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

25/05/2019

Introduction

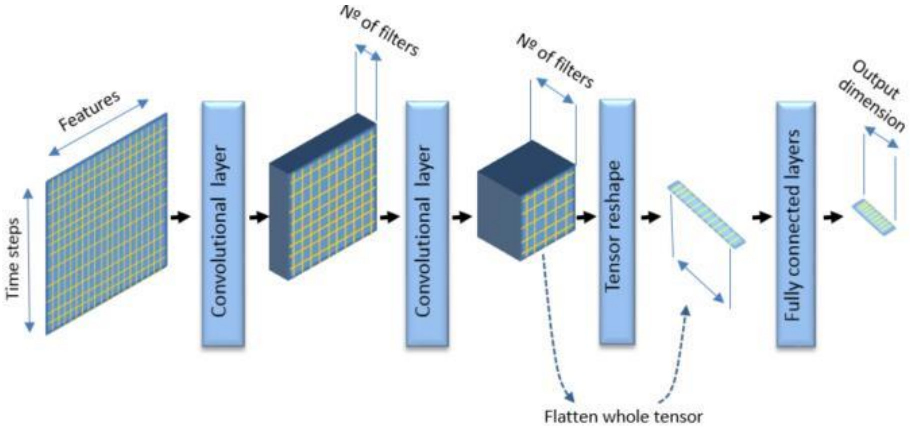
- **Good performance** of neural networks in side-channel analysis
- Improvement possible using **batch normalization** and **regularization**
- **No deep learning metric** usable to evaluate networks for SCA
- Proposition of a **metric** to tell how well a given architecture could perform

Content

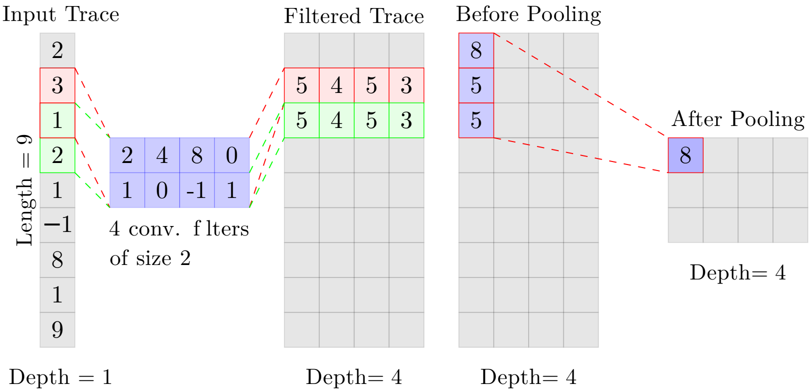
Content

Convolutional Neural Networks (CNNs)

Convolutional neural network architecture

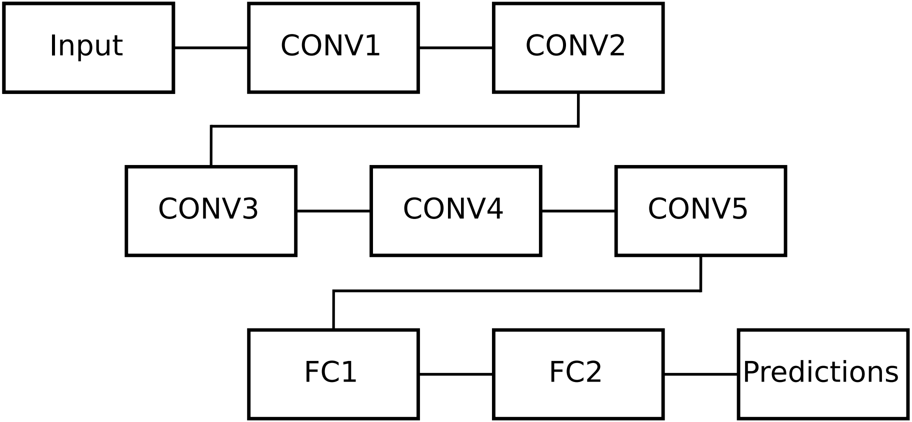


Convolution operation example



Base architecture: CNN_{best}

Network architecture with Batch Normalization



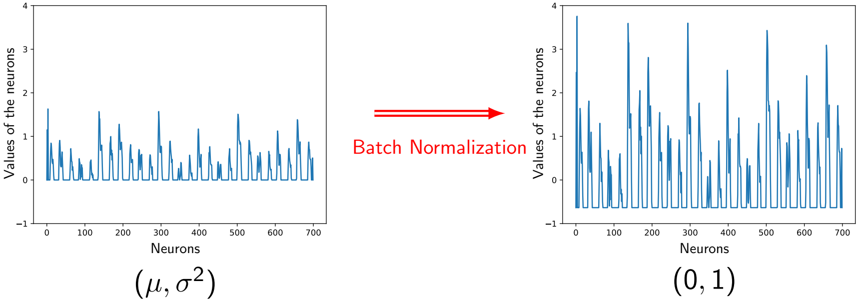
Batch Normalization

Goal

Standardize the data representation across all layers

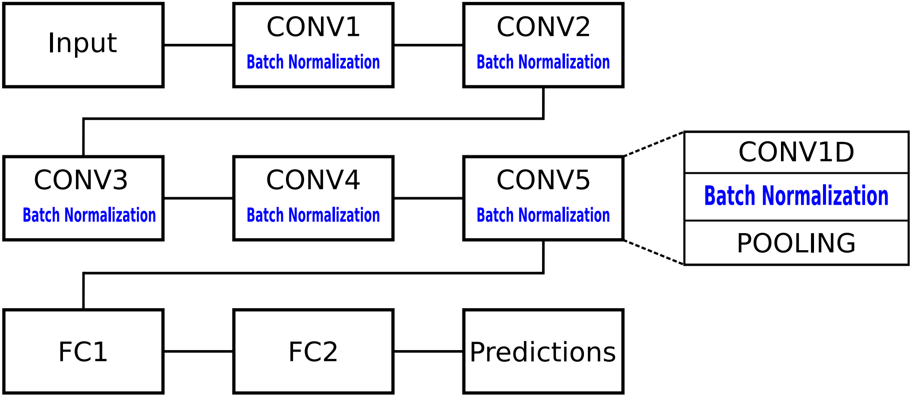
Consequence

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



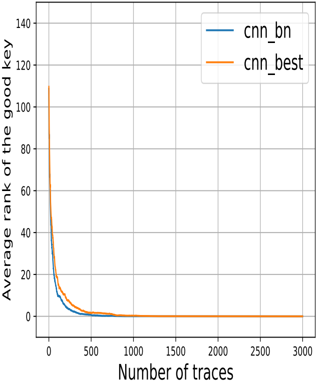
Updated architecture: CNN_{bn}

Network architecture with Batch Normalization



Training on ASCAD desynchronized traces

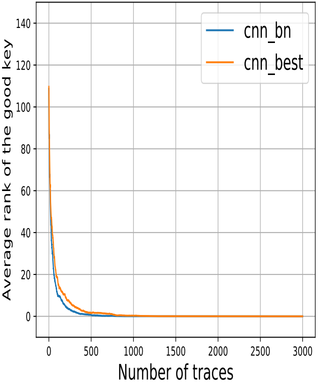
- Desync$N$: random shift between 0 and N applied to the 700 points of the traces



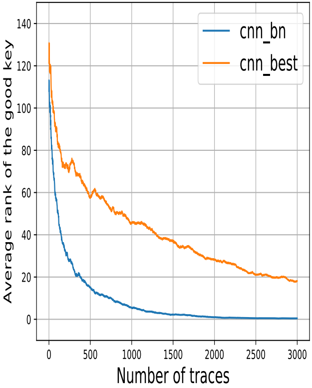
Desync0

Training on ASCAD desynchronized traces

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



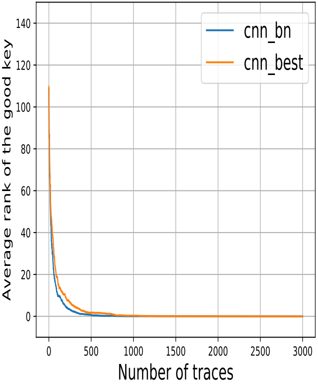
Desync0



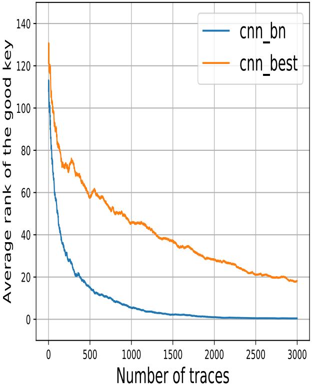
Desync50

Training on ASCAD desynchronized traces

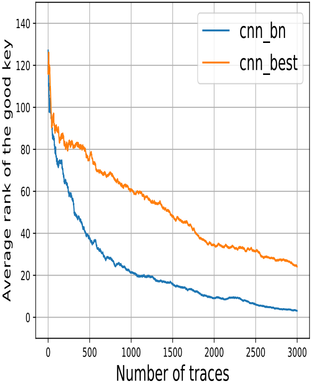
- Desync$N$: random shift between 0 and N applied to the 700 points of the traces



Desync0

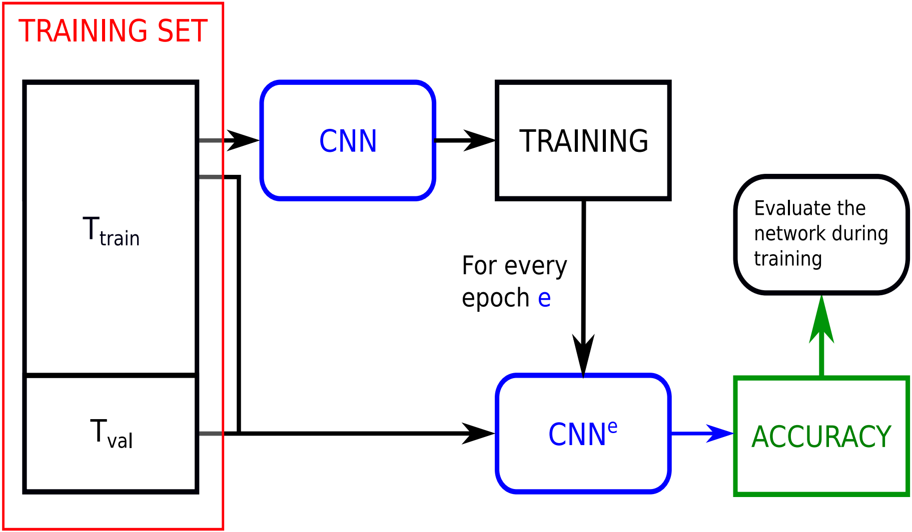


Desync50



Desync100

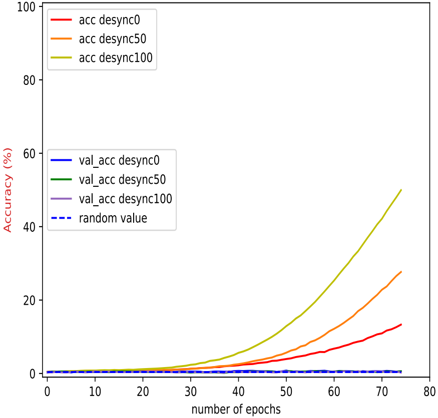
Evaluate the performance of a network



Training Acc. vs. Validation Acc.

Goal

Evaluate the networks during training

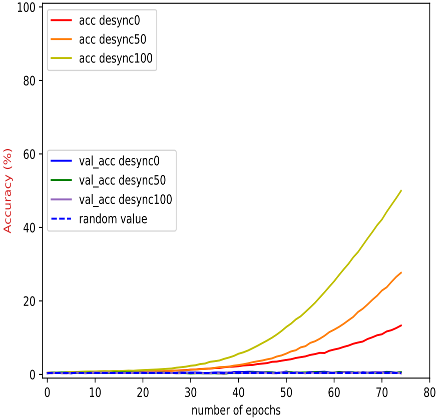


CNN_{best}

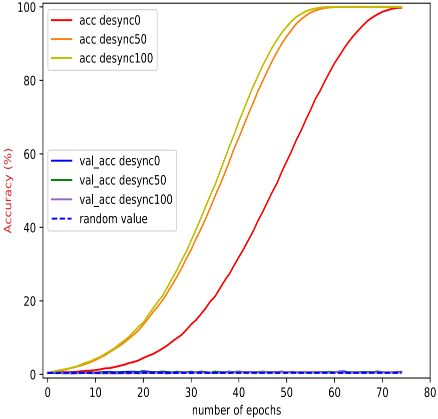
Training Acc. vs. Validation Acc.

Goal

Evaluate the networks during training



CNN_{best}

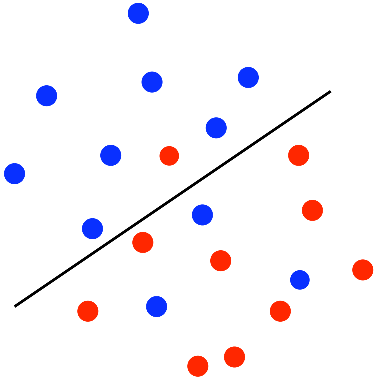


CNN_{bn}

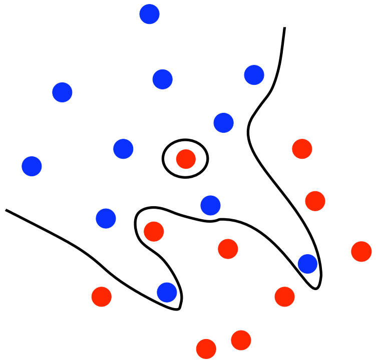
Content

1. CNNs and Batch Normalization
2. $\Delta_{train,val}$: an SCA metric to evaluate performances
3. Regularization
4. Conclusion

The overfitting phenomena



Good estimation



Overfitting

$\Delta_{train,val}$: evaluation of the generalization capacity

Goal

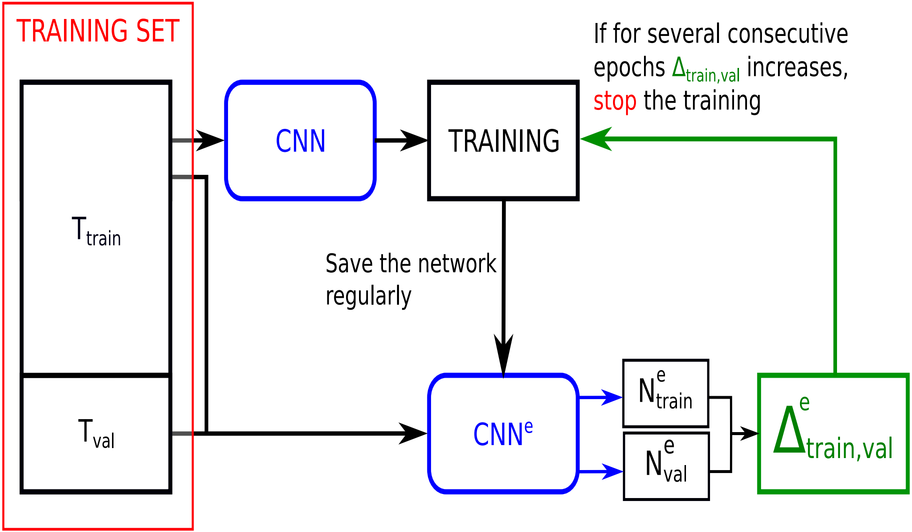
Have a clear indication if the network is overfitting/underfitting and if the performance of the network can be improved

Notations

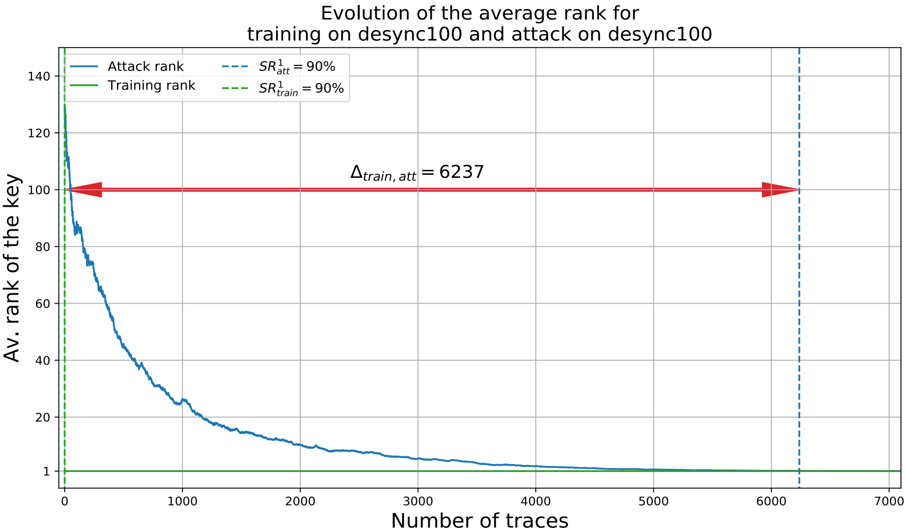
- T_{train} = Set of traces the network used to train
- T_{val} = Set of traces the network has never seen
- $N_{train}(model) := min\{n_{train} \mid \forall n \geq n_{train}, SR^1_{train}(model(n)) = 90\%\}$
- $N_{val}(model) := min\{n_{val} \mid \forall n \geq n_{val}, SR^1_{val}(model(n)) = 90\%\}$

Metric

$$\Delta_{train,val}(model) = \mid N_{val}(model) - N_{train}(model) \mid$$


TRAINING SET
T_{train}
T_{val}
CNN
TRAINING
If for several consecutive epochs $\Delta_{train,val}$ increases, stop the training
Save the network regularly
CNN^e
N^e_{train}
N^e_{val}
$\Delta^e_{train,val}$

Representation of $\Delta_{train,att}$ for CNN_{bn}



Content

Regularization

Goal

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

Means

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

Regularization

Goal

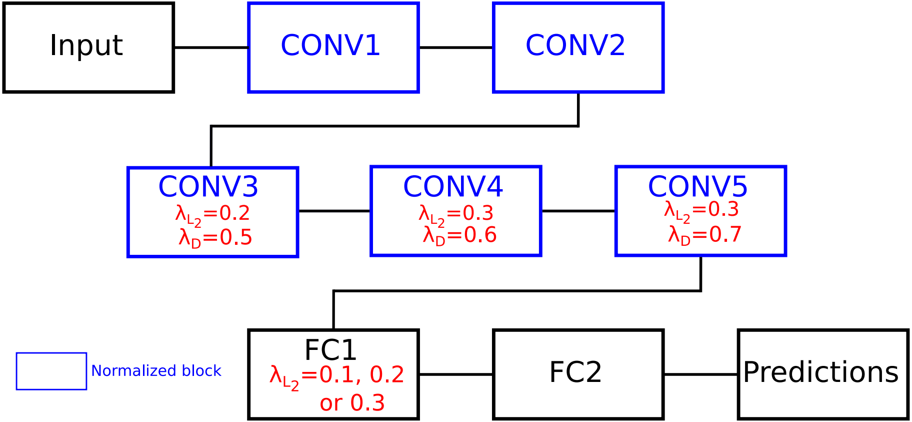
Reduce $\Delta_{train,att}$ even further using regularization

Means

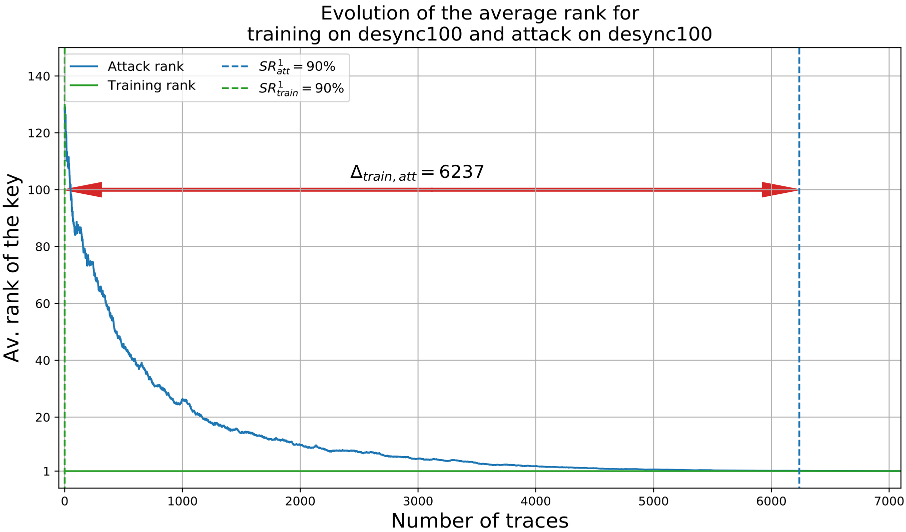
- Dropout with parameter $\boldsymbol{\lambda_D}$
- L_2-Norm regularization with parameter $\boldsymbol{\lambda_{L_2}}$

	Test ($step = 0.1$)		Choice for desync100	
	$\boldsymbol{\lambda_D}$	$\boldsymbol{\lambda_{L_2}}$	$\boldsymbol{\lambda_D}$	$\boldsymbol{\lambda_{L_2}}$
$CONV1\&2$	[0, ..., 0.3]	[0, ..., 0.3]	0	0
$CONV3$	[0, ..., 0.8]	[0, ..., 0.3]	0.5	0.2
$CONV4$	[0, ..., 0.8]	[0, ..., 0.3]	0.6	0.3
$CONV5$	[0, ..., 0.8]	[0, ..., 0.3]	0.7	0.3
$FC1$	[0, ..., 0.8]	[0, ..., 0.3]	0	0.3
$FC2$	[0, ..., 0.3]	[0, ..., 0.3]	0	0

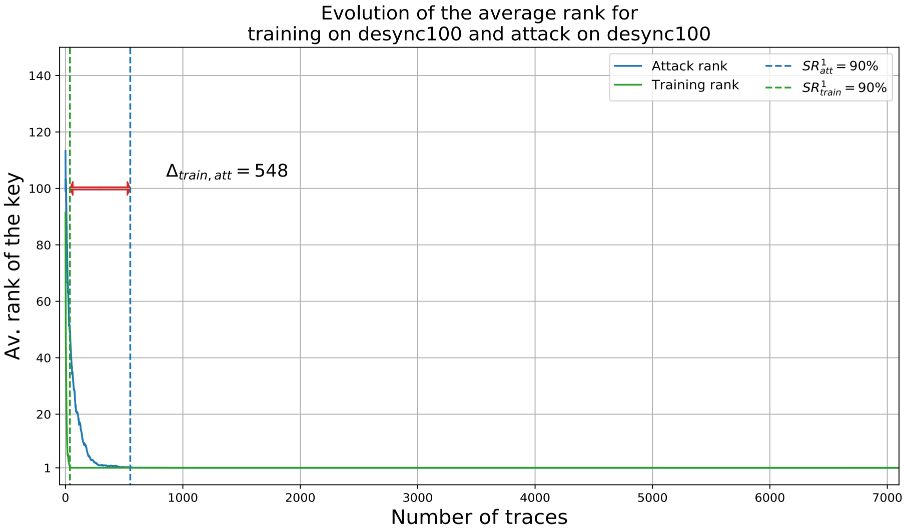
Architecture with regularization: CNN_{bn+reg}



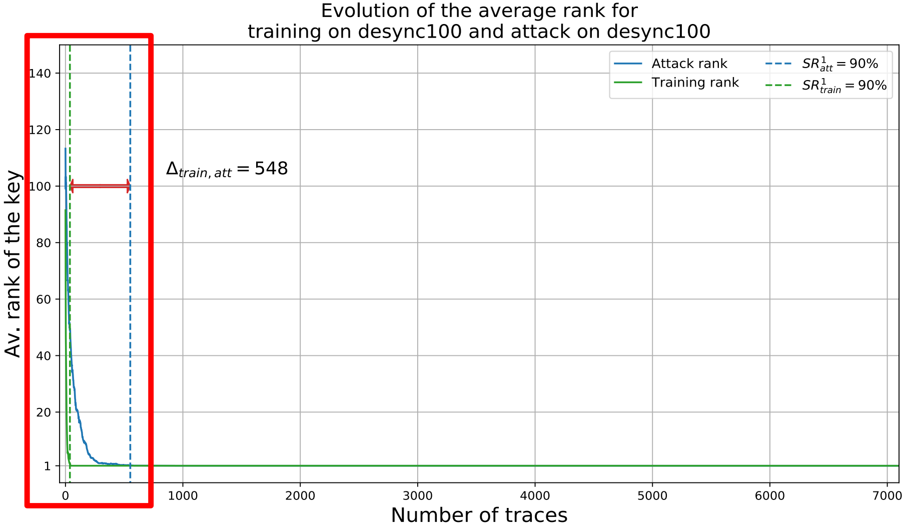
Results without regularization: CNN_{bn}



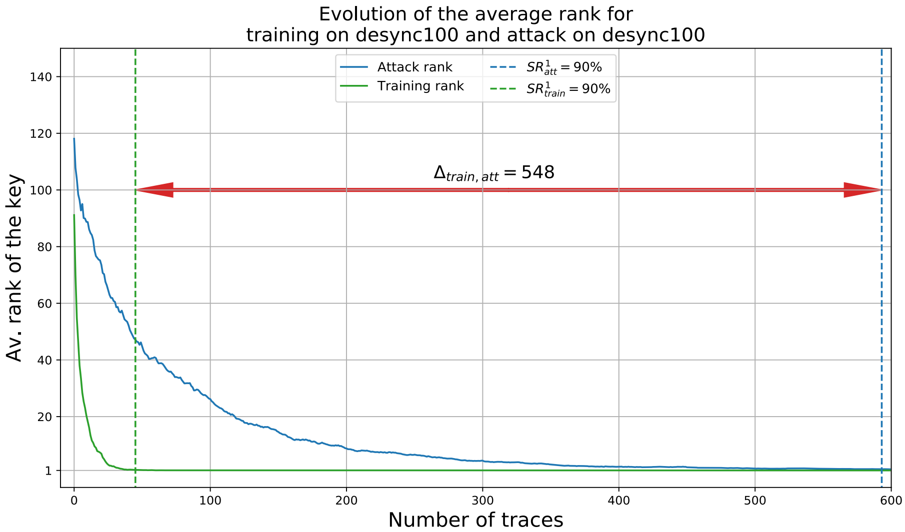
Results with regularization: CNN_{bn+reg}



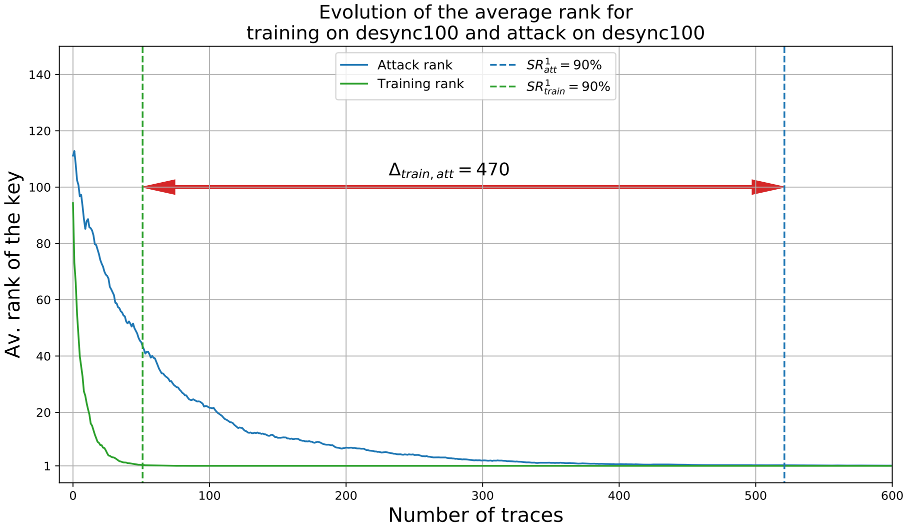
Results with regularization: CNN_{bn+reg}



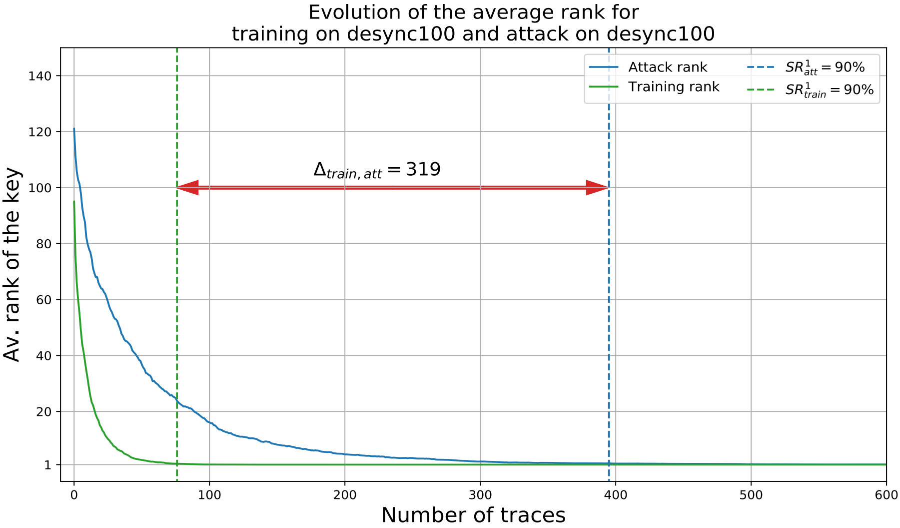
Attack on desync100 using $\lambda_{L_2} = 0.1$ for CNN_{bn+reg}



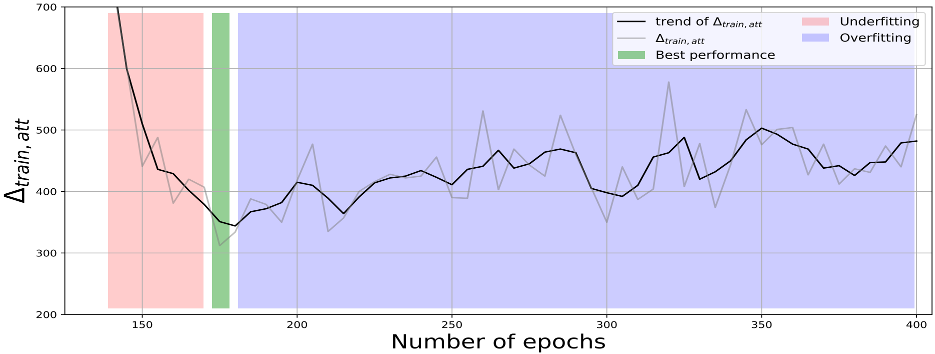
Attack on desync100 using $\lambda_{L_2} = 0.2$ for CNN_{bn+reg}



Attack on desync100 using $\lambda_{L_2} = 0.3$ for CNN_{bn+reg}



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



Best results on other desynchronizations

	N_{train}	N_{att}	$\Delta_{train,att}$	FC1: λ_{L_2}	Nb epochs
Desync0	104	272	**168**	0.1	125
Desync50	21	279	**258**	0.1	200
Desync100	76	395	**319**	0.3	175

Content

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

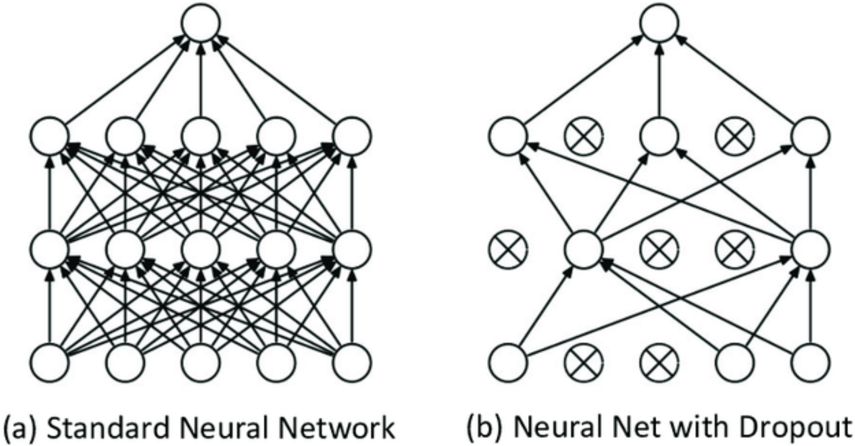
Thank you for listening. Do you have questions ?

LABORATOIRE HUBERT CURIEN

UMR • CNRS • 5516 • SAINT-ETIENNE

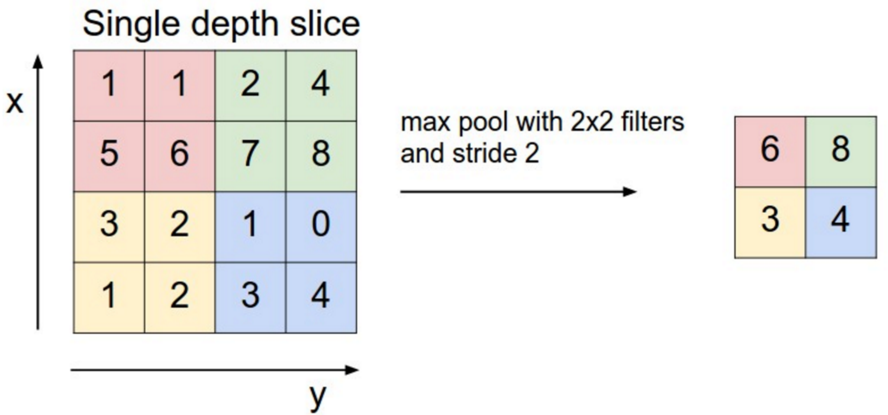
UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Dropout example



Ref.: Roffo, Giorgio. (2017). Ranking to Learn and Learning to Rank: On the Role of Ranking in Pattern Recognition Applications.

Pooling example



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