Mixup Data Augmentation for Deep Learning Side-Channel Attacks

Karim M. Abdellatif

Ledger, France
karim.abdellatif@ledger.fr

Abstract. Following the current direction in Deep Learning (DL), more recent papers have started to pay attention to the efficiency of DL in breaking cryptographic implementations. Several works focus on techniques to boost the efficiency of existing architectures by data augmentation, regularization, etc. In this work, we investigate using mixup data augmentation [21] in order to improve the efficiency of DL-based Side-Channel Attacks (SCAs). We validated the soundness of the mixup on real traces collected from the ChipWhisperer board [14] and from the ASCAD database [1]. The obtained results have proven that using mixup data augmentation decreases the number of measurements needed to reveal the secret key compared to the non-augmented case.

Keywords: Deep Learning · Side-Channel Attacks · Data Augmentation · Mixup · AES.

1 Introduction

Hardware security has become an important system and application metric. It is considered as a key requirement for smart cards, smart phones, Internet of Things (IoT) devices, hardware wallets, and so on. With the increase in the number and forms of such systems, the devices that store critical information become more approachable to malicious attackers. These devices are potentially susceptible to physical attacks that aim at breaking cryptosystems by gaining information from their implementation instead of using theoretical weaknesses.

Physical threats appear at circuit level, where an attacker can measure or physically influence the computation/operation performed by the circuit. Side channel attacks exploit additional sources of information (physical observations), including timing information [9], power consumption [8], electromagnetic emissions (EM) [16], remote monitoring [23], etc.

Different SCAs like Differential Power Analysis (DPA) [1] and Correlation Power Analysis (CPA) [3] have been invented and demonstrated to be realistic threats to many critical embedded systems. They exploit the correlation between the intermediate data in algorithms and the power consumption of implementations to reveal sensitive information.

Recently, profiled side-channel attacks using machine learning techniques have received a significant amount of attention in the SCA community. Such attacks [10] proved to be very efficient compared to classical attacks, like template attacks [3]. The strength of such attacks come from their capability to fully characterize the DUT. The attacker has a full control over a cloned device, which can be used to build the profiling model. This model is then used by the attacker to target similar devices to extract the secret information. The standard machine learning algorithms like Random Forest[15, 11] and Support Vector Machine [5] have been the common choice for such attacks.

More recently, as an efficient alternative to machine learning techniques, DL has been used as a powerful technique for side-channel attacks [13, 2]. The related practical results have showed that such technique is very efficient even under the presence of countermeasures. Cagli et al. [2] presented a deep-learning based approach utilizing Convolutional Neural Networks (CNNs) to perform a successful attack, even in presence of trace misalignments. Masking-based countermeasures were also shown to be broken using Multi-Layer Perception (MLP) and CNN as shown in [4, 12].

Moreover, to strengthen DL-based SCAs, recent works showed techniques to further improve their attacking strength. The authors in [7] showed that adding zero-mean Gaussian noise is helping to generalize the DL model and improves the success of the attack. Cagli et al. [2] showed that applying Data Augmentation (DA) techniques can overcome the jitter and noise countermeasures.

Our Contributions. In this paper, we present using mixup DA technique for improving DL-based SCAs. To the best of our knowledge, this has not been studied yet. Experimental validation is performed on real traces from ChipWhisperer [14] and ASCAD database [1]. The idea of Mixup is to generate new traces from the main traces that can boost the DL performance. We show that using such DA methodology improves the performance of DL-based SCAs by reducing the number of traces needed for a successful key recovery.

Paper Organization. This paper is organized as follows. **Section 2** provides a background on the previous work on data augmentation. Afterwards, we explain mixup data augmentation for SCAs in **Section 3**. **Section 4** and **Section 5** highlight the experimental results of using mixup DA on ASCAD and ChipWhisperer, respectively. **Section 6** presents a short discussion about the obtained results. Finally, we provide the conclusion and further works in **Section 7**.

2 Data Augmentation

Data augmentation (DA) has been demonstrated to achieve considerable performance improvement for deep learning (DL) by increasing accuracy and stability with overfitting reduction [6, 22, 19]. From the prospective of SCAs, DA was investigated as a solution to break jitter-based countermeasures [2]. In [2], authors

applied random shift to existing traces to perform DA and they proved avoiding overfitting, resulting in a better training of CNN.

The methodology of random shifting presented by [2] was based on a dedicated code. However, in [20], the authors used ImageDataGenerator1 class in the Keras DL library to provide DA.

3 Mixup Data Augmentation for SCAs

Mixup data augmentation was first proposed by [22]. It trains a neural network on convex combinations of pairs of examples and their labels. In a nutshell, mixup can be presented as follows:

$$\widehat{x} = \lambda x_i + (1 - \lambda) x_j \tag{1}$$

$$\widehat{y} = \lambda y_i + (1 - \lambda) y_j \tag{2}$$

where (x_i,y_i) and (x_j,y_j) are two examples taken randomly from the training data (x is like an image and y is its hot encoding label) and $\lambda \sim$ Beta (α,α) for each pair of examples, with α a hyperparameter. For the sake of clarity, Fig. 1 shows an example of Mixup in case of $\lambda = 0.5$.

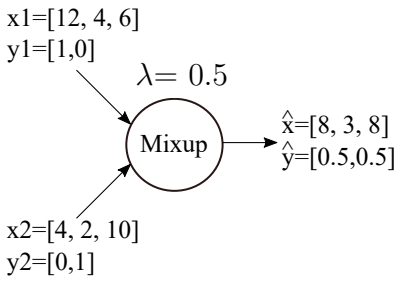


Fig. 1: Example of Mixup

Zhang et al. [22] used the value $\alpha = 1$, which results in a uniform distribution between 0 and 1. It was found also by [22] that on larger datasets such as ImageNet [17], a smaller value of α was required due to underfitting.

The motivation behind mixup comes from the linearity between training examples. As a conclusion from [22], the linearity is an effective inductive bias for most models. Indeed, mixup was shown to be useful across a wide variety of tasks and models [22, 18].

Therefore, mixup extends the training data-set by the prior linear knowledge between dataset samples. It can be implemented in a few lines of code, and introduces minimal computation overhead.

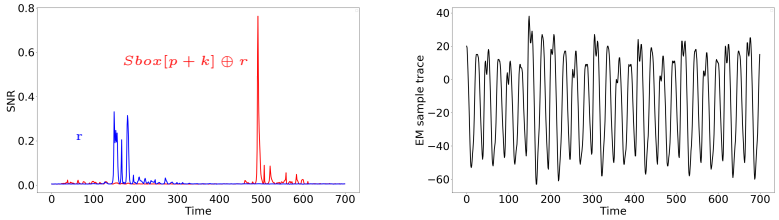


Fig. 2: ASCAD dataset

In order to investigate the effect of mixup data augmentation, we target CNN and MLP models which are the most common DL methods used by the SCA community. For each method (MLP or CNN), we compare the model efficiency (accuracy and key rank) with and without data augmentation. The key rank is calculated by the guessing entropy which gives the average ranking of the secret key K within a vector of key guesses. The vector of key guesses $g_{x,1}$, ... , $g_{x,k}$ for the xth measurement is calculated by mapping each key guess k to a label y with probability $P_{x,y}$ and applying the maximum-likelihood principle over 1 to N, where N is the number of traces.

4 Experimental Results on ASCAD

ASCAD is a public database introduced by Benadjila et al. in [1]. It is used as a common database for research on DL-based SCAs. It is based on a first order protected Software AES implementation running on an 8-bit ATMega8515 board. It is composed of two set of traces: a profiling set of 50k traces and attack set of 10k traces.

Each trace of ASCAD dataset consists of 700 samples focusing on the manipulation of the third byte of the masked state $Sbox(p \oplus k) \oplus r$, where *p, k and* r are respectively the plaintext, the key and the mask values.

To evaluate the amount of leakage in ASCAD database, we use the Signal-to-Noise-Ratio (SNR), which is calculated by Eq. 3. It gives the ratio between the deterministic data-dependent leakage and the remaining noise. Fig. 2 shows the leakage detection (left) and a trace example of ASCAD dataset (right).

$$SNR = \frac{Var(E(X|Y))}{E(Var(X|Y))} \tag{3}$$

where X is the captured trace, Y is the label that is determined, E is the expectation, Var is the variance of a random variable.

DL-based techniques like CNN and MLP have showed a high efficiency for attacking this kind of countermeasures as shown in [1, 12]. Combining the two leakages of $Sbox(p \oplus k) \oplus r$ and r will be performed by the DL architecture in order to act as a first order profiling.

ASCAD contains 50k traces which are used for profiling. We use Mixup DA (see Eq. 1 and Eq. 2), using $\alpha = 0.2$ to generate 100k additional traces from the 50k traces already available. The choice of this value for α is motivated by the results of [17].

4.1 MLP performance

Different MLP architectures were reported in the previous works for ASCAD database as shown in [1, 12]. Our MLP model is shown in List. 1.1. It is composed of five dense layers and one SoftMax layer. We added Dropout layers to avoid overfitting. Batch size equals 128, the number of epochs is 100, and the learning rate is 0.0001. We used the model accuracy as a metric to evaluate the efficiency of the DL model.

```
def mlp_ascad(node=600,hidden_layer_nb=4):
    model = Sequential()
    model.add(Dense(node, input_dim=700, activation='relu'))
    for i in range(hidden_layer_nb):
        model.add(Dense(node, activation='relu'))
        Dropout(0.2)
    model.add(Dense(256, activation='softmax'))
    optimizer = RMSprop(lr=0.00001)
    model.compile(loss='categorical_crossentropy', optimizer=
    optimizer, metrics=['accuracy'])
    return model
```

Listing 1.1: ASCAD MLP

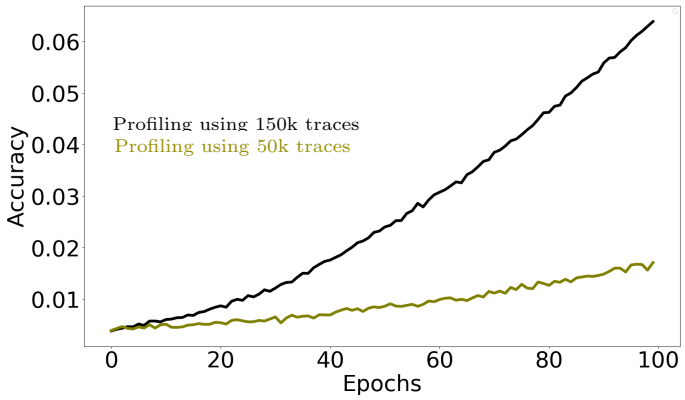


Fig. 3: MLP performance on ASCAD

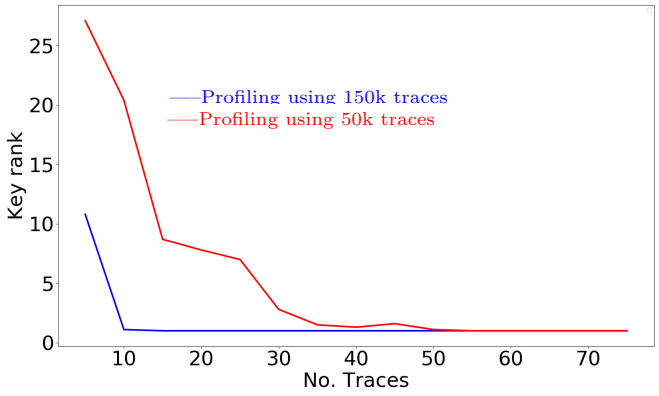


Fig. 4: MLP success rate of ASCAD

As shown in Fig. 3, the architecture performance in case of using data augmentation (profiling using 150k traces) outperforms the case of not using data augmentation (profiling using 50k traces). In order to evaluate the attack efficiency, the correct key rank was computed as shown in Fig. 4. The number of traces needed for a successful key recovery in case of using mixup DA is 10 compared to 55 traces in case of non-augmented profiling.

4.2 CNN performance

For CNN, we consider the architecture shown in List. 1.2. It is composed of two convolutional blocks and two fully-connected layers. The first convolutional layer has a filter size of 32 and the second layer decreases the filter size by a factor of 2. Batch size equals 128, the number of epochs is 100, and the learning rate is 0.0001.

```
def cnn_ascad():
    input_shape = (700,1)
    img_input = Input(shape=input_shape)
    x = Conv1D(32, 32, activation='relu', padding='same',
    name='block1_conv1')(img_input)
    Dropout(0.1)
    x = AveragePooling1D(2, strides=2, name='block1_pool')(x)
    x = Conv1D(16, 16, activation='relu', padding='same',
    name='block2_conv1')(x)
    Dropout(0.1)
    x = AveragePooling1D(2, strides=2, name='block2_pool')(x)
    x = Flatten(name='flatten')(x)
    x = Dense(400, activation='relu', name='fc1')(x)
    x = Dense(400, activation='relu', name='fc2')(x)
```

```
    x = Dense(256, activation='softmax', name='predictions')(x)
    inputs = img_input
    model = Model(inputs, x, name='cnn_test')
    optimizer = RMSprop(lr=0.00001)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model
```

Listing 1.2: ASCAD CNN

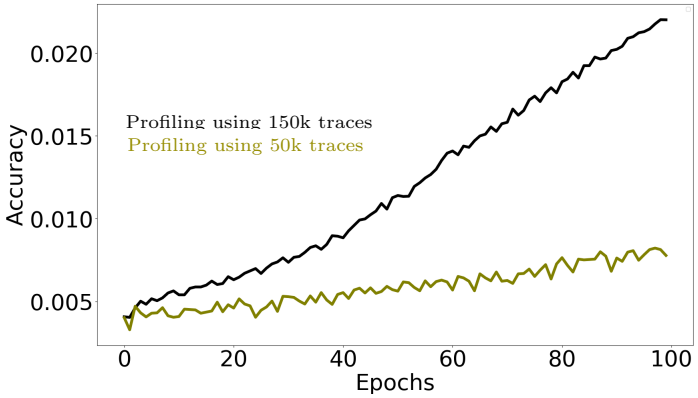


Fig. 5: CNN performance on ASCAD

The result for the accuracy is shown in Fig. 5. The obtained results for the architecture performance confirm the previous results in case of MLP. Moreover, mixup DA reaches the key rank to 0 in less than 45 measurements compared to 200 traces in case of not using any DA (see Fig. 6).

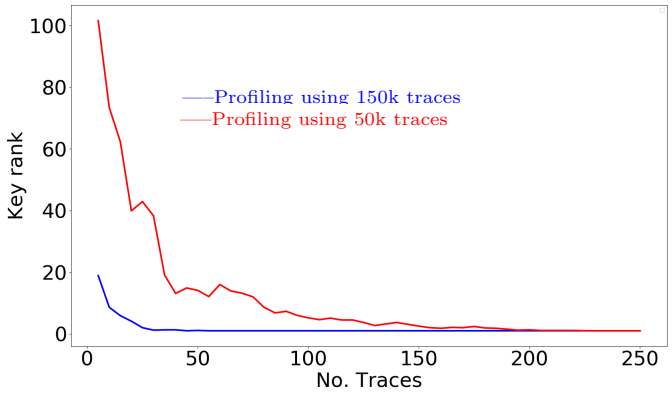


Fig. 6: Success rate

5 Experimental results on CW

In addition to using ASCAD, we present experimental results on traces from ChipWhisperer. A first order protected AES was implemented on ChipWhisperer-lite [14]. We collected 100k traces for profiling and 5k traces for testing. Fig. 7 shows the SNR and trace sample (POI). By using mixup DA, additional 200k traces were generated.

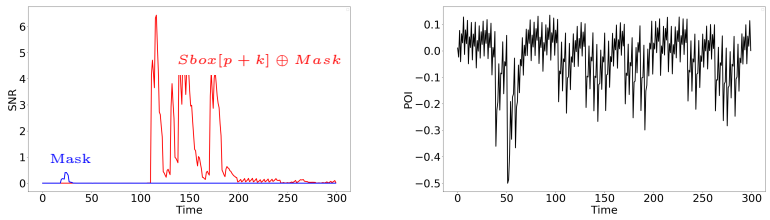


Fig. 7: CW leakage

5.1 MLP performance

We used the same architecture shown in List. 1.1 but we increased the number of epochs to be **200**. As shown in Fig. 8, the architecture performance in case of using the mixup DA (profiling using 300k traces) outperforms the case of not using DA (profiling using 100k traces). Also, from Fig. 9, the number of needed traces for a successful key recovery with mixup DA is less than the non-augmented case.

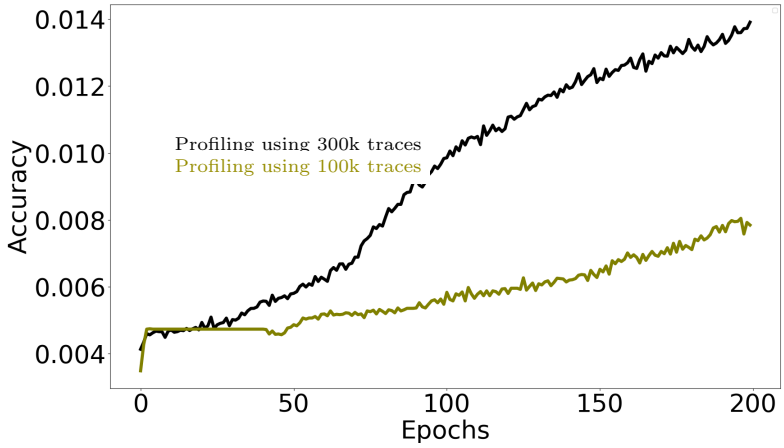
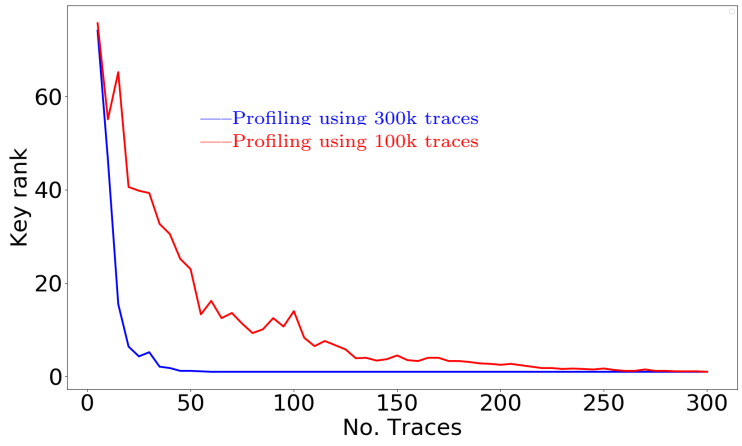


Fig. 8: MLP performance of CW



hh

Fig. 9: MLP Success rate of CW

5.2 CNN performance

For CNN, we also consider the same architecture shown in List. 1.2. However, the number of epochs is increased to **500**. The results for the accuracy are shown in Fig. 10. The obtained results for the architecture performance confirm the previous results in case of MLP. In addition, mixup data DA reaches the key rank to 0 with 60 traces compared to 350 traces in case of not using the mixup DA (see Fig. 11).

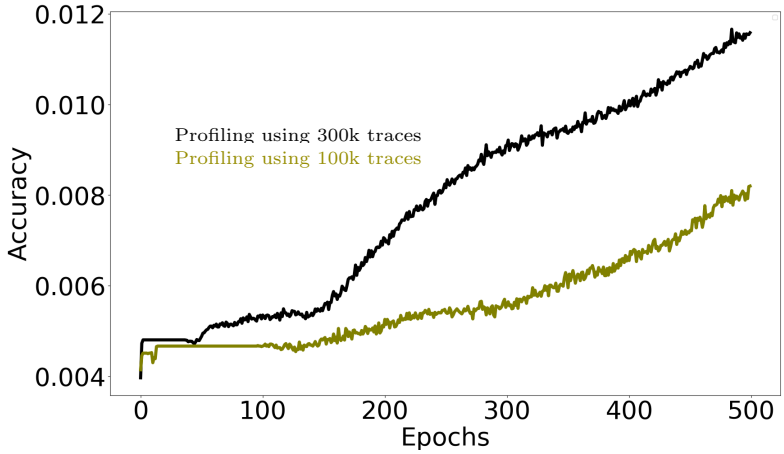


Fig. 10: CNN performance

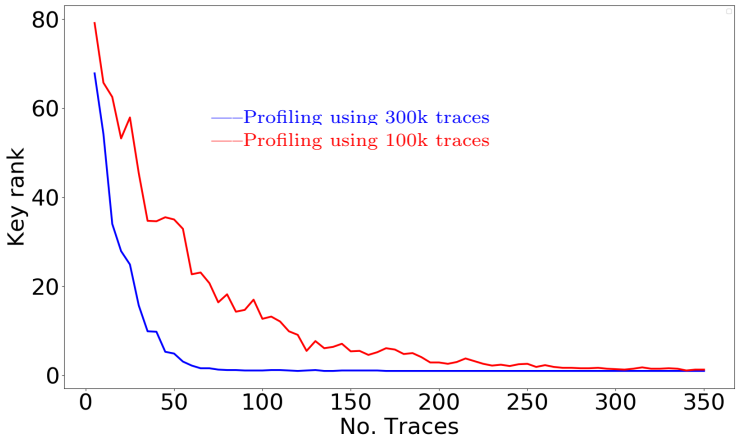


Fig. 11: CNN success rate of CW

6 Discussion

From the previous results of MLP and CNN on ASCAD and Chipwhisperer traces, we can find the positive impact of mixup DA on the attack success. On the other hand, the mixup which was used in this work is based on Eq. 1 and Eq. 2 and it is a linear combination between any two samples from the dataset. However, several non-linear methods were proposed in [18] like vertical and horizontal concatenations, random elements,..., and noisy mixup. The two

datasets used in this work (ASCAD and Chipwhisperer) are based on a first order protected AES (masked AES). However, Jitter-based countermeasures are based on creating non-synchronized traces, which make the attack more difficult. Therefore, it is motivating to implement non-linear mixup DA for jitter-based countermeasures.

7 Conclusion and Further works

In this work, we tackle DA for SCAs. We investigated for the first time, the effect of mixup DA on SCAs. By using MLP and CNN on two different databases, we showed how such technique can improve the attack efficiency (key rank).

The mixup used in this paper is based on a linear combination between samples, which won't be suitable in case of jitter-based countermeasures. Therefore, studying the non-linear mixup methods proposed by [18] on jitter-based datasets is very motivating as a future work.

References

1. Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep Learning for Side-Channel Analysis and Introduction to ASCAD Database. *Journal of Cryptographic Engineering*, 10(2):163–188, 2020.
2. Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional Neural Networks with Data Augmentation against Jitter-based Countermeasures. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 45–68. Springer, 2017.
3. Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template Attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.
4. Richard Gilmore, Neil Hanley, and Maire O'Neill. Neural Network Based Attack on a Masked Implementation of AES. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 106–111. IEEE, 2015.
5. Annelie Heuser and Michael Zohner. Intelligent Machine Homicide. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 249–264. Springer, 2012.
6. Hiroshi Inoue. Data Augmentation by Pairing Samples for Images Classification. *arXiv preprint arXiv:1801.02929*, 2018.
7. Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make Some Noise. Unleashing The power of Convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 148–179, 2019.
8. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.
9. Paul C Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and other Systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

10. Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. Power Analysis Attack: An Approach Based on Machine Learning. *International Journal of Applied Cryptography*, 3(2):97–115, 2014.
11. Liran Lerman, Romain Poussier, Gianluca Bontempi, Olivier Markowitch, and François-Xavier Standaert. Template Attacks vs. Machine Learning Revisited (and the curse of dimensionality in side-channel analysis). In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 20–33. Springer, 2015.
12. Houssem Maghrebi. Deep Learning based Side Channel Attacks in Practice. *IACR Cryptol. ePrint Arch.*, 2019:578, 2019.
13. Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking Cryptographic Implementations Using Deep Learning Techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.
14. Newae. *Chipwhisperer-Lite*, 2020. http://store.newae.com/chipwhisperer-lite-cw1173-basic-board.
15. Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-Channel Evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):1–29, 2019.
16. Jean-Jacques Quisquater and David Samyde. Electromagnetic Analysis (ema): Measures and Counter-Measures for Smart Cards. In *International Conference on Research in Smart Cards*, pages 200–210. Springer, 2001.
17. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet Large Scale Visual Recognition Challenge. *International journal of computer vision*, 115(3):211–252, 2015.
18. Cecilia Summers and Michael J Dinneen. Improved Mixed-Example Data Augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1262–1270. IEEE, 2019.
19. Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Learning from Between-Class Examples for Deep sound Recognition. *arXiv preprint arXiv:1711.10282*, 2017.
20. Yoo-Seung Won, Dirmanto Jap, and Shivam Bhasin. Push For More: On Comparison of Data Augmentation and SMOTE With Optimised Deep Learning Architecture For Side-Channel.
21. Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. *arXiv preprint arXiv:1710.09412*, 2017.
22. Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. *arXiv preprint arXiv:1710.09412*, 2017.
23. Mark Zhao and G Edward Suh. FPGA-Based Remote Power Side-Channel Attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 229–244. IEEE, 2018.