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

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

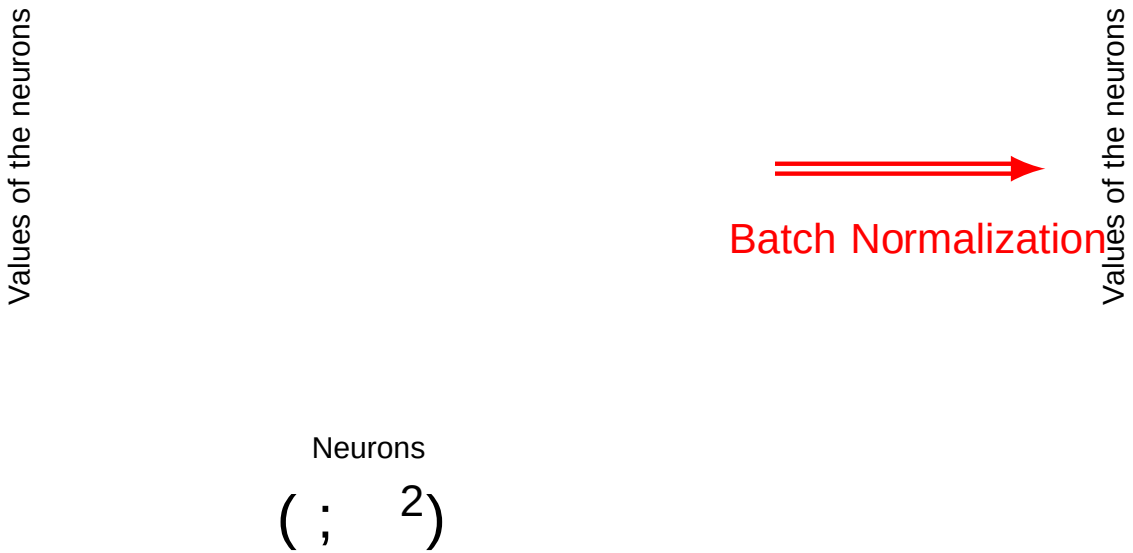
Batch Normalization

Goal

Standardize the data representation across all layers

Consequence

The network focuses on the relative di erences 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

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

The over tting phenomena

Good estimation

Over tting

$_{train;val}$: evaluation of the generalization capacity

Goal

Have a clear indication if the network is over tting/under tting 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

$$_{train;val}(model) = | N_{val}(model) - N_{train}(model) |$$

How to use the metric

Representation of $_{\text{train;att}}$ for CNN$_{\text{bn}}$

Content

Regularization

Goal

Reduce $_{\text{train};\text{att}}$ even further using regularization

Means

- Dropout with parameter $_{D}$
- L_2-Norm regularization with parameter $_{L_2}$

Regularization

Goal

Reduce $_{train;att}$ even further using regularization

Means

Dropout with parameter $_{D}$

L_2-Norm regularization with parameter $_{L_2}$

	Test (step = 0:1)		Choice for desync100	
	D	L_2	D	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 $\ell_2 = 0{:}1$ for $\text{CNN}_{\text{bn+ reg}}$

Attack on desync100 using $\ell_2 = 0.2$ for $\text{CNN}_{\text{bn+ reg}}$

Attack on desync100 using $\lambda_2 = 0.3$ for CNN_{bn+reg}

Evolution of $_{train;att}$ for di erent numbers of epochs

Best results on other desynchronizations

	N_{train}	N_{att}	$_{train;att}$	FC1: 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

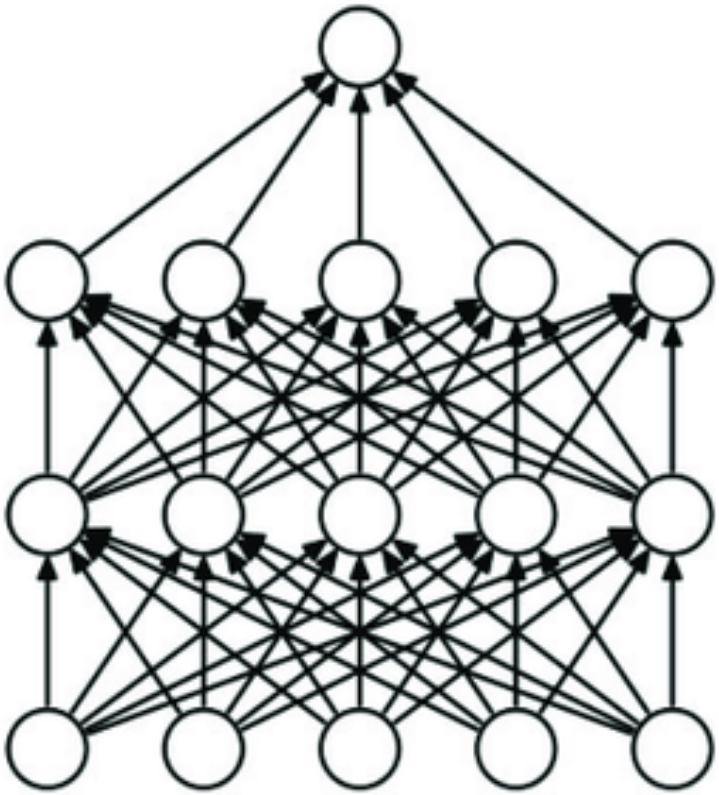
Given the amount of regularization needed to obtain those results, a better architecture probably exists

Apply this technique to other networks

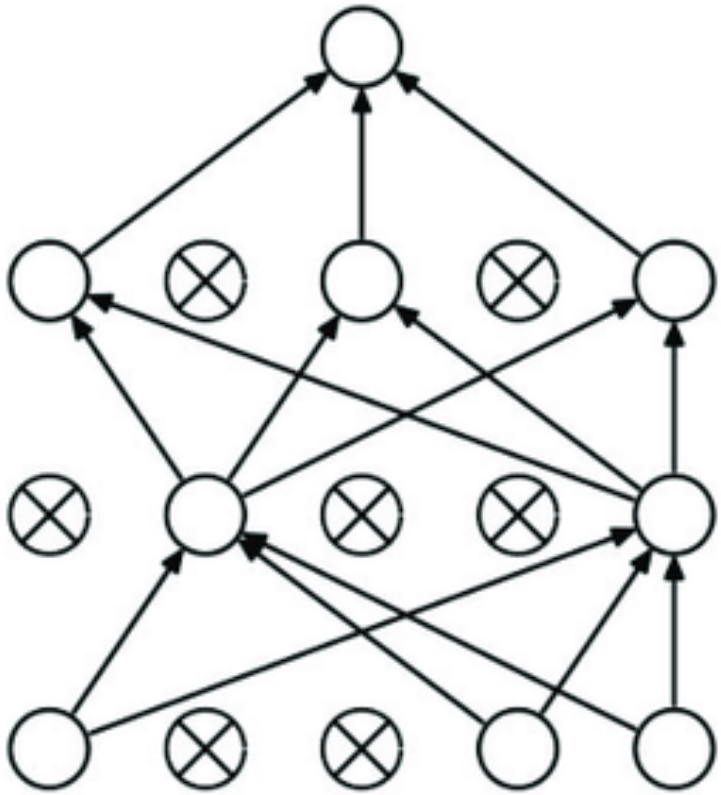
Improved Deep-Learning Side-Channel Attacks using Normalization Layers

Thank you for listening. Do you have questions ?

Dropout example



(a) Standard Neural Network



(b) Neural Net with Dropout

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/