Improved Deep-Learning Side-Channel Attacks using Normalization Layers

Damien Robissout, Gabriel Zaid, Lilian Bossuet, Amaury Habrard

damien.robissout@univ-st-etienne.fr

Laboratoire Hubert Curien
Université Jean Monnet

16/04/2019
• Good performance of neural networks in side-channel analysis

• Improvement possible using batch normalization and regularization

• No deep learning metric usable to evaluate networks for SCA

• Proposition of a metric to tell how well a given architecture could perform
1. Batch Normalization

2. $\Delta_{train, val}$: an SCA metric to evaluate performances

3. Regularization

4. Conclusion
1 Batch Normalization

2 $\Delta_{train,val}$: an SCA metric to evaluate performances

3 Regularization

4 Conclusion
Batch Normalization

**Goal**
Standardize the data representation across all layers

**Consequence**
The network focuses on the relative differences of the values rather than on the numerical values

\[(\mu, \sigma^2)\] 

\[(0, 1)\]
Updated architecture: $CNN_{bn}$

Network architecture with Batch Normalization

- Input
- CONV1 $\text{Batch Normalization}$
- CONV2 $\text{Batch Normalization}$
- CONV3 $\text{Batch Normalization}$
- CONV4 $\text{Batch Normalization}$
- CONV5 $\text{Batch Normalization}$
- CONV1D $\text{Batch Normalization}$
- POOLING
- FC1
- FC2
- Predictions
Training on ASCAD desynchronized traces

- **DesyncN**: random shift between 0 and $N$ applied to the 700 points of the traces

![Graph showing average rank of the good key against number of traces for cnn_bn and cnn_best](image-url)
Training on ASCAD desynchronized traces

- **Desync\(N\):** random shift between 0 and \(N\) applied to the 700 points of the traces

![Graphs showing the average rank of the good key for Desync0 and Desync50](image-url)
Training on ASCAD desynchronized traces

- Desync\(N\): random shift between 0 and \(N\) applied to the 700 points of the traces

![Graphs showing the average rank of the good key vs. number of traces for Desync0, Desync50, and Desync100.](image)
Evaluate the performance of a network
Training Acc. vs. Validation Acc.

Goal

Evaluate the networks during training

![Graph showing training and validation accuracy over epochs for different models labeled as `CNN_best` with legend indicating lines for acc desync0, acc desync50, acc desync100, val_acc desync0, val_acc desync50, and val_acc desync100.]
Goal

Evaluate the networks during training

---

**CNN**

- **CNN**
  - **best**
  - **bn**

Robissout, D. (LabHC)
1. Batch Normalization

2. $\Delta_{\text{train,val}}$: an SCA metric to evaluate performances

3. Regularization

4. Conclusion
The overfitting phenomena

Good estimation

Overfitting
\( \Delta_{\text{train}, \text{val}} \): evaluation of the generalization capacity

**Goal**

Have a clear indication if the network is overfitting/underfitting and if the performance of the network can be improved.

**Notations**

- \( T_{\text{train}} = \text{Set of traces the network used to train} \)
- \( T_{\text{val}} = \text{Set of traces the network has never seen} \)
- \( N_{\text{train}}(model) := \min \{ n_{\text{train}} \mid \forall n \geq n_{\text{train}}, SR_{\text{train}}^{1}(model(n)) = 90\% \} \)
- \( N_{\text{val}}(model) := \min \{ n_{\text{val}} \mid \forall n \geq n_{\text{val}}, SR_{\text{val}}^{1}(model(n)) = 90\% \} \)

**Metric**

\[
\Delta_{\text{train}, \text{val}}(model) = \left| N_{\text{val}}(model) - N_{\text{train}}(model) \right|
\]
How to use the metric

If for several consecutive epochs $\Delta_{\text{train,val}}$ increases, stop the training.

Save the network regularly.

$T_{\text{train}}$

$T_{\text{val}}$
Representation of $\Delta_{train,att}$ for $CNN_{bn}$
1. Batch Normalization

2. $\Delta_{train,val}$: an SCA metric to evaluate performances

3. Regularization

4. Conclusion
Regularization

Goal
Reduce $\Delta_{\text{train,att}}$ even further using regularization

Means
- Dropout with parameter $\lambda_D$
- $L_2$-Norm regularization with parameter $\lambda_{L_2}$
Regularization

Goal
Reduce $\Delta_{train,att}$ even further using regularization

Means
- Dropout with parameter $\lambda_D$
- $L_2$-Norm regularization with parameter $\lambda_{L_2}$

<table>
<thead>
<tr>
<th></th>
<th>Test ($step = 0.1$)</th>
<th>Choice for desync100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\lambda_D$</td>
<td>$\lambda_{L_2}$</td>
</tr>
<tr>
<td>CONV1&amp;2</td>
<td>[0, ..., 0.3]</td>
<td>[0, ..., 0.3]</td>
</tr>
<tr>
<td>CONV3</td>
<td>[0, ..., 0.8]</td>
<td>[0, ..., 0.3]</td>
</tr>
<tr>
<td>CONV4</td>
<td>[0, ..., 0.8]</td>
<td>[0, ..., 0.3]</td>
</tr>
<tr>
<td>CONV5</td>
<td>[0, ..., 0.8]</td>
<td>[0, ..., 0.3]</td>
</tr>
<tr>
<td>FC1</td>
<td>[0, ..., 0.8]</td>
<td>[0, ..., 0.3]</td>
</tr>
<tr>
<td>FC2</td>
<td>[0, ..., 0.3]</td>
<td>[0, ..., 0.3]</td>
</tr>
</tbody>
</table>
Architecture with regularization: $CNN_{bn+reg}$

Input ➔ CONV1 ➔ CONV2

CONV3
$\lambda_{L2} = 0.2$
$\lambda_D = 0.5$

CONV4
$\lambda_{L2} = 0.3$
$\lambda_D = 0.6$

CONV5
$\lambda_{L2} = 0.3$
$\lambda_D = 0.7$

Normalized block
FC1
$\lambda_{L2} = 0.1$, 0.2 or 0.3

FC2 ➔ Predictions
Results without regularization: $CNN_{bn}$

Evolution of the average rank for training on desync100 and attack on desync100

$\Delta_{train, att} = 6237$
Results with regularization: $\text{CNN}_{bn+\text{reg}}$

Evolution of the average rank for training on desync100 and attack on desync100

$\Delta_{\text{train, att}} = 548$

Number of traces

Average rank of the key

$SR_{\text{train}} = 90\%$

$SR_{\text{att}} = 90\%$
Results with regularization: $CNN_{bn+reg}$

Evolution of the average rank for training on desync100 and attack on desync100

$\Delta_{train, att} = 548$
Attack on desync100 using $\lambda_{L_2} = 0.1$ for $CNN_{bn+reg}$

Evolution of the average rank for training on desync100 and attack on desync100

$\Delta_{train, att} = 548$
Attack on desync100 using $\lambda_{L_2} = 0.2$ for $\text{CNN}_{bn+\text{reg}}$

Evolution of the average rank for training on desync100 and attack on desync100

$\Delta_{\text{train, att}} = 470$
Attack on desync100 using $\lambda_{L_2} = 0.3$ for $CNN_{bn+reg}$

Evolution of the average rank for training on desync100 and attack on desync100

$\Delta_{train, att} = 319$
Evolution of $\Delta_{\text{train,att}}$ for different numbers of epochs

Best results on other desynchronizations

<table>
<thead>
<tr>
<th>Desync</th>
<th>$N_{\text{train}}$</th>
<th>$N_{\text{att}}$</th>
<th>$\Delta_{\text{train,att}}$</th>
<th>FC1: $\lambda_{L2}$</th>
<th>Nb epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desync0</td>
<td>104</td>
<td>272</td>
<td>168</td>
<td>0.1</td>
<td>125</td>
</tr>
<tr>
<td>Desync50</td>
<td>21</td>
<td>279</td>
<td>258</td>
<td>0.1</td>
<td>200</td>
</tr>
<tr>
<td>Desync100</td>
<td>76</td>
<td>395</td>
<td>319</td>
<td>0.3</td>
<td>175</td>
</tr>
</tbody>
</table>
1 Batch Normalization

2 $\Delta_{\text{train, val}}$: an SCA metric to evaluate performances

3 Regularization

4 Conclusion
Conclusion

- **New metric** to evaluate the possible improvement of an architecture

- **Normalization and regularization** improve CNN performance in SCA

- Given the amount of regularization needed to obtain those results, a better architecture probably exists

- Apply this technique to other networks
Improved Deep-Learning Side-Channel Attacks using Normalization Layers

Thank you for listening. Do you have questions?
Dropout example

Pooling example

Ref.: Max pooling in CNN.
Source: http://cs231n.github.io/convolutional-networks/