com um ajuste de hiperparâmetro adequado) pode facilmente quebrar as implementações com contramedidas de atraso aleatório ou mascaramento. O estudo de Weissbart (2020) é bastante relevante, pois como comentado pelos autores MLPs são mais simples que CNNs. Assim, com o ajuste adequado, essas redes aplicadas aos SCAs podem se tornar uma ameaça real aos dispositivos criptográficos. Conforme visto neste trabalho, alguns parâmetros não foram testados nos experimentos, como por exemplo, outras funções de ativação. MLPs mais heterogêneas poderiam ser testadas. Também, direrentes números de *perceptrons* nas camadas, etc. Ainda, outras técnicas de redução de dimensionalidade ou pré-processamento diferentes do DoM poderiam ser testadas para melhorar o desempenho da MLP. Desse modo, muitos testes ainda podem ser realizados para verificar e melhorar a eficiência das MLPs no contexto de SCA. Em um caminho inverso, e este é o caminho adotado por esta proposta, podem-se aplicar técnicas às redes CNN com o intuito de reduzi-las, mantendo sua eficiência superior com menos esforço computacional.

É sabido que os pesos de uma rede neural precisam ser inicializados, geralmente com valores baixos e aleatórios. Entretanto, Li; Krček; Perin (2020) investigaram como a escolha dos inicializadores de peso influencia o desempenho das redes neurais profundas, mais especificamente CNNs, na análise de canal lateral. Notavelmente, os autores observaram que o *grid search* pula muitos valores possíveis, limitando a confi-

guração a apenas certos hiperparâmetros, desconsiderando completamente a influência de outros. Os autores utilizaram duas arquiteturas CNN encontradas na literatura para realizar seus experimentos. Os resultados mostram que inicializadores de peso diferentes fornecem comportamento radicalmente diferentes em termos de *guessing entropy* quando os conjuntos de dados são mais difíceis de atacar. Foi observado que mesmo os inicializadores de alto desempenho podem atingir um desempenho significativamente diferente ao realizar várias fases de treinamento. Também, os autores descobriram que esse hiperparâmetro depende mais da escolha do conjunto de dados do que outros hiperparâmetros comumente examinados. Ao avaliar as conexões com outros hiperparâmetros, a maior conexão é observada com as funções de ativação. Os resultados mostram que inicializadores de peso diferentes fornecem comportamento radicalmente diferentes em termos de *guessing entropy* quando os conjuntos de dados são mais difíceis de atacar. Foi observado que mesmo os inicializadores de alto desempenho podem atingir um desempenho significativamente diferente ao realizar várias fases de treinamento. Por fim, os autores descobriram que esse hiperparâmetro depende mais da escolha do conjunto de dados do que outros hiperparâmetros comumente examinados. Ao avaliar as conexões com outros hiperparâmetros, a maior conexão é observada com as funções de ativação.

Pessl; Mangard (2016) mostram que usando TA e a estrutura algébrica simples de multiplicação, o problema de recuperação de chave pode ser convertido para o conhecido problema de Paridade de Aprendizagem com Ruído (do inglês, *Learning Parity with Noise* - LPN). No entanto, em vez de usar algoritmos de solução LPN padrão, é apresentado um método que faz uso extensivo de confiabilidade de *bits* derivada de informações de canal lateral. Isso permite diminuir o tempo de execução do ataque em casos com probabilidades de erro de baixa a média. Em um experimento prático, foi atacada com sucesso uma implementação de *Fresh Re-Keying* protegida de 8 *bits* usando apenas 512 traços, destacando os autores que seu ataque também se aplica a outros cenários que usam a multiplicação de campo binário, como AES-GCM (*Galois Counter Mode*). Os resultados mostram que a estrutura simples da função de *Re-Keying* torna os ataques algébricos de canal lateral uma ameaça real. Além disso, mudar a tarefa de segurança do DPA para uma função de *Re-Keying* dedicada não é trivial. O vazamento de sua saída deve ser considerado em todas as operações subsequentes e mecanismos de proteção simples, como embaralhamento, podem não ser suficientes para proteção. Existem várias maneiras imagináveis de proteger o *Fresh Re-Keying* contra os ataques apresentados. Uma óbvia é adicionar outras contramedidas ao AES, o que, entretanto, aumenta o *overhead* de proteção. Alternativamente, pode-se alterar a função de *Re-Keying*, por exemplo, para multiplicação polinomial sobre um campo primo em vez de $GF(2^8)$. Embora apresente uma nova forma de SCA, esse trabalho é tangente à pesquisa realizada nessa tese. Ainda, como os pró-

prios autores afirmam, não são apresentados dados sobre o tempo de execução dos algoritmos.

O trabalho apresentado por Ozgen; Papachristodoulou; Batina (2016) combina técnicas de aprendizado de máquina com TA, com o intuito de melhorar a eficiência do segundo, focando na fase de correspondência de modelos (*matching*). São comparados três algoritmos de classificação em um conjunto de dados de *template* construído durante a execução de um algoritmo de multiplicação escalar regular. Assim, o ataque proposto usa algoritmos de classificação como um *distinguisher* alternativo para a fase de *matching* de *templates* do TA, a fim de fornecer *templates* precisos para distinguir entre traços de *template* e, eventualmente, recuperar a chave. Os autores mencionam que seu modelo de ataque é bastante genérico e pode funcionar com algoritmos simétricos e assimétricos e para vazamentos HW ou HD. O ataque em um cenário OTA (do inglês, *Online Template Attacks*), mostrou ser possível recuperar com sucesso *bits* do algoritmo de multiplicação escalar. Os métodos de classificação *Naive Bayes*, kNN e SVM podem dar 100% de sucesso na classificação correta dos traços de *template*, quando os *templates* são escolhidos e construídos corretamente. O fato do cenário de ataque utilizado assumir uma classificação binária torna os resultados realmente precisos e pode explicar a taxa de sucesso absoluta. Embora os autores tenham mencionado que seu método é genérico, o trabalho apresentado não ataca o AES, o que pode ser realizado em experimentos de trabalhos futuros. Além disso, não são testados dispositivos com contramedidas, ainda que os autores acreditem que o método obteria sucesso em tais dispositivos.

Gao, Si et al., em (GAO et al., 2017) questionam se é possível desenvolver um método baseado em inteligência artificial que aprenda os estados intermediários do algoritmo criptográfico atacado, a partir dos vazamentos de canal lateral. Sob certas circunstâncias, os autores descobrem que os estados intermediários podem ser recuperados de forma eficiente com a conhecida técnica chamada Análise de Componentes Independentes (do inglês, *Independent Component Analysis* - ICA). Especificamente, foram propostos vários métodos para converter os vazamentos do canal lateral em observações ICA eficazes. Para uma recuperação mais robusta, também foi apresentado um algoritmo ICA especializado que explora as características específicas dos sinais de circuito. Experimentos mostraram que a análise funciona bem em certos modelos ICA, recuperando corretamente mais de 80% dos estados intermediários, com apenas algumas centenas de traços. Além disso, o SCA baseado em ICA traz novas possibilidades para o atual estudo de SCA *non-profiled*, incluindo o ataque à criptografia intermediária e à engenharia reversa com menos restrições. Considerando que a ICA é uma ferramenta mais agressiva do que a maioria das técnicas anteriores de SCA, os autores acreditam que a SCA baseada em ICA é uma ferramenta promissora para o futuro estudo de SCA.

Uma abordagem intermediária entre aprendizado supervisionado e não-supervisionado é empregada em SCA por Picek; Heuser; Jovic (2018). A chamada abordagem semi-supervisionada, consiste em usar um pequeno número de medições rotuladas da fase de perfil, bem como as medições não rotuladas da fase de ataque para construir um modelo mais confiável. Na aprendizagem supervisionada, foram utilizados os algoritmos TA e sua versão agrupada *pooled* TAp, RF, MLP e *Naive Bayes*. Na aprendizagem semi-supervisionada (do inglês, *Semi-Supervised Learning* – SSL) baseada em gráfico, foi usado o algoritmo kNN (ou seja, o método para atribuir rótulos), uma vez que produz uma matriz esparsa que pode ser calculada muito rapidamente. Os resultados obtidos mostram que a SSL pode ajudar em muitos cenários. Melhorias significativas são obtidas para quase todos os classificadores, incluindo TA no cenário de baixo ruído para o pequeno número de traços no conjunto de dados de aprendizagem. Além disso, o TA foi aprimorado para a maioria dos tamanhos de conjunto de dados usando métodos SSL. Em geral, quando medido sobre todos os cenários considerados, o classificador MLP demonstra os melhores resultados, seguido por TAp e NB. A abordagem apresentada em (PICEK; HEUSER; JOVIC, 2018) é uma saída interessante quando o atacante se depara com cenários em que o tamanho do conjunto de dados de treinamento é restrito. Entretanto, como os próprios autores mencionam, poderiam ser exploradas mais contramedidas, já que esse é um problema preocupante nos SCAs (o ataque em dispositivos protegidos).

Uma forma diferente de buscar uma melhoria nos ataques baseados em aprendizado de máquina e aprendizado profundo é apresentada por Van der valk et al. (2020). A ideia é explorar a imitação para comparar modelos aprendidos por uma rede neural. Os autores exploram uma solução alternativa para alcançar compressão do modelo enquanto limita a perda de precisão. A solução proposta é baseada no conceito de mímica. Portanto, em vez de treinar uma pequena rede (de alunos) com base no conjunto de dados original, a simulação visa treinar a rede de alunos com a saída de uma grande rede (de professores). Ao treinar nos resultados da rede do professor, a rede do aluno pode alcançar uma precisão muito melhor em comparação com o treinamento nos rótulos originais. O ataque tem como alvo o valor intermediário dependente da chave de um *byte* do algoritmo cuja atividade de canal lateral é observada. Nos experimentos, foi utilizado o valor intermediário como rótulo (256 classes) ou o peso de *Hamming* (HW) do *byte* (9 classes). Os experimentos mostraram que é mais fácil imitar com MLP do que com CNN, mas ambas as configurações podem dar bons resultados. Melhor desempenho com redes rasas é possível quando arquiteturas profundas se ajustam com *overfitting*.

Diferente dos trabalhos apresentados até aqui, uma linha de pesquisa adotada por alguns autores refere-se à combinar mais de um tipo de informações de entrada para o algoritmo de inteligência artificial que realizará o ataque de canal lateral. Essa

abordagem, busca melhorar a eficiência dos SCA, através do aumento de informação útil entregue para os algoritmos de aprendizado de máquina ou aprendizado profundo.

O primeiro estudo sob esse paradigma, encontrado através da RSL aqui realizada foi (YU; CHEN, 2018). Neste trabalho, os traços de potência dissipada e de emissão eletromagnética são capturados juntos. Então, uma rede neural profunda (do inglês, *Deep Neural Network* – DNN) é usada para modelar a relação entre o ruído dos traços de potência e o ruído dos traços de EM, analisando os perfis de consumo e EM capturados. A partir disso, um ataque é realizado utilizando o ruído da EM para filtrar o ruído da potência. O dispositivo alvo dos ataques é dotado de contramedida de DVS (*Dynamic Voltage Scaling*). Segundo os autores, os resultados mostram que para SCAs convencionais, mesmo se 1 milhão de textos claros forem utilizados não é possível atacar um AES protegido. Ao passo que analisando somente 32500 de textos claros, através do método proposto, a chave criptográfica é revelada. Embora os resultados tenham se mostrado muito superiores as outras técnicas apresentadas, uma quantidade muito grande de traços ainda é necessária para obter sucesso no ataque. Assim, a proposta é interessante, entretanto são necessários estudos futuros para desenvolver melhor o método.

Em (HETTWER; GEHRER; GÜNEYSU, 2019), os autores apresentam uma nova arquitetura CNN para SCAs com perfil que permite codificar informações específicas de domínio (do inglês, *Domain Knowledge* – DK). Ao fazer isso, é possível alimentar o texto claro ou texto cifrado como uma fonte adicional de informações na rede (além das medidas de potência). O CNN com DK é dedicado a aprender de forma autônoma o vazamento do dispositivo em relação à chave secreta. Para inserir informações de conhecimento de domínio ao ataque, são introduzidos neurônios DK em uma rede CNN projetada para SCA. Assim, são unidas informações de características extraídas dos traços através das camadas convolucionais da CNN com informações de domínio específicas (DK). Melhorando assim, a eficiência dos ataques. Os autores demonstraram por meio de experimentos com dois conjuntos de dados diferentes que o CNN proposto com DK efetivamente consegue capturar de forma autônoma a função com o maior vazamento para quebrar a chave secreta diretamente. A ideia de Hettwer; Gehrer; Güneysu (2019) é bastante importante para a área de SCA. Entretanto, como os autores mencionam, outros testes devem ser feitos para consolidar a técnica apresentada. É interessante levar em conta também o tempo de processamento da rede final, o que não foi explorado no trabalho. Sabe-se que CNNs são redes neurais bastante poderosas, contudo seu poder computacional está atrelado a um custo relativamente alto de tempo e recursos.

O trabalho de Hettwer et al. (2020) combina o vazamento dependente da localização, de vários capacitores de desacoplamento em um SoC (*System on Chip*). São combinadas informações (vazamento) de diversas fontes usando a abordagem

de *deep learning information fusion*. Assim, os autores propõem um método de ataque baseado em aprendizado profundo e medições de EM multi-canal. Essa é uma nova estratégia de ataque que extrai o vazamento de informações dependente da localização que captura a atividade de partes específicas do *hardware* projetado usando diferentes capacitores de desacoplamento da fonte de alimentação. As medições obtidas são processadas por uma DNN de multiplas entradas. A DNN é treinada para combinar informações relacionadas ao compartilhamento para fazer previsões sobre os dados mascarados que estão sendo processados. Foram avaliadas muitas abordagens de fusão de dados para fornecer um guia de como a informação extraída pode ser combinada de uma maneira ótima. Para testar a efetividade da proposta, os autores compararam o método proposto com medidas de EM multi-canal obtidas diretamente da superfície de um *die* e do estado da arte de TAs. No total, o método proposto necessitou de 2750 traços para alcançar sucesso em um ataque de primeira ordem em uma implementação segura do AES. Portanto, os resultados mostram uma vantagem em relação ao número de traços necessários para sucesso na recuperação da chave comparado aos *profiled attacks* do estado da arte. É destacado que foram obtidos melhores resultados com MLP do que com CNN. Apesar de apresentar uma proposta interessante, salienta-se que o método proposto não superou TA quando se trata de utilizar EM do *die* como vazamento. Também, os resultados mostram a quantidade de traços para um *ranking*<10 e *ranking*<2. Não mostra quantos traços foram necessários para *ranking* 0. A abordagem da referência (SPECHT et al., 2018) necessitou menos traços para chegar ao *ranking*<2 do que o proposto no artigo.

Zhang et al. (2020) apresentam um método SCA com aprendizado profundo *multi-label*. O método baseia-se em uma modificação da camada de saída de uma rede neural. Na classificação *multi-label*, cada traço do consumo no *dataset* de treino é marcado com um conjunto de *labels* de *bits*, assim um grupo de probabilidades de *labels* é predito para um traço de ataque. Os experimentos com a classificação *multi-label* são realizados com o *dataset* de *benchmark* ASCAD. Para uma comparação justa com os resultados do artigo original do (PROUFF et al., 2018) ASCAD, somente a camada de saída é alterada para poder ser ajustada a um modelo *multi-label*. Modelos *monobit* também são testados. Os resultados mostram que $MLP_{monobit}$ consegue quebrar todos os *bits*, exceto o *bit1* quando os traços estão alinhados. Quando os traços estão desalinhados de 50 amostras só o *bit7* foi quebrado e quando os traços estão desalinhados de 100 amostras nenhum *bit* foi quebrado. Já $CNN_{monobit}$ consegue quebrar todos os *bits* quando os traços estão alinhados, mas nem todos quando os traços estão desalinhados. Entretanto, como mencionado no artigo, *multi-label* pode ser considerado como um *ensemble* de vários *monobits*. Assim, o *ensemble* de MLP *monobit* obtém melhores resultados que $MLP_{best}$. O *ensemble* do $CNN_{monobit}$ executa muito melhor que o *ensemble* $MLP_{monobit}$ e $CNN_{best}$, e aproximadamente 150, 300, 500 tra-

ços são necessários respectivamente para quebrar o *dataset* ASCAD. $MLP_{multi-label}$ tem melhores resultados que $MLP_{best}$. $CNN_{multi-label}$ obtém o melhor resultado e somente 150, 250 e 350 traços são necessários para quebrar o ASCAD. O modelo *multi-label* obtém desempenho semelhante ao conjunto ingênuo de modelos *monobit* com apenas um modelo treinado. $MLP_{multi-label}$ tem um certo grau de degradação de desempenho quando os traços são dessincronizados, enquanto $CNN_{multi-label}$ tem um desempenho um pouco melhor. E é intrigante que $MLP_{multi-label}$ e $CNN_{multi-label}$ obtenham o mesmo desempenho de ataque quando os traços estão alinhados. A modificação da camada de saída das redes neurais para melhoria da eficiência do SCA mostrou-se eficaz. Contudo, como mencionado pelos autores, o método apresentado necessita ainda de otimizações, como por exemplo, a inclusão de uma etapa de pré-processamento.

Uma estratégia interessante, e até onde se sabe, abordada pela primeira vez em Perin; Wu; Picek (2021) no contexto de SCA, consiste em aplicar uma técnica de poda (do inglês, *pruning*) para melhorar o desempenho de redes neurais no ataque. A ideia central desta técnica é realizar um treinamento em uma rede neural relativamente grande, com a qual obtém-se sucesso nos ataques a canais laterais. Em seguida, aplica-se o processo de poda removendo a atividade de alguns pesos da rede. Neste trabalho, os autores removem os pesos de menor valor, embora existam outras técnicas de poda que podem ser aplicadas em trabalhos futuros. Depois disso, a rede é reinicializada com os mesmos pesos originais. Com este processo, os autores dizem que a rede reinicializada mostra desempenho igual ou, na maioria das vezes, superior em comparação com a grande rede treinada. Como visto em (PERIN; WU; PICEK, 2021), outra opção de inicialização da rede podada pode ser através da escolha aleatória de valores, como visto na Figura 15.

O trabalho de Perin; Wu; Picek (2021) baseia-se na hipótese do bilhete de loteria (do inglês, *Lotery Ticket Hypothesis*), o qual assume que redes neurais profundas inicializadas aleatoriamente contêm sub-redes que, quando treinadas isoladamente (sem considerar outras partes da rede), alcançam precisão de teste comparável à rede original em um número semelhante de iterações. As sub-redes são obtidas por poda da rede original. Os autores de Frankle; Carbin (2018) observaram que a sub-rede obtida após a poda (com um nível de esparsidade de P%) fornece desempenho superior quando reinicializada com os pesos usados para inicializar os pesos originais. Essas sub-redes de melhor desempenho são então chamadas de *tickets* vencedores. A esparsidade denota a porcentagem da rede removida (por exemplo, 90% de esparsidade em um MLP consistindo em uma camada oculta com 100 neurônios removeria 90 neurônios).

Sabe-se que o SCA baseado em aprendizado profundo requer, idealmente, a seleção da menor arquitetura de rede neural possível que forneça boa generalização. Mo-

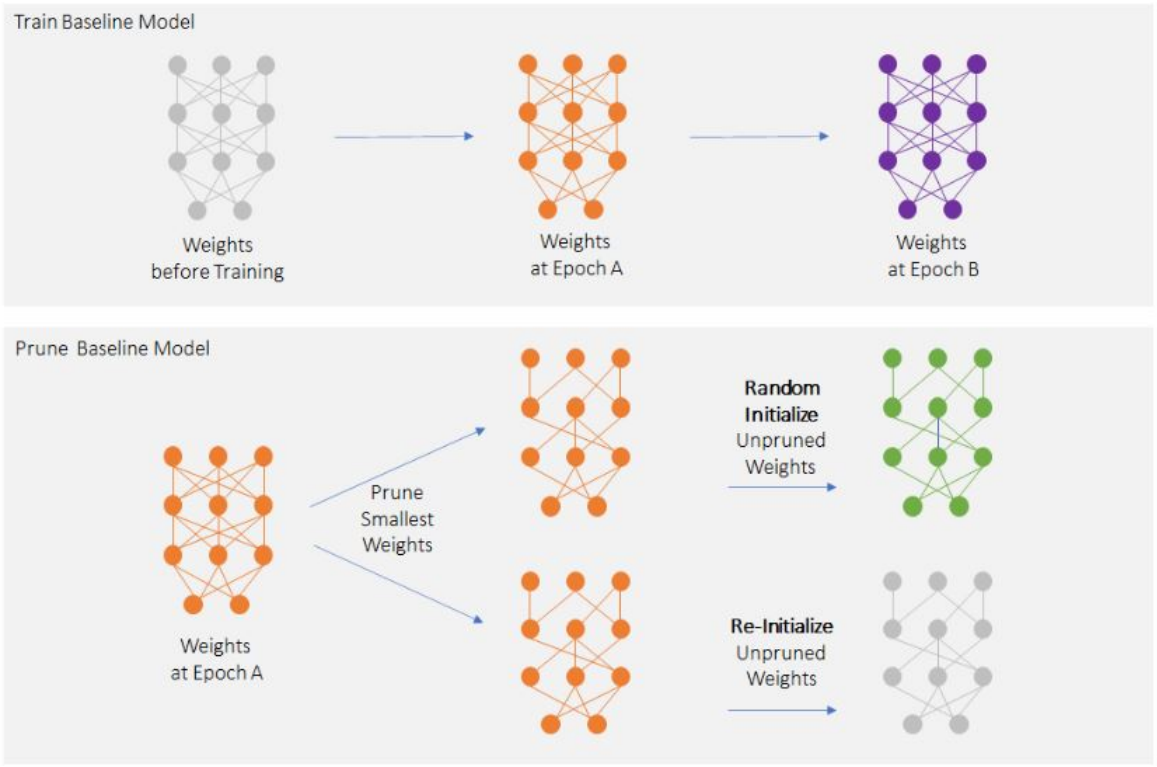


Figura 15 – Procedimento de Poda para LTH. Fonte: (PERIN; WU; PICEK, 2021)

delos pequenos são mais rápidos de treinar e mais fáceis de interpretar. O desafio de encontrar uma arquitetura pequena com bom desempenho pode crescer proporcionalmente à dificuldade do conjunto de dados do canal lateral avaliado (desalinhamento, ruído, contramedidas). No entanto, os traços de canal lateral geralmente fornecem uma relação sinal-ruído baixa e as técnicas de regularização desempenham um papel importante na capacidade de aprendizagem de vazamento. Os modelos pequenos são autorregularizados, principalmente por oferecerem menos capacidade de *overfit* ao conjunto de treinamento. De acordo com os autores, isso justifica a importância de encontrar bilhetes vencedores no SCA. Independentemente do conjunto de dados avaliado, começar a partir de um grande modelo de linha de base e aplicar a hipótese do bilhete de loteria aumenta as chances de criar um modelo de rede neural pequeno e eficiente.

Os autores enfatizam que a poda é conveniente para grandes redes neurais. Encontrar redes eficientes e pequenas é mais difícil do que começar com um modelo grande e depois reduzi-lo. Neste artigo, foram consideradas arquiteturas de rede neural com até 1 milhão de parâmetros treináveis. Segundo os autores, a poda tem duas vantagens principais para SCA:

*i)* Se for encontrado um modelo grande que generaliza bem, a poda favorece a explicabilidade e a interpretabilidade.

*ii)* A poda atua como um regularizador forte, o que é importante para conjuntos de dados SCA pequenos e com ruído.

São também discutidos resumidamente os limites que a poda e a hipótese do bilhete de loteria oferecem em relação aos resultados e sua explicabilidade:

*i)* A poda permite fazer redes neurais menores com desempenho no nível ou até melhor do que redes neurais maiores.

*ii)* A hipótese do bilhete de loteria pressupõe que haverá sub-redes menores e de bom desempenho, os chamados bilhetes premiados

*iii)* No contexto de SCA, os *tickets* vencedores são pequenas sub-redes com bom desempenho de ataque, medido com GE e SR.

*iv)* A poda e o LTH não são métodos para fornecer explicabilidade. No entanto, um modelo podado representa um pequeno modelo que favorece a explicabilidade (por exemplo, técnicas de visualização) e interpretabilidade (por exemplo, quais classes são melhor classificadas pelo modelo podado).

Neste estudo, foi investigado como a poda pode ser uma estratégia útil ao usar uma única rede neural para atacar a chave inteira (e não apenas um único *byte* de chave como comumente relatado na literatura). Na verdade, os autores observaram que a remoção e a reinicialização provaram ser opções muito poderosas para ajustar a rede neural a diferentes configurações. Sua investigação experimental permite podar até 90% dos pesos e ainda alcançar um bom desempenho de ataque. Assim, eles conseguem atingir o mesmo desempenho de ataque para redes significativamente menores (mais fáceis de ajustar e mais rápidas de treinar). Para os experimentos os autores consideraram o modelo de vazamento baseado no peso *Hamming* devido aos *datasets* escolhidos (ASCAD; e CHES CTF 2018).

Com base nos experimentos realizados, os autores fornecem várias observações gerais:

- Se o modelo de linha de base funcionar mal para um conjunto limitado de traços de ataque, a remoção ainda pode melhorar o desempenho;
- Se a linha de base funciona bem e não acontece *overfit*, a poda mantém o desempenho, mas produz redes menores;
- Se não houver traços de perfil suficientes para a capacidade do modelo, acontece *overfit* e a poda pode ajudar a evitar isso;
- Mais traços de criação de perfil melhora os resultados de poda, mas também reduz as diferenças entre as técnicas de inicialização de peso;
- O procedimento de poda e reinicialização de peso funciona melhor, desde que as arquiteturas de rede neural sejam grandes o suficiente para utilizar os *tickets* vencedores;

- A poda pode melhorar os resultados do ataque, conforme indicado por várias métricas de desempenho de SCA;
- A poda representa uma opção forte ao considerar um modelo de rede neural treinado para um *byte* de chave a ser aplicado a outros *bytes* de chave.

Como trabalho futuro, os autores planejam considerar técnicas de poda mais sofisticadas. Finalmente, como discutido, a poda permite redes neurais menores e com bom desempenho, mas não fornece *insights* sobre a explicabilidade das redes neurais. Pode ser interessante considerar várias técnicas de visualização de características para avaliar as características importantes antes e depois da poda.

A técnica apresentada por Perin; Wu; Picek (2021) é bastante popular na área de inteligência artificial e traz muitos benefícios, como revisado acima. Portanto, sua aplicação aos *side channel attacks* pode culminar em redes neurais pequenas e eficientes, apontando uma ameaça aos dispositivos criptográficos. Aqui nesta Tese, foram investigadas formas de realizar a poda, diferentes da aqui apresentada, como será visto em seções sobsequentes. Pois percebeu-se aqui um caminho promissor a ser trilhado com base nesse método.

## 4.4 Comparação dos Trabalhos Relacionados

A Tabela 5 apresenta uma comparação dos trabalhos relacionados que utilizam técnicas de aprendizado de máquina e aprendizado profundo no contexto de SCAs. Para cada um dos trabalhos foram elencadas: *(i)* as ferramentas utilizadas, *(ii)* quais algoritmos de inteligência artificial foram aplicados, *(iii)* que conjuntos de dados foram usados, *(iv)* se o dispositivo atacado é dotado de contramedidas, *(v)* as etapas de pré-processamento empregadas e *(vi)* que métricas de avaliação foram aplicadas, parâmetros estes que ajudaram na tomada de decisão e o respectivo método a ser aplicado.

Tabela 5 – Comparação entre os trabalhos relacionados.

| **Avaliação do Ataque baseado em Aprendizado Profundo sob diferentes cenários** | | | | | | |
|---|---|---|---|---|---|---|
| **Estudo** | **Ferramenta(s)** | **Algoritmo(s) ML/DL** | ***Dataset(s)*** | **Contramedida(s)** | **Pré-Processamento** | **Métrica(s)** |
| (WANG et al., 2020) | Não informado | MLP e CNN | Próprio | Não | *max_min scaling* [0,1] | *Rank*, GE e PGE |
| (WEISSBART; PICEK; BATINA, 2019) | *Keras* | CNN | Próprio | Não | PCA | SR e acurácia |
| (HEUSER et al., 2017) | Não informado | NB, C4.5 e MLP | Próprio | Não | Não | *Confusion Coefficient* e SR |
| (ZHANG et al., 2020) | Não informado | DNN | Próprio e DPA *Contest* v.4 | RSM | FFT | *Loss* e GE |
| (DAS et al., 2019) | *Keras* com *TensorFlow* como *backend* | DNN | Próprio | Não | Não | acurácia |
| (XU et al., 2019) | Não informado | CNN | Próprio | Não | PCA, LDA | *Rank* |
| (BRISFORS; FORSMARK; DUBROVA, 2021) | Não informado | CNN | Próprio | Não | Não | *Rank* e GE |
| (MUKHERJEE, 2020) | Não informado | LSTM | Próprio | Ocultação e Mascaramento | Não | SR |
| (KUBOTA et al., 2019) | Não informado | CNN | Próprio | RSM | Extração de POIs (não explica) | *Rank* |
| (MAGHREBI, 2019) | Não informado | CNN, MLP e LSTM | Próprio | *shuffling* e *1-amongst-N* | Não | *Rank* |

| | | | | | | |
|---|---|---|---|---|---|---|
| (ALIPOUR et al., 2020) | *Keras* | MLP | ASCAD e Próprio | *Correlated Noise Generation* para *hiding* | Não | acurácia |
| (**?**) | *Keras* | CNN | Próprio | *jitter* (RDI) | Não | acurácia e GE |
| **Comparação entre Métodos de Ataque** | | | | | | |
| (LERMAN; MARTINASEK; MARKOWITCH, 2017) | Não informado | SVM, RF e MLP | DPA *Contest* v.2 e DPA *Contest* v.4 | Desalinhamento nos traços e RSM | Não | SR |
| (CARLET et al., 2016) | *Keras* e *Sci-Kit Learn* | RF, *Autoencoder*, CNN, MLP e LSTM | DPA *Contest* v.2 | mascaramento e *hiding* | PCA | GE |
| (ROBISSOUT et al., 2021) | Não informado | $CNN_{best}$ | ASCAD | mascaramento | Não | O artigo baseia-se em criar uma métrica adequada aos SCA. |
| **Melhoria da Eficiência do Ataque** | | | | | | |
| (MUKHTAR; KONG, 2019) | *Weka* | RF, BN, SVM e MLP | Próprio | Não | PCA e extração de características no domínio da frequencia | acurácia |
| (WEISSBART, 2020) | *Keras* | MLP | ASCAD e $AES_RD$ | *Random Delay* e mascaramento | DoM | GE e acurácia |
| (LI; KRČEK; PERIN, 2020) | *Keras* com *TensorFlow* como *backend* | CNN | DPA *Contest* v.4, ASCAD e $AES_RD$ | *Random Delay* e mascaramento | Não | GE |

| | | | | | | |
|---|---|---|---|---|---|---|
| (PESSL; MANGARD, 2016) | Não informado | Um novo algoritmo baseado em LPN | Próprio | *shuffling* | Não | Complexidade do ataque, Meta-probabilidade e Taxa de ocorrência |
| (OZGEN; PAPACHRISTODOULOU; BATINA, 2016) | *Matlab* | NB, kNN e SVM | Próprio | Não | Não | SR |
| (PICEK; HEUSER; JOVIC, 2018) | *Python* | RF, NB, kNN e MLP | DPA *Contest* v.4, $AES\_HD$, $AES\_RD$ e *Random Delay Dataset* | *Random Delay* | Não | GE |
| (VAN DER VALK et al., 2020) | Não informado | MLP e CNN | DPA *Contest* v.4, $AES\_HD$, $AES\_RD$ e ASCAD | *Random Delay* e mascaramento | Não | GE |
| (YU; CHEN, 2018) | Não informado | DNN | Próprio | DVS | Utilizar o ruído de EM para filtrar os traços de consumo | Coeficiente de Correlação |
| (HETTWER; GEHRER; GÜNEYSU, 2019) | *Keras* e *Sci-Kit Learn* | CNN | DPA *Contest* v.2 e DPA *Contest* v.4 | RSM e *shuffling* | PCA para atacar TA | KGE |

| | | | | | | |
|---|---|---|---|---|---|---|
| (HETTWER; GEHRER; GÜNEYSU, 2020) | Não informado | MLP e DNN | Próprio | mascaramento | média com um fator de 250 e PCA no TA | GE |
| (ZHANG et al., 2020) | *Keras* com *TensorFlow* como *backend* | MLP e CNN | ASCAD | mascaramento e desalinhamento | Não | GE |
| (PERIN; WU; PICEK, 2021) | Não informado | MLP e CNN | ASCAD e CHESS CTF 2018 | mascaramento | Não | GE |

## 4.5 Considerações sobre o Capítulo

Este Capítulo apresentou o processo de Revisão Sistemática da Literatura realizada para esta Tese, com o intuito de investigar o estado da arte de trabalhos que empregam Aprendizado Profundo em SCAs. Depois disso, foi feita uma extensa revisão de trabalhos divididos em categorias tais como aplicação de DL em SCA sob diferentes cenários, comparação entre métodos de ataque baseados em DL e melhorias na eficiência do ataque proposto. Destas, a última categoria recebe especial destaque por estarem relacionados com o escopo desta proposta.

Através deste Capítulo, foi possível confirmar a eficiência das redes neurais quando aplicadas em SCAs para atacar dispositivos dotados de contramedidas. Descobriu-se uma maior dificuldade na realização de ataques em dispositivos dotados de contramedidas temporais, como por exemplo a inserção de atrasos aleatórios. Contudo, um tipo específico de rede neural, a CNN mostrou-se eficiente ao atacar até mesmo tais dispositivos. Porém, o sucesso das CNNs se dá às custas de um esforço computacional excessivo. Por isso, esta Tese propõe o uso de métodos de redução do tamanho das redes neurais, mais especificamente as técnicas de poda (em inglês *pruning*), para que se obtenha sucesso aos ataques por canais laterais baseados em redes neurais, sem a necessidade de um esforço computacional excessivo.

# 5 TÉCNICAS DE PODA

No Capítulo 4 foram resgatados da literatura diversos trabalhos que mostram a larga aplicação de algoritmos de aprendizado profundo aplicados a SCAs. Isto evidencia ainda mais o fato de ser interessante apontar métodos para a redução de tais redes. Portanto, neste Capítulo será apresentado um apanhado sobre os principais métodos de poda encontrados na literatura até a data de escrita desta Tese. Tais trabalhos, serviram como base para o desenvolvimento do trabalho aqui proposto. Para a busca dos trabalhos relacionados a este assunto, foi seguida a mesma metodologia de RSL descrita na Seção 4.1.

Nesta etapa foram analisados desde trabalhos que realizam a poda de pesos (HAN et al., 2015) e (GUO; YAO; CHEN, 2016), e também trabalhos propostos especificamente para a remoção de filtros de camadas convolucionais de CNNs como em (POLYAK; WOLF, 2015), (HE; ZHANG; SUN, 2017) e (YAN et al., 2021). Além disso, buscou-se trabalhos que utilizassem uma abordagem mais geral, ou seja, estratégias que fazem a poda em neurônios (sejam filtros de camadas convolucionais (CNNs), ou até mesmo neurônios de camadas densas encontradas tanto em CNNs como em redes MLP).

Neste contexto temos o trabalho de Kim; Kwok (2019) que apresenta um método destinado à poda de filtros, mas que pode ser adaptado para neurônios de camadas densas. O inconveniente é que além de apresentar um método iterativo, Kim et al., realizam a poda de forma suave, em que os neurônios podem até mesmo ser reestabelecidos. Isso torna o método bastante custoso.

Outro trabalho encontrado na literatura, que apresenta a poda de neurônios é o apresentado por Fan; Tang; Ma (2022). Neste trabalho, os autores utilizam a Norma L1, que se baseia na soma do valor absoluto dos pesos do neurônio, como métrica para selecionar os neurônios a serem removidos. Como mostraremos em nossos experimentos, essa métrica pode não ser interessante em alguns casos. Além disso, o método empregado é iterativo, necessitando de muitos treinamentos da rede para atingir o objetivo.

Outros exemplos de trabalhos que empregam a poda sobre neurônios foram encon-

trados: (LAURET; FOCK; MARA, 2006), (BABAEIZADEH; SMARAGDIS; CAMPBELL, 2016), (TAKEDA; NAKADAI; KOMATANI, 2017) e (EVANS, 2018). Dentre os trabalhos aqui revisados, o trabalho apresentado por (HU et al., 2016) mostrou-se bastante interessante, apesar de tratar-se de um método iterativo, como outros encontrados nesta RSL. O método proposto neste trabalho, consiste na realização da poda de neurônios baseado na sua taxa de ativação, descrita através da Equação 30. Esta métrica pareceu-nos bastante adequada para representar a importância dos neurônios dentro da rede. Por esta razão, o trabalho de Hu et al. (2016) foi utilizado como referência para o desenvolvimento da presente Tese, como veremos em seções subsequentes.

## 5.1 Técnicas de Redução de Redes Neurais através de Poda

Nesta Seção são apresentados trabalhos relacionados a técnicas de poda (do inglês, *pruning*). Entretanto, antes de iniciar a apresentação desses trabalhos, seria interessante estabelecer alguns conceitos relativos à poda em redes neurais artificiais. Inicialmente, é interessante saber que a poda pode ser realizada com base em diferentes elementos dentro das redes neurais. O termo utilizado para definir quais elementos da rede neural serão o foco da poda, chama-se granularidade (KOLLEK et al., 2021). Assim, o alvo da poda podem ser os pesos da rede (conexões entre os neurônios), os canais (neurônios de camadas densas, ou filtros das camadas convolucionais), ou até mesmo camadas inteiras. Outro aspecto importante a considerar quando estamos nos referindo a técnicas de poda é se iremos proceder com uma poda estruturada, ou não-estruturada. Kollek et al. (2021) além de outros trabalhos encontrados na literatura nos traz esses conceitos:

- A poda não-estruturada remove os pesos individuais, baseada em determinados critérios. Existem inúmeros critérios encontrados na literatura;
- A poda estruturada, conta com a remoção de conjuntos de pesos, como por exemplo, a remoção de neurônios. Quando removemos um neurônio da rede neural, estamos removendo muitos pesos de uma só vez, pois este neurônio possuía interconexões com neurônios de suas camadas adjacentes. Mais neurônios são removidos, quando abordamos a poda estruturada com foco nas camadas da rede.

Outra característica que deve-se observar quanto ao processo de poda, diz respeito ao processo em si. Portanto, se a poda é realizada através de um procedimento iterativo no qual uma determinada quantidade de neurônios é removida a cada iteração, ou se todos os neurônios são removidos de uma única vez (essa remoção em uma única etapa é também chamada de abordagem *One-Shot*). Como pode-se perceber, o processo iterativo necessita de sucessivos retreinamentos da rede. Assim,

neurônios são removidos em um passo, e no passo seguinte a rede é treinada. Em seguida uma nova remoção é realizada e a rede é novamente treinada, e assim sucessivamente. É possível notar que o processo iterativo é muito mais custoso em termos de tempo de processamento, sendo portanto, recomendável evitá-lo. A partir desses conceitos, pode-se apresentar os trabalhos encontrados na literatura relacionados a técnicas de poda.

Han et al. (2015) apresentam um método que reduz tanto o armazenamento de memória (que figura um dos gargalos do uso de redes neurais), quanto o esforço computacional requeridos por uma rede neural em ordem de magnitude sem afetar sua acurácia por aprender somente as conexões importantes. Conforme destacam os autores, as redes neurais geralmente são super-parametrizadas e existe uma redundância significativa dentro dos seus modelos. O método proposto realiza a poda de conexões redundantes usando um método de três passos. Primeiro, a rede é treinada para aprender quais conexões são importantes. Depois, as conexões não-importantes são podadas. A poda dos pesos se dá através de limiar. Dessa forma, pesos com valor menor do que o limiar, são removidos. Finalmente, a rede resultante é retreinada para um ajuste-fino dos pesos das conexões restantes. Através desse processo, que pode ser repetido, os pesos menos importantes são podados, transformando uma camada totalmente conectada em uma camada esparsa. Este trabalho trata apenas a poda de pesos (conexões) da rede. Isso pode tornar o processo menos eficiente frente a outras técnicas que utilizam uma granularidade maior, como por exemplo neurônios ou até mesmo camadas inteiras. Também por se tratar de um método iterativo, pode se tornar altamente custoso, visto que conexões (pesos) individuais são removidos a cada iteração.

Guo; Yao; Chen (2016) propõem um método que realiza poda de conexões juntamente com emendas de conexões (restauração de conexões) para evitar a poda de conexões erradas (que diminuem a acurácia da rede). Os processos de poda e emenda acontecem de uma forma cíclica, ou seja, acontecem constantemente dentro do processo inteiro. Os resultados mostram que sem qualquer perda de precisão, o método apresentado pode comprimir eficientemente o número de parâmetros em LeNet-5 e AlexNet por um fator de 108× e 17,7× respectivamente. Isso, segundo os autores, supera o método de poda do estado da arte com margem considerável. Apesar de aparentemente atraente, o método aqui apresentado pode ser superado quando aplicamos a poda em granularidades maiores, como por exemplo a canais (neurônios), uma vez que muitas conexões (pesos) são eliminadas de uma única vez. Além disso, por realizar remoções de pesos individuais a cada iteração, esse método torna-se excessivamente custoso comparado a outras técnicas que aplicam maior granularidade nas remoções.

O artigo apresentado por Polyak; Wolf (2015) traz como estudo de caso o pro-

blema encontrado para realizar a biometria, com *hardware* limitado. São propostos dois novos métodos de compressão: um baseado em eliminar canais menos ativos e outro em acoplar poda com uso repetido de elementos já computados. Os autores destacam que a poda de um canal inteiro é uma ideia atraente, uma vez que isso leva diretamente a economia de tempo de execução em quase todas as arquiteturas razoáveis. A escolha do neurônio a ser removido é feita através da medição da variância da ativação da saída do neurônio em questão. Através do método proposto, os autores destacam que foi alcançada uma redução de 2.65 vezes no tempo de execução com uma perda de acurácia muito moderada. É mencionado que o método proposto não altera a arquitetura da rede original. O método aqui apresentado é bastante interessante, pois visa a remoção de neurônios, o que é mais eficiente do que remover pesos individualmente. A métrica de escolha do neurônio a ser removido também é muito pertinente, uma vez que deriva da ativação dos neurônios. Contudo, esse trabalho visa a remoção de filtros ou camadas de filtro inteiras. Como o método baseia-se na informação da saída dos filtros, ou mapas de características, seria difícil a conversão desta técnica para torná-la mais abrangente, como por exemplo, lidar com a remoção de neurônios de camadas densas.

Um novo método de poda de neurônios para aceleração de CNNs é apresentado por He; Zhang; Sun (2017). É proposto um algoritmo iterativo de dois passos para o treinamento de uma CNN. Esse algoritmo efetivamente poda cada camada com uma seleção de neurônios baseada em regressão LASSO (TIBSHIRANI, 1996) e em *least square reconstruction*. Esse algoritmo é generalizado para casos multicamadas e multiramos. Os autores mencionam que esse método reduz o erro acumulado e melhora a compatibilidade com várias arquiteturas. A rede VGG-16 foi utilizada como estudo de caso, e os resultados mostram um aumento na velocidade da rede de até 5x com apenas 0.3% de aumento nos erros. Os autores destacam ainda que, o método por eles proposto é capaz de acelerar redes modernas como ResNet, Xception e sofrer somente 1.4%, 1.0% de perda de acurácia respectivamente, com um aumento de 2x na velocidade das redes. Em He; Zhang; Sun (2017) os autores utilizam uma abordagem que explora redundância dentro do mapa de características gerado pelos filtros da rede. Entretanto, apesar de ser um método interessante, está voltado exclusivamente à remoção de filtros das camadas convolucionais das CNNs. Isso restringe o trabalho a aplicações mais gerais, como redução de MLPs, por exemplo.

No trabalho apresentado em (KIM; KWOK, 2019) os autores propõem uma abordagem que realiza a poda de uma maneira suave. Esse método, diferente de outros encontrados na literatura, permite recuperar componentes anteriormente podados. A proposta consiste em uma nova técnica para a poda de filtros chamada *Dynamic Unit Surgery* - DUS. Com esse método os autores dizem reduzir a degradação e o tempo de poda/ajuste fino através de dois mecanismos: (1) permitir que os componentes

podados se recuperem durante o ajuste fino e (2) podar cada componente de forma continuamente descendente, em vez de eliminar abruptamente o componente. Os resultados apontam que o método foi aplicado à rede VGG-16 (com o *dataset* CIFAR10), e a rede resultante ficou com apenas 5% dos elementos da rede original e 23% dos seus FLOPs (*Float Points Operations*), enquanto a taxa de erro alcançou 6.65% sem degradação da rede original. Embora o método apresentado seja aplicado unicamente à filtros das camadas convolucionais das CNNs, este poderia ser aplicado a neurônios de camadas densas, aumentando sua abrangência. Entretanto, o método é iterativo, e ainda mais, realiza a poda dos filtros de forma suave, podendo os mesmos serem recuperados, o que torna o método mais custoso em termos de tempo de processamento.

Chen et al. (2021) propõem um método dinâmico de poda, o qual poda canais não-importantes na fase inicial do treinamento. Contudo, ao invés de utilizar critérios indiretos como peso normalizado, soma absoluta dos pesos e erro de reconstrução para guiar a poda, os autores criaram um critério diretamente relacionado com a acurácia final de uma rede para avaliar a importância de cada canal. Assim, uma estrutura foi projetada para ativar ou desativar aleatoriamente cada canal para que as alterações de precisão condicional (do inglês, *Conditional Accuracy Changes* - CACs) pudessem ser estimadas sob a condição de cada canal desativado. Em cada iteração do método 5% dos canais foi desabilitado por vez. Os resultados em vários conjuntos de dados (CIFAR, ImageNet e MNIST) para redes com vários tipos de arquitetura (ou seja, ResNet, VGGNet e MLP) demonstram a eficácia do método de poda de canais. Sem perda de precisão em comparação com a rede de base, mais de 30% e 40% de FLOPs para múltiplos ResNet e VGGNet no CIFAR, respectivamente, mais de 10% de FLOPs para ResNet-18 no ImageNet, e 80% de FLOPs para MLP no MNIST podem ser podados pelo método de poda de canal proposto. Como a acurácia da rede deve ser aferida a cada iteração, a rede deve ser treinada pelo menos vinte vezes para que se saiba quais neurônios devem ser podados. Isso tende a tornar o processo muito custos. Além do mais, no contexto de SCA a acurácia não é a métrica mais adequada para definir o bom funcionamento de uma rede neural.

Fan; Tang; Ma (2022) adaptam o método de *channel pruning* às características da unidade de convolução separável em profundidade e suas variantes. Segundo os autores, usando a configuração de taxa de poda unida, a esparsidade adicional contida, e certo pré-processamento no *dataset*, os experimentos mostram uma acurácia mais alta no *dataset* CIFAR-10 em alguns casos. Inicialmente, o modelo é pré-treinado. Em seguida, o processo de seleção dos neurônios não importantes, através da norma L1, e sua remoção é realizada $L$ vezes. Por fim um ajuste fino é realizado para recuperar a acurácia e o modelo podado está disponível. Este método consiste em método iterativo, o que por si só é mais custoso, devido aos sucessivos treinamentos necessários

para colocar o processo em funcionamento. Além disso, a métrica escolhida para seleção dos neurônios a serem removidos poderia ser substituída por uma métrica que refletisse mais a importância de cada neurônio, como por exemplo, sua ativação.

O trabalho apresentado por Yan et al. (2021) realiza a poda dos filtros das camadas convolucionais das CNNs. Os autores propõem um algoritmo de poda de canal através de vários critérios baseado na dependência de peso, *Channel Pruning Method via Multi-Criteria* - CPMC, que pode comprimir um modelo pré-treinado diretamente. O CPMC define a importância do canal em três aspectos, incluindo seu valor de peso associado, custo computacional e quantidade de parâmetros. De acordo com os autores, (YAN et al., 2021) utiliza como critérios para formar o *multi-criteria* tanto as entradas como as saídas dos filtros a serem removidos, o que restringe o uso desse método exclusivamente a filtros das camadas convolucionais das CNNs. Além do mais, para que possamos obter melhores resultados, existe a possibilidade de tornarmos o método iterativo com etapas de poda e treino, o que torna o método bastante custoso computacionalmente.

Lauret; Fock; Mara (2006), propõe um novo algoritmo de poda para obter o número ideal de unidades ocultas de uma única camada de uma rede neural totalmente conectada (NN). A técnica conta com uma análise global de sensibilidade da produção do modelo. A relevância dos nós ocultos é determinada pela análise da decomposição de Fourier da variância da saída do modelo. Cada unidade oculta recebe uma razão (a fração de variância que a unidade responde) que dá sua classificação. Essa informação quantitativa, portanto, leva a uma sugestão das unidades mais favoráveis para eliminar. Resultados experimentais sugerem que o método pode ser visto como uma ferramenta eficaz, capaz de controlar a complexidade em NNs. O método apresentado neste trabalho é iterativo. Assim, além de muitos retreinamentos, é necessário calcular a *Transformada de Fourier* para cada nó da rede, o que é custoso computacionalmente, inviabilizando o uso deste método em muitos casos. Com este método nos experimentos, são podados mais de 85% dos parâmetros mantendo-se a acurácia. (LAURET; FOCK; MARA, 2006) traz ideias interessantes sobre como realizar a poda de neurônios de camadas totalmente conectadas, baseado na sua atividade. Entretanto, nesta abordagem o processo de poda e retreinamento da rede é repetido até que a acurácia da rede caia abaixo de um determinado limiar, fazendo com que o método se torne custoso para alcançar a rede podada.

Babaeizadeh; Smaragdis; Campbell (2016) propõem o *NoiseOut*, um algoritmo de poda totalmente automatizado baseado na correlação entre ativações de neurônios nas camadas ocultas. É mostrado que adicionar neurônios de saída adicionais com alvos totalmente aleatórios resulta em maior correlação entre neurônios, o que torna a poda por *NoiseOut* ainda mais eficaz. O método foi testado em várias redes e conjuntos de dados. Os experimentos apresentam altas taxas de poda, mantendo a

acurácia da rede original.

Hu et al. (2016) mostram que, observando uma NN, é possível perceber que uma parcela significativa de seus neurônios tem sua saída, na maioria das vezes, igual a zero, independentemente dos valores de entrada. Esses neurônios são redundantes e podem ser removidos sem afetar diretamente a eficiência da rede, conforme descrito na Fig. 16, onde podemos ver que a remoção do neurônio selecionado se traduz no corte de vários pesos simultaneamente.

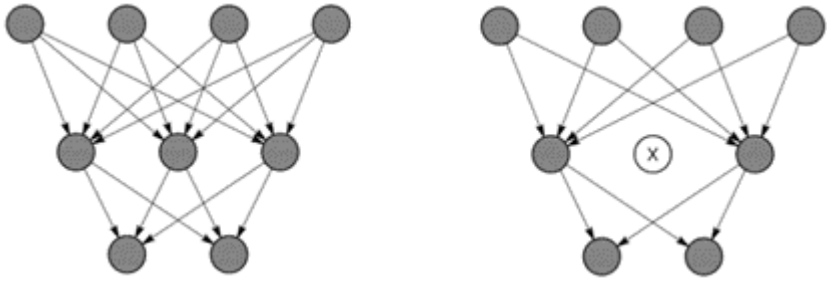

Figura 16 – Rede Neural antes da poda (esquerda) e depois da poda (direita). Fonte: (HU et al., 2016)

Para determinar a taxa de ativação zero, os autores definiram uma métrica chamada Porcentagem Média de Zeros (do inglês, *Average Percentage of Zero* - APoZ). Esta métrica é calculada de acordo com a Equação 30.

$$APoZ_c^{(i)} = APoZ(O_c^{(i)}) = \frac{\sum_k^N \sum_j^M f(O_{c,j}^{(i)} = 0)}{NXM} \tag{30}$$

Onde $O_c^{(i)}$ denota a saída do *c-ésimo* canal na *i-ésima* camada, $f(.)$ será '1' se o neurônio foi ativado e '0' para o contrário, $M$ denota a dimensão do mapa de características de saída de $O_c^{(i)}$ e $N$ denota o número total de exemplos de validação. Depois de identificar neurônios com APoZ mais alto, estes são removidos.

O método proposto por Hu et al. (2016) é composto por três etapas, conforme vemos na Figura 16. Assim, primeiro a rede é treinada sob o processo convencional e o número de neurônios em cada camada é configurado empiricamente. Em seguida, a rede é treinada em um grande conjunto de dados de validação para obter o APoZ de cada neurônio. Neurônios com APoZ alto são podados de acordo com certos critérios. As conexões de e para o neurônio são removidas quando um neurônio é podado (ver Figura 16). Após a poda do neurônio, a rede podada é inicializada usando os pesos antes da poda. A rede podada exibe algum nível de queda de desempenho. Assim, na etapa final, ela é treinada novamente para fortalecer os neurônios restantes, a fim de melhorar o desempenho da rede podada. Os autores mencionam ainda que a inicialização dos pesos é necessária para que a rede obtenha o mesmo desempenho que tinha antes da poda. Se uma rede podada é treinada a partir do zero, percebe-se que

ela contém maior porcentagem de neurônios de ativação zero do que a contraparte com inicialização de peso. Isso significa que uma rede treinada novamente sem inicialização de peso é muito menos eficiente. Como resultados, os autores destacam que seu método reduz a quantidade de parâmetros das redes neurais entre 2 e 3 vezes.

Embora os resultados, tanto com relação a acurácia quanto redução da quantidade de parâmetros da rede, sejam bastante satisfatórios, trata-se de um método iterativo e, portanto, custoso em termos de tempo e esforço computacional. Assim, seria interessante testar uma versão *One-Shot* desse método, evitando múltiplos retreinamentos.

Takeda; Nakadai; Komatani (2017) apresenta uma poda de nós para um modelo acústico baseado em redes neurais. Uma função é definida para medir a importância de cada nó da rede, sendo os nós menos importantes removidos. A função calcula a entropia de atividade de cada nó para descobrir os nós cuja saída não muda. Os autores introduzem a entropia de pesos de cada nó para considerar o número de pesos e seus padrões em cada nó. Como o número de pesos e os padrões diferem em cada camada, a importância do nó também deve ser medida usando os pesos relacionados ao nó alvo. Os autores relatam os resultados experimentais mostrando que o método de poda proposto reduziu com sucesso o número de parâmetros em cerca de 6% sem qualquer perda de precisão em comparação com uma função *score* baseada apenas na entropia da atividade do nó. A ativação dos neurônios é uma métrica bastante eficaz para a definição de sua importância. Entretanto, este trabalho apresenta baixa redução da quantidade de parâmetros em relação à linha de base. Parece ser mais interessante utilizar uma métrica que represente mais diretamente a ativação dos neurônios para realizar a seleção de quais devem ser removidos.

Evans (2018) realiza uma poda de nós nas camadas totalmente conectadas (*batch normalization*) de uma CNN. Isso é feito através de um método chamado *pruning by distinctiveness*. Esta técnica calcula o ângulo entre os vetores de ativação de pares de unidades e combina unidades que são consideradas não distintas ou exclui unidades que são consideradas opostas ou complementares. Como resultado, os autores dizem ter conseguido 94.52% (contra os 94.25% da linha de base) de acurácia na classificação utilizando o *dataset* EMNIST com uma redução de 12% no tamanho das camadas totalmente conectadas da rede. Aqui também é apresentada baixa redução da rede neural com relação à linha de base. Além disso, conforme relatam os autores, o passo de poda leva muito tempo para ser executado, devido ao cálculo dos ângulos vetoriais para cada par possível de unidades. À medida que o número de neurônios ocultos cresce, o tempo necessário para calcular os ângulos cresce (na melhor das hipóteses) exponencialmente, o que torna essa técnica bastante lenta para um grande número de neurônios ocultos. Também, os ângulos calculados são sensíveis ao ponto em que se escolhe medir. Embora seja despendido esforço para escolher um ponto intermediário válido, é difícil comparar os resultados alcançados nesses limiares com

outros estudos, a menos que o ângulo seja medido a partir da média das ativações.

Tabela 6 – Comparativo entre estudos sobre redução de redes neurais.

| **Trabalho** | **Granularidade** | **Métrica de seleção dos neurônios a remover** | **Método iterativo – Necessita retreinamentos** | **Desvantagens** |
|---|---|---|---|---|
| (HU et al., 2016) | Pesos | Limiar | Sim | Método iterativo baseado na remoção de Pesos. É difícil determinar o valor correto para o Limiar utilizado como parâmetro para remoção dos Pesos. |
| (GUO; YAO; CHEN, 2016) | Pesos | Limiar | Sim | Método iterativo baseado na remoção de Pesos. É difícil determinar o valor correto para o Limiar utilizado como parâmetro para remoção dos Pesos. |
| (POLYAK; WOLF, 2015) | Filtros (neurônios de camadas convolucionais) | Variância de Ativação da Saída (filtro) e Variância da Saída (camada) | Sim | O método baseia-se na informação da saída dos filtros (mapas de características). Assim, esse método é limitado, pois não remove neurônios de camadas densas, fazendo com que a redução não seja máxima. |
| (HE; ZHANG; SUN, 2017) | Filtros | Análise dos mapas de características gerados pelos filtros | Sim | Esse trabalho utiliza a redundância dentro do mapa de características gerado pelos filtros da rede como parâmetro de seleção dos filtros a serem removidos. Ou seja, está voltado exclusivamente à remoção de filtros das camadas convolucionais das CNNs. Isso restringe o trabalho a aplicações mais gerais, como redução de MLPs por exemplo. |

| | | | | |
|---|---|---|---|---|
| (KIM; KWOK, 2019) | Filtros | Fator de Escala de Normalização em Lote | Sim | O método foi aplicado à filtros, mas poderia ser aplicado também à neurônios de Camadas Totalmente Conectadas. Contudo, o método é iterativo. E ainda mais, realiza a poda dos filtros de forma suave, podendo os mesmos serem recuperados, o que torna o método mais custoso em termos de tempo de processamento. |
| (CHEN et al., 2021) | Neurônios | Alterações de precisão condicionais | Sim | A acurácia da rede é aferida a cada iteração. Assim, a rede deve ser treinada muitas vezes para que se descubra quais neurônios devem ser podados. Isso tende a tornar o processo muito custos. Além do mais, no contexto de SCA a acurácia não é a métrica mais adequada para definir o bom funcionamento de uma rede neural. |
| (FAN; TANG; MA, 2022) | Neurônios | Norma L1 | Sim | O método é iterativo, necessitando de muitos retreinamentos da rede. Além disso, o método utilizado para seleção dos neurônios poderia ser mais próximo a sua utilidade na rede, como por exemplo, sua ativação. |

| | | | | |
|---|---|---|---|---|
| (YAN et al., 2021) | Filtros | Multi-critérios | Sim | O método utiliza como critérios para formar o multi-criteria, tanto as entradas como as saídas dos filtros a serem removidos, o que restringe o uso desse método exclusivamente a filtros das camadas convolucionais das CNNs. Além do mais, para que possamos obter melhores resultados, existe a possibilidade de tornarmos o método iterativo com etapas de poda e treino. O que torna o método bastante custoso computacionalmente. |
| (LAURET; FOCK; MARA, 2006) | Neurônios | Análise da decomposição de *Fourier* da variância da saída do modelo. | Sim | O método apresentado neste trabalho é iterativo. Assim, além de muitos retreinamentos, é necessário calcular a *Transformada de Fourier* para cada nó da rede, o que é custoso computacionalmente, inviabilizando o uso deste método em muitos casos. |
| (BABAEIZADEH; SMARAGDIS; CAMPBELL, 2016) | Neurônios | Correlação entre ativações de neurônios nas camadas ocultas | Sim | O processo de poda e retreinamento da rede é repetido até que a acurácia da rede caia abaixo de um determinado limiar. Fazendo com que o método se torne custoso para alcançar a rede podada. Além disso, monitorar a acurácia pode não ser a melhor forma de avaliar uma rede no contexto de SCA. |

| | | | | |
|---|---|---|---|---|
| (HU et al., 2016) | Neurônios | Ativação | Sim | Trata-se de um método iterativo e, portanto, custoso em termos de tempo e esforço computacional. Assim, seria interessante testar uma versão *One-Shot* desse método, evitando múltiplos retreinamentos. |
| (TAKEDA; NAKADAI; KOMATANI, 2017) | Neurônios | Entropia de Atividade | Sim/Não | Este trabalho apresenta baixa redução (apenas 6%) em relação à linha de base. Parece ser mais interessante utilizar uma métrica que represente mais diretamente a ativação dos neurônios para realizar a seleção de quais devem ser removidos. |

| | | | | |
|---|---|---|---|---|
| (EVANS, 2018) | Neurônios | Ângulo entre os vetores de ativação de pares de neurônios | Sim | Baixa redução da rede neural com relação à linha de base. À medida que o número de neurônios ocultos cresce, o tempo necessário para calcular os ângulos cresce (na melhor das hipóteses) exponencialmente, o que torna essa técnica bastante lenta para um grande número de neurônios ocultos. Os ângulos calculados são sensíveis ao ponto em que se escolhe medir. Embora seja despendido um esforço para escolher um ponto intermediário válido, é difícil comparar os resultados alcançados nesses limiares com outros estudos, a menos que o ângulo seja medido a partir da média das ativações. |

## 5.2 Considerações sobre o Capítulo

Neste Capítulo de revisão da literatura, foram exploradas diversas técnicas de poda aplicadas em redes neurais, com um foco particular na análise comparativa entre os estudos identificados. Esta análise comparativa foi essencial para identificar padrões, assim como lacunas nos trabalhos do estado da arte, fornecendo uma base para a pesquisa aqui desenvolvida.

Na Tabela 6, apresentamos um resumo dos principais estudos selecionados, destacando suas abordagens em relação à granularidade da poda, métrica de seleção de neurônios e natureza iterativa do método de poda. Esta Tabela oferece uma visão panorâmica das variações nas abordagens adotadas pelos estudos revisados. Nota-se que a granularidade da poda varia entre neurônios, pesos e camadas, refletindo diferentes níveis de eficiência entre as técnicas abordadas. A métrica de seleção de neurônios a serem podados varia desde critérios baseados em magnitude até métodos mais sofisticados, como a taxa de ativação.

Nesta Tese, buscamos encontrar um método eficiente tanto na redução da rede neural ao qual é aplicado, como também quanto ao esforço computacional para realizar a tarefa de redução das redes. Assim, trabalhos que aplicam a poda sobre os pesos da rede podem não ser tão eficientes, pois mesmo pesos com alto valor de magnitude a princípio poderiam ser removidos se forem constituintes de neurônios menos ativos dentro da rede. Da mesma forma, procuramos apresentar uma abordagem mais abrangente. Portanto, trabalhos que apresentam técnicas aplicáveis a somente um tipo de nó da rede, como filtros por exemplo, não se enquadram adequadamente ao que buscamos. Como buscamos também a eficiência do processo de redução das redes, métodos iterativos, se possível, são evitados, por serem menos eficientes, devido aos muitos retreinamentos para termos a rede final reduzida. Dentre os estudos revisados, destacamos o trabalho apresentado por Hu et al. (2016), que apesar de se tratar de um método iterativo, apresenta características buscadas. Assim, consideramos uma opção interessante, a busca por um método baseado no trabalho de Hu et al. com uma abordagem *One-Shot*, evitando assim, os múltiplos ciclos de treinamento e remoção inerentes ao processo iterativo.

# 6 MÉTODO PROPOSTO E EXPERIMENTOS REALIZADOS

Este Capítulo apresenta a estratégia de redução de redes neurais aplicadas no âmbito dos SCAs, proposta nesta Tese. Também são apresentados os experimentos realizados com o intuito de comprovar o funcionamento e eficiência do método proposto, bem como os *setups* utilizados para implementação destes experimentos. Como veremos, foram realizados experimentos com as principais redes neurais aplicadas à SCA encontradas na literatura. Além disso, é apresentado o *dataset* utilizado como entrada para as redes neurais testadas aqui. Trata-se de um *dataset* largamente utilizado nos trabalhos relacionados a estudos envolvendo aprendizado profundo aplicado à SCA.

## 6.1 Método Proposto

Com base na RSL apresentada na Seção 5.1 foi possível elaborar uma estratégia baseada em técnicas de poda que se adequasse a SCAs. Assim, o objetivo desta Tese consiste em demonstrar que as redes neurais empregadas no âmbito dos ataques a canais laterais podem ter seu tamanho reduzido, e por consequência, ter menor consumo de recursos como uso de memória e tempo de treinamento. Com isso, pretende-se mostrar uma potencial ampliação da ameaça representada pelos ataques conhecidos como DL-SCAs. A possibilidade de realizar ataques com menor custo computacional, ou seja, com menor exigência de memória e tempo de treinamento, pode contribuir para a ampla disseminação desses tipos de ataques, acessíveis também a agressores com recursos computacionais limitados.

Na presente Tese, propõe-se um *framework*, fundamentado no trabalho de Hu et al. (2016), para abordar essa problemática.

Como vimos na Seção 5.1, a abordagem original de Hu et al. (2016) implementa uma técnica de poda, chamada pelos autores de cirurgia, baseada na Porcentagem Média de Zeros (APoZ) calculada através da Equação 30. Esta função mede a porcentagem em que um neurônio fornece uma saída igual a zero durante a etapa de predição, ou seja, as vezes em que o neurônio em questão não é ativado. Sob essa estra-

tégia, a remoção dos neurônios ocorre somente ao final do processo de treinamento. Depois disso, os pesos são atualizados com seus pesos iniciais $\theta_0$ Frankle; Carbin (2018), e o processo se repete. Dessa forma, o processo de poda apresentado por Hu et al. (2016) gera redes neurais menores a cada iteração de poda-treinamento, ajudando a mitigar o *overfitting*. Contudo, o processo em si, acarreta um grande *overhead* devido às ações de treinamento e poda sucessivos.

A partir do exposto, surgem duas questões:

1. É possível obter informações suficientes acerca da identificação dos neurônios a remover (valores aceitáveis de APoZ), treinando-se a rede original por uma quantidade de épocas inferior ao estabelecido para realizar a tarefa para qual ela foi projetada (ou seja, o processo de treinamento da rede neural no contexto de ataques por canais laterais)?

2. É possível realizar a remoção (poda) de todos os neurônios identificados como menos importantes (maior APoZ) de uma única vez (*One-Shot*), eliminando-se os múltiplos treinamentos inerentes ao processo?

### 6.1.1 Análise e identificação dos Neurônios a Remover

O cálculo de APoZ de acordo com o trabalho de Hu et al. (2016), é realizado a partir da predição da rede neural analisada. Pensando nisso, criamos uma função de *callback* para calcular APoZ no final de cada época. Sendo assim, é possível monitorar quais neurônios seriam apontados para remoção a cada época computada. Com isto, propõe-se uma estratégia para buscar responder à primeira questão apresentada na Seção 6.1. Foi desenvolvido um algoritmo que executa as etapas seguintes:

1. Conta-se o número de vezes que cada neurônio foi selecionado para remoção durante as épocas de treinamento.

2. Busca-se os neurônios que apareceram mais vezes durante as épocas de treinamento. Adota-se uma taxa de 75% do número total de épocas de treinamento.

3. Identifica-se em quais épocas os neurônios do passo (2) aparecem.

4. Com base no passo (3) pode-se afirmar que, se temos os mesmos neurônios identificados para remoção por um número específico de épocas (5, por exemplo), então obtivemos os neurônios a serem removidos.

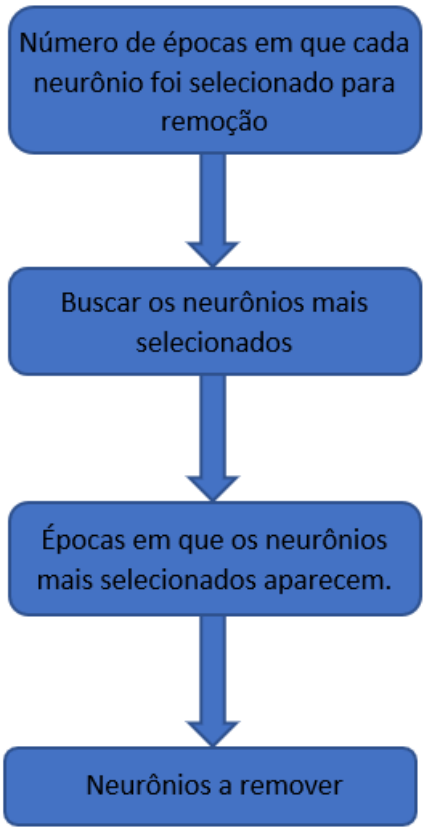


Figura 17 – Fluxo para determinar do número de épocas de treinamento necessários para identificar os neurônios a remover.

Este estudo foi aplicado a diferentes redes, conforme a Seção 6.3. Assim, é possível concluir que a partir de certo momento, os mesmos neurônios são indicados para remoção em épocas consecutivas. Com isso, observa-se que é possível parar o treinamento antes de executar o número de épocas pré-estabelecido para a rede neural, ou ainda, estabelecer uma nova quantidade de épocas para o treinamento da rede (menor que a originalmente recomendada para a execução dos ataques) para obtermos os neurônios a serem removidos através da poda.

### 6.1.2 Procedimento de Remoção dos Neurônios

Conforme discutido na Seção 5.1, embora o método apresentado por Hu et al. (2016) alcance excelentes resultados quanto à redução do tamanho de redes neurais, o processo possui um custo computacional muito elevado. Isso se deve ao fato deste método ser um processo iterativo, em que são realizadas extrações dos neurônios seguidas de treinamentos da rede submetida ao método, repetidas vezes.

Dessa forma, neste ponto do trabalho buscou-se responder à segunda questão da Seção 6.1. Assim, procurou-se aqui desenvolver um método para remover neurônios de redes neurais aplicadas em SCA, reduzindo seu tamanho, com menor esforço computacional em relação ao apresentado na literatura. Para isto, optou-se por uma abordagem *One-Shot*, em que a remoção de todos os neurônios com menor atividade

é realizada de uma vez, evitando-se *loops* de treinamento e remoção, reduzindo-se com isto, o esforço computacional atribuído ao processo de poda.

O método de poda aplicado a redes neurais dedicadas a SCA desenvolvido nesta Tese é apresentado em Lellis; Soares; Perin (2022), e pode ser dividido nas seguintes etapas:

1. Primeiramente, busca-se na literatura uma rede neural consolidada, capaz de realizar SCAs bem sucedidos.

2. Uma vez determinada a rede que será submetida ao processo de poda, define-se um limiar de APoZ que servirá como métrica para decisão de remoção ou não dos neurônios. Através de estudos preliminares vimos que valores acima de 0.6 indicam inatividade, ou seja, neurônios com ativação zero acima de 60% tem uma contribuição insignificante na fase de predição.

3. Em seguida, os pesos iniciais $\theta_0$ da rede neural escolhida são armazenados, e a rede é então treinada. Após o treinamento, o APoZ de cada neurônio é calculado. Lembramos que, conforme vimos na Seção 6.1.1, não é necessário treinar a rede a quantidade total de épocas estabelecida para a realização do ataque.

4. Com base no limiar estabelecido na etapa (2) e no valor de APoZ (calculado na etapa (3)), determina-se quais neurônios devem ser removidos.

5. Realizada-se a remoção dos neurônios selecionados.

6. Para concluir o processo de poda, os pesos iniciais $\theta_0$ armazenados na etapa (2) são atribuídos às conexões resultantes após o processo de poda.

O método proposto, descrito acima, é resumido pelo Algoritmo mostrado na Figura 18:

## 6.2 Base de dados ANSSI - ASCAD

ASCAD (ANSSI SCA Database) [1] é uma base de dados criada para pesquisa em ataques por canais laterais baseados em aprendizado profundo. ASCAD contém dados de treinamento e dados de validação ou teste. Três conjuntos de dados (ou traços) estão disponíveis nestas bases de dados, todos coletados a partir de implementações em *software*:

1. ASCADv1 com chave fixa: ao todo, são 60.000 traços coletados, que representam o consumo de energia de operações de cifração AES executando em um

[1] https://github.com/ANSSI-FR/ASCAD

1: Escolher uma rede neural consolidada para SCA $M$ e armazenar seus pesos iniciais $\theta_0$
2: Treinar $M$:
3: $j \leftarrow 0$; $k \leftarrow 0$
4: **for** ($i = 0$ **to** $total\_neuronios$) **do**
5:  **if** ($get\_apoz(neuronio[i]) > limiar$) **then**
6:  $neuronios\_del[j] \leftarrow neuronio[i]$
7:  $j \leftarrow j + 1$
8:  **else**
9:  $neuronios\_restantes[k] \leftarrow neuronio[i]$
10:  $k \leftarrow k + 1$
11:  **end if**
12: **end for**
13: Deletar os neurônios listados em $neuronios\_del$
14: $remaining\_neurons \leftarrow \theta_0$ (recolocar os pesos iniciais nos neuronios restantes)

Figura 18 – Algoritmo proposto.

microcontrolador da família ATmega. Cada traço possui 100.000 amostras, com texto claro aleatório. O algoritmo AES implementado para este conjunto de dados é protegido contra ataques de primeira ordem através de contramedida por mascaramento (do inglês *masking*).

2. ASCADv1 com chave aleatória: este segundo conjunto de dados consiste de 300.000 traços coletados de uma implementação idêntica à anterior (algoritmo AES protegido contra ataques de primeira ordem). Dentro desse conjunto, 200.000 traços possuem chave de cifração aleatória e os restantes 100.000 traços foram coletados com chave fixa. Assim, o primeiro conjunto é utilizado na fase de treinamento e o segundo conjunto, com chave fixa, é adequada para fases de validação e teste.

3. ASCADv2: esta é uma base de dados maior, consisitindo de 800000 traços coletados pela monitoração do consumo de potência de um microntrolador STM32F303RCT7 executando o algoritmo AES. Nesse caso, a implementação AES é protegida contra ataques de segunda ordem através de contramedidas de mascaremnto e embaralhamento (comumente referenciados pela literatura em língua inglesa como *affine masking* e *shuffling*). Cada traço contem 100.0000 de amostras. Dentre os 800.000 traços coletados, 700.000 foram obtidos com chaves de cifração aleatória e os restantes 100000 traços com uma chave fixa.

Para os três conjuntos de dados ASCAD, o intervalo mensurado de consumo de energia representa as operações referentes à primeira rodada da cifração do AES128.

Para os experimentos realizados nesta Tese, são utilizados os traços disponibilizados pela base de dados do ANSSI, ASCADv1 com chave fixa Prouff et al. (2018). Este conjunto de dados é dividido em subconjuntos de 60.000 traços, conforme descrito abaixo:

- Traços originais, isto é, traços adquiridos e sem aplicação de pré-processamento;
- Extração de 700 POIs dos traços originais com contramedida de mascaramento;
- Extração de 700 POIs dos traços originais com mascaramento e deslocamento aleatório (*jitter*) de até 50 amostras;
- Extração de 700 POIs dos traços originais com mascaramento e deslocamento aleatório (*jitter*) de até 100 amostras;

Os traços do ASCADv1 são rotulados com o terceiro *byte* da chave criptográfica durante a primeira rodada de encriptação, e previamente divididos em 50.000 traços para treino e 10.000 traços para o teste da rede empregada no ataque.

## 6.3 Experimentos Realizados

Esta seção tem como objetivo apresentar todos os experimentos realizados nesta Tese, validando desta forma, o fucionamento e eficiência da metodologia apresentada na Seção 6.1. Através dos resultados obtidos a partir desses experimentos, veremos que é possível reduzir-se o tamanho das redes neurais que realizam SCA, através de um processo menos custoso do que os, até então, apresentados na literatura. Isto nos mostra a possibilidade de expansão dessa classe de ataques, uma vez que para efetuar tais ataques, um usuário mal intencionado necessita de menos recursos computacionais.

### 6.3.1 Aplicação da técnica de Cirurgia em DL-SCA

Como vimos também através da Seção 6.1, o trabalho aqui proposto consiste em uma forma de realização de poda a fim de reduzir o tamanho das redes neural empregadas em DL-SCA. Dentre os trabalhos revisados, nos baseamos no estudo apresentado por Hu et al. (2016). A partir disso, os primeiros experimentos visam adaptar o método empregado por Hu et al., ao contexto de SCA, verificando em primeira instância, se é possível aplicar tais técnicas neste contexto.

Antes de apresentarmos esse conjunto de experimentos, é importante destacarmos que estes foram realizados em um computador Dell Vostro 3681 – M20M 10ª Geração Intel Core i5 8GB 1TB. Tendo como ambiente de desenvolvimento o *Jupyter Notebook* 6.2.0 Kluyver et al. (2016). A linguagem de programação utilizada foi o

*Python* 3.8.5 Van rossum; Drake jr (1995), apoiada pelas bibliotecas mais usuais disponíveis para essa linguagem, além do *Keras* 2.4.3 Chollet (2015) e *TensorFlow* 2.3.0 Abadi et al. (2015).

Assim, inicialmente foram realizados testes a partir de um *framework* para realização de poda bastante similar ao apresentado em Hu et al. (2016). O intuito desses experimentos foi verificar a aplicabilidade da técnica apresentada por Hu et al. em redes projetadas para realizar SCA. O *framework* proposto e baseado em Hu et al. (2016) utilizado em nossos experimentos é mostrado na Figura 19:

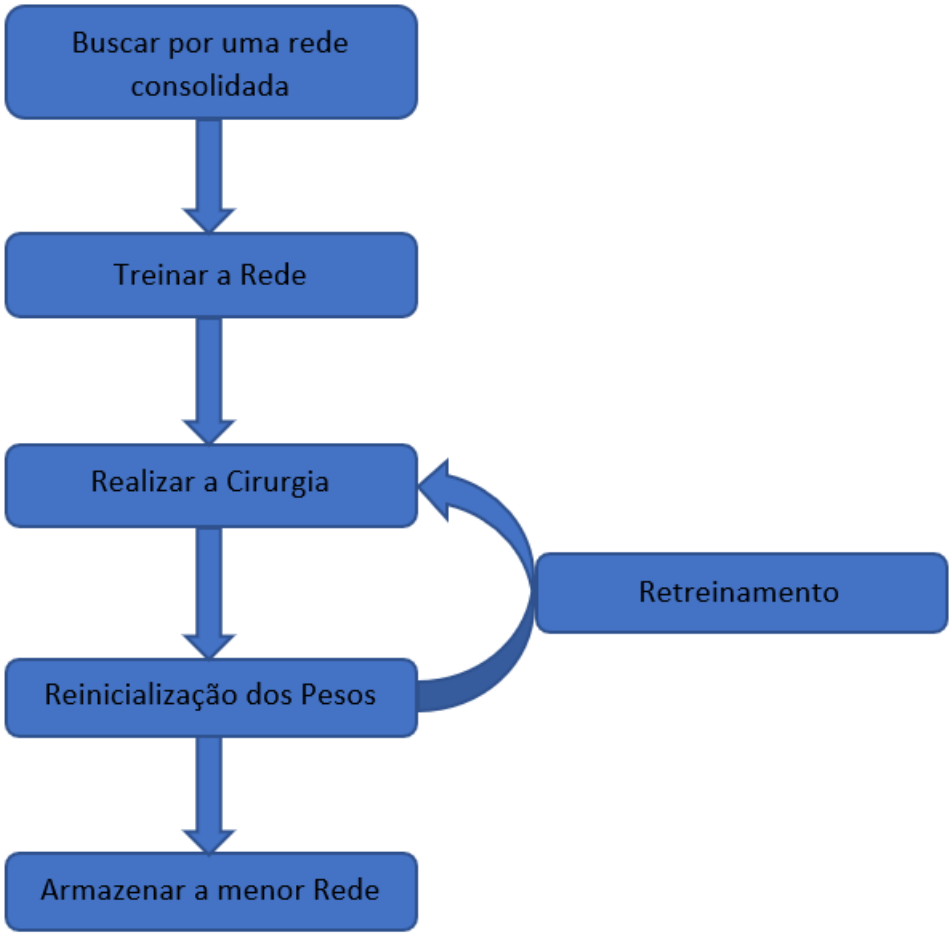


Figura 19 – *Framework* adaptado de Hu et al. (2016) Fonte: Própria

Este *framework* é aplicado as redes MLPs e CNNs como prova de conceito a fim de obter-se redes que possuem fases de treinamento e ataque com esforço computacional reduzido. Como mostrado na Figura 19, o *framework* apresentado é formado pelas seguintes etapas:

1. O primeiro passo é buscar uma rede neural já consolidada e capaz de realizar com sucesso ataques a um conjunto de dados conhecido. Os procedimentos realizados nas etapas seguintes são executados na rede escolhida;

2. A rede neural é treinada pela primeira vez. Nesta etapa, os pesos iniciais são armazenados. Em seguida, um ataque é realizado para validar o treinamento obtido;

3. A partir daí, o procedimento cirúrgico é realizado, removendo os neurônios da rede de acordo com suas taxas de ativação (APoZ) de acordo com a Equação 30 Hu et al. (2016);

4. A rede resultante é reinicializada com os pesos iniciais ($\theta_0$), retreinada, e realizado um ataque usando essa rede resultante;

5. Se o ataque da etapa (4) for bem-sucedido, as etapas (3) e (4) serão executadas novamente. Esse loop é repetido $n$ vezes até que o ataque na etapa (4) não seja mais bem-sucedido;

6. Finalmente, os pesos restantes na rodada $n-1$ do loop descrito, são restaurados com os pesos iniciais $\theta_0$. Obtendo-se assim, a menor rede capaz de executar um SCA sobre o conjunto de dados apresentado.

Os experimentos realizados nesta etapa utilizaram redes neurais apresentadas em dois estudos recentes encontrados na literatura Zaid et al. (2020)Prouff et al. (2018). Como entrada, o conjunto de dados ASCADv1 Prouff et al. (2018) apresentado na Seção 6.2 foi utilizado.

Abaixo, temos uma descrição das redes neurais utilizadas nestes primeiros experimentos. Lembrando que maiores detalhes sobre tais redes podem ser encontrados em Prouff et al. (2018) e Zaid et al. (2020).

A MLP apresentada por Prouff et al. Prouff et al. (2018) é utilizada nos primeiros experimentos. Esta NN tem uma estrutura relativamente pequena em comparação com outras redes utilizadas em SCA. O tamanho dessa rede depende dos dados de entrada. Prouff et al. (2018) utiliza o *dataset* ASCADv1, onde os traços são obtidos de uma implementação em *software* do algoritmo criptográfico AES que conta apenas com uma contramedida de mascaramento (contramedida de primeira ordem). A tarefa executada pela rede é menos complexa do que atacar dispositivos protegidos com uma contramedida temporal, fato que permite uma rede menor para executar o ataque. A rede de Prouff et al. Prouff et al. (2018) é composta pelas seguintes camadas:

- Uma camada de entrada densa contém 700 entradas de acordo com as amostras dos traços de destino disponíveis no conjunto de dados. Esta camada é adaptada a 100 neurônios, como será discutido mais adiante (200 na rede original). A função de ativação nesta camada é a ReLU;

- Quatro camadas ocultas densas com 100 neurônios. Aqui também foi feita uma adaptação como será explicado a seguir. Para essas camadas, a função de ativação usada também é a ReLU;

- Finalmente, uma camada de saída com 256 neurônios em relação às 256 possibilidades de classificação para o *byte* da chave secreta. Para esta camada, a função de ativação empregada é *Softmax*;

A taxa de aprendizado é 0,00001 com o otimizador *RMSprop*. A descrição dos hiperparâmetros desta rede também pode ser encontrada em Prouff et al. (2018). Embora essa rede seja pequena, ela ainda é superdimensionada para a tarefa a ser executada. Isso será demonstrado com os resultados obtidos a partir do processo cirúrgico descrito acima. No entanto, vale ressaltar que antes de realizar esse processo, o número de neurônios das camadas ocultas foi reduzido de 200 (da rede original) para 100, pois, de antemão, notamos um superdimensionamento da rede apresentado por Prouff et al. Prouff et al. (2018).

Como mencionado anteriormente, os experimentos iniciais também foram aplicados a uma CNN proposta por Zaid et al. (2020). Neste caso, temos uma rede mais complexa capaz de realizar um ataque a dispositivos equipados com contramedidas de tempo. Esta rede é basicamente formada pelas seguintes camadas:

- Uma camada de entrada densa contendo as 700 entradas e a função de ativação SeLU;

- Três blocos contendo uma camada Convolucional 1D, uma camada de *Batch Normalization* e uma camada de *Average Pooling*. Essas camadas têm respectivamente 32, 64 e 128 neurônios e sua função de ativação é SeLU;

- Em seguida, há uma camada de achatamento (*Flatten*) e três camadas densas compostas por 20 neurônios com função de ativação SeLU cada uma;

- Para a camada de saída temos uma camada com 256 neurônios para as 256 possibilidades de valores que um *byte* da chave criptográfica pode assumir. Para esta camada, a função de ativação empregada também é *Softmax*;

Como na MLP de Prouff et al., a taxa de aprendizagem é 0,00001. No entanto, o otimizador usado aqui é *Adam*. Mais informações sobre esta rede podem ser encontradas em Zaid et al. (2020).

Para a CNN de Zaid et al. (2020), também é utilizado o conjunto de dados ASCADv1 Prouff et al. (2018). Dentro deste *dataset*, para esta rede é utilizado um subconjunto de traços oriundos de um dispositivo protegido por uma contramedida de deslocamento de tempo aleatório de até 100 amostras.

A partir do exposto, inicialmente foram realizados experimentos aplicando-se o *framework* proposto às redes neurais MLP e CNN já caracterizadas. Os resultados obtidos são mostrados na Figura 20. A métrica usada para avaliação da eficiência da rede neural é a classificação média da chave ou a entropia de adivinhação Standaert; Malkin; Yung (2006) (do inglês, *guessing entropy*). A classificação média mede os traços necessários para um ataque bem-sucedido. Uma classificação média igual a zero garante que a chave correta seja recuperada com êxito.

A Figura 20 (a) apresenta os valores médios de classificação obtidos a partir da MLP original, nossa linha de base, representada em azul. Os valores médios de classificação para a rede reduzida, após a aplicação do Algoritmo **??** aparecem em laranja. Os pontos identificam quando a classificação média chega a zero. Embora o desempenho do ataque seja bastante semelhante, a rede reduzida apresenta uma ligeira melhoria, atingindo o rank médio 0 com aproximadamente 700 traços de ataque, enquanto a linha de base atinge essa marca com mais traços de ataque. A Figura 20 (a) também mostra a classificação tanto para a CNN original quanto para a rede reduzida. Aqui há uma pequena diferença no número de traços para atingir a classificação média 0: 325 traços para a CNN original e 250 com a rede reduzida. Isso sugere que as redes reduzidas podem realizar ataques bem-sucedidos mesmo depois de serem reduzidas. Também em termos de parâmetros treináveis, a MLP começa com 352456 e termina com 234376, representando uma redução de 33,5%. Na CNN, a redução dos parâmetros treináveis passa de 142044 para 59126, o que representa uma redução de 58,37%. Os tempos de treinamento medidos para as redes originais e as redes reduzidas, excluindo o tempo de execução do método cirúrgico, revelam uma redução de 23,4 e 36,33% no tempo de treinamento, respectivamente, para os casos de MLP e CNN.

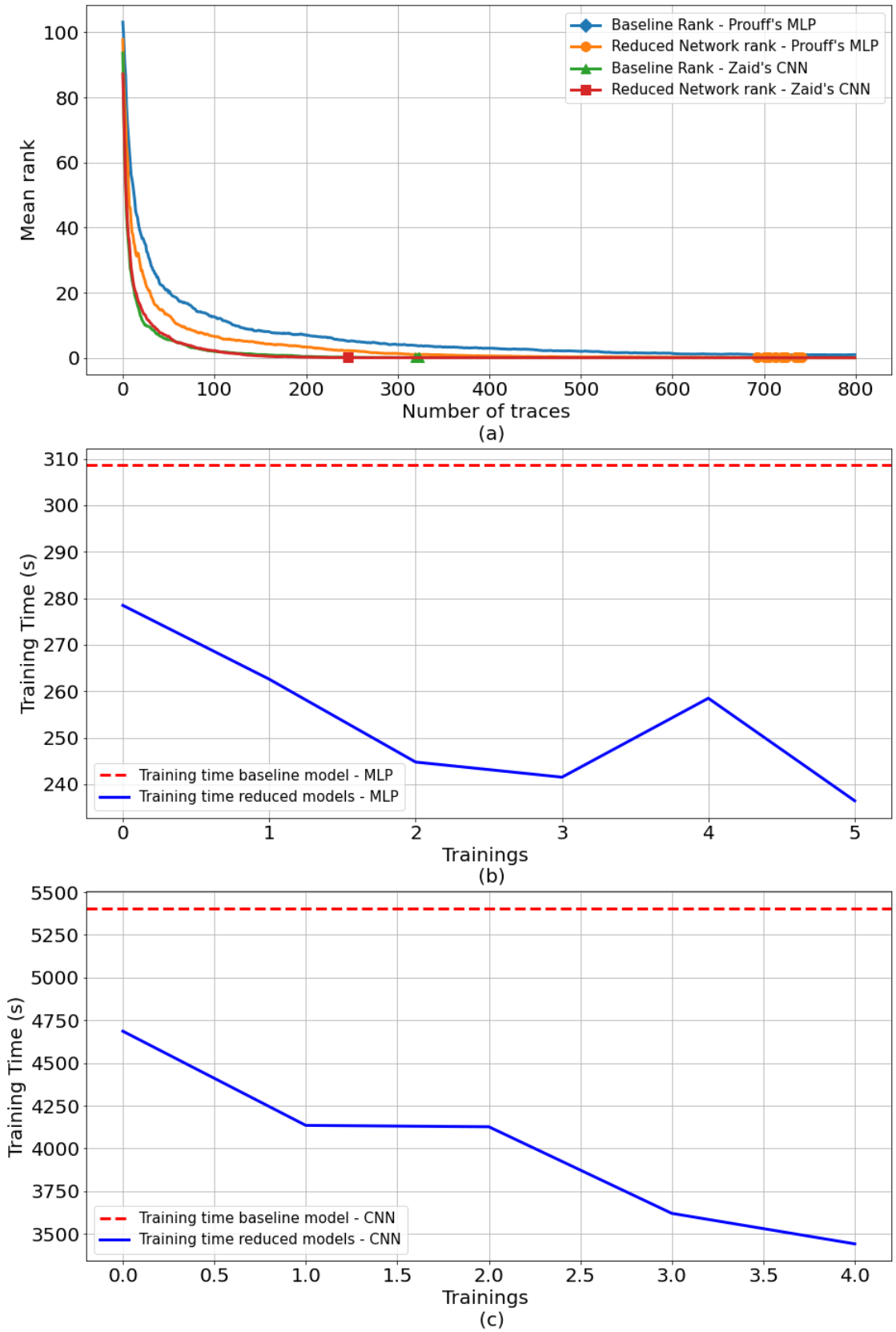


Figura 20 – *Rank* médio vs. Traços do consumo – MLP de Prouff Prouff et al. (2018) e CNN de Zaid Zaid et al. (2020) (a) e Tempos de treinamento das redes – MLP (b) e CNN (c) – Método original de cirurgia Hu et al. (2016) Fonte: Própria

Tais resultados são relevantes para o SCA principalmente porque os ataques po-

dem ser executados mais rapidamente, exigindo menos recursos computacionais e tornando-se mais ameaçadores. No entanto, o processo cirúrgico é bastante longo devido ao treinamento iterativo. A Figura 20 (c) mostra o tempo para 5 treinamentos: (i) em vermelho, para CNN original; (ii) em azul, para a CNN reduzida iterativa. Em SCA, esse tempo é justificável, uma vez que o ataque revela apenas um *byte*, e é muito provável que a mesma rede seja capaz de revelar outros *bytes* da chave.

### 6.3.2 Método de Cirurgia de Extração Única

Embora a técnica de cirurgia tenha alcançado resultados significativos, sentimos a necessidade de aprimorar o processo como um todo. Conforme acabamos de revisar, são necessários muitos ciclos de treinamento e poda para atingirmos a rede ideal, 5 para a MLP e 4 para a CNN, sendo que o treinamento das CNNs é mais custoso geralmente. Dessa forma, buscamos um método para realizar a poda das redes de uma única vez (*One-Shot*), sem a necessidade de ciclos de treinamento e poda.

Portanto, como apresentado em Lellis; Soares; Perin (2022) e descrito na Seção 6.1, aqui abordamos um método de poda através da técnica de cirurgia baseado em Hu et al. (2016), que não necessita de múltiplos treinamentos.

Para estes experimentos, as redes utilizadas como referência são a MLP Prouff et al. (2018) e a CNN Zaid et al. (2020), cujas características foram apresentadas anteriormente. Destas, seis redes foram criadas, sendo 3 MLPs: (i) MLP de Prouff et al. (2018), (ii) MLP v2 - baseado em (i), mas com 300 neurônios nas camadas densas ao invés de 200, e (iii) MLP v3 - também baseado em (i), mas com uma camada densa extra. Além dessas, foram utilizadas 3 CNNs: (i) CNN de Zaid et al. (2020), (ii) CNN v2 - com base em (i), mas com o dobro de neurônios nas camadas totalmente conectadas em comparação com (i); e (iii) CNN v3 - também com base em (i), mas com uma camada extra totalmente conectada.

O método proposto é avaliado em um segundo estudo de caso, onde apenas uma operação cirúrgica é realizada, ao invés de remoções iterativas de neurônios, como sugerido em 16. Para validar o método, as três versões de MLPs descritas anteriormente foram usadas. Primeiro, verificamos se as NNs continuaram a realizar ataques bem-sucedidos após a remoção dos neurônios.

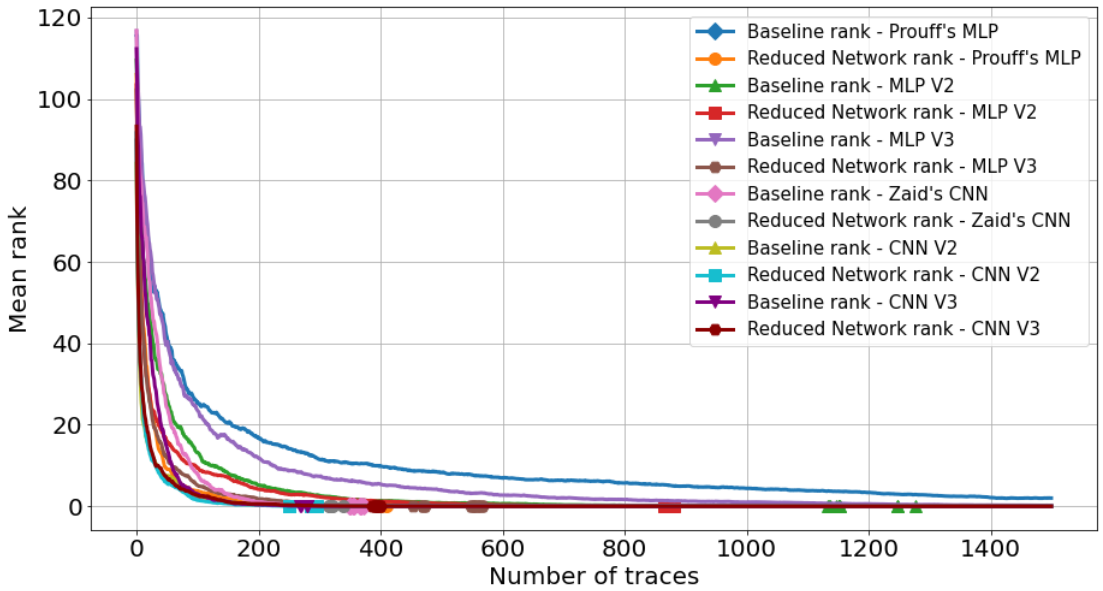


Figura 21 – *Rank* médio vs. Traços do consumo – Método proposto. Fonte: Própria

A Figura 21 mostra o desempenho do ataque medido com a classificação média para as três MLPs, onde observamos que a eficiência dos ataques é muito próxima dos modelos originais de linha de base. Todas as redes reduzidas atingem a classificação média igual a 0 mais cedo do que suas respectivas linhas de base. Na Figura 21, a MLP de linha de base não atinge a classificação média 0 com pelo menos 1400 traços. Enquanto isso, as redes reduzidas são capazes de recuperar a chave correta com o uso de 500 traços. Portanto, as melhorias são verificadas para as redes reduzidas MLP, MLP v2 e MLP v3.

Em seguida, o método proposto também é aplicado às três CNNs descritas anteriormente. As classificações médias obtidas para as redes CNN, CNN v2, CNN v3 reduzidas correspondentes são semelhantes, com pequenas melhorias nas redes reduzidas em comparação com suas CNNs originais, pois as redes reduzidas atingem a classificação média 0 antes das redes originais, como podemos ver através da Figura 21.

A Tabela 7 resume a comparação entre o método cirúrgico original e o método proposto, quando consideradas as redes MLP de Prouff Prouff et al. (2018) e CNN de Zaid Zaid et al. (2020). Destaca a redução do tempo de treinamento e do número de parâmetros treináveis obtidos em termos percentuais. Além disso, apresenta o tempo de processamento para a execução completa de cada método. Obtivemos melhorias significativas em todos os casos. O método cirúrgico melhorado é 9 vezes mais rápido do que o método original para MLP e 6,38 vezes mais rápido para o caso da CNN.

A Tabela 8 apresenta os resultados obtidos com o método proposto nas seis NNs previamente definidas. Destaca, em termos percentuais, as reduções obtidas no tempo de treinamento e no número de parâmetros treináveis. Observa-se que os percentuais de redução tanto para MLPs quanto para CNNs são próximos. Olhando para

Tabela 7 – Comparação entre os métodos de cirurgia original e o proposto (nosso).

| | **MLP Prouff et al. (2018)** | | **CNN Zaid et al. (2020)** | |
|---|---|---|---|---|
| **Reduções** | **Hu et al. (2016)** | **Nosso** | **Hu et al. (2016)** | **Nosso** |
| Tempo (%) | 23.4 | 28.52 | 36.33 | 48.33 |
| Param. | 33.5 | 41.29 | 58.37 | 70.6 |
| **Tempo Proc. (s)** | **1522.22** | **169.05** | **20010.98** | **3135.38** |

os resultados obtidos para as redes MLP, a melhor redução do tempo de treinamento, 28,52%, é alcançada para o MLP proposto por Prouff et al. (2018). Em comparação, a redução mais significativa no número de parâmetros treináveis, 44,7%, ocorreu na MLP v2. Enquanto isso, nas redes CNN, temos uma redução mais significativa no tempo de treinamento, 51,57% para CNN v2, e uma redução maior no número de parâmetros, 76,36%, é observada na CNN v3.

Tabela 8 – Reduções nas Redes Neurais - Método Proposto.

| **Rede** | **Redu. de Tempo (%)** | **Redu. em Param. (%)** |
|---|---|---|
| **MLP Prouff et al. (2018)** | 28.52 | 41.29 |
| **MLP V2** | 28.15 | 44.7 |
| **MLP V3** | 20.08 | 41.1 |
| **CNN Zaid et al. (2020)** | 48.33 | 70.6 |
| **CNN V2** | 51.57 | 70.28 |
| **CNN V3** | 46.41 | 76.36 |

Assim, podemos concluir que o método cirúrgico, especialmente sua versão *One-Shot*, é bastante efetivo para reduzir o tamanho da rede neural, levando a tempos de treinamento mais curtos e tornando as redes neurais mais eficientes contra ataques de perfil. O método proposto mostrou-se bem-sucedido de modo que o tempo gasto no processo cirúrgico foi reduzido 9 vezes para a rede MLP e 6,38 vezes para a rede CNN em comparação com a aplicação direta do processo cirúrgico, conforme proposto na literatura de aprendizagem profunda Hu et al. (2016).

### 6.3.3 Aceleração das CNNs Reduzidas por Esparsidade

Apesar de alcançarmos bons resultados em termos de redução das redes neurais aplicadas à SCA em Lellis; Soares; Perin (2022), o método utilizado não é aplicável às camadas *Batch Normalization* presentes na CNN proposta por Zaid et al. Zaid et al. (2020), e utilizada como estudo de caso nestes experimentos. Portanto, com o intuito de encontrarmos a CNN mais rápida, em termos de tempo de treinamento, possível a partir de Zaid et al. (2020), buscamos uma técnica que de alguma forma acelerasse o processamento realizado por esse tipo de camadas.

A partir de buscas na literatura, verificamos que no trabalho apresentado por Yu et al. (2021) os autores propõem uma abordagem de poda global *One-Shot* chamada *Gate Trimming* (GT) que estima globalmente o efeito dos canais das camadas de *Batch Normalization* (BN) através de uma métrica denominada Informação de Ganho (*Information Gain* - IG). Assim, se um canal de uma camada BN possui IG alto, ele tem uma maior contribuição para a saída da rede, e sua exclusão causará grandes danos no desempenho. Em contraste, podar um canal de uma camada BN com o menor IG causará pouco efeito na rede. A poda dos canais BN é feita zerando-se seus fatores de escala e fatores de deslocamento.

Com base no estudo de Yu et al. (2021), foram realizados experimentos a fim de incorporar uma etapa adicional ao nosso fluxo de redução de CNNs aplicadas a SCA. Ou seja, após aplicarmos o método apresentado em Lellis; Soares; Perin (2022), foi aplicado o método de Yu et al. (2021) para alcançar uma maior aceleração da CNN de Zaid et al. (2020).

Como resultado, observamos uma redução adicional de 5,84% no tempo de treinamento da rede reduzida anteriormente através de Lellis; Soares; Perin (2022). Assim, incorporando esta etapa, alcançamos uma redução total de 51,35% no tempo de execução do ataque em relação à rede original proposta por Zaid et al. (2020). Ainda, uma melhora na performance do ataque foi observada, uma vez que a quantidade de traços necessários para revelar a chave criptográfica com a CNN reduzida por Lellis; Soares; Perin (2022) que era de 400 passou para 292, conforme podemos ver na Figura 22.

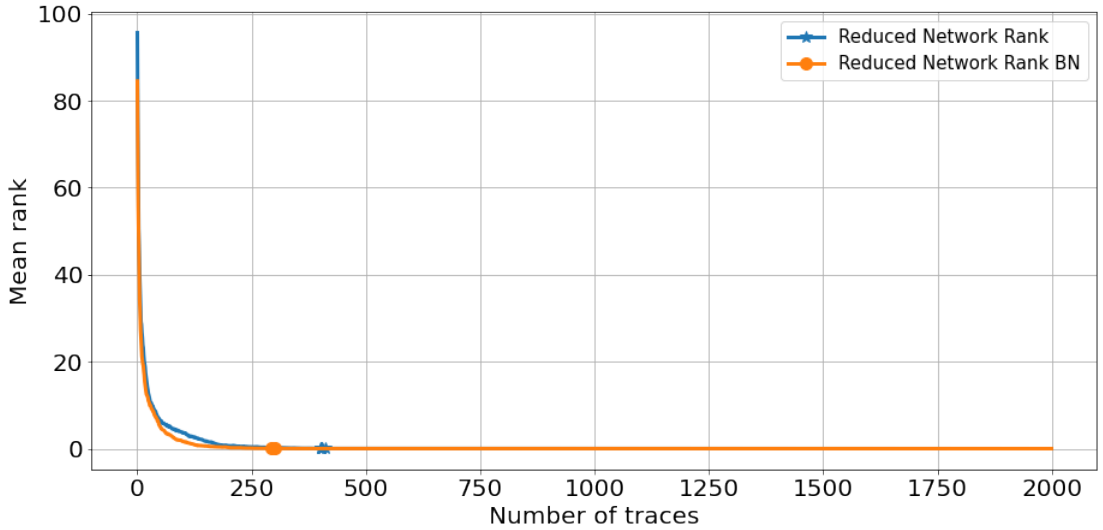


Figura 22 – *Rank* médio vs. Traços do consumo para CNN – Método de Lellis; Soares; Perin (2022) (azul) Método proposto com etapa adicional de aceleração (laranja). Fonte: Própria

Cabe ressaltar que o tempo gasto no processo de cirurgia teve um aumento de apenas 0,0018%, ou seja, a etapa incorporada ao fluxo de redução introduz um *overhead* desprezível ao processo como um todo. Com isto, concluímos que é bastante inte-

ressante incorporar a aceleração proposta por Yu et al. (2021) ao método de redução de redes neurais apresentado em Lellis; Soares; Perin (2022) para CNNs aplicadas à SCA.

## 6.4 Análise Detalhada do Método Proposto

Até aqui, foram realizados testes que mostraram ser possível a adaptação da proposta de Hu et al. (2016) ao contexto de SCAs. Como vimos na Seção 6.3.2, é possível aplicar uma abordagem *One-Shot* baseada em Hu et al. (2016) sobre redes neurais utilizadas para realizar esses tipos de ataques. Com isso, é possível obter resultados que mostram reduções em termos de tamanho das redes, ou seja, uma menor quantidade de parâmetros treináveis e menor tempo de treinamento necessários para realizar um ataque bem sucedido. Nesta Seção, além destas análises iniciais, procuramos estabelecer uma relação quanto à quantidade de traços utilizados no ataque entre as redes originalmente propostas e suas respectivas redes reduzidas. Ainda, realizamos um estudo com relação à quantidade de épocas de treinamento necessárias para realizar-se os ataques, para as redes encontradas na literatura, antes e depois do processo de cirurgia.

Partindo destas constatações, buscou-se extender os experimentos aplicando-se a abordagem aqui proposta a quatro redes neurais encontradas na literatura e consolidadas na comunidade de SCA. Assim, para esta rodada de experimentos, testamos nosso método em duas redes MLP: Uma rede MLP apresentada em Perin; Picek (2020) com 9.966.256 parâmetros, e também uma MLP com tamanho menor $MLP_{best}$ revisada nos experimentos anteriores, proposta por Prouff et al. (2018), composta por 352.456 parâmetros. Também foram realizados testes com duas redes neurais convolucionais (CNNs): A rede chamada $CNN_{best}$ também proposta por Prouff et al. (2018), contendo 66.652.544 parâmetros, assim como uma CNN menor encontrada em Perin; Picek (2020). Esta CNN contém 6.797.246 parâmetros. Ou seja, nesta Seção foram realizados experimentos que aplicam o método de cirurgia *One-Shot* nos dois tipos de redes neurais mais recorrentes na área de SCA (MLPs e CNNs), conforme constatados através da RSL apresentada no Capítulo 4, testando MLPs e CNNs de diferentes tamanhos (uma rede maior e uma rede menor de cada tipo).

Como a rede $MLP_{best}$ de Prouff et al. (2018) já foi apresentada na Seção 6.3, nesta Seção são apresentadas apenas as configurações das outras redes usadas por Perin; Picek (2020) e Prouff et al. (2018). Inicialmente é apresentanda a arquitetura da rede MLP usadas por Perin; Picek (2020):

- Uma camada de entrada densa contendo as 700 entradas e a função de ativação ReLU;

- Dez camadas ocultas densas com 1000 neurônios cada. Para essas camadas, a função de ativação utilizada também é a ReLU;

- Por último, uma camada de saída com 256 neurônios, correspontentes às 256 classes possíveis, conforme descrito na Seção 6.3. Esta camada possui a função de ativação *Softmax*;

Cabe aqui ressaltar que esta rede neural possui uma taxa de aprendizado de 0,001 com um *batch size* de 400. O trabalho apresentado em Perin; Picek (2020) realiza um estudo relacionado à diferentes otimizadores. Para este trabalho, escolheu-se trabalhar com *RMSProp*.

A seguir, apresentamos a arquitetura da CNN presente no trabalho de Perin; Picek (2020):

- Uma camada de entrada densa contendo as 700 entradas e a função de ativação. ReLU;

- Quatro camadas convolucionais 1D com respectivamente 10, 20, 40 e 80 neurônios. O *kernel* de cada uma dessas camadas é setado com tamamho 4, e sua função de ativação é a ReLU;

- Depois, temos quatro camadas ocultas totalmente conectadas de 1000 neurônios cada. Essas camadas também tem como função de ativação ReLU;

- A camada de saída é composta de 256 neurônios com função de ativação de *Softmax*.

Esta rede tem como otimizador o *RMSPop*. A taxa de aprendizado é igual a 0,001 e o *batch size* é igual a 400.

Por fim, descrevemos as configurações da rede $CNN_{best}$ de Prouff et al. (2018):

- Uma camada de entrada densa contendo as 700 entradas e a função de ativação. ReLU;

- Em seguida, são implementados cinco blocos com uma camada convolucional 1D com filtros de tamanhos: 64, 128, 256, 512 e 512. O tamanho para todos os *kernels* dos filtros dessas camadas é 11. A função de ativação para estas camadas é a ReLU. Ainda, dentro de cada bloco existe uma camada de agrupamento *Average Pooling*, com *strides* igual a dois;

- Depois dos blocos convolucionais, temos duas camadas densas contendo 4096 neurônios cada uma. Estas camadas também contam com a função de ativação ReLU;

- Por fim, assim como nas outras redes apresentadas, esta também conta com uma camada de saída com 256 neurônios e função de ativação *Softmax*.

Para a rede acima descrita, a taxa de aprendizado é igual a 0,00001 e o otimizador utilizado foi o *RMSProp*. O *batch size* definido por Prouff et al. (2018) é igual a 200.

Antes de apresentar os experimentos realizados nesta etapa do trabalho, gostaríamos de mencionar que o *setup* utilizado é constituído de um PC equipado com uma CPU contendo um processador *Intel Core* i7 - 12700K operando a uma frequência de relógio de 3.60 GHz, com 16GB de memória RAM. Este PC também possui uma placa de processamento gráfico *NVIDIA* RTX 3070. Além disso, para as redes descritas nessa Seção, a função de perdas definida foi a $categorical\_crossentropy$.

### 6.4.1 Identificação dos Neurônios a Remover

A etapa inicial consiste em descobrir qual o número de épocas de treinamento é necessário para cada uma das redes de modo a corretamente identificar os neurônios a serem removidos. Através dessa etapa, é possível ver que não é necessário realizar o treinamento completo das redes, isto é, um treinamento longo com o mesmo número de épocas considerados nos artigos de referência para descobrirmos quais neurônios são menos relevantes, e que portanto podem ser removidos. Assim, aplicamos o estudo descrito na Seção 6.1.1 para cada uma das redes da Seção 6.4. A partir desse processo, obtivemos como resultado as quantidades de épocas apresentadas na Tabela 9.

Tabela 9 – Resultados do Estudo de Épocas para Selecionar os Neurônios a Remover.

| Rede | Épocas sugeridas | Épocas estudo |
|---|---|---|
| **CNN Prouff** | 130 | 128 |
| **CNN Perin** | 400 | 350 |
| **MLP Perin** | 200 | 149 |
| **MLP Prouff** | 200 | 113 |

Assim, nos experimentos a seguir, as redes neurais foram treinadas pelos respectivos números de épocas apresentados na Tabela 9.

### 6.4.2 Análise de MLPs Aplicadas à SCA

Uma vez determinada as quantidades de épocas para treinar as redes neurais, a etapa seguinte nesta Tese consistiu em aplicar o método de poda proposto à MLP de Prouff et al. (2018). Portanto, foram realizados 10 experimentos utilizando-se como entrada o *dataset* ASCADv1, composto por 50.000 traços de 700 amostras oriundos de um dispositivo criptográfico dotado de contramedida de mascaramento aditivo de

primeira ordem. Detalhes sobre esse conjunto de dados são apresentados na Seção 4.

Dessa forma, a Tabela 10 apresenta os valores de redução em termos de tempo de treinamento, quantidade de parâmetros e também na quantidade de traços da MLP original com relação à esta MLP reduzida para os dez experimentos realizados nesta etapa. Além desses dados, a Tabela 10 traz o tempo de execução do processo de redução da rede neural.

Os tempos de treinamento da rede neural reduzida atingem reduções de 8.45 até 30.26% em relação à original, ou seja, os dez experimentos realizados obtiveram reduções em termos de tempo de treinamento. Para alcançar essas reduções, os tempos de processamento de redução dessa rede em particular estão entre 143.63s a 279.51s. Como explicado na Seção 6.1, a técnica utilizada para remoção dos neurônios, e consequente redução da rede neural, é realizada de uma só vez (*One-Shot*), o que justifica os baixos tempos obtidos no processo de redução. As reduções temporais alcançadas neste experimento estão apoiadas pelas reduções na quantidade de parâmetros da rede reduzida em relação à rede original. A Tabela 10 também apresenta as reduções na quantidade de parâmetros para os dez experimentos, que vão de 28.87 à 38.95%. Conforme vemos na Tabela 10, todos os experimentos aqui realizados apresentam reduções significativas em termos de parâmetros aplicando-se o método proposto.

As redes neurais reduzidas também apresentaram melhor desempenho quanto à quantidade de traços necessários para obtenção de ataques por canais laterais bem sucedidos. A última coluna da Tabela 10 mostra os valores de redução na quantidade de traços. Sob esse paradigma, temos reduções que vão de 52.25 à 81.35%. Assim, vemos que as redes neurais reduzidas necessitam de pelo menos 50% de traços a menos para realizar os SCAs. Isso traduz-se em um modelo de ataque de perfil mais potente, permitindo um atacante recuperar a chave com menos traços do dispositivo alvo.

Tabela 10 – Reduções na MLP de Prouff - Método Proposto.

| Tempo (s) | | | | Parâmetros | | | Traços | | |
|---|---|---|---|---|---|---|---|---|---|
| **Proc.** | **Orig.** | **Nossa** | **Redu. (%)** | **Orig.** | **Nossa** | **Redu. (%)** | **Orig.** | **Nossa** | **Redu. (%)** |
| 148.82 | 270.58 | 199.27 | 26.35 | 352,456 | 235,607 | 33.15 | 1999 | 848 | 57.58 |
| 143.63 | 260.00 | 216.75 | 16.64 | 352,456 | 221,520 | 37.15 | 1937 | 570 | 70.57 |
| 153.33 | 276.96 | 229.15 | 17.26 | 352,456 | 230,589 | 34.58 | 1781 | 683 | 61.65 |
| 158.59 | 282.88 | 258.99 | 8.45 | 352,456 | 224,161 | 36.4 | 1988 | 463 | 76.71 |
| 177.73 | 317.69 | 234.04 | 26.33 | 352,456 | 224,718 | 36.24 | 1999 | 508 | 74.79 |
| 189.76 | 337.74 | 251.75 | 25.46 | 352,456 | 233,495 | 33.75 | 1980 | 871 | 56.01 |
| 166.49 | 293.62 | 254.02 | 13.49 | 352,456 | 250,713 | 28.87 | 1995 | 372 | 81.35 |
| 179.21 | 324.02 | 268.64 | 17.09 | 352,456 | 236,238 | 32.97 | 1997 | 758 | 62.04 |
| 279.51 | 502.75 | 434.68 | 13.54 | 352,456 | 223,017 | 36.72 | 1354 | 582 | 57.02 |
| 174.89 | 314.34 | 219.22 | 30.26 | 352,456 | 215,175 | 38.95 | 1952 | 932 | 52.25 |

Sob o ponto de vista de *rank* da chave criptográfica, pudemos perceber que as redes neurais reduzidas neste experimento apresentam valores *rank* mínimo da chave criptográfica menores ou iguais à rede original. Dessa forma, segundo esse quesito, de um modo geral, as redes neurais reduzidas apresentam reduções com relação ao *rank* mínimo da chave. Com isto, podemos concluir que a MLP, após passar pelo processo de reudução, ou cirurgia, são ainda mais eficientes do que a MLP original. Isso é mostrado na Tabela 11.

Tabela 11 – Ranks da MLP de Prouff - Método Proposto.

| **Rank Min. Original** | **Rank Min. Reduzidas** |
|---|---|
| 0.075 | 0.0 |
| 0.005 | 0.0 |
| 0.015 | 0.0 |
| 1.03 | 0.0 |
| 0.205 | 0.0 |
| 0.125 | 0.0 |
| 0.125 | 0.0 |
| 0.055 | 0.0 |
| 0.0 | 0.0 |
| 0.015 | 0.0 |

A Tabela 11 confirma a afirmação de que as MLPs reduzidas tem uma eficiência maior do que a MLP original. Nesta Tabela, podemos ver que em apenas um dos 10 experimentos executados sobre as MLPs de Prouff et al. (2018) a rede proposta pelos autores consegue atingir *rank* zero da chave criptográfica, enquanto que para todos os experimentos as redes reduzidas atingiram esse *rank*.

Essa maior eficiência das redes neurais reduzidas, MLPs nesse caso, pode ser vista plotando-se os gráficos de *rank* da chave criptográfica das redes reduzidas com a rede original para cada um dos dez experimentos executados nesta etapa. Podemos ver isso através da Figura 23.

Na Figura 23, os gráficos dos *ranks* da MLP original para os dez experimentos realizados (*baselines*) estão representados na cor azul, enquanto os gráficos correspondentes para as redes reduzidas encontram-se em vermelho. Observando-se a Figura 23 percebe-se que, de um modo geral, as curvas dos *ranks* x traços de consumo para as MLPs reduzidas estão posicionadas mais abaixo do que suas respectivas curvas da rede original, em relação ao eixo $y$, eixo representante dos *ranks* das chaves criptográficas. Esta Figura ilustra os resultados de melhor eficiência das redes reduzidas geradas com relação à MLP original.

Em resumo, com base nos experimentos, aplicando-se o método proposto nesta Tese à MLP proposta por Prouff et al. (2018), podemos concluir que as redes reduzi-

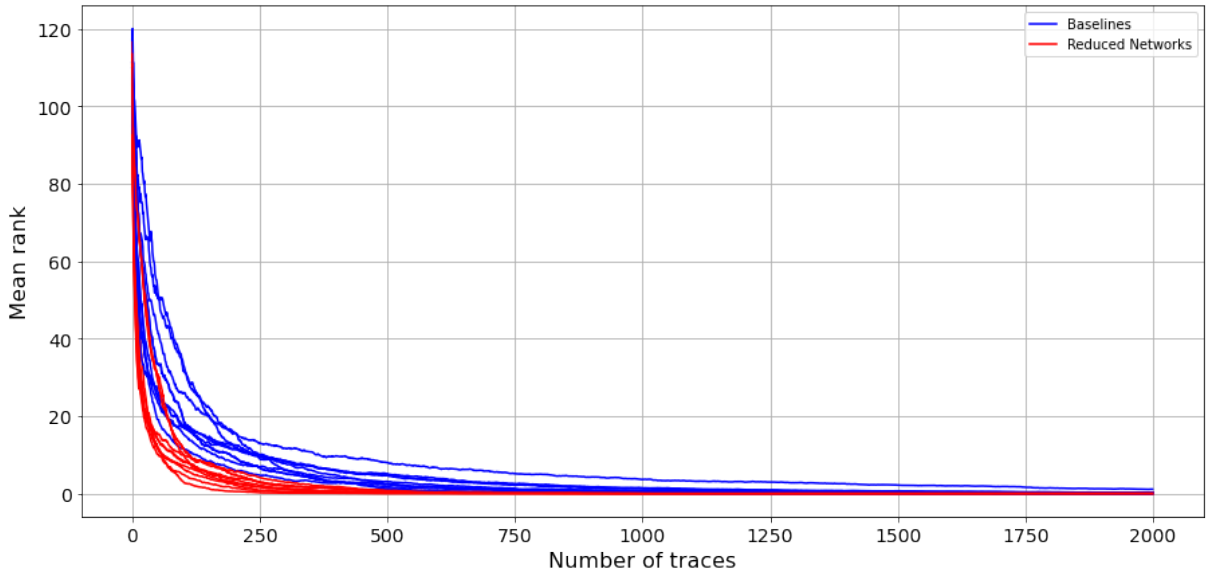


Figura 23 – Ranks MLP de Prouff - 10 execuções com a rede original (azul) e 10 execuções com as redes reduzidas (vermelho).

das resultantes do processo proposto possuem menores tamanhos em relação à MLP de Prouff et al. (2018) em termos de quantidade de parâmetros treináveis. Como consequencia, o tempo de treinamento para essas redes na realização de SCAs também são menores em relação à respectiva rede original. Com relação à quantidade de traços necessários para ataques bem sucedidos, vemos que as MLPs reduzidas tem melhores resultados do que a rede original. Isto deve-se, conforme vimos através da Tabela 11 e do gráfico da Figura 23, ao menor *rank* da chave criptográfica obtido com as redes reduzidas com relação à MLP original.

Através da revisão apresentada no Capítulo 5.1, foi possível perceber que alguns trabalhos encontrados na literatura sobre métodos de poda utilizam métricas de seleção dos neurônios a serem removidos diferentes de APoZ (utilizada nesta Tese). Dentre eles, o método mais recorrente utiliza a norma L1 para determinar a importância dos neurônios Li et al. (2017). A norma L1, consiste em calcular a soma dos valores absolutos dos pesos de cada neurônio. Esta métrica é correlacionada à ativação dos neurônios, pois conforme revisamos no Capítulo 3.1 a ativação de um neurônio se dá através da submissão do somatório dos seus pesos multiplicados por suas entradas à uma função de ativação. Nesta Seção são apresentados experimentos aplicando-se o método de poda proposto à MLP de Prouff et al. (2018), utilizando a métrica L1 para seleção e remoção de neurônios. Os resultados são apresentados na Tabela 12.

Tabela 12 – Reduções na MLP de Prouff utilizando a norma L1 como métrica para seleção dos neurônios a remover - Método Proposto.

| Tempo (s) | | | | Parâmetros | | | Traços | | |
|---|---|---|---|---|---|---|---|---|---|
| **Proc.** | **Orig.** | **Nossa** | **Redu. (%)** | **Orig.** | **Nossa** | **Redu. (%)** | **Orig.** | **Nossa** | **Redu. (%)** |
| 185.97 | 309.15 | 251.01 | 18.80 | 352,456 | 226,326 | 35.79 | 1999 | 669 | 66.53 |
| 227.33 | 383.44 | 278.86 | 27.28 | 352,456 | 235,265 | 32.25 | 1994 | 454 | 77.23 |
| 175.61 | 287.74 | 246.48 | 14.34 | 352,456 | 270,733 | 23.19 | 1984 | 941 | 52.57 |
| 199.0 | 333.41 | 210.15 | 36.97 | 352,456 | 223,852 | 36.49 | 1419 | 538 | 62.09 |
| 156.82 | 252.64 | 202.79 | 19.73 | 352,456 | 232,139 | 34.14 | 1992 | 838 | 57.93 |
| 157.76 | 258.53 | 226.61 | 12.35 | 352,456 | 280,159 | 20.51 | 1980 | 870 | 56.06 |
| 153.17 | 247.7 | 228.79 | 7.63 | 352,456 | 282,311 | 19.9 | 1999 | 628 | 68.58 |
| 168.91 | 278.72 | 220.01 | 21.06 | 352,456 | 229,860 | 34.78 | 1975 | 918 | 53.52 |
| 211.39 | 349.38 | 282.5 | 19.14 | 352,456 | 235,163 | 33.28 | 1982 | 1187 | 40.11 |
| 172.67 | 281.59 | 209.81 | 25.49 | 352,456 | 220,898 | 37.33 | 1895 | 956 | 49.55 |

Vemos na Tabela 12 que dentre os dez experimentos executados, temos reduções de 7.63 a 36.97% quanto ao tempo de treinamento, de 19.9 a 37.33% em termos de quantidade de parâmetros e, com relação à quantidade de traços são apresentadas reduções de 40.11 à 77.23%. Comparando-se os resultados aqui obtidos com os apresentados na Tabela 10, vemos que os resultados de ambos os experimentos (com APoZ e com L1) são bastante próximos, com resultados levemente superiores utilizando-se APoZ como métrica de seleção.

Quanto ao *rank* mínimo da chave criptográfica, os resultados para estes experimentos são mostrados na Tabela 13.

Tabela 13 – Ranks da MLP de Prouff utilizando a norma L1 como métrica de seleção dos neurônios a remover - Método Proposto.

| **Min. Rank Baseline** | **Min. Rank Reduced** |
|---|---|
| 0.535 | 0.0 |
| 0.98 | 0.0 |
| 0.125 | 0.0 |
| 0.0 | 0.0 |
| 0.205 | 0.0 |
| 0.02 | 0.0 |
| 0.25 | 0.0 |
| 0.05 | 0.0 |
| 0.385 | 0.055 |
| 0.015 | 0.0 |

Mais uma vez, vemos que os resultados mostrados na Tabela 13, ou seja, os resultados para norma L1 são muito próximos aos obtidos com a métrica APoZ mostrados na Tabela 11. Podemos ver que a rede original atinge *rank* zero em somente um dos experimentos, enquanto as respectivas redes reduzidas atingem o *rank* zero em nove dos dez experimentos realizados.

As curvas dos *ranks* da chave por números de traços para este experimento são mostrados na Figura 24, onde é possível observar que as curvas das redes reduzidas (em vermelho) possuem *rank* médio inferior ao *rank* da rede original (azul). Isso corrobora os resultados mostrados nas Tabelas 12 e 13.

Como vimos, os resultados utilizando a norma L1 como métrica de seleção e remoção de neurônios são muito próximos aos resultados baseados com a métrica APoZ, havendo uma leve perda de desempenho em relação a APoZ. Embora os resultados alcançados sejam semelhantes, o custo computacional para realização do cálculo da norma L1 para redes maiores torna-se muito significativo. Como o intuito desta Tese baseia-se em propor um método para reduzir redes neurais com menor custo computacional, seguimos realizando os próximos experimentos utilizando somente APoZ como forma de selecionar os neurônios a remover.

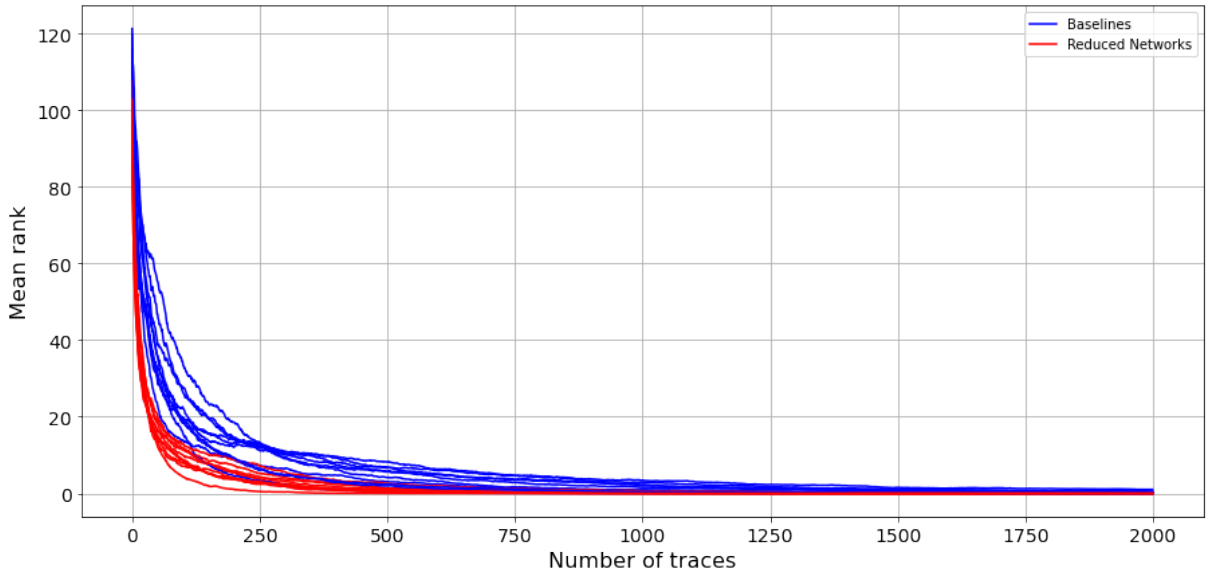


Figura 24 – Ranks MLP de Prouff (L1) - 10 execuções com a rede original (azul) e 10 execuções com as redes reduzidas (vermelho).

Dentro das MLPs aplicadas à SCA, existem redes maiores do que a MLP de Prouff et al. (2018). Como um exemplo, é possível citar a MLP de Perin; Picek (2020). Assim, para os próximos experimentos, buscou-se testar o método proposto nesta MLP, a fim de verificar seu funcionamento para uma rede neural com um número maior de parâmetros.

Assim, para esta rede, foram também executados 10 experimentos com aplicação do método de poda proposto à MLP de Perin; Picek (2020). Num primeiro momento, foram registradas as reduções em termos de tempo de treinamento, quantidade de parâmetros e quantidade de traços necessários para sucesso no ataque das redes reduzidas em relação à MLP original de Perin; Picek (2020). Isto é mostrado na Tabela 14.

Tabela 14 – Reduções na MLP de Perin - Método Proposto.

| Tempo (s) | | | | Parâmetros | | | Traços | | |
|---|---|---|---|---|---|---|---|---|---|
| **Proc.** | **Orig.** | **Nossa** | **Redu. (%)** | **Orig.** | **Nossa** | **Redu. (%)** | **Orig.** | **Nossa** | **Redu. (%)** |
| 920.09 | 869.79 | 382.12 | 56.07 | 9,966,256 | 5,678,616 | 43.02 | 3934 | 2711 | 31.09 |
| 733.99 | 851.94 | 366.59 | 56.97 | 9,966,256 | 5,546,573 | 44.35 | 3985 | 3359 | 15.71 |
| 741.1 | 843.71 | 362.78 | 57.0 | 9,966,256 | 5,460,232 | 45.21 | 3990 | 1218 | 69.47 |
| 650.5 | 840.27 | 359.87 | 57.17 | 9,966,256 | 5,385,677 | 45.96 | 3830 | 1231 | 67.86 |
| 767.81 | 844.09 | 369.75 | 56.2 | 9,966,256 | 5,543,789 | 44.37 | 3962 | 747 | 81.15 |
| 587.39 | 843.99 | 375.12 | 55.55 | 9,966,256 | 5,478,293 | 40.03 | 3985 | 3151 | 20.93 |
| 751.01 | 848.66 | 369.75 | 56.43 | 9,966,256 | 5,477,203 | 45.04 | 3960 | 1458 | 63.18 |
| 557.66 | 851.78 | 360.33 | 57.7 | 9,966,256 | 5,465,070 | 45.25 | 3772 | 2314 | 38.65 |
| 648.11 | 857.31 | 382.04 | 55.43 | 9,966,256 | 5,622,091 | 43.59 | 3948 | 1330 | 66.31 |
| 597.5 | 837.4 | 388.57 | 53.6 | 9,966,256 | 5,594,355 | 43.87 | 3953 | 998 | 74.53 |

A Tabela 14 nos mostra que temos reduções desde 53.26 a 57.7% em termos de tempo de treinamento das MLPs reduzidas em relação à MLP de Perin; Picek (2020). Como mencionado anteriormente, essas reduções em tempos de treinamento, se devem às reduções do tamanho das redes reduzidas em relação à rede original de Perin; Picek (2020). Isso é mostrado na 8ª coluna da Tabela 14. Nesta coluna, podemos ver que reduções na quantidade de parâmetros vão desde 40.03 à 45.96% dentre os 10 experimentos realizados à partir dessa MLP. A última coluna da Tabela 14 mostra que em termos de traços necessários para ter-se sucesso no ataque também obtivemos reduções em todos os experimentos realizados. Sobre esse ponto, temos reduções de 15.71 à 81.15%.

Esses resultados mostram que para uma MLP do tamanho da rede encontrada em Perin; Picek (2020), o método aqui proposto, apresenta reduções significativas. Vemos também através da Tabela 14 que o processo de redução para esta MLP vão de 557.66 à 920.09s, que representam tempos relativamente baixos, devido à utilização de uma abordagem *One-Shot*, como mencionado anteriormente.

Com relação ao *rank* mínimo da chave criptográfica das redes reduzidas, aplicando-se o método aqui proposto à MLP de Perin; Picek (2020) obtivemos melhores resultados em todos os 10 experimentos realizados. Isso pode ser visto na Tabela 15.

Tabela 15 – Ranks da MLP de Perin - Método Proposto.

| Rank Min. Original | Rank Min. Reduzidas |
|---|---|
| 0.035 | 0.0 |
| 0.045 | 0.015 |
| 0.72 | 0.005 |
| 0.08 | 0.0 |
| 0.51 | 0.0 |
| 0.62 | 0.395 |
| 0.255 | 0.0 |
| 0.065 | 0.005 |
| 0.105 | 0.0 |
| 0.21 | 0.0 |

Através da Tabela 15, vemos que, para todos os experimentos realizados com a MLP de Perin; Picek (2020), o *rank* mínimo alcançado com as redes reduzidas é menor do que para a MLP original. As redes reduzidas atingem *rank* zero em seis dos dez experimentos realizados, enquanto a MLP de Perin; Picek (2020) não atinge esse *rank* em nenhum dos experimentos realizados.

A Figura 25, mostra os gráficos de *rank* x número de traços necessários para ataques bem sucedidos. Mais uma vez, vemos que as redes reduzidas (em vermelho)

são mais eficientes para SCAs do que a rede proposta por Perin; Picek (2020).

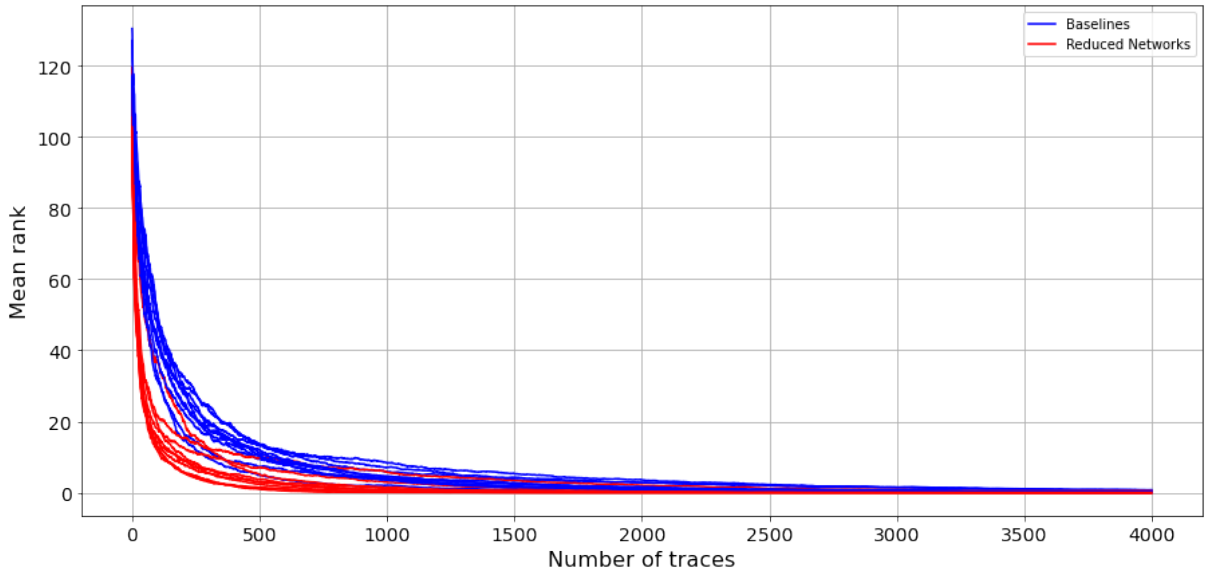


Figura 25 – Ranks MLP de Perin - 10 execuções com a rede original (azul) e 10 execuções com as redes reduzidas (vermelho).

Portanto, como vimos através dos experimentos anteriores, a abordagem aqui proposta atinge resultados promissores quanto à redução de redes MLP aplicadas aos SCAs. Seguimos os testes verificando o funcionamento do método proposto para CNNs que realizam esse tipo de ataques. Com isto, pretende-se atestar o funcionamento satisfatório do método desenvolvido nesta Tese sobre as duas principais arquiteturas de redes neurais aplicadas à SCA até o presente momento na literatura.

### 6.4.3 Análise de CNNs Aplicadas à SCA

Esta Seção apresenta o processo de cirurgia *One-Shot* aplicado à rede CNN de Perin; Picek (2020) executando um ataque ao *dataset* ASCADv1, com contramedida de mascaramento e contramedida temporal. Assim, para esta arquitetura de rede foram obtidos os seguintes resultados de reduções apresentados na Tabela 16.

Tabela 16 – Reduções na CNN de Perin - Método Proposto.

| Tempo (s) | | | | Parâmetros | | | Traços | | |
|---|---|---|---|---|---|---|---|---|---|
| **Proc.** | **Orig.** | **Nossa** | **Redu. (%)** | **Orig.** | **Nossa** | **Redu. (%)** | **Orig.** | **Nossa** | **Redu. (%)** |
| 475.45 | 558.93 | 509.33 | 8.87 | 6,797,246 | 4,120,056 | 39.39 | 2042 | 1902 | 20.82 |
| 499.53 | 553.28 | 487.21 | 11.94 | 6,797,246 | 3,599,353 | 47.05 | 2723 | 1533 | 43.7 |
| 520.48 | 563.59 | 503.56 | 10.65 | 6,797,246 | 4,089,818 | 39.83 | 2384 | 1731 | 27.39 |
| 505.63 | 559.65 | 492.16 | 12.06 | 6,797,246 | 3,495,454 | 39.83 | 1649 | 989 | 40.02 |
| 507.57 | 557.78 | 494.62 | 11.33 | 6,797,246 | 3,682,848 | 45.82 | 1458 | 1325 | 9.12 |
| 514.35 | 564.08 | 494.34 | 12.36 | 6,797,246 | 3,513,517 | 48.31 | 1973 | 1097 | 44.4 |
| 511.88 | 564.6 | 483.88 | 14.3 | 6,797,246 | 2,956,093 | 56.51 | 2319 | 1744 | 24.8 |
| 510.07 | 558.15 | 497.85 | 10.8 | 6,797,246 | 3,960,294 | 41.74 | 2359 | 1737 | 26.37 |
| 514.87 | 563.31 | 491.44 | 12.76 | 6,797,246 | 3,608,795 | 46.91 | 2006 | 1790 | 10.77 |
| 510.63 | 558.71 | 492.78 | 11.8 | 6,797,246 | 3,528,464 | 48.09 | 2704 | 1583 | 41.46 |

A Tabela 16 mostra que para os dez experimentos executados sobre esta CNN, temos reduções no tempo de treinamento desde 8.87 até 12.76%, reduções na quantidade de parâmetros de 39.39 a 56.51% e, reduções na quantidade de traços para o ataque de 9.12 até 44.4%. Mais uma vez, percebemos que o método aqui proposto apresenta ótimos resultados de redução e eficiência das redes neurais aplicadas à SCA. Outra observação que gostaríamos de fazer, diz respeito aos baixos valores de tempo de execução do processo de redução desta CNN. Tempos estes que vão de apenas 475.45 a 520.48s.

Da mesma forma que foi feito para os outros tipos de redes neurais, aqui vamos visualizar os resultados obtidos através do procedimento de poda abordado nesta Tese com relação ao *rank* da chave criptográfica para os dez experimentos executados nesta etapa. Isto pode ser visto, através da Tabela 17.

Tabela 17 – Ranks da CNN de Perin - Método Proposto.

| **Rank Min. Original** | **Rank Min. Reduzidas** |
|---|---|
| 0.0 | 0.0 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |

A partir da Tabela 17 vemos que tanto a CNN de Perin; Picek (2020), quanto as redes reduzidas através do processo aqui proposto alcançam *rank* zero para todos os dez experimentos aqui realizados, ou seja, o processo de redução da CNN em questão não reduz a eficiência desta rede neural.

Apesar dos resultados apresentados no parágrafo anterior nos sugerirem uma mesma eficiência em termos de *rank* da chave para a CNN vista em Perin; Picek (2020) e para as redes neurais reduzidas a partir do processo aqui proposto, é possível verificar através da Figura 26 que as CNNs reduzidas apresentam melhor eficiência do que a rede original encontrada em Perin; Picek (2020).

Aqui na Figura 26, mais uma vez, podemos ver que os gráficos dos *ranks* das chaves criptográficas para a CNN de Perin; Picek (2020) (vistos em azul) mostram uma menor eficiência em relação às redes reduzidas resultantes do processo de poda proposto nesta Tese (curvas na cor vermelha). Através da Figura 26 vemos que as redes reduzidas apresentam uma melhor eficiência em relação a CNN encontrada na

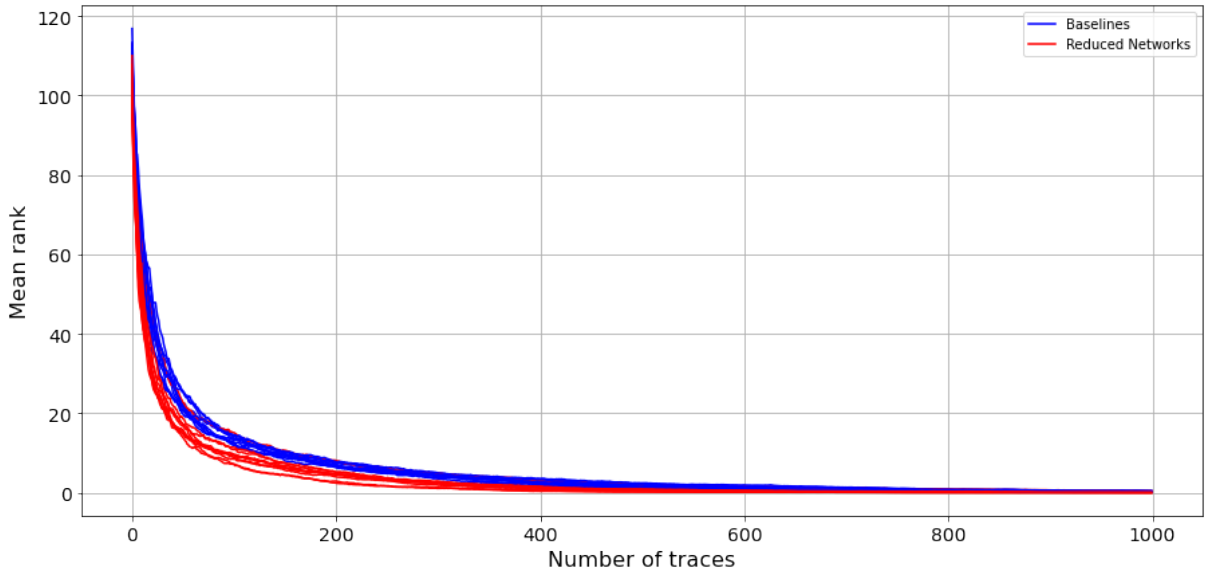


Figura 26 – Ranks CNN de Perin - 10 execuções com a rede original (azul) e 10 execuções com as redes reduzidas (vermelho).

literatura Perin; Picek (2020) quando se trata dos *ranks* da chave secreta.

Em última análise, buscamos testar nosso método aplicando-o a uma CNN aplicada à SCA que apresenta uma arquitetura com um tamanho maior à encontrada em Perin; Picek (2020). Para isto, os mesmos testes empregados até então, foram aplicados à CNN de Prouff et al. (2018).

Desse modo, assim como realizado para as outras redes neurais utilizadas como estudo de caso, o método proposto nesta Tese foi aplicado à CNN de Prouff et al. (2018). Para os experimentos realizados nesta etapa, considera-se a aplicação ao *dataset* ASCADv1 com contramedida de mascaramento e contramedida temporal. A partir disso, a Tabela 18 mostra as reduções obtidas através dos dez experimentos realizados nesta parte desta Tese.

Tabela 18 – Reduções na CNN de Prouff - Método Proposto.

| Tempo (s) | | | | Parâmetros | | | Traços | | |
|---|---|---|---|---|---|---|---|---|---|
| **Proc.** | **Orig.** | **Nossa** | **Redu. (%)** | **Orig.** | **Nossa** | **Redu. (%)** | **Orig.** | **Nossa** | **Redu. (%)** |
| 1341.36 | 2114.05 | 1539.97 | 27.16 | 66,652,544 | 36,042,227 | 45.93 | 14379 | 9344 | 35.02 |
| 1341.13 | 2117.59 | 1372.87 | 35.16 | 66,652,544 | 32,268,597 | 51.59 | 14933 | 9664 | 35.28 |
| 1371.13 | 2114.76 | 1525.53 | 27.86 | 66,652,544 | 33,334,302 | 49.99 | 14999 | 4367 | 70.88 |
| 1423.41 | 2116.8 | 1513.28 | 28.51 | 66,652,544 | 37,131,097 | 44.29 | 14987 | 4334 | 71.08 |
| 1333.58 | 2117.37 | 1569.61 | 25.87 | 66,652,544 | 36,476,651 | 45.27 | 14826 | 11411 | 23.03 |
| 1355.98 | 2119.81 | 1383.10 | 24.75 | 66,652,544 | 32,162,890 | 51.75 | 14927 | 9587 | 35.77 |
| 1380.17 | 2112.15 | 1445.20 | 31.58 | 66,652,544 | 34,568,700 | 48.14 | 14950 | 11302 | 24.4 |
| 1446.48 | 2119.18 | 1574.09 | 25.72 | 66,652,544 | 38,637,303 | 42.03 | 14880 | 6427 | 56.8 |
| 1469.88 | 2118.77 | 1580.38 | 25.41 | 66,652,544 | 40,447,633 | 39.32 | 14996 | 10826 | 27.8 |
| 1402.98 | 2116.93 | 1553.02 | 26.64 | 66,652,544 | 35,847,318 | 46.22 | 14635 | 7803 | 46.68 |

A Tabela 18 indica reduções no tempo de treinamento de 24.75 até 35.16%, em termos de quantidade de parâmetros (tamanho da rede neural) que vão desde 39.32 a 51.75%. Também vemos que houve redução com relação à quantidade de traços necessários para o ataque de 23.03 até 71.08%. Assim, os resultados apresentados na Tabela 18 confirmam que o método de redução das redes neurais proposto nesta Tese apresenta resultados muito bons quando aplicado à CNN de Prouff et al. (2018). Mais uma vez podemos perceber, através da primeira coluna da Tabela 18, que o processo de redução da rede neural consome baixos valores de tempo, indo de 1333.58 à 1446.48s. Conforme mencionamos anteriormente, isto deve-se à abordagem *One-Shot* para a remoção dos neurônios menos ativos de tais redes.

Do ponto de vista de *rank* da chave secreta, vemos através da Tabela 19, que tanto a CNN de Prouff et al. (2018) quanto a CNN reduzida não alcançam *rank* zero em nenhum dos experimentos realizados. Embora, para todos os dez experimentos executados para esta rede, o *rank* mínimo da chave criptográfica é menor para as redes reduzidas do que para a CNN original (Prouff et al. (2018)). Assim sendo, vemos que o menor *rank* mínimo da chave é alcançado pela rede neural reduzida (0.02), conforme destacado na cor amarela na Tabela 18.

Tabela 19 – Ranks da CNN de Prouff - Método Proposto.

| **Rank Min. Original** | **Rank Min. Reduzidas** |
|---|---|
| 0.755 | 0.205 |
| 0.745 | 0.26 |
| 1.61 | 0.11 |
| 0.79 | 0.02 |
| 0.345 | 0.19 |
| 3.27 | 2.135 |
| 0.41 | 0.15 |
| 0.37 | 0.04 |
| 0.945 | 0.525 |
| 0.94 | 0.25 |

Como vimos, os dados da Tabela 19 mostram uma maior eficiência das redes neurais reduzidas em relação à CNN de Prouff et al. (2018). Isto é confirmado através da Figura 27, onde vemos que as curvas dos *ranks* da CNN de Prouff et al. (2018) (em azul) apresentam, em geral, *ranks* menores do que suas respectivas redes neurais reduzidas, para os dez experimentos aqui executados.

Com isto, podemos concluir a partir dos experimentos aqui realizados, que o método proposto nesta Tese para redução de redes neurais alcança ótimos resultados para as redes neurais mais utilizadas no âmbito de *Side Channel Attacks*. Técnicas como a aqui apresentada podem tornar o risco desta classe de ataques muito maior,

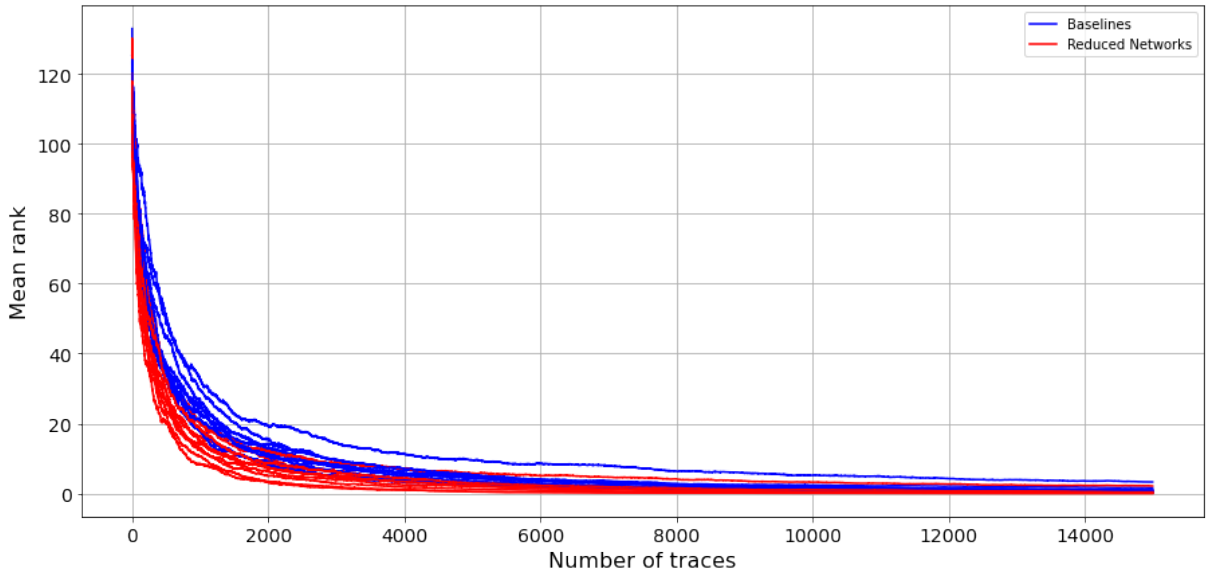


Figura 27 – Ranks CNN de Prouff - 10 execuções com a rede original (azul) e 10 execuções com as redes reduzidas (vermelho).

pois conforme aqui apresentado, esse tipo de ataques pode ser realizado até mesmo em dispositivos dotados de contramedidas através de redes neurais menores do que as principais redes propostas na literatura. Assim, um atacante necessita de muito menos recursos computacionais e tempo de ataque, e isto pode popularizar ainda mais esse tipo de ataques.

## 6.5 Análise da Quantidade de Épocas de Treinamento para SCA

Nesta Seção, mais alguns experimentos foram realizados. Desta vez, testamos as menores redes neurais (em termos de quantidade de parâmetros) alcançadas através do processo de poda aqui proposto, para cada uma das redes consideradas nos experimentos anteriores. Portanto, utilizamos como estudo de caso a menor MLP atingida a partir da MLP de Prouff et al. (2018), assim como a menor MLP alcançada a partir de Perin; Picek (2020). Os testes descritos a seguir, também foram realizados para as menores redes encontradas a partir da CNN de Perin; Picek (2020) e Prouff et al. (2018). Os experimentos, cujos resultados são mostrados a seguir, consistem em verificar quantas épocas de treinamento são necessárias para que seja possível realizar um ataque bem sucedido em uma etapa posterior do fluxo de ataques. Assim, realizamos testes para verificar a quantidade de épocas para realizar os ataques das redes originais em relação à suas respectivas redes reduzidas (menores redes alcançadas). Como veremos a seguir, os resultados mostram que as redes neurais reduzidas necessitam de menos épocas de treinamento do que as redes originais para realizar-se SCAs.

### 6.5.1 Treinamento das MLPs Reduzidas

Os testes com relação à quantidade de épocas foram inicialmente realizadas com a MLP de Prouff et al. (2018). Os experimentos aqui realizados foram balizados pelo *rank* mínimo da chave criptográfica após a execução dos ataques. Para esta rede, iniciamos ataques com 200 épocas (número de épocas recomendadas por Prouff et al. (2018)) e fomos reduzindo esse número até que o *rank* mínimo da chave possuísse um valor igual ou superior a 1. A Tabela 20, mostra os resultados para a MLP de Prouff et al. (2018).

Tabela 20 – MLP de Prouff Original - Épocas para um ataque bem sucedido - Método Proposto.

| Épocas | Traços | Rank |
|---|---|---|
| 200 | 1952 | 0.015 |
| 180 | 1991 | 1.31 |
| 160 | 1998 | 0.915 |
| 150 | 1945 | 0.005 |
| 145 | 1620 | 0.0 |
| 140 | 1991 | 0.265 |
| 135 | 1990 | 2.59 |
| 130 | 1912 | 0.12 |
| 129 | 1997 | 0.77 |
| 128 | 1976 | 6.22 |
| 127 | 1998 | 13.69 |

Na Tabela 20 vemos que para 180, 135, 128 e 127 épocas o *rank* mínimo da chave é maior do que 1. Através dos resultados mostrados na Tabela 20, podemos considerar que precisamos de 129 (destacado em amarelo), das 200 épocas recomendadas, para realizar SCA. Vemos que apenas para 145 épocas o *rank* zero é alcançado para a quantidade de traços considerados, conforme destacado em verde.

Os gráficos correspondentes às curvas dos *ranks* x quantidade de traços par a MLP de Prouff et al. (2018) são mostrados na Figura 28.

A Figura 28 nos mostra que para treinamentos desta MLP com menos de 129 épocas o *rank* da chave secreta não atinge valores considerados satisfatórios para um ataque bem sucedido, dentro do número de traços considerado para este experimento.

Depois desse experimento, a partir do qual extraímos nossa *baseline*, realizamos experimentos relacionados à quantidade de épocas necessárias para realização de SCA para a menor MLP obtida através do processo de cirurgia aqui proposto, aplicado à MLP de Prouff et al. (2018). Os resultados para este experimentos são vistos a partir da Tabela 21.

Como podemos ver através da Tabela 21, necessitamos treinar a MLP em questão por apenas 97 épocas para que o *rank* da chave criptográfica mantenha-se a um valor

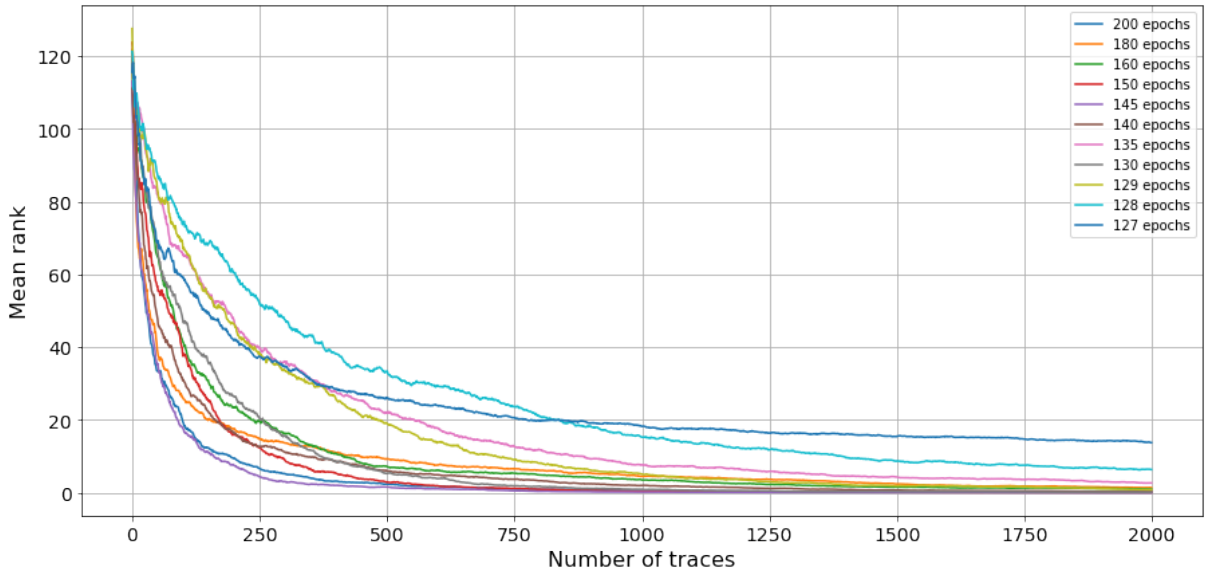


Figura 28 – Ranks por épocas - MLP de Prouff Original.

Tabela 21 – MLP de Prouff Reduzida - Épocas para um ataque bem sucedido - Método Proposto.

| Épocas | Traços | Rank |
|---|---|---|
| 200 | 932 | 0.0 |
| 150 | 2285 | 0.0 |
| 140 | 1662 | 0.0 |
| 130 | 3506 | 0.0 |
| 120 | 1524 | 0.0 |
| 110 | 3894 | 0.005 |
| 105 | 1610 | 0.0 |
| 100 | 2521 | 0.0 |
| 97 | 3882 | 0.095 |
| 96 | 3990 | 3.605 |
| 95 | 3992 | 12.15 |

abaixo de 1 (conforme definimos como um *rank* aceitável).Assim, considerando-se um *rank* menor do que 1, essa rede pode realizar um ataque bem sucedido com menos da metade de épocas sugeridas para a rede original.

A Figura 29, mostra as curvas de *rank* x número de traços para as quantidades de épocas de treinamento consideradas na Tabela 21.

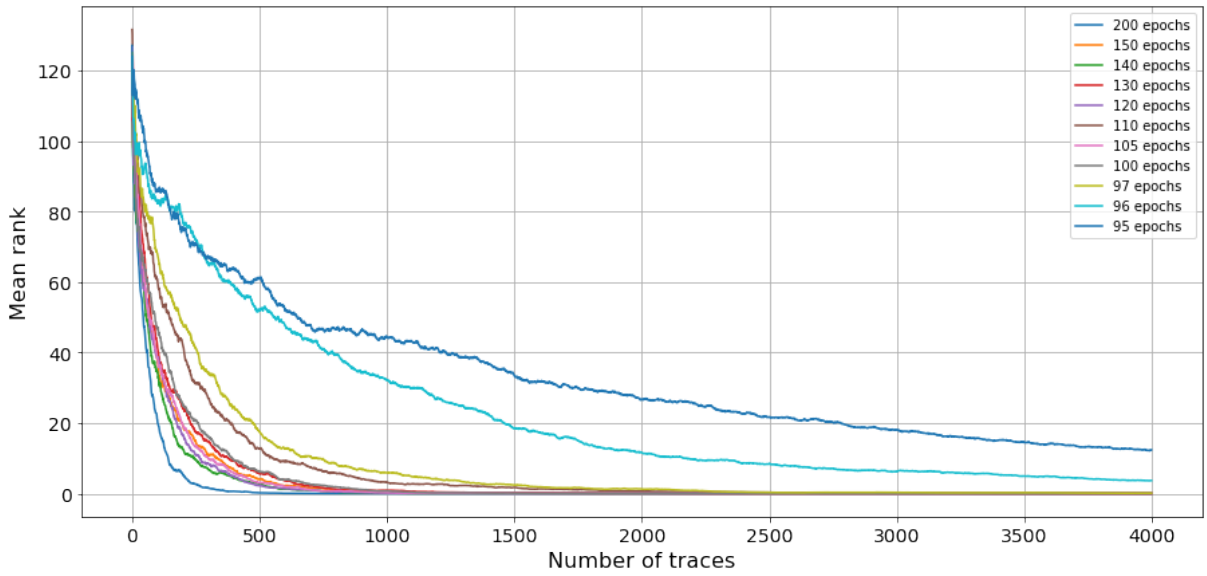


Figura 29 – Ranks por épocas - MLP de Prouff.

Vemos na Figura 23, que para ataques realizados a partir da MLP testada em que a rede foi treinada com menos de 97 épocas, os resultados em termos de *rank* não convergem para valores que caracterizam ataques bem sucedidos para a quantidade de traços considerada.

Estes primeiros experimentos nos mostra que, uma vez que o atacante pondere realizar o treinamento da MLP de Prouff et al. (2018) para realização de SCA, por uma quantidade de épocas menor do que o sugerido pelos autores, com o intuito de reduzir o tempo total dos ataques, a MLP reduzida pode ser treinada por uma quantidade de épocas menor do que a MLP original.

A seguir, os mesmos experimentos foram executados, utilizando-se a MLP de Perin; Picek (2020) como referência. Assim, primeiramente foram realizados testes com relação à quantidade mínima de épocas de treinamento para ataques bem sucedidos, com relação à MLP de Perin; Picek (2020). Esses resultados são apresentados na Tabela 22.

Conforme vemos na Tabela 22, são necessárias aproximadamente 80 épocas (amarelo) para realizarmos um SCA bem sucedido. Aqui, afrouxando-se a métrica anteriormente estabelecida, podemos considerar um ataque aceitável com 70 épocas (verde) de treinamento da MLP. Vemos também na Tabela 22 que em nenhum dos experimentos realizados nesta etapa, o *rank* da chave mínimo alcançou o valor zero. Na melhor das hipóteses, com 175 épocas (destacado em verde) treinando a rede neural

Tabela 22 – MLP de Perin Original - Épocas para um ataque bem sucedido - Método Proposto.

| Épocas | Traços | Rank |
|---|---|---|
| 200 | 3830 | 0.08 |
| 175 | 3962 | 0.02 |
| 150 | 3966 | 0.085 |
| 125 | 3618 | 0.04 |
| 100 | 3988 | 0.66 |
| 90 | 3902 | 0.1 |
| 80 | 3997 | 0.56 |
| 70 | 3993 | 2.27 |
| 69 | 3998 | 5.32 |
| 68 | 3944 | 14.18 |

o *rank* mínimo da chave é de 0.02. Este foi o melhor resultado atingido pela MLP de Perin; Picek (2020).

As curvas mostrando os *ranks* por número de traços são mostradas na Figura 30.

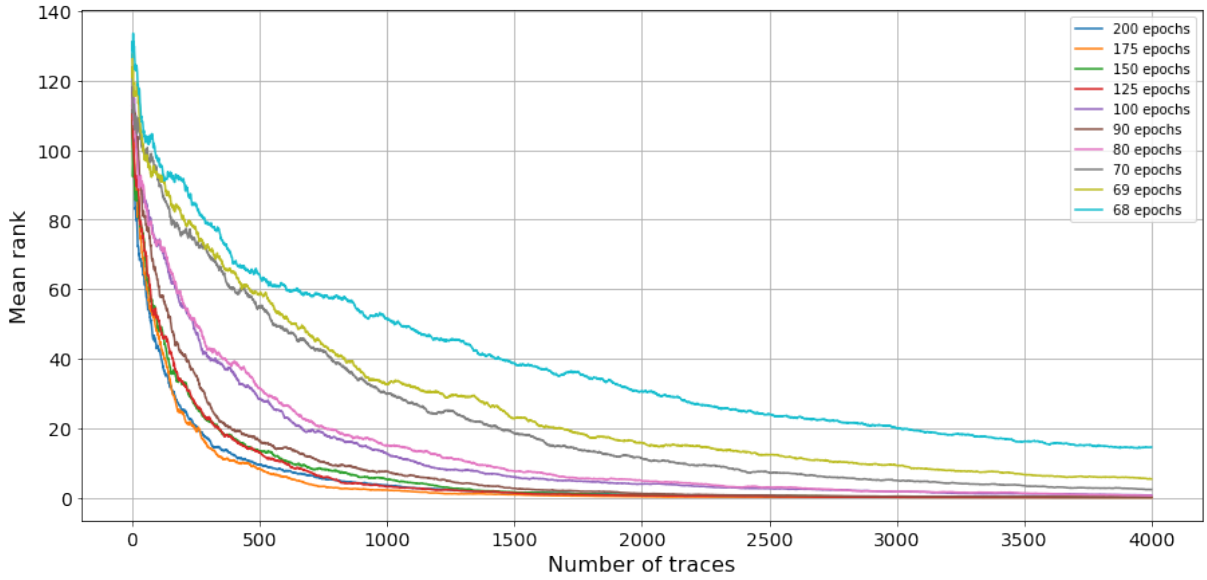


Figura 30 – Ranks por épocas - MLP de Perin Original.

Agora que temos os dados da *baseline* (MLP de Perin; Picek (2020)), partiremos para a descoberta da quantidade de épocas que precisamos treinar a menor MLP obtida através do método proposto sobre a MLP de Perin; Picek (2020), para que esta realize um SCA bem sucedido. Esta rodada de experimentos tem seus resultados mostrados na Tabela 23.

A Tabela 23 nos mostra que a rede neural reduzida, diferentemente da MLP de Perin; Picek (2020), atinge o *rank* zero para a chave criptográfica quando treinamos a MLP reduzida por mais de 30 épocas (em verde). E, considerando *rank* 1 um resultado satisfatório para termos um ataque bem sucedido (conforme estabelecemos anterior-

Tabela 23 – MLP de Perin Reduzida - Épocas para um ataque bem sucedido - Método Proposto.

| Épocas | Traços | Rank |
|---|---|---|
| 200 | 1231 | 0.0 |
| 150 | 1790 | 0.0 |
| 100 | 1553 | 0.0 |
| 70 | 1289 | 0.0 |
| 50 | 2390 | 0.0 |
| 30 | 2270 | 0.0 |
| 20 | 3767 | 0.33 |
| 19 | 3968 | 0.86 |
| 18 | 3953 | 7.045 |
| 17 | 3980 | 16.3 |

mente), percebemos que se treinarmos essa rede reduzida por 19 épocas (destacado em amarelo) ou mais, temos um SCA bem sucedido. Aqui, percebemos que mais uma vez a MLP reduzida pode realizar um ataque bem sucedido, sendo treinada por uma quantidade de épocas inferior à sua correspondente MLP original.

Vemos as curvas dos *rank* da chave para a MLP reduzida na Figura 31.

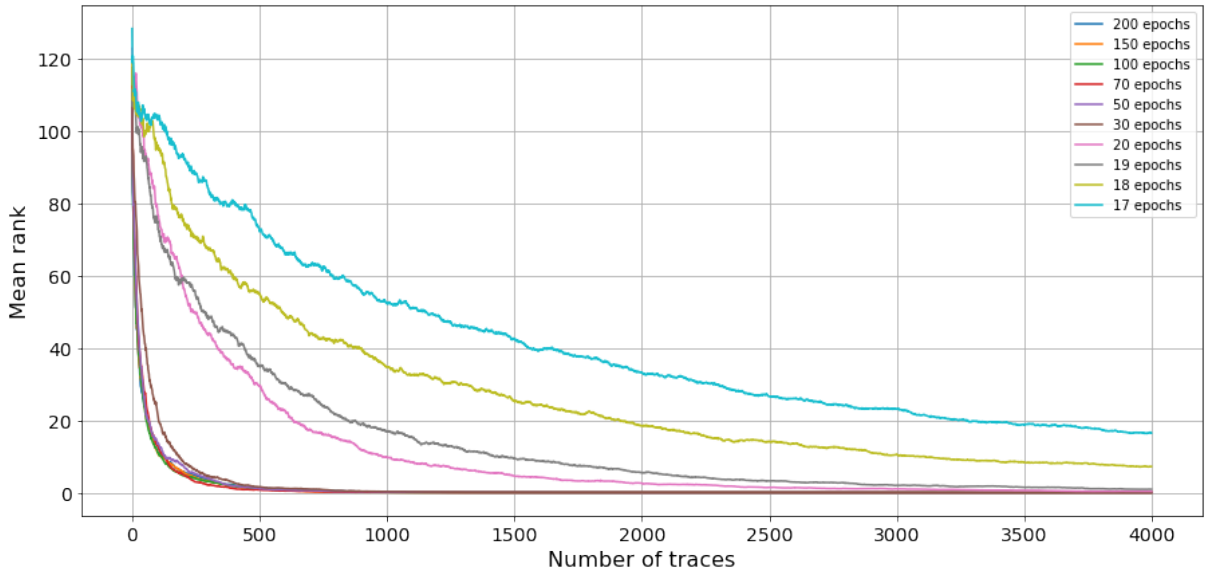


Figura 31 – Ranks por épocas - MLP de Perin.

É possível notar na Figura 31 que o *rank* médio das curvas vão aumentando conforme vamos reduzindo o número de épocas de treinamento, principalmente quando realizamos o treinamento da rede neural com 20 épocas ou menos. Com trinta épocas ou mais de treinamento, vemos que as curvas de *rank* são bastante próximas, e vão se distanciando em função da redução de épocas, conforme esperávamos.

### 6.5.2 Treinamento das CNNs Reduzidas

Como vimos, as MLPs reduzidas através do método aqui proposto podem ser treinadas por uma quantidade de épocas inferior às suas respectivas MLPs originais. Nos próximos passos, testamos as CNNs utilizadas como estudo de caso nos exemplos anteriores. Primeiramente, foram realizados experimentos a partir da CNN encontrada em Perin; Picek (2020), descrita em parágrafos anteriores. Os resultados para esta rede são mostrados na Tabela 24.

Tabela 24 – CNN de Perin Original- Épocas para um ataque bem sucedido - Método Proposto.

| Épocas | Traços | Rank |
|---|---|---|
| 400 | 2319 | 0.0 |
| 200 | 1752 | 0.0 |
| 160 | 2664 | 0.0 |
| 155 | 3105 | 0.0 |
| 154 | 3727 | 0.0 |
| 153 | 1379 | 0.0 |
| 152 | 3987 | 0.07 |
| 151 | 3997 | 0.27 |
| 150 | 3914 | 8.41 |
| 149 | 3993 | 23.03 |

Com base nos dados da Tabela 24 vemos que para a CNN de Perin; Picek (2020) apresenta *rank* mínimo da chave criptográfica abaixo de 1 quando treinamos essa rede neural com 151 épocas ou mais. Podemos ver também que o *rank* zero é atingido quando a rede é treinada por pelo menos 153 épocas (destacado em verde). Assim, notamos que com menos de 153 épocas de treinamento o *rank* da chave criptográfica é maior do que zero, crescendo rapidamente quando reduzimos o número de épocas abaixo de 151 épocas. Confirmando com isso, a necessidade de treinar-se a rede com pelo menos 151 épocas para obtermos SCAs bem sucedidos.

As curvas do *rank* da chave para estes experimentos são mostradas na Figura 32.

Como mencionamos anteriormente, vemos através da Figura 32 que quando reduzimos o número de épocas de treinamento abaixo de 151 épocas, as curvas dos *ranks* da chave secreta tem seus valores médios aumentados consideravelmente. Isto corrobora a necessidade de treinamento de pelo menos 151 épocas para a realização de ataques bem sucedidos.

Partindo dessa *baseline*, os mesmos experimentos em termos de número de épocas foram realizados considerando a menor CNN obtida aplicando-se o método aqui proposto à CNN de Perin; Picek (2020). Os resultados estão mostrados na Tabela 25.

Através da Tabela 25 vemos que apesar de o *rank* da chave criptográfica alcançar valor igual a zero somente para o treinamento da rede reduzida por 400 épocas, o *rank*

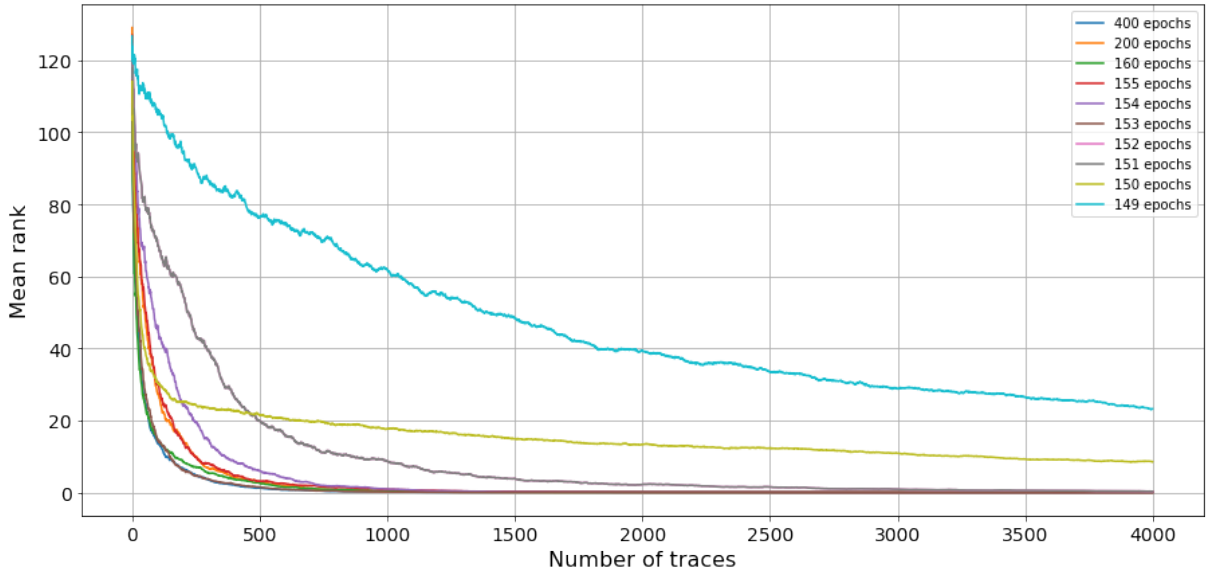


Figura 32 – Rank por épocas - CNN de Perin Orig.

Tabela 25 – CNN de Perin Reduzida - Épocas para um ataque bem sucedido - Método Proposto.

| Épocas | Traços | Rank |
|---|---|---|
| 400 | 1744 | 0.0 |
| 300 | 3668 | 0.015 |
| 200 | 3983 | 0.05 |
| 150 | 3792 | 0.005 |
| 149 | 3900 | 0.12 |
| 148 | 3992 | 0.94 |
| 147 | 3993 | 2.23 |
| 146 | 3991 | 2.85 |
| 145 | 3977 | 9.67 |
| 144 | 3999 | 14.36 |

abaixo de 1 é atingido quando treinamos essa rede por 148 épocas ou mais. Com um número menor de épocas, verificamos que os valores de *rank* da chave criptográfica aumentam substancialmente, colocando esse número de épocas como sendo o valor mínimo de épocas de treinamento para realizar SCA. Percebemos, portanto, que apesar de haver uma pequena diferença neste caso, a CNN reduzida necessita ser treinada por uma quantidade de épocas menor do que a CNN original de Perin; Picek (2020).

As curvas dos *ranks* da chave criptográfica para a CNN reduzida através do método aqui proposto são mostradas na Figura 33.

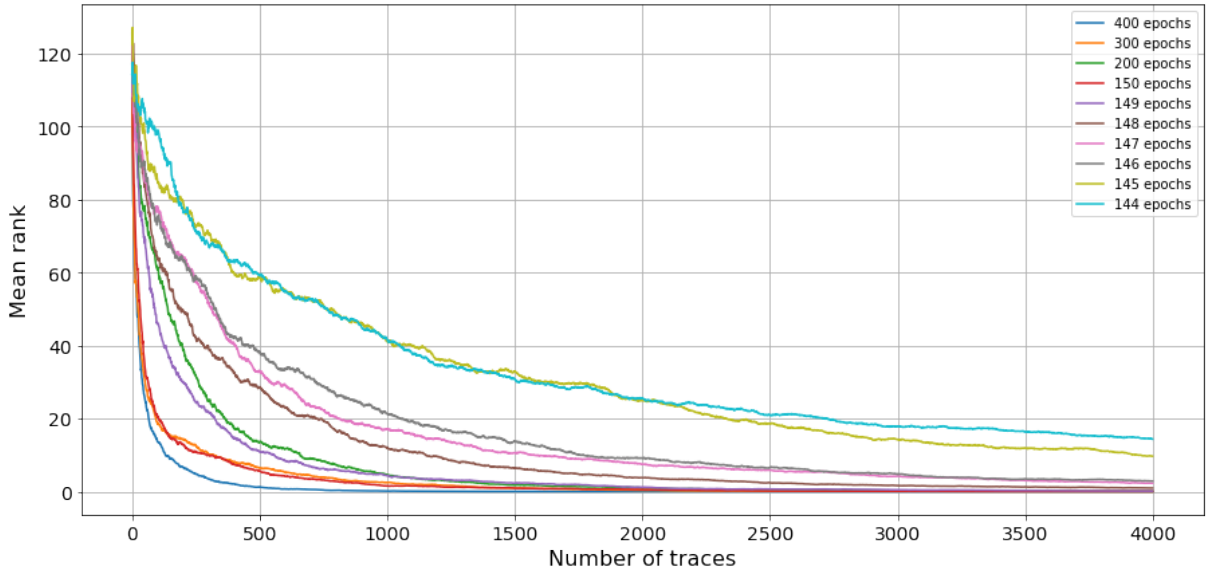


Figura 33 – Rank por épocas - CNN de Perin.

Observando as curvas dos *ranks* apresentadas na Figura 33, podemos notar que embora as curvas mostradas na Figura sejam bastante próximas neste experimento, é possível perceber que para treinamentos com 148 épocas ou mais, o *rank* da chave atinge valores muito próximos a zero (menores do que 1, conforme definimos).

Por último foram realizados testes com relação à quantidade de épocas necessárias para um ataque bem sucedido a partir da CNN de Prouff et al. (2018). Seguindo-se os mesmos procedimentos para os experimentos anteriores, começaremos mostrando os resultados obtidos para esta rede neural, na Tabela 26.

Vemos através da Tabela 26, que para obtermos o *rank* mínimo da chave menor do que 1, conforme definido para ter-se sucesso no ataque anteriormente, precisamos treinar a rede neural por 55 épocas (em amarelo). Inclusive, para esse número de épocas de treinamento o resultado do ataque (*rank*) é melhor do que para as 130 épocas recomendada pelos autores. Se formos considerar o *rank* (3.27) obtido com 130 épocas de treinamento como referência, poderíamos considerar um ataque bem sucedido treinando a rede com 52 épocas de treinamento, ou mais (destacado em verde).

Tabela 26 – CNN de Prouff - Épocas para um ataque bem sucedido - Método Proposto.

| Épocas | Traços | Rank Min. |
|---|---|---|
| 130 | 14927 | 3.27 |
| 100 | 14981 | 1.17 |
| 60 | 14983 | 2.295 |
| 55 | 14973 | 0.5 |
| 54 | 14594 | 1.88 |
| 53 | 14802 | 1.99 |
| 52 | 14997 | 1.355 |
| 51 | 13549 | 5.83 |
| 50 | 14991 | 6.61 |
| 49 | 14776 | 13.53 |

Como fizemos para as outras redes, aqui mostramos na Figura 34 as curvas dos *ranks* para os experimentos realizados.

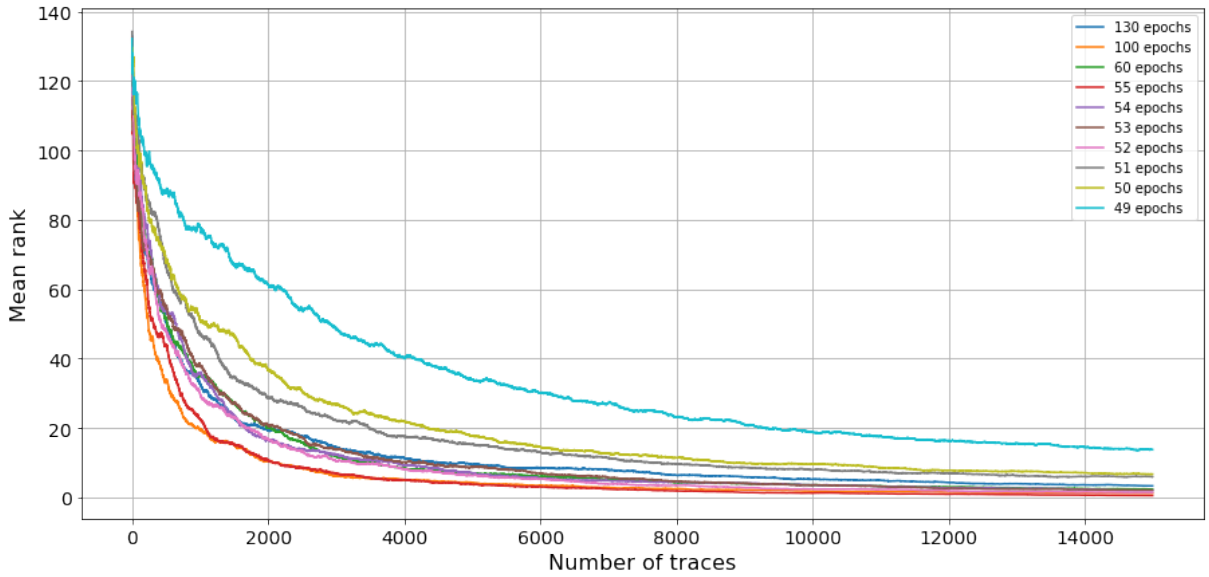


Figura 34 – Rank por épocas - CNN de Prouff Orig.

Através da Figura 34 percebemos visualmente através das curvas, que para casos em que a rede foi treinada por menos de 52 épocas o *rank* mínimo da chave não atinge os menores valores, confirmando os resultados da Tabela 26.

Agora, estes experimentos são realizados com a menor rede obtida a partir da CNN de Prouff et al. (2018). A Tabela 27 mostra os resultados obtidos em termos de *rank* mínimo da chave.

A Tabela 27 mostra que para obtermos *rank* menor do que 1, precisamos treinar a rede neural por 40 épocas (em amarelo). No entanto, se considerarmos o *rank* obtido com 130 épocas de treinamento (2.14) como referência, vemos que se treinarmos a rede por 29 épocas (verde) ou mais pode-se considerar um ataque bem sucedido.

Tabela 27 – CNN de Prouff Reduzida - Épocas para um ataque bem sucedido - Método Proposto.

| Épocas | Traços | Rank Min. |
|---|---|---|
| 130 | 9587 | 2.14 |
| 100 | 14927 | 0.72 |
| 70 | 14995 | 1.51 |
| 50 | 13939 | 0.24 |
| 40 | 14974 | 0.32 |
| 30 | 14932 | 1.57 |
| 29 | 14799 | 1.29 |
| 28 | 14963 | 5.06 |
| 27 | 14936 | 7.35 |
| 26 | 14506 | 22.15 |

As curvas dos *ranks* para este experimento, podem ser vistas na Figura 35.

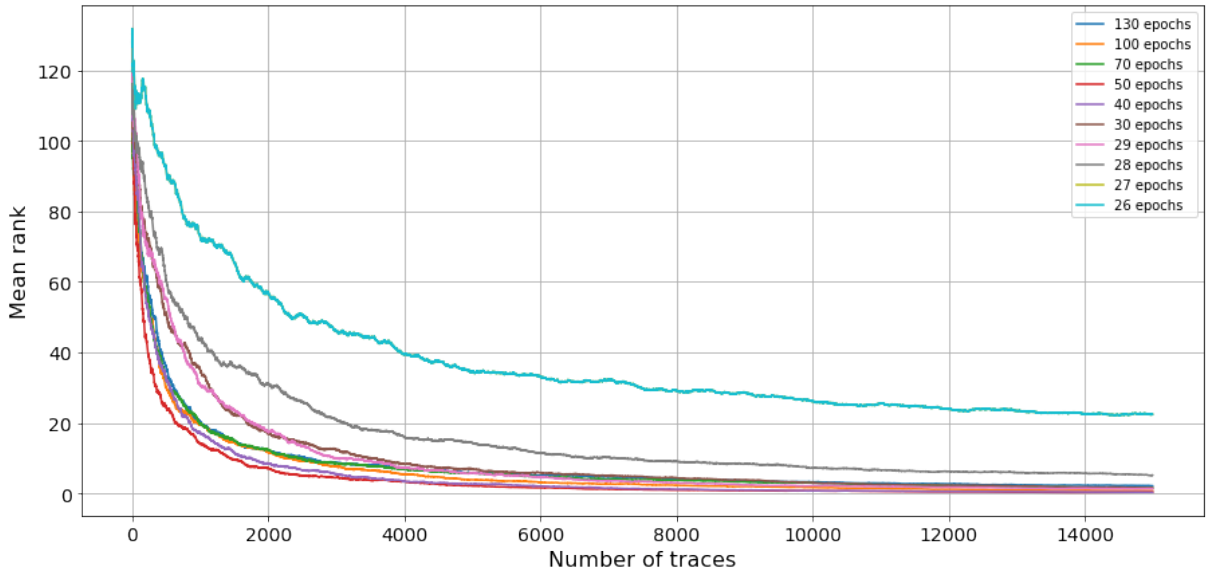


Figura 35 – Rank por épocas - CNN Prouff.

Como podemos ver através da Figura 35, as curvas são bastante próximas, atingindo maiores valores de *rank* médio quando treinamos a rede neural por menos de 29 épocas.

Com base nos experimentos realizados, pudemos ver que para todas as redes neurais aqui testadas, suas versões reduzidas, através do método aqui proposto, necessitam ser treinadas por uma quantidade menor de épocas do que suas respectivas redes originais, para obtermos sucesso nos SCAs. A Tabela 28, traz um resumo dos resultados obtidos de modo que possamos fazer uma comparação entre as redes.

Com base nestes experimentos, concluímos que é possível um atacante adotar a estratégia de diminuir a quantidade de épocas de treinamento da rede para reduzir

Tabela 28 – Comparação entre resultados de números de épocas.

| Rede | Épocas Orig. | Épocas Red. | Redução% |
|---|---|---|---|
| **MLP Prouff** | 129 | 97 | 24.81 |
| **MLP Perin** | 80 | 19 | 76.25 |
| **CNN Perin** | 151 | 148 | 1.99 |
| **CNN Prouff** | 55 | 40 | 27.27 |

o tempo de execução do SCA, visto que as redes neurais reduzidas pelo método proposto apresentam melhores desempenho.

## 6.6 Considerações sobre o Capítulo

Este Capítulo apresentou o método de poda proposto nesta Tese, o qual consiste no desenvolvimento de um fluxo para redução de redes neurais capazes de realizar ataques a canais laterais. Com isto, mostramos que os SCAs podem ser realizados através de redes neurais com menor esforço computacional, e com menores tempos de treinamento do que com as redes apresentadas na literatura, apontando um potencial aumento no nível de ameaça dessa classe de ataques.

Como vimos, a abordagem aqui adotada baseou-se na técnica de cirurgia apresentada por Hu et al. (2016). Assim, os primeiros experimentos consistiram em aplicar o método de Hu et al. (2016) no contexto de SCA. A partir desses testes iniciais, foi possível constatar que tal método pode reduzir as redes disponíveis na literatura usadas para esta finalidade.

Em um segundo momento, foram realizados um conjunto de experimentos aplicando-se o método proposto, que teve como objetivo reduzir o esforço computacional e consequentemente o tempo despendido no processo de redução da rede neural em si. Com base nesses experimentos, finalmente pudemos comprovar que é possível obter-se redes neurais reduzidas para realização de SCA com um custo computacional e temporal inferiores aos indicados na literatura.

Por fim, mostramos que as redes neurais reduzidas através do nosso método podem ser treinadas por menos épocas do que as redes originais para realização dos ataques. Assim, indicamos que caso o atacante utilize a redução do número de épocas de treinamento para realizar ataques mais rápidos, as redes neurais reduzidas apresentam melhor desempenho que as redes originais.

# 7 CONSIDERAÇÕES FINAIS

Este Capítulo visa destacar as principais conclusões decorrentes dos experimentos realizados a partir do método de redução de redes neurais aplicadas à SCAs proposto nesta Tese. Além disso, são caracterizadas alternativas para continuidade dos trabalhos, tendo por base os estudos e pesquisa realizados.

## 7.1 Principais Conclusões

O expressivo aumento na quantidade de dispositivos que usam criptografia embarcada, trouxe consigo uma nova classe de ameaças que são os ataques a canais laterais (KOCHER, 1996) e (KOCHER; JAFFE; JUN, 1999). Embora existam proteções contra esse tipo de ataques, chamadas contramedidas (BOEY et al., 2010), (CHOU; LU, 2019), (DAS et al., 2020), (LIU; CHANG; LEE, 2010) e (LIU; CHANG; LEE, 2012), estudos mostraram que etapas de pré-processamento podem neutralizar tais contramedidas (LELLIS; SOARES, 2017), (LODER, 2014), (NAGASHIMA et al., 2007) e (LE et al., 2007).

Além disso, devido à crescente evolução da área de inteligência artificial, algoritmos de aprendizado de máquina e aprendizado profundo foram também empregados no contexto dos SCAs (HETTWER; GEHRER; GÜNEYSU, 2020), (YANG et al., 2012) e (LERMAN et al., 2013). Inicialmente, os trabalhos encontrados na literatura realizavam ataques baseados em inteligência artificial sobre dispositivos desprotegidos. Contudo, também são encontrados trabalhos que realizam ataques em dispositivos dotados de contramedidas (LERMAN; MARTINASEK; MARKOWITCH, 2017), (**?**), (PROUFF et al., 2018), (TIMON, 2018) e (TIMON, 2019).

Embora comprovado através dos estudos apresentados na literatura, que algoritmos de aprendizado profundo possam recuperar a chave criptográfica através de um ataque a canal lateral, o custo computacional envolvido é bastante excessivo, pois a rede neural utilizada para essa tarefa pode conter uma quantidade de parâmetros treináveis da ordem de milhões. Alguns experimentos mencionam semanas de tempo de execução dos seus algoritmos. Isto pode tornar o ataque inviável em muitos casos.

Por outro lado, técnicas de redução de redes neurais são encontradas na literatura, como por exemplo a poda, que busca eliminar elementos da rede (pesos, neurônios ou camadas) baseando-se em algum critério de utilidade de tais elementos (GUO; YAO; CHEN, 2016), (HE; ZHANG; SUN, 2017), (KIM; KWOK, 2019), (CHEN et al., 2021), (FAN; TANG; MA, 2022) e (HU et al., 2016). Através dos experimentos aqui relatados, pudemos comprovar que técnicas, como a apresentada por Hu et al. (2016) podem ser empregadas no contexto de SCA com sucesso, encontrando-se redes muito menores, e por tanto mais eficientes, capazes de realizar ataques bem sucedidos.

Contudo, muitos dos métodos encontrados da literatura, como por exemplo (HU et al., 2016), são bastante custosos por tratarem-se de métodos iterativos contendo repetidos ciclos de poda-treinamento. Assim, aqui propomos um método *One-Shot* eliminando os diversos retreinamentos capaz de obter-se redes menores, além de mais eficientes, do que as encontradas na literatura (LELLIS; SOARES; PERIN, 2022).

Experimentos foram realizados com diferentes configurações de redes neurais mais recorrentes no campo de SCA (MLPs e CNNs), confirmando a eficiência do método aqui proposto para reduzir redes neurais capazes de realizar ataques a canais laterais. Como vimos, as redes reduzidas através da abordagem desenvolvida nesta Tese possuem uma quantidade menor de parâmetros treináveis e, consequentemente menores tempos de treinamento. Além disso, como pudemos ver através dos resultados apresentados no Capítulo 6.3, as redes neurais reduzidas são capazes de realizar SCAs necessitando para isso uma quantidade menor de traços do consumo, métrica bastante coerente na área de *Side Channel Attacks*. Ainda, experimentos realizados com relação à quantidade de épocas de treinamento das redes necessárias para realizar-se ataques bem sucedidos. Nesses casos, as redes reduzidas também se mostraram mais eficientes do que suas respectivas redes neurais.

Com base nos resultados alcançados no desenvolvimento desta Tese, podemos apontar o crescente potencial desse tipo de ataques, uma vez que através de redes neurais de tamanho reduzido um atacante é capaz de realizar ataques com menos recursos computacionais e de tempo consumido, até mesmo em dispositivos dotados de contramedidas. Assim, SCAs podem tornar-se mais ameaçadores, pois estes podem tornar-se acessíveis a uma quantidade cada vez maior de usuários mal intencionados.

## 7.2 Trabalhos Futuros

Como concluímos, os ataques por canais laterais podem tornar-se mais ameaçadores se conseguirmos realizar tais ataques ao custo de cada vez menos recursos computacionais. Aqui nesta Tese exploramos a redução de tais redes através de um método de poda (*pruning*). Como vimos, os métodos de poda apresentam algumas características que apresentam alguns aspectos que podem ser explorados, como fi-

zemos por exemplo, variando-se a forma de selação dos neurônios (APoZ, L1, etc.). A granularidade consiste em outra característica da poda que apresenta variações. Aqui, entendemos que para as redes testadas a melhor granularidade consiste na poda de neurônios. Porém, técnicas que mesclam podas em diferentes granularidades podem ser testadas futuramente. Além disso, outras formas de redução de redes neurais como a quantização também são possíveis. Assim, para passos futuros tais técnicas podem ser exploradas e até mesmo integradas a fluxos juntamente com técnicas de poda.

Ainda, as técnicas de redução encontradas na literatura têm a função de diminuir o tamanho de redes neurais pré-existentes. Assim, é necessário que se tenha uma rede prévia para realizar a sua redução. Sabe-se que o projeto de redes neurais é uma tarefa extremamente complexa, e que exige profundo conhecimento de seus projetistas, além de horas de treinamento para chegar-se ao resultado desejado. Para mitigar esses problemas, técnicas automáticas de projeto de redes neurais, chamadas de *Neural Architecture Search* – NAS são encontradas na literatura. Através de NAS redes neurais são projetadas sem a necessidade de intervenção humana, chegando-se a resultados superiores ao estado da arte até o presente momento. Tais técnicas podem também ser aplicadas no contexto de SCA em momentos futuros.

Tais frentes de pesquisa elencadas configuram o panorama para o aprimoramento e expansão do trabalho aqui realizado, ou seja, a busca por técnicas de realização de ataques por canais laterais baseados em redes neurais com menor necessidade de recursos computacionais e de tempo.

# REFERÊNCIAS


ABADI, M. et al. **TensorFlow**: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>.

AGRAWAL, D. et al. The EM Side–Channel(s):Attacks and Assessment Methodologies. **Cryptographic Hardware and Embedded Systems - CHES 2002**, Redwood Shores, CA, USA, v.6917, p.29–45, 2002.

ALIPOUR, A. et al. On the Performance of Non-Profiled Differential Deep Learning Attacks against an AES Encryption Algorithm Protected using a Correlated Noise Generation based Hiding Countermeasure. **Proceedings of the 2020 Design, Automation and Test in Europe Conference and Exhibition, DATE 2020**, Alpexpo Congress Centre in Grenoble, p.614–617, 2020.

BABAEIZADEH, M.; SMARAGDIS, P.; CAMPBELL, R. H. NoiseOut: A Simple Way to Prune Neural Networks. **arXiv:1611.06211**, arXiv preprint, 2016.

BOEY, K. H.; LU, Y.; O'NEILL, M.; WOODS, R. Random clock against differential power analysis. **2010 IEEE Asia Pacific Conference on Circuits and Systems**, Kuala Lumpur, Malaysia, p.756–759, 2010.

BRIER, E.; CLAVIER, C.; OLIVIER, F. Correlation Power Analysis with a Leakage Model. In: CRYPTOGRAPHIC HARDWARE AND EMBEDDED SYSTEMS - CHES 2004: 6TH INTERNATIONAL WORKSHOP CAMBRIDGE, MA, USA, AUGUST 11-13, 2004. PROCEEDINGS, 2004. **Anais...** Springer, 2004. v.3156, p.16–29.

BRISFORS, M.; FORSMARK, S.; DUBROVA, E. How Deep Learning Helps Compromising USIM. In: SPRINGER, 2021, Virtual. **Anais...** Springer, 2021. v.12609 LNCS, p.135–150.

CAGLI, E.; DUMAS, C.; PROUFF, E. Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Preprocessing. In: CRYPTOGRAPHIC HARDWARE AND EMBEDDED SYSTEMS -

CHES 2017 - 19TH INTERNATIONAL CONFERENCE, TAIPEI, TAIWAN, SEPTEMBER 25-28, 2017, PROCEEDINGS, 2017. **Anais...** Springer, 2017. p.45–68. (Lecture Notes in Computer Science, v.10529).

CARLET, C. et al. Breaking Cryptographic Implementations Using Deep Learning Techniques. In: SPRINGER, 2016, Hyderabad, India. **Anais...** Springer, 2016. v.10076 LNCS, n.December, p.3–26.

CHARI, S.; RAO, J. R.; ROHATGI, P. Template Attacks. **4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'02)**, Redwood Shores, CA, USA, p.13–28, 2002.

CHAWLA, N.; SINGH, A.; KAR, M.; MUKHOPADHYAY, S. Application Inference using Machine Learning based Side Channel Analysis. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), 2019., 2019, Budapest, Hungary. **Anais...** IEEE, 2019. p.1–8.

CHEN, Z. et al. Dynamical Channel Pruning by Conditional Accuracy Change for Deep Neural Networks. **IEEE Transactions on Neural Networks and Learning Systems**, IEEEexplore, v.32, n.2, p.799–813, 2021.

CHOI, J.-B.; KIM, D.-S.; CHOE, J.-Y.; SHIN, K.-W. Hardware Implementation of ECIES Protocol on Security SoC. In: INTERNATIONAL CONFERENCE ON ELECTRONICS, INFORMATION, AND COMMUNICATION (ICEIC), 2020., 2020, Barcelona, Spain. **Anais...** IEEE, 2020. p.1–4.

CHOLLET, F. **Keras**. Repository: GitHub, 2015. `https://github.com/fchollet/keras`.

CHONG, K.-S. et al. Dual-Hiding Side-Channel-Attack Resistant FPGA-Based Asynchronous-Logic AES: Design, Countermeasures and Evaluation. **IEEE Journal on Emerging and Selected Topics in Circuits and Systems**, IEEEexplore, v.11, n.2, p.343–356, 2021.

CHOU, Y. H.; LU, S. L. L. A high performance, low energy, compact masked 128-Bit AES in 22nm CMOS technology. **2019 International Symposium on VLSI Design, Automation and Test, VLSI-DAT 2019**, Hsinchu, Taiwan, p.1–4, 2019.

CORON, J.-S.; GOUBIN, L. On Boolean and Arithmetic Masking against Differential Power Analysis. **Cryptographic Hardware and Embedded Systems - CHES 2000**, Worcester, MA, USA, p.231–237, 2000.

CRISTIANI, V.; LECOMTE, M.; MAURINE, P. Leakage assessment through neural estimation of the mutual information. In: IEEE, 2020, Rome, Italy. **Anais...** Spring, 2020. v.12418 LNCS, p.144–162.

DAEMEN, J.; RIJMEN, V. AES submission document on Rijndael, Version 2. , USA, 1999.

DAS, D. et al. X-DeepSCA. In: ANNUAL DESIGN AUTOMATION CONFERENCE 2019, 56., 2019, New York, NY, USA. **Proceedings...** ACM, 2019. n.DI, p.1–6.

DAS, D. et al. Deep Learning Side-Channel Attack Resilient AES-256 using Current Domain Signature Attenuation in 65nm CMOS. **Proceedings of the Custom Integrated Circuits Conference**, Boston, MA, USA, v.2020-March, p.2–5, 2020.

DEOTTE, C. **How to choose cnn architecture mnist @ONLINE**. Disponível em: <https://www.kaggle.com/code/cdeotte/how-to-choose-cnn-architecture-mnist/notebook>.

DES. Data Encryption Standard. In: IN FIPS PUB 46, FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, 1977, USA. **Anais...** NIST (National Institute of Standards and Technology), 1977. p.46–2.

EVANS, N. Pruning the fully connected layers of a convolutional neural network by distinctiveness. **ABCs 2018 - 1st ANU Bio-inspired Computing conference**, Canberra, Australia, v.2018, p.13, 2018.

FAN, Y.; TANG, X.; MA, Z. A Weight-Based Channel Pruning Algorithm for Depth-Wise Separable Convolution Unit. In: INTERNATIONAL CONFERENCE ON ALGORITHMS, COMPUTING AND ARTIFICIAL INTELLIGENCE, 2021., 2022, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2022. (ACAI '21).

FRANKLE, J.; CARBIN, M. The Lottery Ticket Hypothesis: Training Pruned Neural Networks. **CoRR**, arvix, v.abs/1803.03635, 2018.

FRIEDEL, J. E.; HOLZER, T. H.; SARKANI, S. Development, Optimization, and Validation of Unintended Radiated Emissions Processing System for Threat Identification. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, IEEExplore, v.50, n.6, p.2208–2219, 2020.

FUHR, T.; JAULMES, E.; LOMNE, V.; THILLARD, A. Fault attacks on AES with faulty ciphertexts only. **Proceedings - 10th Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2013**, Los Alamitos, CA, USA, p.108–118, 2013.

GAO, S. et al. My traces learn what you did in the dark: Recovering secret signals without key guesses. **CT-RSA 2017: Topics in Cryptology – CT-RSA 2017**, an Francisco, CA, USA, v.10159, p.363–378, 2017.

GENKIN, D.; SHAMIR, A.; TROMER, E. Acoustic Cryptanalysis. **Journal of Cryptology**, Berlin, Heidelberg, v.30, n.2, p.392–443, 2017.

GHOSH, S. et al. Deep Network Pruning for Object Detection. **International Conference on Image Processing - ICIP**, Taipei, Taiwan, p.3915–3919, 2019.

GKALELIS, N.; MEZARIS, V. Structured Pruning of LSTMs via Eigenanalysis and Geometric Median for Mobile Multimedia and Deep Learning Applications. **Proceedings - IEEE International Symposium on Multimedia - ISM**, Naples, Italy, p.122–126, 2020.

GOLIC, J. D. Techniques for random masking in hardware. **IEEE Transactions on Circuits and Systems I: Regular Papers**, IEEExplore, v.54, n.2, p.291–300, 2007.

GOODFELLOW, I. J.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA, USA: MIT Press, 2016. `http://www.deeplearningbook.org`.

GUAN, T.; ZHANG, C. Mixed Pruning Method for Vehicle Detection. **Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS**, Beijing, China, v.2020-October, p.232–235, 2020.

GUO, Y.; YAO, A.; CHEN, Y. Dynamic Network Surgery for Efficient DNNs. **CoRR**, arxiv, v.abs/1608.04493, 2016.

HAN, S.; POOL, J.; TRAN, J.; DALLY, W. J. Learning both Weights and Connections for Efficient Neural Networks. **CoRR**, arxiv, v.abs/1506.02626, 2015.

HAYKIN, S. **Redes Neurais**: princípios e prática - 2.ed. Porto Alegre: Bookman, 2001.

HE, Y.; ZHANG, X.; SUN, J. Channel Pruning for Accelerating Very Deep Neural Networks. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 2017., 2017, Venice, Italy. **Anais...** IEEE, 2017. p.1398–1406.

HERON, S. Advanced Encryption Standard (AES). **Network Security**, NLD, v.2009, n.12, p.8–12, 2009.

HETTWER, B.; GEHRER, S.; GÜNEYSU, T. Profiled Power Analysis Attacks Using Convolutional Neural Networks with Domain Knowledge. In: IEEE, 2019, Calgary, AB, Canada. **Anais...** [S.l.: s.n.], 2019. v.11349 LNCS, p.479–498.

HETTWER, B.; GEHRER, S.; GüNEYSU, T. Applications of machine learning techniques in side-channel attacks: a survey. **Journal of Cryptographic Engineering**, Springer Link, v.10, n.2, p.135–162, 2020.

HETTWER, B. et al. Lightweight Side-Channel Protection using Dynamic Clock Randomization. In: INTERNATIONAL CONFERENCE ON FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 2020., 2020, Gothenburg, Sweden. **Anais...** IEEE, 2020. p.200–207.

HETTWER, B. et al. Deep learning multi-channel fusion attack against side-channel protected hardware. **Proceedings - Design Automation Conference - DAC**, Moscone West, San Francisco, v.2020-July, 2020.

HEUSER, A.; PICEK, S.; GUILLEY, S.; MENTENS, N. Side-channel analysis of lightweight ciphers: Does lightweight equal easy? In: IEEE, 2017, Hong Kong, China. **Anais...** Springer, 2017. v.10155 LNCS, n.July, p.91–104.

HORNIK, K.; STINCHCOMBE, M. B.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, GBR, v.2, n.5, p.359–366, 1989.

HU, H.; PENG, R.; TAI, Y.; TANG, C. Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures. **arXiv:1607.03250**, arxiv, v.abs/1607.03250, 2016.

KIM, M.; KWOK, J. T. Dynamic Unit Surgery for Deep Neural Network Compression and Acceleration. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), 2019, Budapest, Hungary. **Anais...** IEEE, 2019. p.1–8.

KIM, N. J.; KIM, H. Mask-Soft Filter Pruning for Lightweight CNN Inference. **International SoC Design Conference (ISOCC)**, Yeosu, Korea, p.2020–2021, 2020.

KIRIMTAT, A.; KREJCAR, O.; KERTESZ, A.; TASGETIREN, M. F. Future Trends and Current State of Smart City Concepts: A Survey. **IEEE Access**, IEEEexplore, v.8, p.86448–86467, 2020.

KITCHENHAM, B. A.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. USA: Keele University and Durham University Joint Report, 2007. (EBSE 2007-001).

KIZHVATOV, I. An Efficient Method for Random Delay Generation in Embedded Software. **Cryptographic Hardware and Embedded Systems - CHES 2009**, Lausanne, Switzerland, p.156–170, 2009.

KLUYVER, T. et al. Jupyter Notebooks - a publishing format for reproducible computational workflows. In: IEEE, 2016, Göttingen, Germany. **Anais...** IOS Press, 2016. p.87 – 90.

KNIGHT, A.; LEE, B. K. Performance Analysis of Network Pruning for Deep Learning based Age-Gender Estimation. **International Conference on Computational Science and Computational Intelligence (CSCI)**, Las Vegas, NV, USA, p.1684–1687, 2021.

KOCHER, P. C. **Timing Attacks on Implementations of Diffie-Hellman**. Santa Barbara, California, USA: Springer, 1996. 104–113p.

KOCHER, P.; JAFFE, J.; JUN, B. Differential Power Analysis. **Wiener M. (eds) Advances in Cryptology — CRYPTO' 99. CRYPTO 1999. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, Heidelberg**, Santa Barbara, California, USA, p.336–338, 1999.

KOLLEK, K.; AGUILAR, M. A.; BRAUN, M.; KUMMERT, A. Improving Neural Network Architecture Compression by Multi-Grain Pruning. In: IEEE INTERNATIONAL CONFERENCE ON PROGRESS IN INFORMATICS AND COMPUTING (PIC), 2021., 2021, Shanghai/Tampere. **Anais...** IEEE, 2021. p.6–13.

KUBOTA, T.; YOSHIDA, K.; SHIOZAKI, M.; FUJINO, T. Deep Learning Side-Channel Attack Against Hardware Implementations of AES. **Proceedings - Euromicro Conference on Digital System Design, DSD 2019**, Kallithea, Greece, p.261–268, 2019.

LAURET, P.; FOCK, E.; MARA, T. A node pruning algorithm based on a Fourier amplitude sensitivity test method. **IEEE Transactions on Neural Networks**, IEEExplore, v.17, n.2, p.273–293, 2006.

LE, T. H.; CLÉDIÈRE, J.; SERVIÈRE, C.; LACOUME, J. L. Efficient solution for misalignment of signal in side channel analysis. **ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings**, Honolulu, HI, USA, v.2, p.257–260, 2007.

LECUN, Y.; HAFFNER, P.; BOTTOU, L.; BENGIO, Y. Object Recognition with Gradient-Based Learning. In: SHAPE, CONTOUR AND GROUPING IN COMPUTER VISION, 1999. **Anais...** Springer, 1999. p.319. (Lecture Notes in Computer Science, v.1681).

LELLIS, R. N.; SOARES, R. I. FLUXO DE ATAQUE DPA/DEMA BASEADO NA ENERGIA DOS TRAÇOS PARA NEUTRALIZAR CONTRAMEDIDAS POR DESALINHAMENTO TEMPORAL EM CRIPTOSISTEMAS. **Dissertação de Mestrado - UFPel**, Pelotas, n.53, p.9610–9610, 2017.

LELLIS, R.; SOARES, R.; PERIN, G. Pruning-based Neural Network Reduction for Faster Profiling Side-Channel Attacks. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2022., 2022, Glasgow, Scotland. **Anais...** IEEE, 2022. p.1–4.

LERMAN, L.; BONTEMPI, G.; BEN TAIEB, S.; MARKOWITCH, O. A time series approach for profiling attack. In: IEEE, 2013, Kharagpur, India. **Anais...** Springer, 2013. v.8204 LNCS, p.75–94.

LERMAN, L.; MARTINASEK, Z.; MARKOWITCH, O. Robust profiled attacks: Should the adversary trust the dataset? **IET Information Security**, USA, v.11, n.4, p.188–194, 2017.

LERMAN, L.; POUSSIER, R.; MARKOWITCH, O.; STANDAERT, F. X. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. **Journal of Cryptographic Engineering**, Springer Link, v.8, n.4, p.301–313, 2018.

LI, H.; KRČEK, M.; PERIN, G. A comparison of weight initializers in deep learning-based side-channel analysis. In: SPRINGER, 2020, Rome, Italy. **Anais...** Springer, 2020. v.12418 LNCS, p.126–143.

LI, H. et al. **Pruning Filters for Efficient ConvNets**. arxiv: ArXiv, 2017.

LI, L.; ZHU, J.; SUN, M. T. Deep learning based method for pruning deep neural networks. **Proceedings - 2019 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2019**, Shanghai, China, p.312–317, 2019.

LIM, J.; HO, W.-g.; CHONG, K.-s.; GWEE, B.-h. DPA-Resistant QDI Dual-Rail AES S-Box Based on. **2017 IEEE International Symposium on Circuits and Systems (ISCAS)**, Baltimore, MD, USA, 2017.

LIU, P. C.; CHANG, H. C.; LEE, C. Y. A low overhead DPA countermeasure circuit based on ring oscillators. **IEEE Transactions on Circuits and Systems II: Express Briefs**, IEEEexplore, v.57, n.7, p.546–550, 2010.

LIU, P. C.; CHANG, H. C.; LEE, C. Y. A true random-based differential power analysis countermeasure circuit for an AES engine. **IEEE Transactions on Circuits and Systems II: Express Briefs**, IEEEexplore, v.59, n.2, p.103–107, 2012.

LODER, L. L. Proposta de um fluxo de ataque DPA para avaliar a vulnerabilidade de arquiteturas criptograficas protegidas por aleatorizacao de processamento. **Dissertação de Mestrado - UFPel**, Pelotas, n.750004, p.2–7, 2014.

MAGHREBI, H. Assessment of common side channel countermeasures with respect to deep learning based profiled attacks. **Proceedings of the International Conference on Microelectronics, ICM**, Cairo, Egypt, v.2019-Decem, p.126–129, 2019.

MAGHREBI, H.; PORTIGLIATTI, T.; PROUFF, E. Breaking Cryptographic Implementations Using Deep Learning Techniques. In: SECURITY, PRIVACY, AND APPLIED CRYPTOGRAPHY ENGINEERING, 2016, Cham. **Anais...** Springer International Publishing, 2016. p.3–26.

MANGARD, S.; OSWALD, E.; POPP, T. Power analysis attacks - revealing the secrets of smart cards. In: SPRINGER INTERNATIONAL PUBLISHING, 2007, New York, NY. **Anais...** Springer Publishing Company: Incorporated, 2007.

MARTINASEK, Z.; MALINA, L.; TRASY, K. Profiling power analysis attack based on multi-layer perceptron network. **Computational Problems in Science and Engineering**, Springer Link, v.343, p.317–339, 2015.

MATLAB. **version 7.10.0 (R2010a)**. Natick, Massachusetts: The MathWorks Inc., 2010.

MUKHERJEE, S. A Power Modeling Approach to Protect GPUs from Side-Channel Attacks. , USA, n.April, 2020.

MUKHTAR, N.; KONG, Y. Hyper-parameter optimization for machine-learning based electromagnetic side-channel analysis. **26th International Conference on Systems Engineering, ICSEng 2018 - Proceedings**, [S.l.], 2019.

NAGASHIMA, S. et al. DPA Using Phase-Based Waveform Matching against Random-Delay Countermeasure. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 2007., 2007, New Orleans, LA, USA. **Anais...** IEEE, 2007. p.1807–1810.

OZGEN, E.; PAPACHRISTODOULOU, L.; BATINA, L. Template attacks using classification algorithms. **Proceedings of the 2016 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016**, Washington DC, USA, p.242–247, 2016.

PARSIFAL. **Parsifal (2.3)**. Disponível em: <https://parsif.al/>.

PERIN, G.; PICEK, S. **On the Influence of Optimizers in Deep Learning-based Side-channel Analysis**. `https://eprint.iacr.org/2020/977`, Cryptology ePrint Archive, Paper 2020/977. Disponível em: <https://eprint.iacr.org/2020/977>.

PERIN, G.; WU, L.; PICEK, S. Gambling for Success : The Lottery Ticket Hypothesis in Deep Learning-based SCA. **Cryptology ePrint Archive, Paper 2021/197**, IACR, p.1–29, 2021.

PESSL, P.; MANGARD, S. Enhancing side-channel analysis of binary-field multiplication with bit reliability. **Proceedings of the RSA Conference on Topics in Cryptology - CT-RSA 2016**, Berlin, Heidelberg, v.9610, p.255–270, 2016.

PICEK, S.; HEUSER, A.; JOVIC, A. Improving Side-Channel Analysis Through Semi-supervised Learning Improving Side-channel Analysis through Semi-supervised Learning. **Smart Card Research and Advanced Applications - CARDIS**, Montpellier, France, n.April 2019, 2018.

POLYAK, A.; WOLF, L. Channel-level acceleration of deep face representations. **IEEE Access**, IEEExplore, v.3, p.2163–2175, 2015.

PROUFF, E. et al. Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. **IACR Cryptol. ePrint Arch.**, IACR, v.2018, p.53, 2018.

PU, K. et al. A Quantitative Analysis of Non-Profiled Side-Channel Attacks Based on Attention Mechanism. **Electronics**, Switzerland, v.12, n.15, 2023.

QADIR, A. M.; VAROL, N. A Review Paper on Cryptography. In: INTERNATIONAL SYMPOSIUM ON DIGITAL FORENSICS AND SECURITY (ISDFS), 2019., 2019, Barcelos, Portugal. **Anais...** IEEE, 2019. p.1–6.

RAMEZANPOUR, K.; AMPADU, P.; DIEHL, W. SCAUL: Power Side-Channel Analysis With Unsupervised Learning. **IEEE Transactions on Computers**, IEEExplore, v.69, n.11, p.1626–1638, 2020.

ROBISSOUT, D. et al. Online Performance Evaluation of Deep Learning Networks for Profiled Side-Channel Analysis. In: IEEE, 2021, Lugano, Switzerland. **Anais...** Springer, 2021. v.12244 LNCS, n.March, p.200–218.

SCHOLAR, G. **Google Acadêmico**. `https://scholar.google.com.br/`.

SINGH, A. et al. Enhanced Power and Electromagnetic SCA Resistance of Encryption Engines via a Security-Aware Integrated All-Digital LDO. **IEEE Journal of Solid-State Circuits**, IEEExplore, v.55, n.2, p.478–493, 2020.

SOARES, R. I. **Arquitetura GALS pipeline para criptografia robusta a ataques DPA e DEMA**. 2010. Tese (Doutorado em Ciência da Computação) — PUC-RS.

SPECHT, R. et al. Dividing the threshold: Multi-probe localized EM analysis on threshold implementations. In: IEEE INTERNATIONAL SYMPOSIUM ON HARDWARE ORIENTED SECURITY AND TRUST (HOST), 2018., 2018, Washington DC, USA. **Anais...** IEEE, 2018. p.33–40.

STANDAERT, F.-X.; MALKIN, T. G.; YUNG, M. **A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks (extended version)**. `https://eprint.iacr.org/2006/139`, Cryptology ePrint Archive, Paper 2006/139. Disponível em: <https://eprint.iacr.org/2006/139>.

TAKEDA, R.; NAKADAI, K.; KOMATANI, K. Node Pruning Based on Entropy of Weights and Node Activity for Small-Footprint Acoustic Model Based on Deep Neural Networks. In: INTERSPEECH, 2017, Stockholm, Sweden. **Anais...** INTERSPEECH, 2017.

TIBSHIRANI, R. Regression shrinkage and selection via the LASSO. **Journal of the Royal Statistical Society Series B: Statistical Methodology**, England, v.58, n.1, p.267–288, 1996.

TIMON, B. Non-Profiled Deep Learning-Based Side-Channel Attacks. **Cryptology ePrint Archive**, IACR, p.1–34, 2018.

TIMON, B. Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis. **IACR Transactions on Cryptographic Hardware and Embedded Systems**, IACR, v.2019, n.2, p.107–131, 2019.

TRICHINA, E.; De Seta, D.; GERMANI, L. Simplified Adaptive Multiplicative Masking for AES. **Cryptographic Hardware and Embedded Systems - CHES 2002**, Redwood Shores, CA, USA, v.2523, p.187–197, 2003.

UNIVERSITY, C. . E. department of the Télécom ParisTech french. **DPA Contest**. Paris: DPA Contest, 2015. `http://www.dpacontest.org/v4/42_traces.php`.

VAN DER VALK, D.; KRCEK, M.; PICEK, S.; BHASIN, S. Learning from a big brother - Mimicking neural networks in profiled side-channel analysis. **Proceedings - Design Automation Conference**, Moscone West, San Francisco, v.2020-July, 2020.

VAN ROSSUM, G.; DRAKE JR, F. L. **Python reference manual**. Amsterdan: Centrum voor Wiskunde en Informatica Amsterdam, 1995.

WANG, B. et al. Against Double Fault Attacks: Injection Effort Model, Space and Time Randomization Based Countermeasures for Reconfigurable Array Architecture. **IEEE Transactions on Information Forensics and Security**, IEEExplore, v.11, n.6, p.1151–1164, 2016.

WANG, H. et al. SCARF: Detecting Side-Channel Attacks at Real-time using Low-level Hardware Features. **Proceedings - 2020 26th IEEE International Symposium on On-Line Testing and Robust System Design, IOLTS 2020**, Virtual, 2020.

WANG, R.; WANG, H.; DUBROVA, E. Far Field EM Side-Channel Attack on AES Using Deep Learning. **ASHES 2020 - Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security**, Orlando, Florida, USA, p.35–44, 2020.

WEISSBART, L. Performance analysis of multilayer perceptron in profiling side-channel analysis. In: INTERSPEECH, 2020, Rome, Italy. **Anais...** Springer, 2020. v.12418 LNCS, p.198–216.

WEISSBART, L.; PICEK, S.; BATINA, L. One trace is all it takes: machine learning-based side-channel attack on EDDSA. In: SPRINGER, 2019, Gandhinagar, India. **Anais...** Springer, 2019. v.11947 LNCS, p.86–105.

WU, L. et al. **Ablation Analysis for Multi-device Deep Learning-based Physical Side-channel Analysis**. https://eprint.iacr.org/2021/717, Cryptology ePrint Archive, Paper 2021/717. Disponível em: <https://eprint.iacr.org/2021/717>.

XU, J.; TANG, Y.; WANG, Y.; WANG, X. A Practical Side-Channel Attack of a LoRaWAN Module Using Deep Learning. **Proceedings of the IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)**, Xiamen, China, v.2019-Octob, p.17–21, 2019.

YAN, Y. et al. Channel Pruning via Multi-Criteria based on Weight Dependency. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), 2021, Virtual. **Anais...** IEEE, 2021. p.1–8.

YANG, S.; ZHOU, Y.; LIU, J.; CHEN, D. Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations. In: IEEE, 2012, Seoul, Korea. **Anais...** Springer Link, 2012. v.7259 LNCS, p.169–185.

YAO, J. B.; ZHANG, T. Insert Random Time-Delay Defense High Order Side-Channel Attack. **Advanced Engineering Forum**, Switzerland, v.6-7, p.169–174, 2012.

YU, F. et al. Gate Trimming: One-Shot Channel Pruning for Efficient Convolutional Neural Networks. **2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**, Toronto, Canada, p.1365–1369, 2021.

YU, W.; CHEN, J. Deep learning-assisted and combined attack: A novel side-channel attack. **Electronics Letters**, USA, v.54, n.19, p.1114–1116, 2018.

ZAID, G.; BOSSUET, L.; HABRARD, A.; VENELLI, A. Methodology for Efficient CNN Architectures in Profiling Attacks. **IACR Transactions on Cryptographic Hardware and Embedded Systems**, IACR, v.2020, n.1, p.1–36, 2020.

ZHANG, F. et al. From homogeneous to heterogeneous: Leveraging deep learning based power analysis across devices. **Proceedings - Design Automation Conference**, Moscone West, San Francisco, v.2020-July, 2020.

ZHANG, L. et al. Multi-label Deep Learning based Side Channel Attack. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEExplore, v.0070, n.c, 2020.

ZHOU, Y.; STANDAERT, F. X. Deep learning mitigates but does not annihilate the need of aligned traces and a generalized ResNet model for side-channel attacks. **Journal of Cryptographic Engineering**, Springer Link, v.10, n.1, p.85–95, 2020.

# 8 ASSINATURAS

Rodrigo Nuevo Lellis
Proponente

Rafael Iankowski Soares
Prof. Orientador

Gilherme Perin
Prof. Coorientador