Train or Adapt a Deeply Learned Profile?

Christophe Genevey-Metat, Annelie Heuser, Gerard Benoit

Train or Adapt a Deeply Learned Profile?

Christophe Genevey-Metat[1][0000−0002−1901−626X], Annelie Heuser[1][0000−0002−1095−5420], and Benoît Gérard[1,2][0000−0002−0598−2387]

[1] Univ Rennes, Inria, CNRS, IRISA, France
[2] Direction Générale de l'Armement

Abstract. In recent years, many papers have shown that deep learning can be beneficial for profiled side-channel analysis. However, to obtain good performance with deep learning, an evaluator or an attacker face the issue of data. Due to the context, he might be limited in the amount of data for training. This can be mitigated with classical Machine Learning (ML) techniques such as data augmentation. However, these mitigation techniques lead to a significant increase in the training time; first, by augmenting the data and second, by increasing the time to perform the learning of the neural network.
Recently, weight initialization techniques using specific probability distributions have shown some impact on the training performances in side-channel analysis. In this work, we investigate the advantage of using weights initialized from a previous training of a network in some different contexts. The idea behind this is that different side-channel attacks share common points in the sense that part of the network has to understand the link between power/electromagnetic signals and the corresponding intermediate variable. This approach is known as Transfer Learning (TL) in the Deep Learning (DL) literature and has shown its usefulness in various domains. We present various experiments showing the relevance and advantage of starting with a pretrained model
In our scenarios, pretrained models are trained on different probe positions/channels/chips. Using TL, we obtain better accuracy and/or training speed for a fixed amount of training data from the target device.
Keywords: Side-channel analysis, profiling attacks, neural networks, electromagnetic emanations, transfer learning

1 Introduction

Since its introduction at the end of the 90's [14], the exploitation of side-channel information observed from a cryptographic device has grown significantly. f Using physical quantities such as time, heat, power, electromagnetic field, photon emission, sound, it is possible to recover secret data. Defenses against side-channel attacks can be found in smart-cards, set-top boxes, video game consoles, or smartphones for instance.

Profiled attacks are considered as the strongest type of side-channel attacks. They are based on the principle that the attacker is able to derive a relevant leakage model for the targeted device. Template attacks [5] are considered as optimal [10] and often the noise distribution is modeled as multivariate Gaussian.

However, this approach may not be the most effective in practice due to some limitations. Among them, the Gaussian model that may not be perfectly suited or the spreading of the leakage points (e.g. due to some jitter) that may lead to intractable computations for model estimation. A new trend in side-channel analysis (SCA) is to explore the use of deep learning (DL) tools for profiled attacks [20,2]. These tools may help in solving problems such as trace misalignment, or high dimensional data in combination with masking countermeasures. However, deep learning tools still require a sufficient amount of data to construct relevant models, which may not be possible in particular scenarios.

Data augmentation [4] has been shown as a solution to cope with insufficient amount of data in some contexts. On the downside, adding data increases the amount of resources needed for both generating and processing this additional data. In this paper, we investigate the opportunity of exploiting information previously learnt from a different context to mount a new attack.

Traditionally, the dataset for profiling is assumed to come from a distribution identical to the one of the target device. In the research community, often only one dataset is measured, which is then divided into profiling and attacking datasets. Lately, the problematic of portability has been brought up and investigated [6,3,8]. In these works, portability refers to the differences between training and attacking dataset distributions and the consequences on the learnt model. The differences investigated in these works mostly arise due to fixed key alterations or variations in the manufacturing process of the device. The authors show that indeed the impact of the differences in distribution are notably in the effectiveness of the side-channel attack.

So far, the changes investigated could be mostly considered as minor compared to scenarios tackled, for example, in the field of image classification or computer vision. In the same context, transfer learning has recently[3] gained attention in the deep learning community [9], which allows to refine models that have been built solving different – but still related – problems. These pretrained models may allow to build accurate models for different tasks in a time-saving way as less data and training may be required.

Following this path, we investigate if data coming from sources that are not identical to the target dataset can actually benefit the side-channel attacker or evaluator. In particular, we adapt the concept of pretrained models to the side-channel community and extend the currently used attacker models. One could argue that pretrained models could be open-sourced and available in the community easily as it is done, for example, in the image classification domain.

In this paper, we investigate if using pretrained weights from different contexts, such as

- different EM probe positions,
- different side-channel sources (power instead of EM), and
- different devices,

could benefit an attacker/evaluator. We experiment both the naive approach of directly using the pretrained model itself and the transfer learning approach

[3] The concept itself has been already discussed in 1995 at NIPS [21].

where the pretrained models are first fine-tuned for the new task with some available labelled data. The latter one can be seen as initializing the network weights with some application-dependent values.

2 State-of-the-art on deep learning techniques for SCA

First works using machine learning techniques in side-channel analysis showed that Support Vector Machine (SVM) and Random Forest (RF) are effective profiled side-channel attacks [17,12]. Particularly, when the size of the training dataset is limited, SVM can be more efficient than Gaussian templates due to the underlying estimation problem [11]. More recently, deep learning techniques have shown to be even more advantageous in several settings. Using the advantages of deep learning in side-channel analysis is becoming a very "fruitful" topic, with newly published works very frequently. Nevertheless, how to use the full potential of deep learning for side channel analysis has not been developed yet. One of the first works [20] showed that when an implementation is protected with a masking countermeasure, neural networks can reveal sensitive key information even without the need of a higher order combination function [22] or an additional step of point of interest selection. Shortly after the introduction of deep learning techniques for side-channel analysis, a database of side-channel measurements (called ASCAD) has been published [2] to facilitate comparable research works in this direction. The database consists of EM measurements of an AES-128 implementation protected with a masking countermeasure. Furthermore, the authors provide a software tool to artificially add a random delay countermeasure. Together with the database, the authors provide a study of neural network architectures, parameter selection, and pretrained neural network models. On the same lines as against masking countermeasures, neural networks are extremely effective against random delay countermeasures [4].

To strengthen profiled side-channel attacks based on neural networks, recent works showed techniques to further improve their attacking strength. The authors [4] highlighted that data augmentation techniques, i.e., the addition of artificial data, are significantly improving the success of an attack when shuffling (jitter-based) protections are present. A practical parameter selection guide is given by Maghrebi [19], i.e. the author provides some recommendations and practical hints to either enhance the efficiency from an adversary's perspective or to strengthen the resistance of the cryptographic implementations against these attacks from a security developer's perspective. A follow-up on parameter selection has also been published by Zaid *et al.* [27] where authors try to find minimal networks to greatly improve the training time at a negligible cost in terms of final accuracy, which was further improved by Wouters *et al.* [26]. The influence of weight initalizations on CNN has been compared in terms of guessing entropy by Li et al. [18]. A realistic real-world study has been performed by Bhasin *et al.* [3], and similarly by Das *et al.* [8], and by Wang *et al.* [25]. These works investigated the scenario when the profiling and attacking devices are different (to some extent), which is relevant in practice, but not always studied in research. In

addition the work [25] also investigate how cryptographic algorithm implementation diversity affects classification accuracy. In these works, researchers trained MLP or CNN in order to study their performance on several chips of the same device, and/or different keys, or different modes of operation. To overcome the problem of overfitting to one specific device chip, the authors [3] trained on one device, validated on a second device, and attacked on the third device. A different approach was discussed by Das *et al.* [8], where the authors trained on multiple chips of the same device to avoid influences from cross-devices dissimilarities. Note that all these papers investigated simple devices running an 8-bit microcontroller. In this paper, we consider another variability source also on more complex devices. We investigate several devices, probe positions and types, and we derive a workaround to diversity by fine-tuning a pretrained model (which is known as "transfer learning" in the machine learning community). Transfer learning has been shown to be successful in a simulated environment by Thapar *et al.* [24], where results show that transfer learning may help to lower the requirement on the number of traces in the learning phase.

3 On transferring side-channel model knowledge

3.1 Approach

Profiled side-channel analysis requires a sufficient amount of representative data in the learning phase, in particular, when using deep learning techniques. Fortunately, the community has put forward open-source datasets and models trained on these datasets recently. Naturally, the available data and pretrained models may not correspond exactly to the conditions an attacker faces on the target device. This problem of data discrepancy between training and testing datasets or data limitation of the training dataset is already known in other research fields and is tackled, for example, with the concept of transfer learning using neural networks. In particular, the idea of transfer learning is to transfer "knowledge" from a previous task or on related data to reduce the complexity of the learning on the actual suitable training dataset. Using this knowledge is simply achieved by using the weights of an already trained network as initialization instead of random weights according to some distributions. Seeing it from an implementation point of view, an attacker simply needs to download the pretrained model and load it into his framework before starting the training. Then, when reusing previously learnt models, the amount of data needed from the target device and the training time can be reduced. Or to put it the other way around, with less data it is possible to achieve higher effectiveness.

Transfer learning has shown to be efficient in several domains such as food classification [13], illustration classification [16] and for saliency [15].

In SCA, the first publicly known pretrained model is the ASCAD model [2], since then some other pretrained models have been published independently [27,26].

Remark 1 (Clone dataset). Conditions that are not identical between training and attacking may not only come from device variations, but also from the

measurement setup (as in our experiments later on). We therefore use the term "clone dataset" instead of the state-of-the-art term "clone device" if we want to refer to a dataset which is identical to the one in the attacking phase.

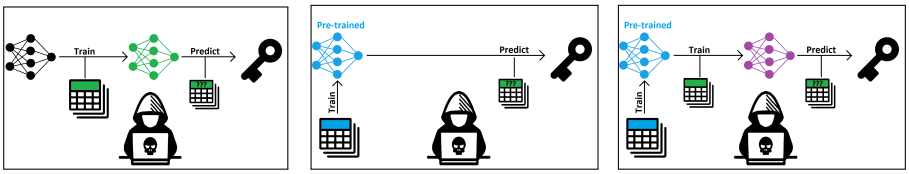
Depending on the available amount of data and the attack/evaluation context, multiple choices are available for training. We define three different training strategies that use throughout the paper:

Definition 1 (Training the model on a clone dataset). *A (small) clone profiling dataset is available. This clone dataset is obtained from a setup that is identical to the attack dataset one (illustrated in Figure 1a and labeled as S0).*

Definition 2 (Training the model on a different dataset). *In this scenario, possessing a clone dataset is not possible at all. One can only take advantage of a pretrained neural network that has been trained on a related (but not identical) setup (illustrated in Figure 1b and labeled as S1).*

Definition 3 (Fine-tuning the pretrained model with a (small) clone dataset). *A (small) clone profiling dataset is available. This dataset is used to fine-tune a model previously trained on some related setup (illustrated in Figure 1c and labeled as S2).*

Remark 2. Training strategy in Definition 1 corresponds to the traditional profiled attacker that has been originally introduced in the context of template attacks. Definition 3 also corresponds to the traditional profiled view, however, in here we additionally consider pretrained models from the outside. Assuming only a limited clone dataset (without the knowledge of pretrained models) has been investigated in recent works on side-channel attacks using machine or deep learning. For instance, authors [4] suggest to use Data Augmentation by generating new traces by the addition of noise (here temporal noise). Data Augmentation is known as a relevant technique to deal with too small datasets, however adding data increases the resource complexity.



(a) Strategy S0 using a clone dataset; traditional profiled attacker

(b) Strategy S1 using only a pretrained model

(c) Strategy S2 using a pretrained model and a clone dataset to finetune

Fig. 1: Training strategies.

In this paper, we do not consider variations due to the manufacturing process of the chip, but differences arising from the measuring setup, side-channel

information source, or from different devices (while still being close enough). We investigate three main scenarios in our experiments.

1. The position and type of the EM probe differ between pretrained model and target dataset.
2. The source of side-channel information differs between pretrained model and target dataset. Here, we investigate the use of power consumption and EM.
3. The device differs between pretrained model and target dataset. In this work, we consider the STM32Fx family as explained in more details later on.

3.2 CNN architecture

Thorough study of CNN was introduced by Benadjila *et al.* [2] and is commonly called ASCAD network. Its architecture was chosen through exhaustive evaluation of many design principles and parameters. The best performing network (in their selection) is relying on the architecture of VGG-16 [23] with five blocks and 1 convolutional layer per block, a number of filters equal to (64, 128, 256, 512, 512) with kernel size 11 ("same" padding), ReLU activation functions and an average pooling layer for each block. The CNN has two final dense layers of 4 096 units. The network is illustrated in Figure 2. The weights of the network are initialized with the "Glorot uniform" initializer. To conform to Benadjila et al.'s use-case [2], we have selected time frames for each experiment to reduce the input trace to 700 points based on the highest value of SNR.

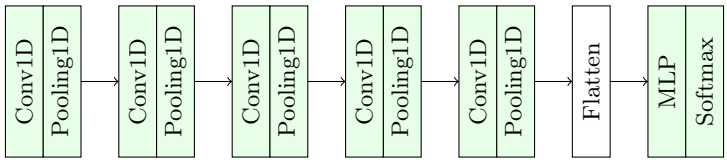
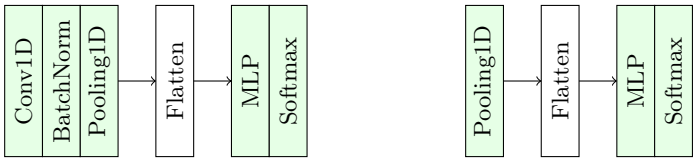


Fig. 2: ASCAD network

A fine-tuned and more efficient version of ASCAD was introduced by Zaid *et al.* [27] where the authors show that their network outperforms the ASCAD network while significantly reducing the network complexity. We label the network as ASCAD++. It is composed of one convolutional layer with a number of filters equal to 4 with kernel size 1 (same padding), one batch normalization layer, and one average pooling layer. The network has two final dense layers of 10 units. The network is illustrated in Figure 3a. As for the ASCAD network, we used the Glorot uniform initializer for weight initialisation and the size of the time frame for each of our experiments is equal to 700 points.

An enhancement of the ASCAD++ was given by Wouters *et al.* [26]. Authors point out that the convolutional layer of ASCAD++ network acts as a normalization and that it is useless on normalized inputs. Hence, this network does not contain any convolutional layers. It is only composed of one average pooling



(a) ASCAD++ network (b) NoConv1 network

Fig. 3: Enhancements of the ASCAD network

layer, and has two final dense layers of 10 units. The network is illustrated in Figure 3b. As ASCAD++ and ASCAD network, we use the network with the Glorot uniform initializer and the size of the time frames for our experiments is equal to 700 points. In the following section, this network is called NoConv1.

3.3 CNN training

Naturally, for S0 and S1, a large enough amount of data is needed to train the neural network, because none of the parameters of both convolutional and dense layers are fitted to the classification problem faced in side-channel analysis when starting from a random initial state.

However, for S2, different strategies using transfer learning are available. For example, an attacker could have only trained dense layers (thus freezing convolutional layers) or could have reset some specific layers while keeping previous parameters for other layers. Those strategies may permit to decrease the amount of data needed to train the neural network. The parameters kept from the first training may be better suited than a random initialization, thus getting the starting point closer to a minimum in the search space. In addition, some strategies where layers are frozen reduce the number of parameters to update in the neural network, which may increase the training speed. We apply two strategies on initial experiments: re-training the complete network during the second step and freezing the convolutional layers[4] (assuming that feature extraction is similar from one context to another but only decision changes). Since both gave similar results on our first runs, we decided to focus on retraining the full network, since then we do not make any hypothesis (that could end up to be false in some cases). Investigating deeply different techniques is clearly an interesting extension of this work.

We used a similar amount of data across experiments. The choices have been made i) to have most of the direct approaches leading to working attacks ii) to show the data requirement drops when performing transfer learning, and iii) to take into account that pretrained models are open-sourced from another party that may have higher available resources than the attacker itself. The number of traces used are hence:

[4] if existent

- 100 000 traces to pretrain a network on a dataset different than the clone dataset,
- 12 500 traces for direct attacks, or to fine-tune a network.

In our experiments, we use a batch size of 128, and the number of epochs is set to 100, for which we observed a convergence of the ASCAD network[5]. For the ASCAD network, the optimizer is the *RMSprop* optimizer with a learning rate equal to 1e-5 as introduced by Benadjila *et al.* [2]. For the ASCAD++ and NoConv1 networks, we used *Adam* optimizer with a learning rate equal to 1e-5. We do not apply an early stopping criterion to stop the training of the neural networks if it reaches a minimal loss error. We simply save the best model according to the lowest (validation) loss error during the training.

3.4 Evaluation Metrics & Targeted Value

Experiments have been performed on first-order leakage from the output of the AES substitution box (SBox) [7]. In Section 5 and Section 6 we use a simple implementation of AES and target the Sbox output that is $y = \text{SBox}(t \oplus k)$ with t being a plaintext byte and k a key byte. In Section 4, we collect datasets with multiple probes for which the setup is similar to the one introduced by Benadjila *et al.* [2]. We then chose the highest first-order leakage source that is the masked output of the Sbox.2

To evaluate the amount of leakage, we use the signal-to-noise ratio (SNR). Let X denote the captured side-channel measurement, let Y be the label that is determined by the plaintext and the secret fixed key, then SNR gives the ratio between the deterministic data-dependent leakage and the remaining noise, i.e. $SNR = \frac{\text{Var}(\text{E}(X|Y))}{\text{E}(\text{Var}(X|Y))}$, where $\text{E}(\cdot)$ is the expectation and $\text{Var}(\cdot)$ the variance of a random variable.

To evaluate the ability to retrieve the key, we use the guessing entropy (GE), by computing the average rank of the secret key k^* within a vector of key guesses. In particular, the vector of key guesses $g_{i,1}, \ldots, g_{i,|K|}$ for the ith measurement is calculated by mapping each key guess k to a label j with probability $\hat{p}_{i,j}$ and applying the maximum-likelihood principle over 1 to m measurements. The rank is then the position of the secret key k^* in the sorted vector of key guesses, where the sorting is applied to the probabilities in descending order. In other words, the guessing entropy gives the expected number of key guesses an attacker needs to perform before he reveals the secret key.

Remark 3. In our experiments, we observed that in some cases, GE inversely converges, i.e., instead of going down towards zero, it increases towards 255. We (and others) have observed this effect already on other datasets and models, yet this observation has not been published or explained. In our experiments, GE is "inverted" as a straightforward solution when seeing this effect.

[5] NoConv1 and ASCAD++ may require a larger number of epochs before convergence and thus the guessing entropy might be still improvable using more epochs. An extended evaluation on the convergence and the corresponding guessing entropies will be provided on eprint.

4 Transferring between EM probe position and type

In this section, we investigate transferring knowledge from a network trained on a dataset obtained using a different probe type or position. The motivation for this scenario is twofold.

First, when using EM as a side-channel source, the type of probe as well as its exact position, i.e., location and angle, play a crucial role (see for example [1]). In some situations, an attacker may not be able to precisely replicate the measurement setup from training on the attacked device.

Second, from an evaluation lab point of view, this scenario is related to the problematical duration of a fine-grained cartography. Leveraging transfer learning between probe positions, it is then possible to decrease the acquisition time since fewer traces are needed to train the network at each new position. The methodology would be to acquire a lot of traces on a seemingly relevant position to train a network, then to measure a few traces on each other position and adapt the network using transfer learning.

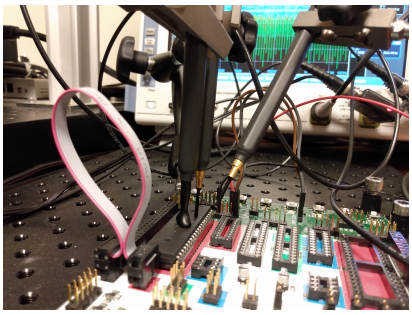


Fig. 4: Multi-probe experiment setup

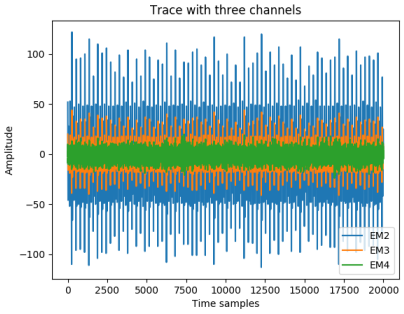


Fig. 5: Measurement trace from each of the three channels

For our experiments, we use a similar measurement setup and device as introduced by Benadjila *et al.* [2], namely a raw AtMega8515 microcontroller on the AVR STK500 platform[6]. We used the same AES-128 encryption than the one from ASCAD, which is protected using a masking countermeasure. The compiler optimization flag was set to `-O0` but we did not embed the SOSSE operating system. The chip frequency was set to 3.686MHz. Recall that for consistency between experiments, we targeted the masked output of the Sbox while knowing the output mask to target a significant first-order leakage. The measurements are obtained using different Langer near-field EM probes (two RF-B 0,3-3 and one RF-K 7-4) connected to 30dB amplifiers while the overall is having a bandwidth maximum frequency of 3GHz. The signal was then digitized by an RTO2014

[6] For ASCAD a smart-card embedding this micro-controller has been used.

oscilloscope from Rohde & Schwarz having a bandwidth of 1GHz (thus being the limiting link).

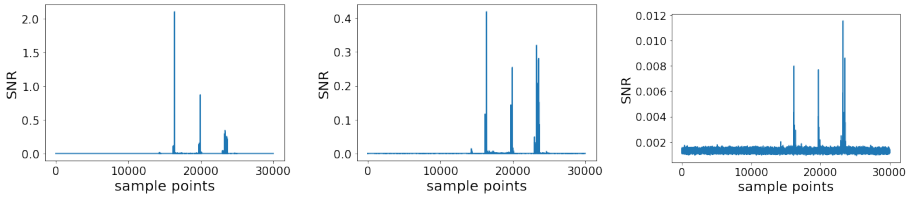


Fig. 6: SNR evaluation for each channel (from left to right: EM2, EM3, EM4)

We focused on the first AES round with a sampling frequency of 1Gs per second and obtained traces containing 20K samples. We observed that the leakages we obtained have a different location than the one reported by Benadjila *et al.* [2], however, we obtained similar leakages for the most informative part of the signal.

We show for example a measurement trace for each of the three channels in Figure 5. The probes on channels 2 and 3 (EM2, EM3) are placed to capture data-dependent leakage signals, whereas channel 4 (EM4) is capturing mostly noise[7]. Moreover, EM2 and EM3/EM4 are different types of probes, which explains the different amplitude as well. This is confirmed in Figure 6 showing the SNRs for EM2, EM3, EM4. One can see that EM2 provides higher SNR levels then EM3 and EM4, however, the leakage positions in time are consistent. Note that, even though EM4 is very noisy, one can still observe minor leakages.

4.1 Experimental results

Figure 7 shows the GE when targeting EM2, which is the channel with the highest SNR value. On the left side, we see that the pretrained model of EM2 (even if limited in traces) is converging quickly towards zero, with ASCAD++ and NoConv1 being slightly more effective. Pretrained models on other EM channels also succeed (even for EM4 which contains a high amount of noise), again on these ASCAD++ is the most effective, closely followed by NoConv1. On the other hand, using transfer learning is not improving the GE in most cases.

Next, Figure 8 shows the GE when targeting EM3, which has a medium SNR and is using the same probe type as EM4. Interestingly, a pretrained model on EM4 is slightly more effective than using a model trained on EM3 when directly used for attacking EM3. Again, ASCAD++ and NoCOnv1 are performing better than ASCAD. When using transfer learning, the results are slightly improved for the pretrained model of EM4, and improved for ASCAD using EM2.

[7] Channel 1 was used for triggering.

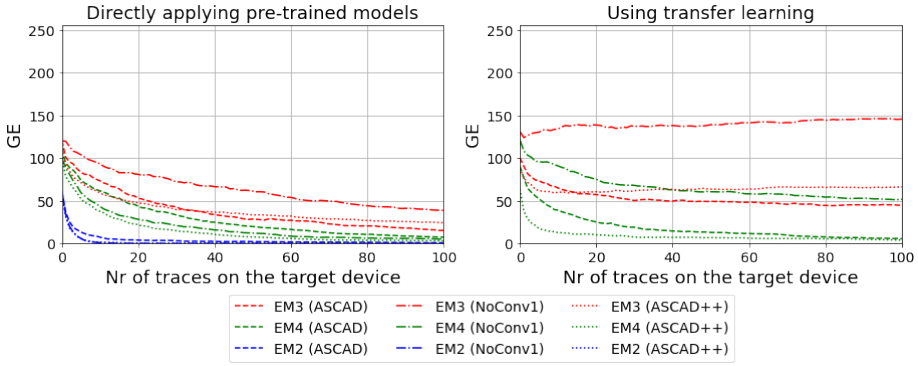


Fig. 7: Guessing entropy when targeting EM2

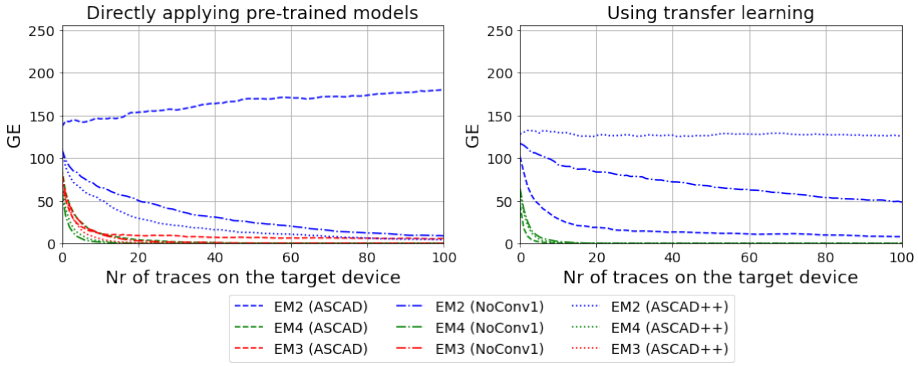


Fig. 8: Guessing entropy when targeting EM3

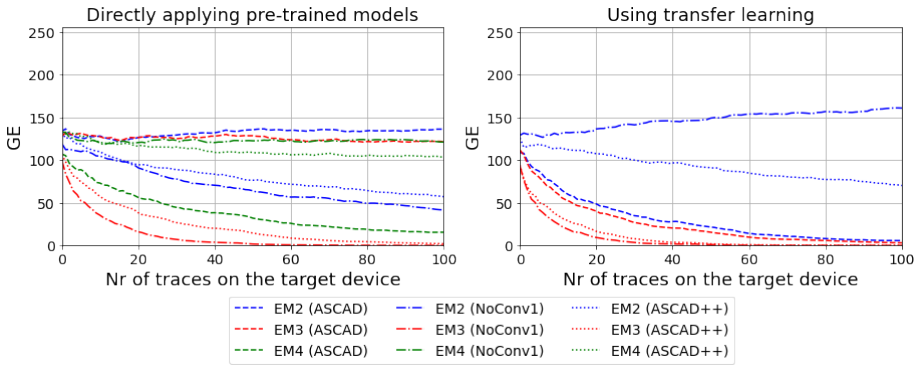


Fig. 9: Guessing entropy when targeting EM4

In Figure 9 we show the GE targeting EM4 that has the lowest SNR from all three EM positions. Directly applying a pretrained model from EM3 using

NoConv1 works sufficiently well, followed by ASCAD++ network from EM3, and a pretrained model on EM4 using ASCAD. Interestingly, using directly the pretrained model EM2, which is the channel containing the highest amount of information, does not perform better than less informative channels. Using transfer learning improves the result for EM3+NoConv1, EM3+ASCAD++, EM3+ASCAD, and EM2+ASCAD.

Summary When targeting noisy channels, our results show that the performance increases when using pretrained models on another channel with more available data, compared to training a model that is identical to the one of the target dataset (clone dataset) with less amount of data. In addition, on this dataset which is related to the original ASCAD dataset, NoConv1 and ASCAD++ (that have been fine-tuned on the ASCAD dataset) perform (slightly) better than the ASCAD network. Even though these two architectures are very specific and restricted, they still improve in some scenarios using transfer learning.

5 Transferring between Power and EM

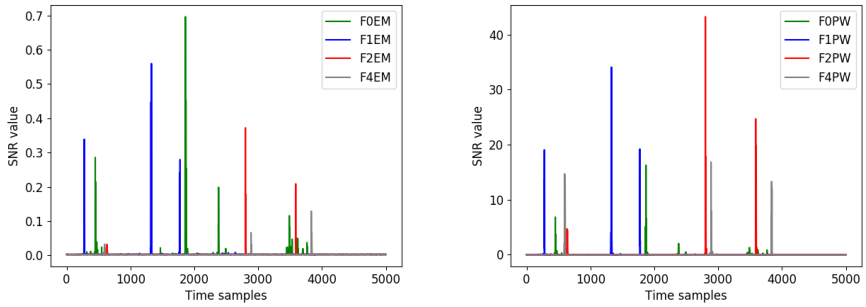
In this scenario, we investigate the transfer between side-channels. The main motivation for this experiment comes from the duality between power and EM side-channels. Usually, measuring power consumption is achieved by adding a resistor to a power line. This implies that the obtained values usually correspond to all (or at least a big part of) the chip (including highly consuming but unrelated features). On the contrary, EM may allow a more precise selection of the leakage, but this induces a risk of not capturing all relevant signals or introducing noise.

In some contexts, the attacker cannot add a resistor to the PCB of the target and thus has to use an EM probe for attacking. On the contrary, he may buy a clone device and build a dedicated card enabling power consumption measurements. He may thus train on power beforehand to build a pretrained model which makes use of all possible leakages, and then attack using EM where only a part of the signal are available (depending on the probe position for instance).

Again, from the evaluation lab point of view, this scenario is linked to the cartography problem. EM source is more localized and may lead to better results than power, but it needs a fine-grained cartography of the chip. First, training a network with power speeds up the cartography as mentioned in the previous experiment. Using power for a first training prevents from making a bad position choice for the first acquisition run (with no signal or a too specialized one).

We use the chipwhisperer lite capture board combined with the CW308 UFO board on STM32Fx target devices. Like in the previous setup, we measure the beginning of an AES-128 encryption, where we used the `TINYAES` implementation integrated in the chipwhisperer software. The chip frequency was set to 7.37MHz and the measurements are sampled at 4×7.37 Ms/s. Power consumption is collected through the measurement shunt on the CW308 UFO board. To capture

EM signals, we used a Langer near-field EM probe (RF-U 5-2) connected to a 20dB amplifier.



(a) Electromagnetic emanation (b) Power consumption

Fig. 10: SNR evaluation for each targeted device

Figure 10a and Figure 10b show the SNRs obtained with EM emissions and power consumption for each device. Depending on the device, we have different SNR levels for power and EM emissions. However, the shapes corresponding to power and EM are close to each other. The SNR values for power are higher than for EM. Seemingly, the EM emission measurements contain more noise than the ones from power. We expect that transfer learning can use the knowledge given by power to improve the models to target EM.

5.1 Experimental results

We now compare the effectiveness of applying pretrained models against using transfer learning when targeting EM measurements and having available pretrained models on power consumption. We consider the four previously introduced devices (namely, F0, F1, F2, and F4).

In Figure 11 we plot the guessing entropy obtained when targeting device F0 with EM (F0em). On the left, one can observe that for all three neural networks the attack does not converge within 1000 traces. When using transfer learning (right side), i.e., using the weights obtained when training on power consumption as initialization, all networks show a decreasing GE, whereas ASCAD is the most effective network nearly reaching a GE of 0.

Figure 12 presents the results when attacking F1 with electromagnetic emission. Similarly as before, when applying pretrained networks directly (even when using EM measurements), none of the networks seem to have a decreasing GE. However, when updating the pretrained model with transfer learning, one can observe that ASCAD is reaching a GE close to 0.

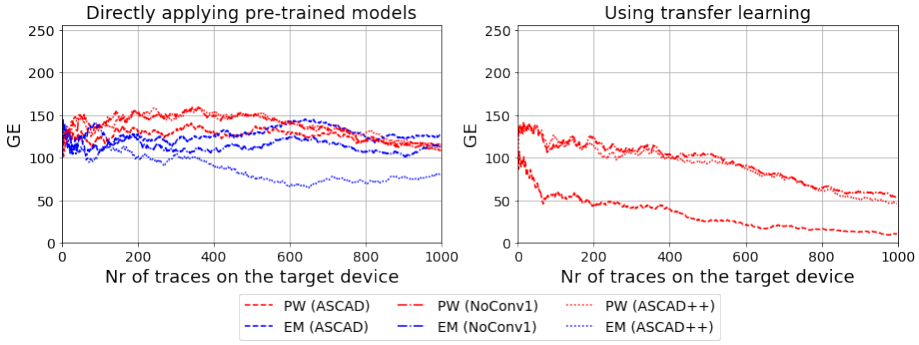


Fig. 11: Guessing entropy when targeting F0em

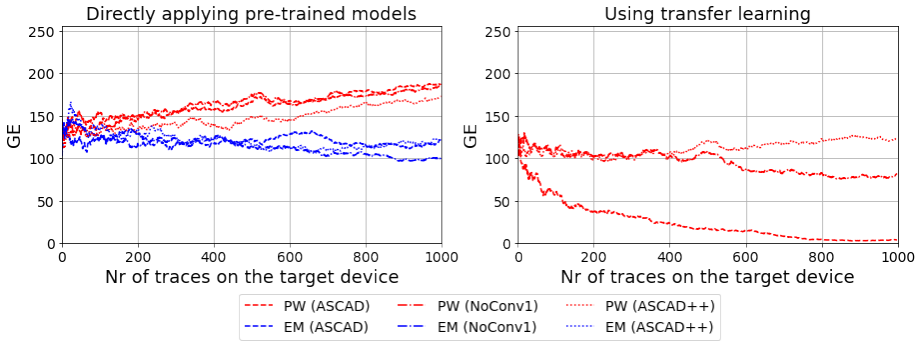


Fig. 12: Guessing entropy when targeting F1em

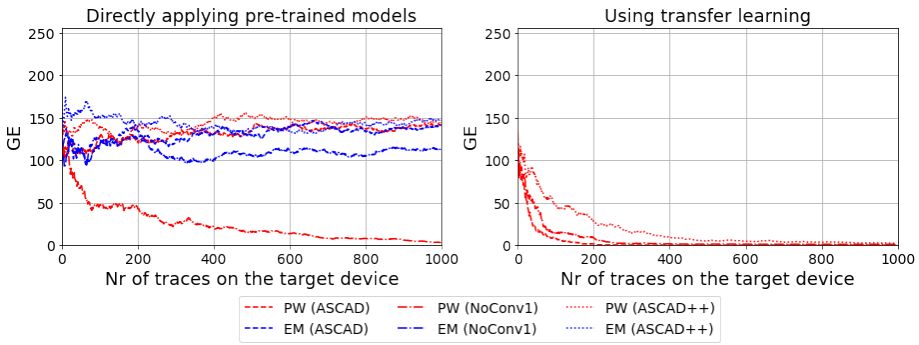


Fig. 13: Guessing entropy when targeting F2em

Targeting F2em, Figure 13 shows that using a pretrained model on power and the NoConv1 network results in converging GE towards 0, whereas the other two networks trained on power as well as all networks on EM fail to decrease

within 1000 traces. Interestingly, we see that in this scenario transfer learning is very effective on all three networks, while ASCAD is the most effective one, reaching a GE of 0 with less than 100 traces.

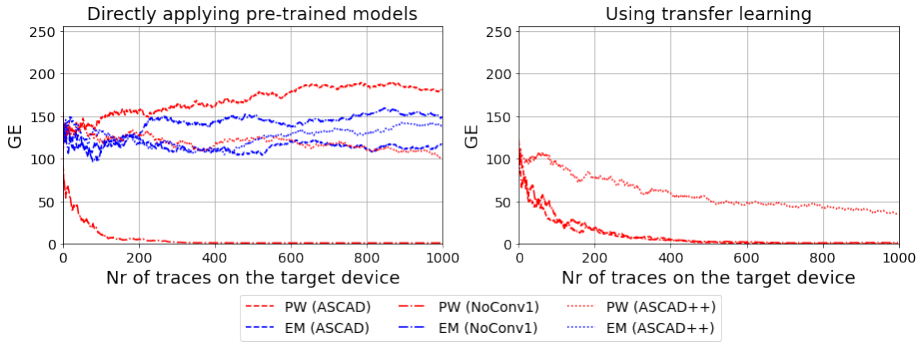


Fig. 14: Guessing entropy when targeting F4

On device F4, directly applying a pretrained model with the NoConv1 network results in an effective attack converging towards GE 0 rather quickly, where all other pretrained models and networks do not converge. When using transfer learning, one can see that all 3 neural networks pretrained on power consumption are decreasing in GE with NoConv1, while ASCAD is the most effective one.

Summary In all of the four scenarios, the effectiveness of using pretrained models with transfer learning instead of training on the dataset corresponding to the target dataset can be seen. Moreover, in most of the cases, using transfer learning instead of applying directly the pretrained network is showing superior results. When using transfer learning, the ASCAD network is the most effective one, whereas when directly applying pretrained networks, only NoConv1 converges in the scenario of F2em.

6 Transferring between different devices

In this scenario, we investigate the possible use of pretrained models stemming from different devices. The motivation here is that an attacker may have access to public datasets and/or publicly trained networks corresponding to some other chip. Then, based on this, he may try to attack another device that is different but still close (same architecture for instance). From an evaluation lab perspective, it boils down to leveraging previous analyses with similar enough chips to speed up the current one.

We use the same measurement setup as in the previous section. In Figure 10b, we show the SNR values obtained from measuring the power consumption of the

devices F0, F1, F2, and F4. We see that the highest SNR levels are obtained by F2, followed by F1, F4, and finally F0.

6.1 Experimental results

In Figure 15 we show the guessing entropy when targeting F0 and pretrained models are built from F0, F1, F2, F4. One can see that when using directly pretrained models, the model on F0 is the most effective, that not all pretrained models converge towards 0, and ASCAD is the best among the three neural networks. When compared to transfer learning, all pretrained models are improved, where using ASCAD with a pretrained model on F2 and F4 is the most effective one. Note that F0 and F2/F4 have different ARM Cortex M versions and the highest values of SNR are at different time locations (see Figure 10), still using knowledge from devices with different architectures brings improvements.

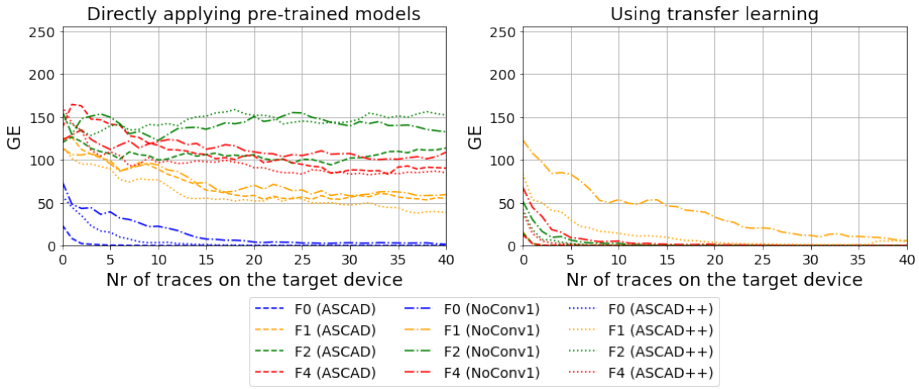


Fig. 15: Guessing entropy when targeting F0

Figure 16 shows a similar trend. When directly applying pretrained models, the most effective pretrained model is the one trained on the target device, where the ones on F2 may also converge given a higher number of attacking traces. Using the pretrained models as weight initialisation improves all networks. Again, the ASCAD network is the most effective one, but the difference between networks is rather marginal.

Figure 17 shows the results when targeting F2, again using a pretrained network on the target device is the best performing one, while transfer learning improves the results for all pretrained models and architectures. Interestingly, in this scenario we see a clear advantage of using transfer learning against the network directly trained on the target device. This shows that initializing weights using a network trained on a similar problem is more suited than using the classical initialization (here default Glorot uniform initialization).

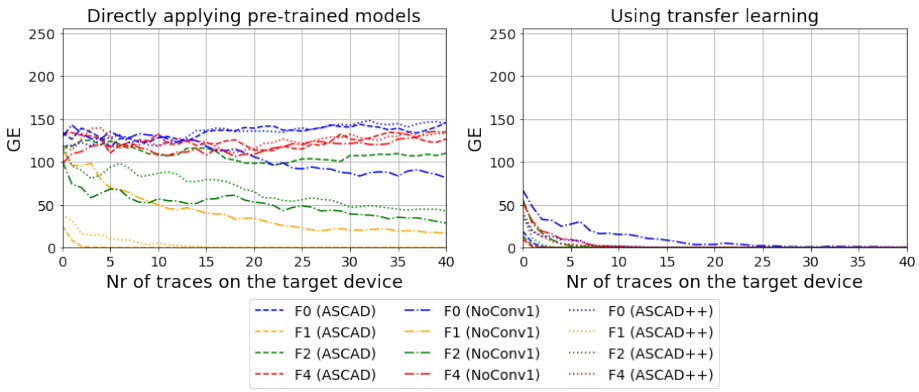


Fig. 16: Guessing entropy when targeting F1

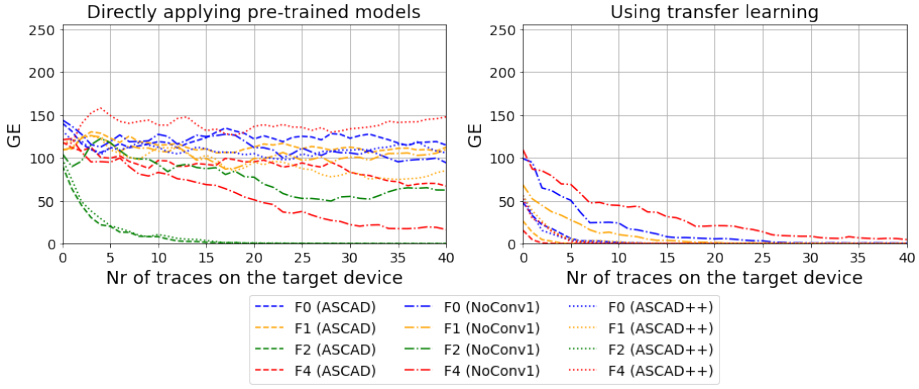


Fig. 17: Guessing entropy when targeting F2

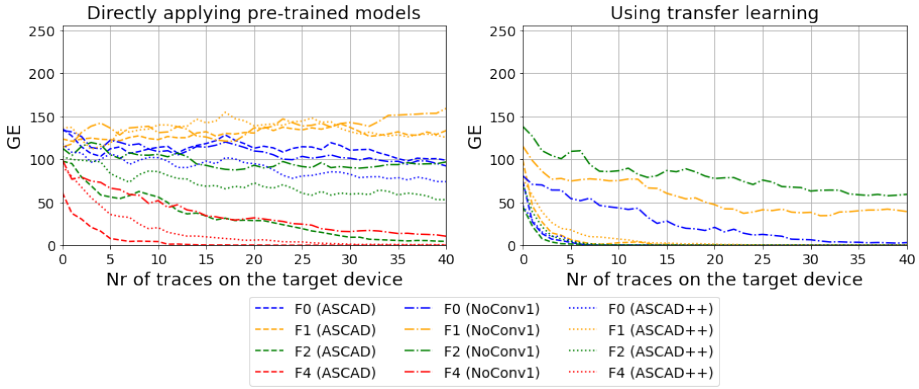


Fig. 18: Guessing entropy when targeting F4

Figure 18 shows the GE of target F4. Using a pretrained model on F2 together with the ASCAD network with transfer learning is the most effective attack, followed by a pretrained network trained on F0 and F1 together with transfer learning. Interestingly, all three are performing better than a pretrained model on F4 directly.

Summary In this scenario, we see a disadvantage of the NoConv1 network for both cases, using directly pretrained models and updating the pretrained model with transfer learning. In all scenarios, the ASCAD network is the most effective one. This would suggest that the ASCAD network is more suited for STM32 targets when the amount of available data is low. Another interesting observation is that, in all scenarios, using the pretrained model on F4 as weight initialization is the best performing one, even though it is not the model with the highest SNR values.

7 Conclusion

In this work, we considered three training strategies:

- S0: training a neural network with a (limited) clone dataset (ie. a dataset that is identical to the one from the target device);
- S1: using an available pretrained model trained on a (large) dataset with different conditions than the target dataset;
- S2: using an available pretrained model as weight initialization and then fine-tuning it using a (smaller) clone dataset.

We compared these training strategies using three state-of-the-art neural network models (ASCAD, ASCAD++, NoConv1) on three different scenarios of data discrepancy: EM probe type and/or positions, side-channel sources (EM vs power) and target devices. Our results show that directly applying pretrained models (S1) can lead to successful attacks in a few investigated scenarios. Much better and stable results can be achieved through transfer learning in all scenarios when using the weights of a pretrained model as initialization and further fine-tune them on a (limited) clone dataset (S2). The improvement of transfer learning can be seen on all three networks, even for the specific and limited network of ASCAD++ and NoConv1. Interestingly, these two networks show only the best effectiveness on a dataset that is close to the ASCAD dataset for which they have been fine-tuned on. On our other datasets, the ASCAD network, which is more general, achieves the most effective results.

In general, we could observe that in the vast majority of cases, using a pretrained model as weight initialization gives better results as the standard uniform Glorot initalizer.

The interest of these results is twofold.

1. First, a profiled attacker scenario should not be rejected due to a too small number of available labeled traces.

2. Second, evaluation process (and particularly trace acquisition) can be significantly speed up and improved by using transfer learning with an already trained network (from a previous evaluation or another position in the case of a cartography).

We chose to train the whole network again to avoid misinterpretations due to wrong choices of the frozen layers. However, to improve even more the efficiency of the training, investigating different fine-tuning strategies may be of interest. This is part of the natural extension of the presented work. Another extension which is less straightforward (but may be linked to layer freezing) would be to leverage transfer learning to help a network to converge on a protected (masked) implementation. A network has to understand the signal structure, then extract the information, and then process it. The hope is that the signal structure knowledge could be obtained from an easier setup (unprotected) to focus the second learning on the extraction of relevant information and its smart combination.

References

1. Andrikos, C., Batina, L., Chmielewski, L., Lerman, L., Mavroudis, V., Papagiannopoulos, K., Perin, G., Rassias, G., Sonnino, A.: Location, location, location: Revisiting modeling and exploitation for location-based side channel leakages. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 285–314. Springer International Publishing, Cham (2019)
2. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. J. Cryptogr. Eng. **10**(2), 163–188 (2020)
3. Bhasin, S., Chattopadhyay, A., Heuser, A., Jap, D., Picek, S., Shrivastwa, R.R.: Mind the portability: A warriors guide through realistic profiled side-channel analysis. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020. The Internet Society (2020)
4. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without preprocessing. In: CHES 2017. pp. 45–68 (2017)
5. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems. pp. 13–28. CHES '02, Springer-Verlag, London, UK, UK (2003)
6. Choudary, O., Kuhn, M.G.: Template attacks on different devices. In: COSADE 2014. pp. 179–198 (2014)
7. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography). Springer, 1 edn. (2002)
8. Das, D., Golder, A., Danial, J., Ghosh, S., Raychowdhury, A., Sen, S.: X-deepsca: Cross-device deep learning side channel attack. In: DAC 2019. pp. 134:1–134:6 (2019)
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press (2016)
10. Heuser, A., Rioul, O., Guilley, S.: Good Is Not Good Enough - Deriving Optimal Distinguishers from Communication Theory. In: CHES 2014. pp. 55–74 (2014)
11. Heuser, A., Zohner, M.: Intelligent Machine Homicide - Breaking Cryptographic Devices Using Support Vector Machines. In: Schindler, W., Huss, S.A. (eds.) COSADE. LNCS, vol. 7275, pp. 249–264. Springer (2012)

12. Hospodar, G., Gierlichs, B., Mulder, E.D., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. J. Cryptographic Engineering **1**(4), 293–302 (2011)
13. Islam, K.T., Wijewickrema, S., Pervez, M., O'Leary, S.: An exploration of deep transfer learning for food image classification. In: 2018 Digital Image Computing: Techniques and Applications (DICTA). pp. 1–5 (Dec 2018)
14. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: CRYPTO '99. pp. 388–397 (1999)
15. Kümmerer, M., Wallis, T.S.A., Bethge, M.: Deepgaze II: reading fixations from deep features trained on object recognition. CoRR **abs/1610.01563** (2016)
16. Lagunas, M., Garces, E.: Transfer learning for illustration classification. CoRR **abs/1806.02682** (2018)
17. Lerman, L., Bontempi, G., Markowitch, O.: Side Channel Attack: an Approach Based on Machine Learning. In: COSADE 2011. pp. 29–41 (2011)
18. Li, H., Krcek, M., Perin, G.: A comparison of weight initializers in deep learning-based side-channel analysis. In: Conti, M., Zhou, J., Ahmed, C.M., Au, M.H., Batina, L., Li, Z., Lin, J., Losiouk, E., Luo, B., Majumdar, S., Meng, W., Ochoa, M., Picek, S., Portokalidis, G., Wang, C., Zhang, K. (eds.) Applied Cryptography and Network Security Workshops - ACNS 2020 Satellite Workshops, Rome, Italy, October 19-22, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12418, pp. 126–143. Springer (2020)
19. Maghrebi, H.: Deep learning based side channel attacks in practice. IACR Cryptology ePrint Archive **2019**, 578 (2019)
20. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: SPACE 2016s. pp. 3–26 (2016)
21. NIPS*95 Post-Conference Workshop: Learning to learn: Knowledge consolidation and transfer in inductive systems. `http://plato.acadiau.ca/courses/comp/dsilver/NIPS95_LTL/transfer.workshop.1995.html` (1995)
22. Prouff, E., Rivain, M., Bevan, R.: Statistical analysis of second order differential power analysis. IEEE Trans. Computers **58**(6), 799–811 (2009)
23. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)
24. Thapar, D., Alam, M., Mukhopadhyay, D.: Deep learning assisted cross-family profiled side-channel attacks using transfer learning. In: 2021 22nd International Symposium on Quality Electronic Design (ISQED). pp. 178–185 (2021)
25. Wang, H., Brisfors, M., Forsmark, S., Dubrova, E.: How diversity affects deep-learning side-channel attacks. Cryptology ePrint Archive, Report 2019/664 (2019)
26. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a methodology for efficient cnn architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(3), 147–168 (Jun 2020)
27. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient cnn architectures in profiling attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 1–36 (2020)