

FROM RESEARCH TO INDUSTRY



## Feature Extraction for Side-Channel Attacks

Eleonora Cagli

05/12/2018, Paris

CESTI - Centre d'Évaluation de la Sécurité des  
Technologies de l'Information - CEA Grenoble

LIP6 - Laboratoire d'Informatique de Paris 6



*PhD Supervisor :* Emmanuel Prouff  
(ANSSI)  
*CEA Supervisor :* Cécile Dumas  
(CESTI - CEA Grenoble)

## Contents

1. Context
2. State of the Art, Objectives, Contributions
3. Kernel Discriminant Analysis against Masking
  - 3.1 Kernel Discriminant Analysis
  - 3.2 Experimental Results
4. Deep Learning against Misalignment
  - 4.1 Data Augmentation
  - 4.2 Experimental Results
5. Conclusions

## Secure Component and Embedded Cryptography

**A piece of hardware with security properties**  
**Embedded cryptography to provide security services**

## Secure Component and Embedded Cryptography

A piece of hardware with security properties  
Embedded cryptography to provide security services



- ▶ Sensitive applications: ID cards, credit cards, transport cards, health cards, SIM
- ▶ Pervasive aspect: several billion smartcards sold per year
- ▶ Hard to update
- ▶ Hostile environment

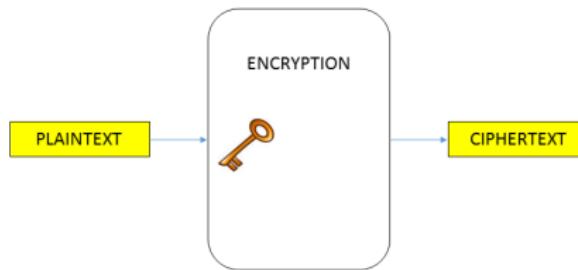
⇒ Requires protection against very high-level attacker

## Security Certification



- ▶ Standardised Evaluation (e.g. ISO/IEC 15408 - Common Criteria)
- ▶ Many evaluation tasks
- ▶ AVA class: vulnerability assessment (penetration testing)

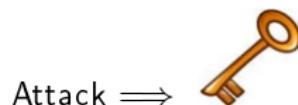
## Side-Channel Vulnerability of Embedded Cryptography



Attack  $\implies$  a secret

Classical Attacks	Side-Channel Attacks
Mathematical vulnerability	
Black Box	

## Side-Channel Vulnerability of Embedded Cryptography



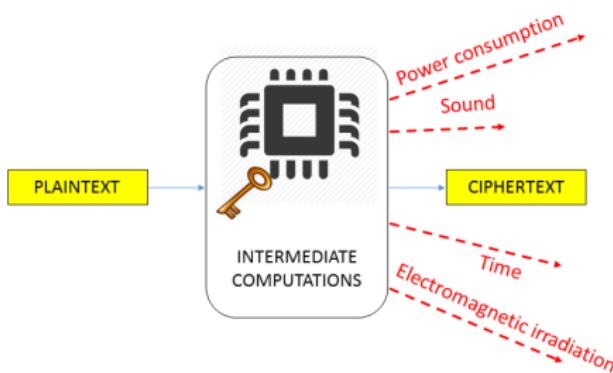
---

Classical Attacks	Side-Channel Attacks
-------------------	----------------------

---

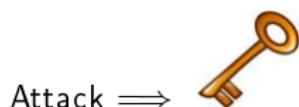
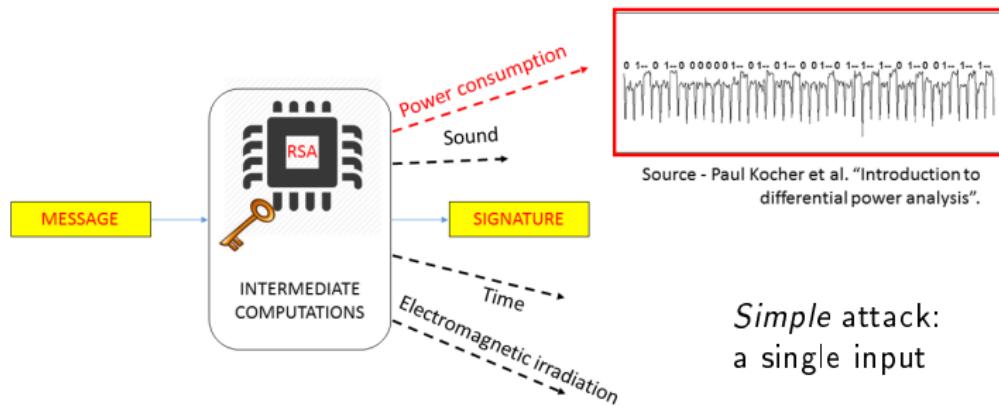
Mathematical vulnerability  
Black Box

## Side-Channel Vulnerability of Embedded Cryptography



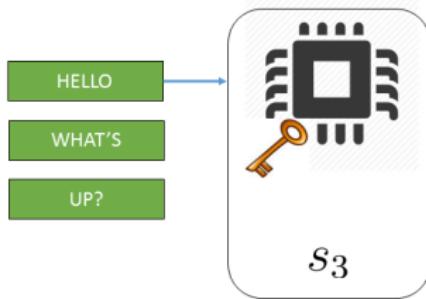
Classical Attacks	Side-Channel Attacks
Mathematical vulnerability	Physical vulnerability
Black Box	Grey Box / Divide-and-conquer

## Side-Channel Vulnerability of Embedded Cryptography

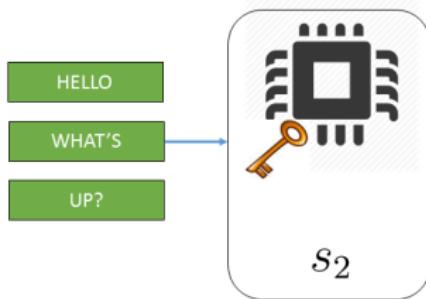


Classical Attacks	Side-Channel Attacks
Mathematical vulnerability	Physical vulnerability
Black Box	Grey Box / Divide-and-conquer

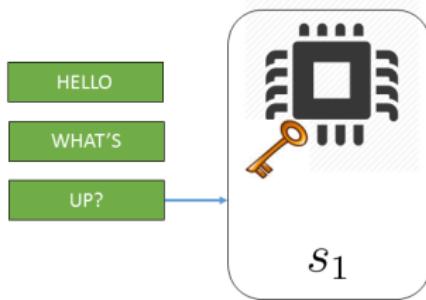
## Advanced Side-Channel Attacks



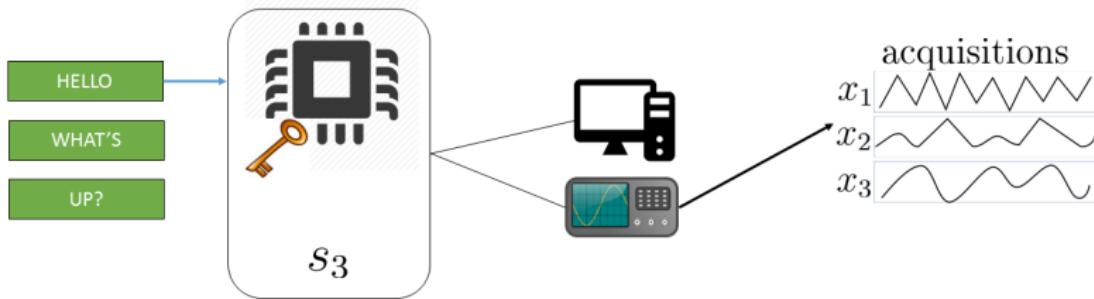
## Advanced Side-Channel Attacks



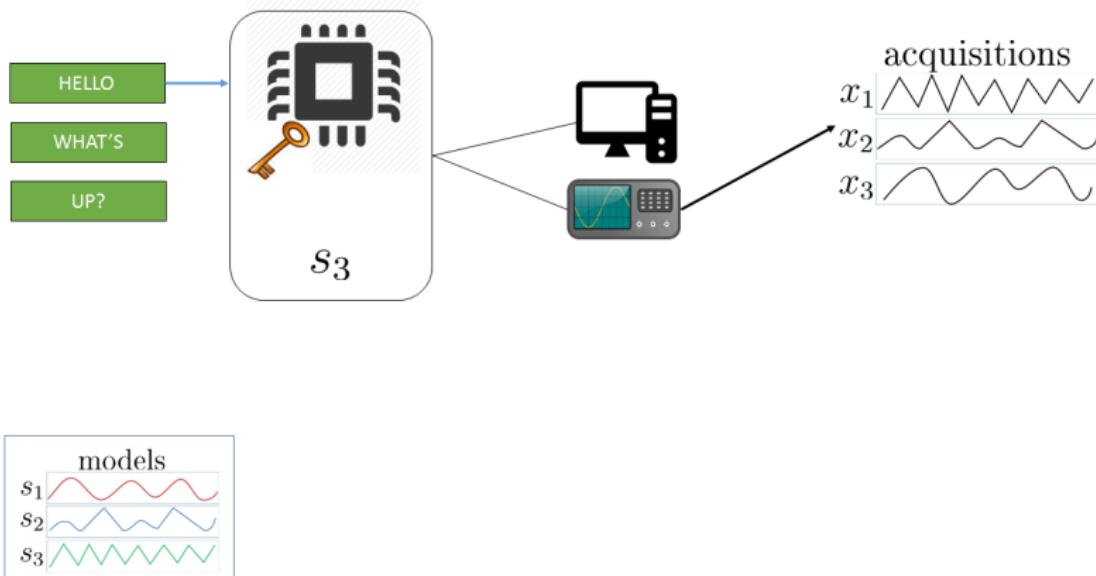
## Advanced Side-Channel Attacks



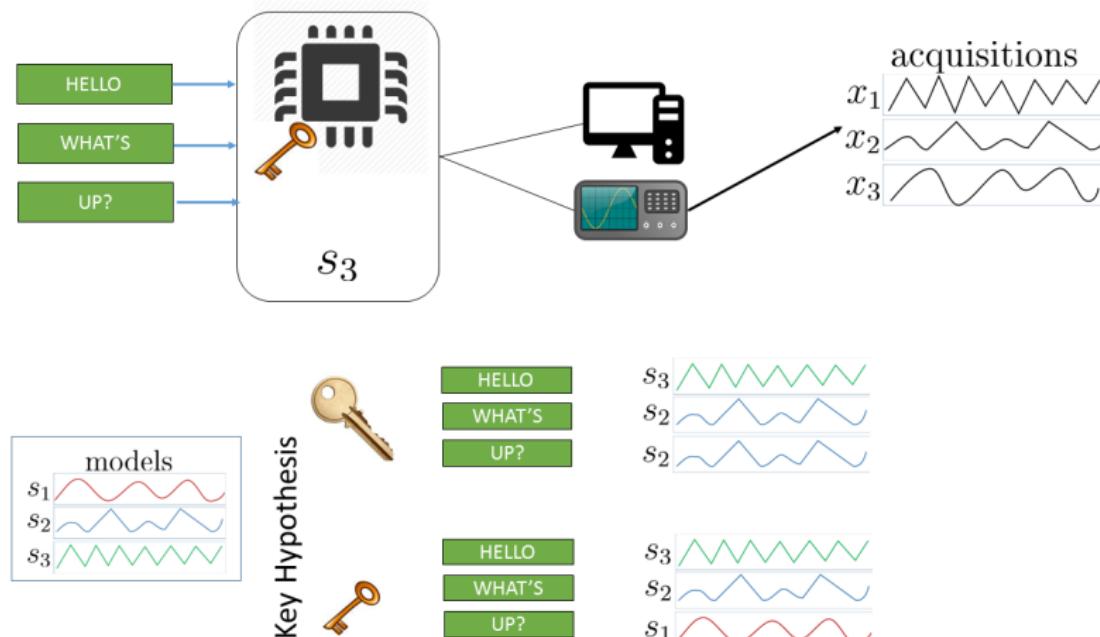
## Advanced Side-Channel Attacks



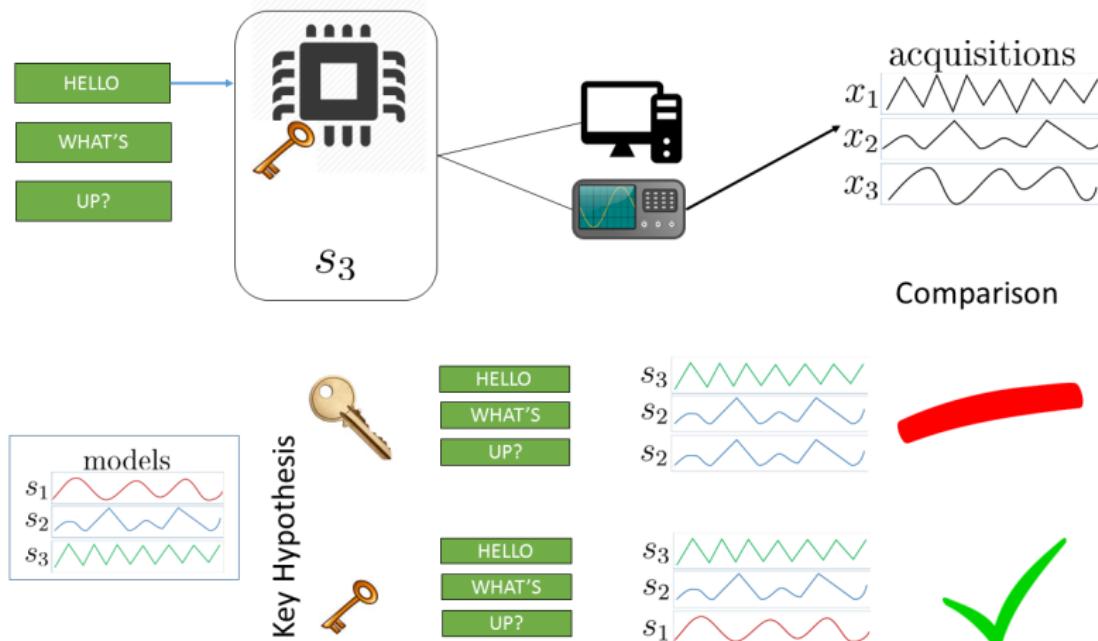
## Advanced Side-Channel Attacks



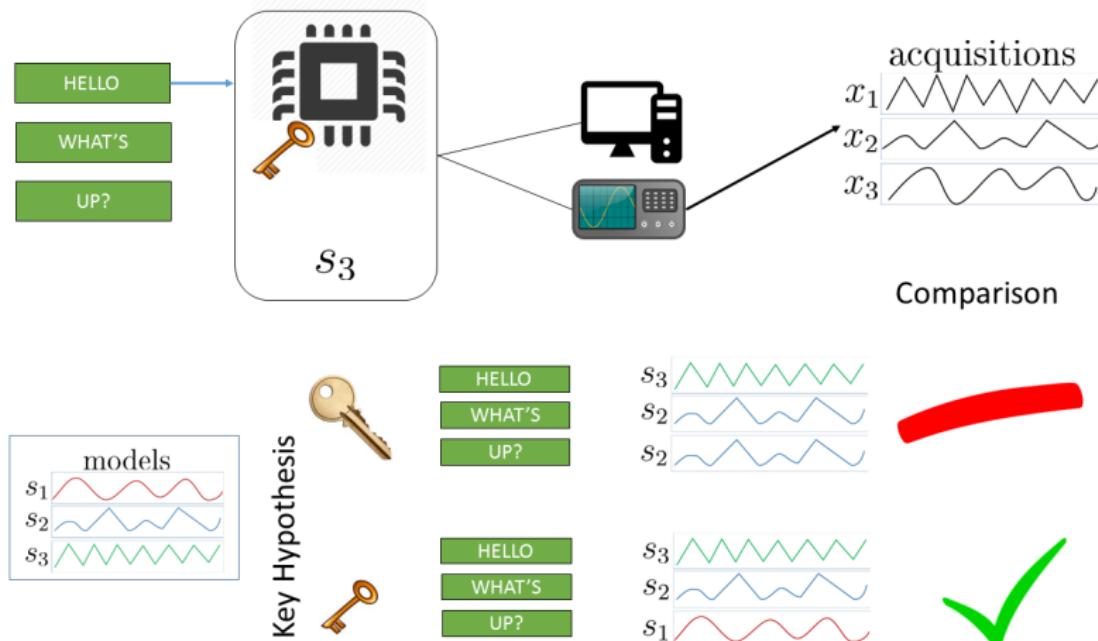
## Advanced Side-Channel Attacks



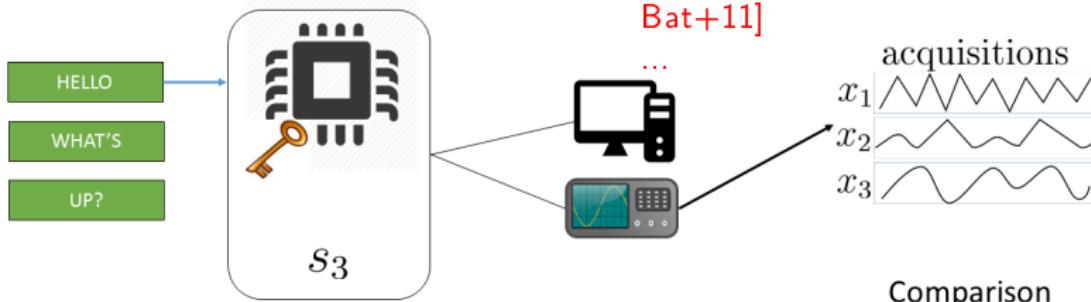
## Advanced Side-Channel Attacks



## Advanced Side-Channel Attacks

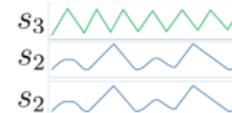
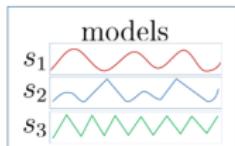


## Advanced Side-Channel Attacks

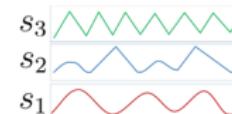


Differential Power Analysis [KJJ99]  
Correlation Power Analysis [BCO04]  
Mutual Information Analysis [Gie+08;  
Bat+11]

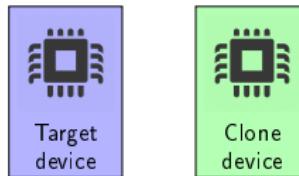
Non-profiling attacks  
Profiling attacks ...



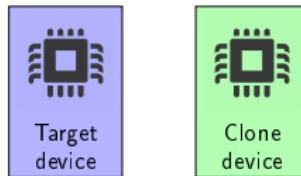
Comparison



## Profiling Attacks...Supervised Learning



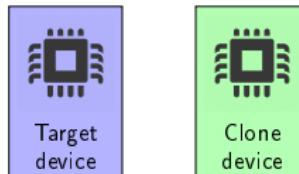
## Profiling Attacks...Supervised Learning



Machine Learning

Supervised Learning

## Profiling Attacks...Supervised Learning



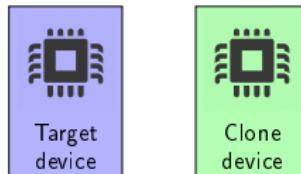
### Machine Learning

*Learn from data via statistic models*

Task - Performance - Experience [TM97]

### Supervised Learning

## Profiling Attacks...Supervised Learning



### Machine Learning

*Learn from data via statistic models*

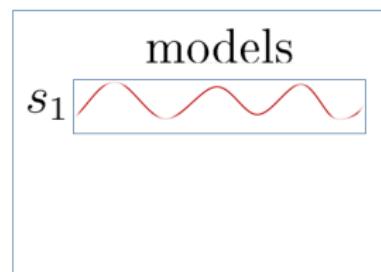
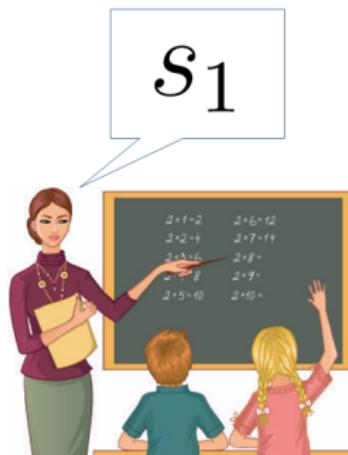
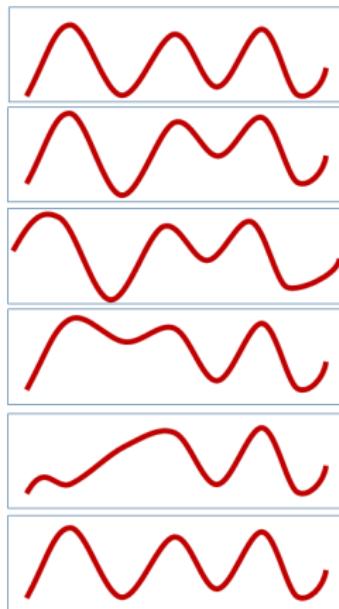
Task - Performance - Experience [TM97]

### Supervised Learning

The *supervised* learning algorithms access to a dataset of examples, each associated in general to a *target* or *label*.



## Classroom Side-Channel Attacks



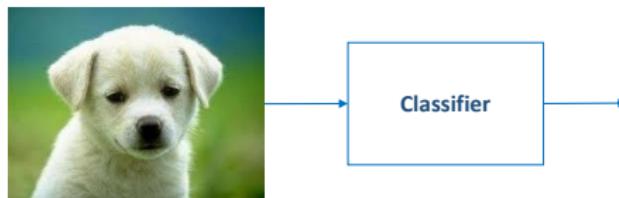
## Classroom Side-Channel Attacks



## Classification

### Classification problem

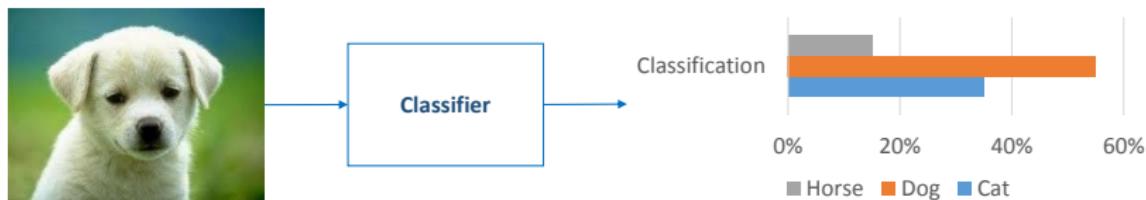
Assign to a datum  $\vec{X}$  (e.g. an image) a label  $Z$  among a set of possible labels  
 $\mathcal{Z} = \{\text{Cat, Dog, Horse}\}$



## Classification

### Classification problem

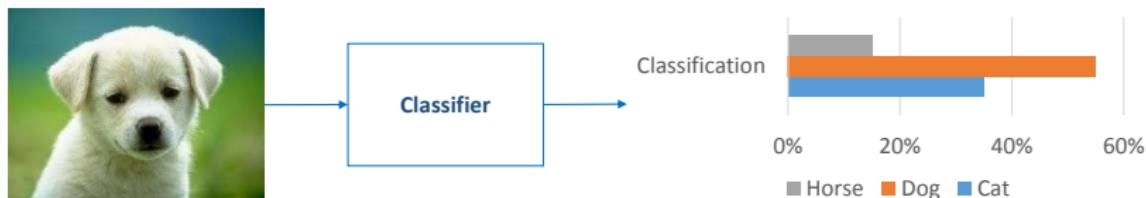
Assign to a datum  $\vec{X}$  (e.g. an image) a label  $Z$  among a set of possible labels  
 $\mathcal{Z} = \{\text{Cat}, \text{Dog}, \text{Horse}\}$



## Classification

### Classification problem

Assign to a datum  $\vec{X}$  (e.g. an image) a label  $Z$  among a set of possible labels  $\mathcal{Z} = \{\text{Cat}, \text{Dog}, \text{Horse}\}$



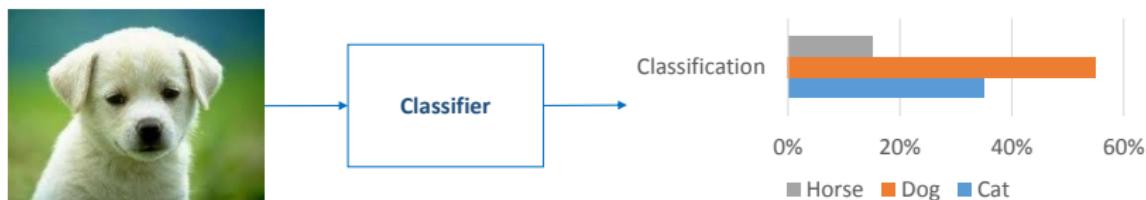
### Advanced Attack as a Classification Problem



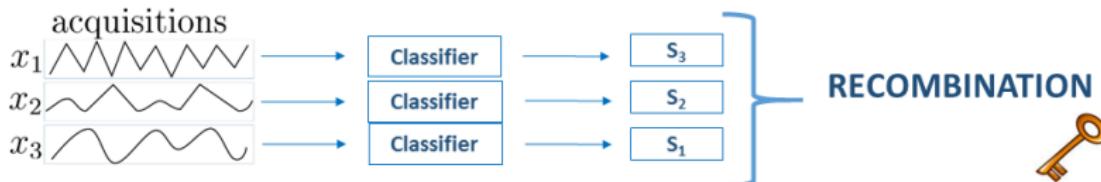
## Classification

### Classification problem

Assign to a datum  $\vec{X}$  (e.g. an image) a label  $Z$  among a set of possible labels  $\mathcal{Z} = \{\text{Cat}, \text{Dog}, \text{Horse}\}$



### Advanced Attack as Multiple Classification Problems

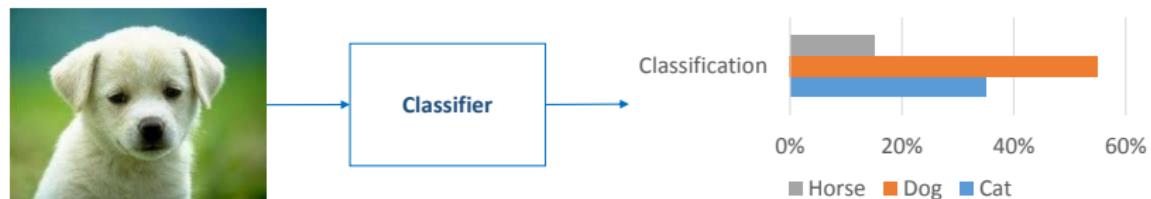


## Classification

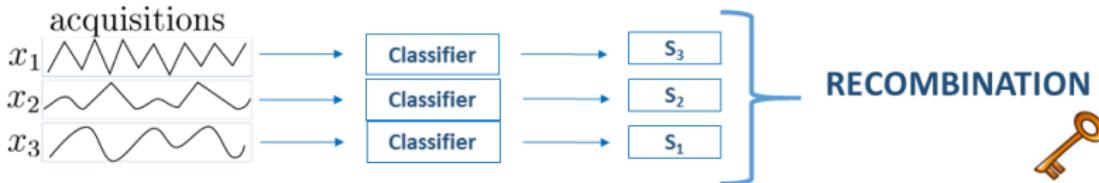
Machine Learning classifiers in Side-Channel literature: SVM ([HZ12; Hos+11]), RF ([LBM14; LBM15])

### Classification problem

Assign to a datum  $\vec{X}$  (e.g. an image) a label  $Z$  among a set of possible labels  $\mathcal{Z} = \{\text{Cat}, \text{Dog}, \text{Horse}\}$



### Advanced Attack as Multiple Classification Problems



## Contents

1. Context
2. State of the Art, Objectives, Contributions
3. Kernel Discriminant Analysis against Masking
  - 3.1 Kernel Discriminant Analysis
  - 3.2 Experimental Results
4. Deep Learning against Misalignment
  - 4.1 Data Augmentation
  - 4.2 Experimental Results
5. Conclusions

## Notations

### Notations and generalities

- ▶ Side-channel traces: realizations of a random vector  $\vec{X} \in \mathbb{R}^D$
- ▶  $D$  is the number of time samples (or features)
- ▶ Target: a *sensitive* variable  $Z = f(e, k)$  in  $\mathcal{Z} = \{s_1, \dots, s_{|\mathcal{Z}|}\}$

### Profiling attack scenario

- ▶ labelled traces  $\mathcal{D}_{\text{train}} = (\vec{x}_i, e_i, k_i)_{i=1}^{N_t}$ , acquired under known secrets
- ▶ attack traces  $\mathcal{D}_{\text{attack}} = (\vec{x}_i, e_i)_{i=1}^{N_a}$  acquired under unknown secrets

## Profiling Attack

### Profiling phase

- ▶ estimate
  - ▶  $p_{\vec{X} \mid Z=z}$

### Attack phase

- ▶ Likelihood score for each key hypothesis  $k$

$$d_k = p_{\vec{X} \mid Z} \left( (\vec{x}_i)_{i=1, \dots, N_a}, (f(e_i, k))_{i=1, \dots, N_a} \right)$$

## Profiling Attack

### Profiling phase

- ▶ estimate
  - ▶  $p_{\vec{X} \mid Z=z} p_{\vec{X}} p_Z$

### Attack phase

- ▶ Likelihood score for each key hypothesis  $k$

$$d_k = p_{\vec{X} \mid Z} \left( (\vec{x}_i)_{i=1, \dots, N_a}, (f(e_i, k))_{i=1, \dots, N_a} \right)$$

- ▶ A-posteriori probability score for each key hypothesis  $k$

$$d_k = p_{Z \mid \vec{X}} \left( f(e_i, k)_{i=1, \dots, N_a}, (\vec{x}_i)_{i=1, \dots, N_a} \right),$$

## Profiling Attack

### Profiling phase

- ▶ estimate
  - ▶  $p_{\vec{X} \mid Z=z} p_{\vec{X}} p_Z$  (generative model)
  - ▶  $p_Z \mid \vec{X}=\vec{x}$  (discriminative model)

### Attack phase

- ▶ Likelihood score for each key hypothesis  $k$

$$d_k = p_{\vec{X} \mid Z} \left( (\vec{x}_i)_{i=1, \dots, N_a}, (f(e_i, k))_{i=1, \dots, N_a} \right)$$

- ▶ A-posteriori probability score for each key hypothesis  $k$

$$d_k = p_{Z \mid \vec{X}} \left( f(e_i, k)_{i=1, \dots, N_a}, (\vec{x}_i)_{i=1, \dots, N_a} \right),$$

## Profiling Attack

### Profiling phase

$$\vec{X} \in \mathbb{R}^D$$

Curse of dimensionality!

- ▶ estimate
  - ▶  $p_{\vec{X} | Z=z}$   $p_{\vec{X}}$   $p_Z$  (generative model)
  - ▶  $p_{Z | \vec{X}=\vec{x}}$  (discriminative model)

### Attack phase

- ▶ Likelihood score for each key hypothesis  $k$

$$d_k = p_{\vec{X} | Z} \left( (\vec{x}_i)_{i=1, \dots, N_a}, (f(e_i, k))_{i=1, \dots, N_a} \right)$$

- ▶ A-posteriori probability score for each key hypothesis  $k$

$$d_k = p_{Z | \vec{X}} \left( f(e_i, k)_{i=1, \dots, N_a}, (\vec{x}_i)_{i=1, \dots, N_a} \right),$$

## Profiling Attack

### Profiling phase

$$\vec{X} \in \mathbb{R}^D$$

Curse of dimensionality!

- ▶ estimate
  - ▶  $p_{\vec{X}} | Z=z$   $p_{\vec{X}} p_Z$  (generative model)
    - ▶ Gaussian hypothesis (**Template Attack**) [CRR03]
  - ▶  $p_Z | \vec{X}=\vec{x}$  (discriminative model)

### Attack phase

- ▶ Likelihood score for each key hypothesis  $k$

$$d_k = p_{\vec{X} | Z} \left( (\vec{x}_i)_{i=1, \dots, N_a}, (f(e_i, k))_{i=1, \dots, N_a} \right)$$

- ▶ A-posteriori probability score for each key hypothesis  $k$

$$d_k = p_{Z | \vec{X}} \left( f(e_i, k)_{i=1, \dots, N_a}, (\vec{x}_i)_{i=1, \dots, N_a} \right),$$

## Profiling Attack

### Profiling phase

$$\vec{X} \in \mathbb{R}^D$$

Curse of dimensionality!

- ▶ mandatory dimensionality reduction  $[D_{\text{train}} \longrightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C]$
- ▶ estimate
  - ▶  $P_{\epsilon(\vec{X}) \mid Z=z} P_{\epsilon(\vec{X})} p_Z$  (generative model)
    - ▶ Gaussian hypothesis (**Template Attack**) [CRR03]
  - ▶  $P_{Z \mid \epsilon(\vec{X})=\epsilon(\vec{x})}$  (discriminative model)

### Attack phase

- ▶ Likelihood score for each key hypothesis  $k$

$$d_k = p_{\epsilon(\vec{X}) \mid Z} \left( (\epsilon(\vec{x}_i))_{i=1, \dots, N_a}, (f(e_i, k))_{i=1, \dots, N_a} \right)$$

- ▶ A-posteriori probability score for each key hypothesis  $k$

$$d_k = p_{Z \mid \epsilon(\vec{X})} \left( f(e_i, k)_{i=1, \dots, N_a}, (\epsilon(\vec{x}_i))_{i=1, \dots, N_a} \right),$$

## Profiling Attack

$$\vec{X} \in \mathbb{R}^D$$

Curse of dimensionality!

### Profiling phase

- ▶ manage de-synchronization problem  $[\mathcal{D}_{\text{train}} \longrightarrow \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D]$
- ▶ mandatory dimensionality reduction  $[\mathcal{D}_{\text{train}} \longrightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C]$
- ▶ estimate
  - ▶  $P_{\epsilon(\rho(\vec{X}))} | Z=z P_{\epsilon(\rho(\vec{X}))} p_Z$  (generative model)
    - ▶ Gaussian hypothesis (**Template Attack**) [CRR03]
  - ▶  $p_Z | \rho(\epsilon(\vec{X})) = \epsilon(\rho(\vec{X}))$  (discriminative model)

### Attack phase

- ▶ Likelihood score for each key hypothesis  $k$

$$d_k = p_{\epsilon(\rho(\vec{X})) | Z} \left( (\epsilon(\rho(\vec{x}_i)))_{i=1, \dots, N_a}, (f(e_i, k))_{i=1, \dots, N_a} \right)$$

- ▶ A-posteriori probability score for each key hypothesis  $k$

$$d_k = p_{Z | \epsilon(\rho(\vec{X}))} \left( f(e_i, k)_{i=1, \dots, N_a}, (\epsilon(\rho(\vec{x}_i)))_{i=1, \dots, N_a} \right),$$

## Objectives

### Profiling phase

- ▶ manage de-synchronization problem  $[\mathcal{D}_{\text{train}} \longrightarrow \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D]$
- ▶ mandatory dimensionality reduction  $[\mathcal{D}_{\text{train}} \longrightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C]$
- ▶ estimate
  - ▶  $P_{\epsilon(\vec{X})} | Z=z P_{\epsilon(\vec{X})} p_Z$  (generative model)
    - ▶ Gaussian hypothesis (**Template Attack**) [CRR03]
  - ▶  $Z | \epsilon(\vec{X}) = \epsilon(\vec{x})$  (discriminative model)

## Objectives

## Objectives

### Profiling phase

- ▶ manage de-synchronization problem  $[\mathcal{D}_{\text{train}} \longrightarrow \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D]$
- ▶ mandatory dimensionality reduction  $[\mathcal{D}_{\text{train}} \longrightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C]$
- ▶ estimate
  - ▶  $P_{\epsilon(\vec{X})} | Z=z P_{\epsilon(\vec{X})} p_Z$  (generative model)
    - ▶ Gaussian hypothesis (**Template Attack**) [CRR03]
  - ▶  $Z | \epsilon(\vec{X}) = \epsilon(\vec{x})$  (discriminative model)

### Objectives

- ▶ Ameliorate the template attack routine by proposing efficient dimensionality reduction techniques

## Objectives

### Profiling phase

- ▶ manage de-synchronization problem [ $\mathcal{D}_{\text{train}} \longrightarrow \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D$ ]
- ▶ mandatory dimensionality reduction [ $\mathcal{D}_{\text{train}} \longrightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C$ ]
- ▶ estimate
  - ▶  $P_{\epsilon(\vec{X})} | Z=z P_{\epsilon(\vec{X})} p_Z$  (generative model)
    - ▶ Gaussian hypothesis (**Template Attack**) [CRR03]
  - ▶  $Z | \epsilon(\vec{X}) = \epsilon(\vec{x})$  (discriminative model)

### Objectives

- ▶ Ameliorate the template attack routine by proposing efficient dimensionality reduction techniques
- ▶ Consider the presence of most-commonly-implemented SCA countermeasures (masking, hiding)

## Objectives

### Profiling phase

- ▶ manage de-synchronization problem  $[D_{\text{train}} \longrightarrow \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D]$
- ▶ mandatory dimensionality reduction  $[D_{\text{train}} \longrightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C]$
- ▶ estimate
  - ▶  $P_{\epsilon(\vec{X})} | Z=z P_{\epsilon(\vec{X})} p_Z$  (generative model)
    - ▶ Gaussian hypothesis (**Template Attack**) [CRR03]
  - ▶  $Z | \epsilon(\vec{X}) = \epsilon(\vec{x})$  (discriminative model)

### Objectives

- ▶ Ameliorate the template attack routine by proposing efficient dimensionality reduction techniques
- ▶ More generally, ameliorate the profiling attack strategy
- ▶ Consider the presence of most-commonly-implemented SCA countermeasures (masking, hiding)

## Objectives

### Profiling phase

- ▶ manage de-synchronization problem  $[\mathcal{D}_{\text{train}} \longrightarrow \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D]$   
**DEEP LEARNING**
- ▶ mandatory dimensionality reduction  $[\mathcal{D}_{\text{train}} \longrightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C]$
- ▶ estimate
  - ▶  $P_{\epsilon(\vec{X})} | Z=z$ ,  $P_{\epsilon(\vec{X})} | p_Z$  (generative model)
    - ▶ Gaussian hypothesis (Template Attack) [CRR03]
  - ▶  $Z | \vec{X} = \vec{x}$  (discriminative model)

### Objectives

- ▶ Ameliorate the template attack routine by proposing efficient dimensionality reduction techniques
- ▶ More generally, ameliorate the profiling attack strategy
- ▶ Consider the presence of most-commonly-implemented SCA countermeasures (masking, hiding)

## Dimensionality Reduction: State of the Art

### Dimensionality Reduction

$$\begin{aligned}\epsilon: \mathbb{R}^D &\rightarrow \mathbb{R}^C \\ \vec{x} &\mapsto \epsilon(\vec{x})\end{aligned}$$

- ▶ Feature selection (Points of Interest selection)
- ▶ Feature extraction

## Dimensionality Reduction: State of the Art

### Dimensionality Reduction

$$\begin{aligned}\epsilon: \mathbb{R}^D &\rightarrow \mathbb{R}^C \\ \vec{x} &\mapsto \epsilon(\vec{x})\end{aligned}$$

- ▶ Feature selection (Points of Interest selection)
- ▶ Feature extraction

### Feature selection

$\epsilon$  performs a sub-sampling

- ▶ SOD [CRR03]
- ▶ SOST [BDP10]
- ▶ SNR [MOP08]/ NICV [Bha+14]
- ▶  $t$ -test,  $F$ -test,... [GLRP06; CK14]

## Dimensionality Reduction: State of the Art

### Dimensionality Reduction

$$\begin{aligned}\epsilon: \mathbb{R}^D &\rightarrow \mathbb{R}^C \\ \vec{x} &\mapsto \epsilon(\vec{x})\end{aligned}$$

- ▶ Feature selection (Points of Interest selection)
- ▶ Feature extraction

### Feature selection

$\epsilon$  performs a sub-sampling

- ▶ SOD [CRR03]
- ▶ SOST [BDP10]
- ▶ SNR [MOP08]/ NICV [Bha+14]
- ▶  $t$ -test,  $F$ -test,... [GLRP06; CK14]

### Linear feature extraction

$\epsilon$  performs linear combinations

$$\epsilon(\vec{x}) = A\vec{x} \text{ with } A \in M_{\mathbb{R}}(C, D)$$

- ▶ Principal Component Analysis (PCA) [Arc+06; BHW12]
- ▶ Linear Discriminant Analysis (LDA) [SA08; Bru+15]
- ▶ Projection Pursuits (PP) [Dur+15]

## Dimensionality Reduction: State of the Art

### Dimensionality Reduction

$$\begin{aligned}\epsilon: \mathbb{R}^D &\rightarrow \mathbb{R}^C \\ \vec{x} &\mapsto \epsilon(\vec{x})\end{aligned}$$

- ▶ Feature selection (Points of Interest selection)
- ▶ Feature extraction

### Feature selection

$\epsilon$  performs a sub-sampling

- ▶ SOD [CRR03]
- ▶ SOST [BDP10]
- ▶ SNR [MOP08]/ NICV [Bha+14]
- ▶  $t$ -test,  $F$ -test, ... [GLRP06; CK14]

### Linear feature extraction

$\epsilon$  performs linear combinations

$$\epsilon(\vec{x}) = A\vec{x} \text{ with } A \in M_{\mathbb{R}}(C, D)$$

- ▶ Principal Component Analysis (PCA) [Arc+06; BHW12]
- ▶ Linear Discriminant Analysis (LDA) [SA08; Bru+15]
- ▶ Projection Pursuits (PP) [Dur+15]

## Contributions

- ▶ **Linear Dimensionality Reduction ([CARDIS 2015]):**
  - ▶ PCA, choice of components ELV
  - ▶ LDA in case of undersampling
- ▶ **Kernel Discriminant Analysis ([CARDIS 2016]):** application of an appropriate kernel trick to LDA, in order to manage masking countermeasure
- ▶ **Convolutional Neural Networks ([CHES 2017]) :**
  - ▶ discriminative model by means of neural network classifiers
  - ▶ convolutional layers to manage desynchronization (a form of hiding)
  - ▶ Data Augmentation techniques to reduce overfitting
- ▶ **ASCAD public dataset** (submitted paper):
  - ▶ deep learning open comparison platform (implementation sources, side-channel traces, attack scripts)
  - ▶ hyper-parametrization methodology proposal and test

## Contributions

- ▶ **Linear Dimensionality Reduction ([CARDIS 2015]):**
  - ▶ PCA, choice of components ELV
  - ▶ LDA in case of undersampling
- ▶ **Kernel Discriminant Analysis ([CARDIS 2016]):** application of an appropriate kernel trick to LDA, in order to manage masking countermeasure
- ▶ **Convolutional Neural Networks ([CHES 2017]):**
  - ▶ discriminative model by means of neural network classifiers
  - ▶ convolutional layers to manage desynchronization (a form of hiding)
  - ▶ Data Augmentation techniques to reduce overfitting
- ▶ **ASCAD public dataset** (submitted paper):
  - ▶ deep learning open comparison platform (implementation sources, side-channel traces, attack scripts)
  - ▶ hyper-parametrization methodology proposal and test

## Contents

1. Context
2. State of the Art, Objectives, Contributions
3. Kernel Discriminant Analysis against Masking
  - 3.1 Kernel Discriminant Analysis
  - 3.2 Experimental Results
4. Deep Learning against Misalignment
  - 4.1 Data Augmentation
  - 4.2 Experimental Results
5. Conclusions

## Dimensionality reduction in presence of masking

### $(d - 1)$ th-order Sharing (or Masking)

Split each sensitive  $Z$  into shares  $Z = M_1 \oplus \dots \oplus M_d$

- ▶ Random *masks*:  $M_1, \dots, M_{d-1}$
- ▶ *Masked variable*:  $M_d = Z \oplus M_1^{-1} \oplus \dots \oplus M_{d-1}$

Shares are handled at time samples

$t_1, \dots, t_d$       (in general different if software countermeasure)

Indistinguishability of  $p_{\vec{X} \mid Z=z}$  up to order  $d - 1$

## Dimensionality reduction in presence of masking

### ( $d - 1$ )th-order Sharing (or Masking)

Split each sensitive  $Z$  into shares  $Z = M_1 \oplus \dots \oplus M_d$

- ▶ Random *masks*:  $M_1, \dots, M_{d-1}$
- ▶ *Masked variable*:  $M_d = Z \oplus M_1^{-1} \oplus \dots \oplus M_{d-1}$

Shares are handled at time samples

$t_1, \dots, t_d$  (in general different if software countermeasure)

Indistinguishability of  $p_{\vec{X} | Z=z}$  up to order  $d - 1$

$f(z) = \mathbb{E} [\vec{X}[t_1] \vec{X}[t_2] \dots \vec{X}[t_d] | Z = z]$  non-constant  $\Rightarrow d$ th-order attack

## Dimensionality reduction in presence of masking

### $(d - 1)$ th-order Sharing (or Masking)

Split each sensitive  $Z$  into shares  $Z = M_1 \oplus \dots \oplus M_d$

- ▶ Random *masks*:  $M_1, \dots, M_{d-1}$
- ▶ *Masked variable*:  $M_d = Z \oplus M_1^{-1} \oplus \dots \oplus M_{d-1}$

Shares are handled at time samples

$t_1, \dots, t_d$  (in general different if software countermeasure)

Indistinguishability of  $p_{\vec{X} | Z=z}$  up to order  $d - 1$

$f(z) = \mathbb{E} [\vec{X}[t_1] \vec{X}[t_2] \dots \vec{X}[t_d] | Z = z]$  non-constant  $\Rightarrow d$ th-order attack  
 $\Rightarrow$  extract features containing  $\vec{X}[t_1] \vec{X}[t_2] \dots \vec{X}[t_d]$  (**Necessary condition**)

## How to detect the $d$ -tuple $t_1, \dots, t_d$ ?

### Feature selection

$\epsilon$  performs a sub-sampling

- ▶ SOD [CRR03]
- ▶ SOST [BDP10]
- ▶ SNR [MOP08]/ NICV [Bha+14]
- ▶  $t$ -test,  $F$ -test, ... [GLRP06; CK14]

- ▶ Point-wise statistics -
- ▶ Exploit  $\mathbb{E}[\vec{X}|Z = z]$  -

### Linear feature extraction

$\epsilon$  performs linear combinations

$$\epsilon(\vec{x}) = A\vec{x} \text{ with } A \in M_{\mathbb{R}}(C, D)$$

- ▶ Principal Component Analysis (PCA) [Arc+06; BHW12]
- ▶ Linear Discriminant Analysis (LDA) [SA08; Bru+15]
- ▶ Projection Pursuits (PP) [Dur+15]

- ▶ Combine all time samples ✓
- ▶ Linear combinations  $\mathbb{E}[A\vec{X}|Z = z]$  -

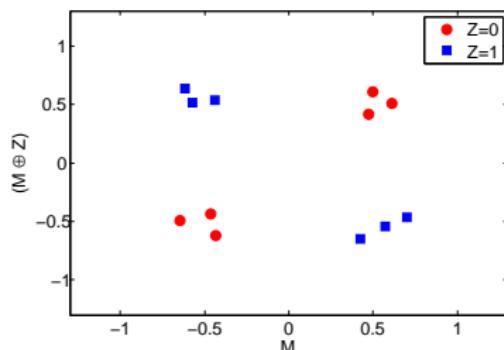
## How to detect the $d$ -tuple $t_1, \dots, t_d$ ?

### Feature selection

$\epsilon$  performs a sub-sampling

- ▶ SOD [CRR03]
- ▶ SOST [BDP10]
- ▶ SNR [MOP08] / NICV [Bha+14]
- ▶  $t$ -test,  $F$ -test, ... [GLRP06; CK14]

- ▶ Point-wise statistics -
- ▶ Exploit  $\mathbb{E}[\vec{X}|Z = z]$  -



### Linear feature extraction

$\epsilon$  performs linear combinations

$$\epsilon(\vec{x}) = A\vec{x} \text{ with } A \in M_{\mathbb{R}}(C, D)$$

- ▶ Principal Component Analysis (PCA) [Arc+06; BHW12]
- ▶ Linear Discriminant Analysis (LDA) [SA08; Bru+15]
- ▶ Projection Pursuits (PP) [Dur+15]

- ▶ Combine all time samples ✓
- ▶ Linear combinations  $\mathbb{E}[A\vec{X}|Z = z]$  -

Toy example: 2 time samples, 1-bit data

$$t_1: M + n, n \sim \mathcal{N}(0, 0.1)$$

$$t_2: M \oplus Z + n \text{ (Boolean masking)}$$

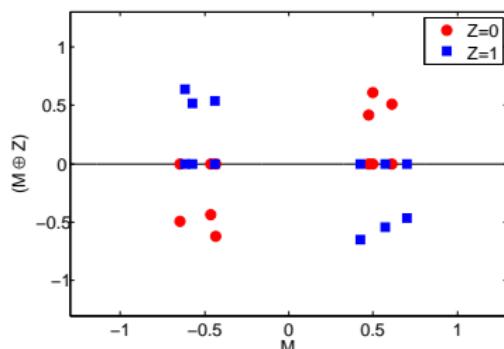
## How to detect the $d$ -tuple $t_1, \dots, t_d$ ?

### Feature selection

ε performs a sub-sampling

- ▶ SOD [CRR03]
- ▶ SOST [BDP10]
- ▶ SNR [MOP08] / NICV [Bha+14]
- ▶  $t$ -test,  $F$ -test, ... [GLRP06; CK14]

- ▶ Point-wise statistics -
- ▶ Exploit  $\mathbb{E}[\vec{X}|Z = z]$  -



### Linear feature extraction

ε performs linear combinations

$$\epsilon(\vec{x}) = A\vec{x} \text{ with } A \in M_{\mathbb{R}}(C, D)$$

- ▶ Principal Component Analysis (PCA) [Arc+06; BHW12]
- ▶ Linear Discriminant Analysis (LDA) [SA08; Bru+15]
- ▶ Projection Pursuits (PP) [Dur+15]

- ▶ Combine all time samples ✓
- ▶ Linear combinations  $\mathbb{E}[A\vec{X}|Z = z]$  -

Toy example: 2 time samples, 1-bit data

$$t_1: M + n, n \sim \mathcal{N}(0, 0.1)$$

$$t_2: M \oplus Z + n \text{ (Boolean masking)}$$

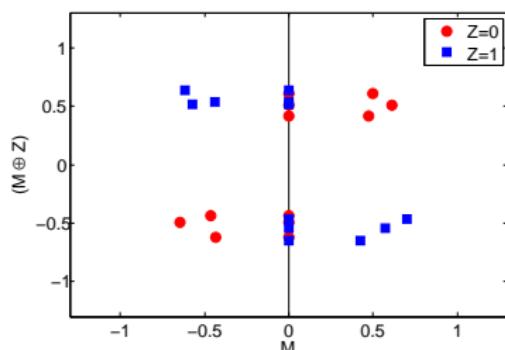
## How to detect the $d$ -tuple $t_1, \dots, t_d$ ?

### Feature selection

ε performs a sub-sampling

- ▶ SOD [CRR03]
- ▶ SOST [BDP10]
- ▶ SNR [MOP08] / NICV [Bha+14]
- ▶  $t$ -test,  $F$ -test, ... [GLRP06; CK14]

- ▶ Point-wise statistics -
- ▶ Exploit  $\mathbb{E}[\vec{X}|Z = z]$  -



### Linear feature extraction

ε performs linear combinations

$$\epsilon(\vec{x}) = A\vec{x} \text{ with } A \in M_{\mathbb{R}}(C, D)$$

- ▶ Principal Component Analysis (PCA) [Arc+06; BHW12]
- ▶ Linear Discriminant Analysis (LDA) [SA08; Bru+15]
- ▶ Projection Pursuits (PP) [Dur+15]

- ▶ Combine all time samples ✓
- ▶ Linear combinations  $\mathbb{E}[A\vec{X}|Z = z]$  -

Toy example: 2 time samples, 1-bit data

$$t_1: M + n, n \sim \mathcal{N}(0, 0.1)$$

$$t_2: M \oplus Z + n \text{ (Boolean masking)}$$

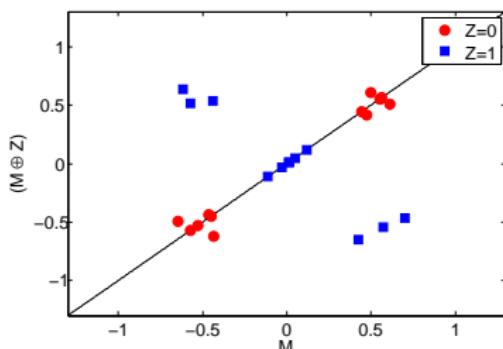
## How to detect the $d$ -tuple $t_1, \dots, t_d$ ?

### Feature selection

ε performs a sub-sampling

- ▶ SOD [CRR03]
- ▶ SOST [BDP10]
- ▶ SNR [MOP08] / NICV [Bha+14]
- ▶  $t$ -test,  $F$ -test, ... [GLRP06; CK14]

- ▶ Point-wise statistics -
- ▶ Exploit  $\mathbb{E}[\vec{X}|Z = z]$  -



### Linear feature extraction

ε performs linear combinations

$$\epsilon(\vec{x}) = A\vec{x} \text{ with } A \in M_{\mathbb{R}}(C, D)$$

- ▶ Principal Component Analysis (PCA) [Arc+06; BHW12]
- ▶ Linear Discriminant Analysis (LDA) [SA08; Bru+15]
- ▶ Projection Pursuits (PP) [Dur+15]

- ▶ Combine all time samples ✓
- ▶ Linear combinations  $\mathbb{E}[A\vec{X}|Z = z]$  -

Toy example: 2 time samples, 1-bit data

$$t_1: M + n, n \sim \mathcal{N}(0, 0.1)$$

$$t_2: M \oplus Z + n \text{ (Boolean masking)}$$

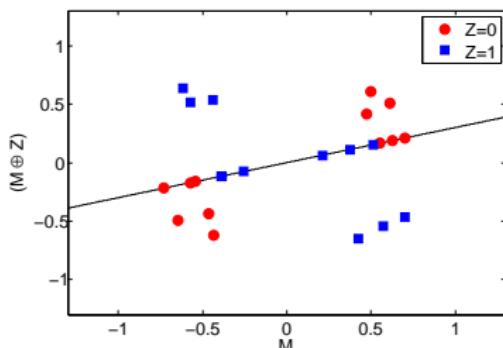
## How to detect the $d$ -tuple $t_1, \dots, t_d$ ?

### Feature selection

ε performs a sub-sampling

- ▶ SOD [CRR03]
- ▶ SOST [BDP10]
- ▶ SNR [MOP08] / NICV [Bha+14]
- ▶  $t$ -test,  $F$ -test, ... [GLRP06; CK14]

- ▶ Point-wise statistics -
- ▶ Exploit  $\mathbb{E}[\vec{X}|Z = z]$  -



### Linear feature extraction

ε performs linear combinations

$$\epsilon(\vec{x}) = A\vec{x} \text{ with } A \in M_{\mathbb{R}}(C, D)$$

- ▶ Principal Component Analysis (PCA) [Arc+06; BHW12]
- ▶ Linear Discriminant Analysis (LDA) [SA08; Bru+15]
- ▶ Projection Pursuits (PP) [Dur+15]

- ▶ Combine all time samples ✓
- ▶ Linear combinations  $\mathbb{E}[A\vec{X}|Z = z]$  -

Toy example: 2 time samples, 1-bit data

$$t_1: M + n, n \sim \mathcal{N}(0, 0.1)$$

$$t_2: M \oplus Z + n \text{ (Boolean masking)}$$

## Pols Research

### A lacking literature

- ▶ many HO attacks papers assume the knowledge of  $t_1, \dots, t_d$
- ▶ Pol research exploiting the masks knowledge in profiling phase
- ▶ Hand selection via educated guess [Osw+06]
- ▶ Feature Selection for Higher-Order Attacks → Projection Pursuits [Dur+15]

### Kernel Discriminant Analysis starting point

Naive strategy: infer over all possible  $d$ -tuples

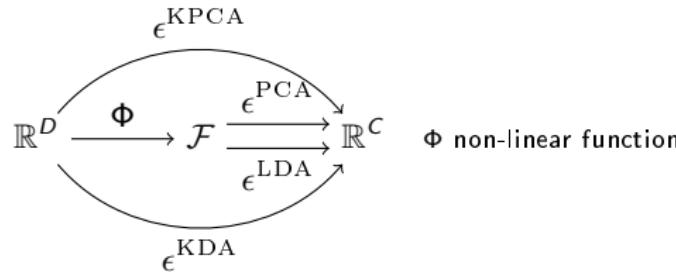
## KDA: the purpose

### Problem

All  $d$ th-degree monomials in the trace coordinates lies in the:

$$\mathcal{F} = \mathbb{R}^{\binom{D+d-1}{d}} \quad \text{feature space}$$

⚠ Dimension increasing combinatorially with  $d$  and  $D$



### KDA

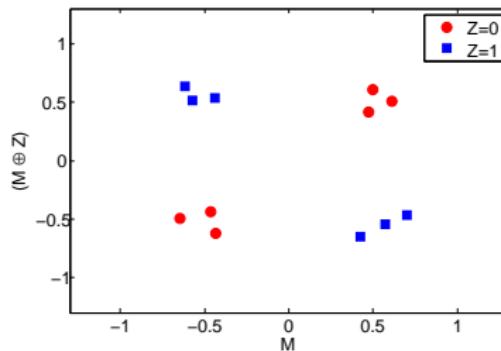
KDA allows performing LDA in  $\mathcal{F}$ , remaining in  $\mathbb{R}^D$ .

## KDA: an intuition

Toy example: 2 time samples, 1-bit data

$t_1: M + n, n \sim \mathcal{N}(0, 0.1)$

$t_2: M \oplus Z + n$  (Boolean masking)

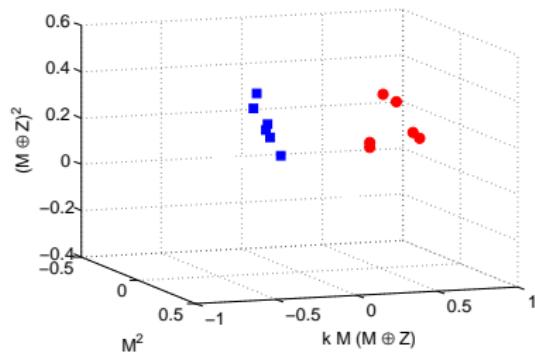
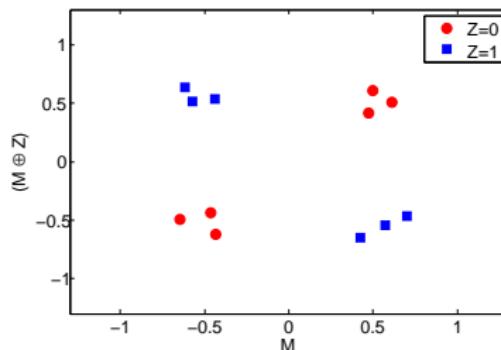


## KDA: an intuition

Toy example: 2 time samples, 1-bit data

$$t_1: M + n, \quad n \sim \mathcal{N}(0, 0.1)$$

$$t_2: M \oplus Z + n \text{ (Boolean masking)}$$



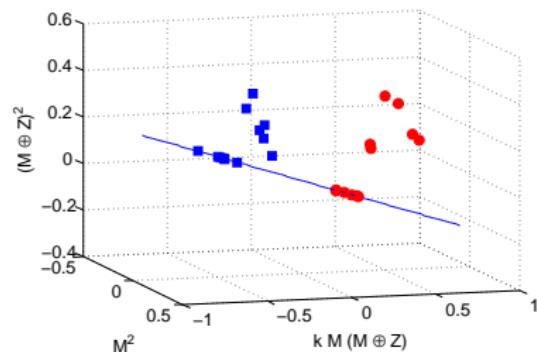
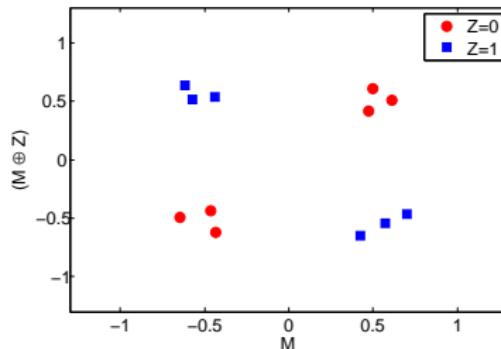
$$\Phi: \mathbb{R}^D \rightarrow \mathbb{R}^{\binom{D+d-1}{d}}$$
$$\Phi(t_1, t_2) = (t_1^2, t_2^2, k t_1 t_2)$$

## KDA: an intuition

Toy example: 2 time samples, 1-bit data

$t_1: M + n, n \sim \mathcal{N}(0, 0.1)$

$t_2: M \oplus Z + n$  (Boolean masking)



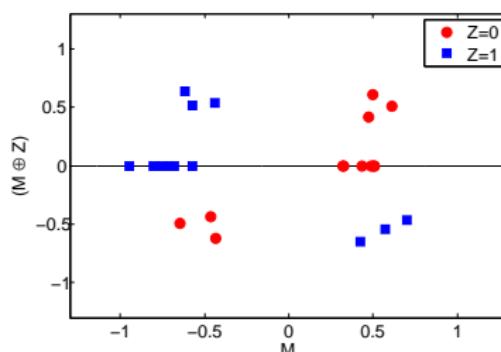
$\Phi \rightarrow \text{LDA}$

## KDA: an intuition

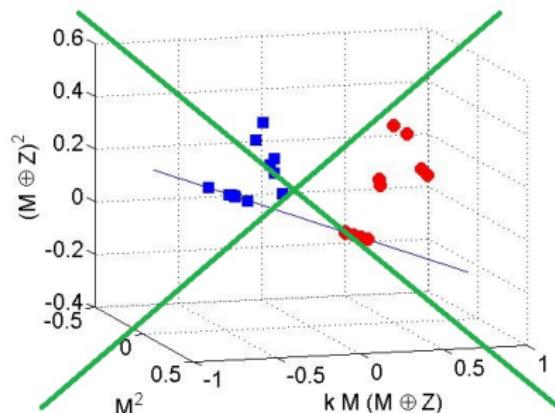
Toy example: 2 time samples, 1-bit data

$$\mathbf{t}_1: M + n, \quad n \sim \mathcal{N}(0, 0.1)$$

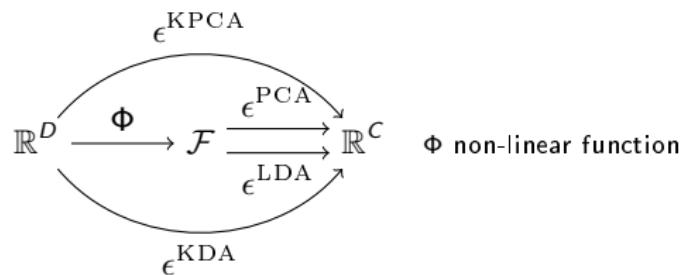
$$\mathbf{t}_2: M \oplus Z + n \text{ (Boolean masking)}$$



KDA  
remains in  $\mathbb{R}^D$



## Kernel Function



### Kernel Function

$$K: \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad (1)$$

### *d-degree Polynomial Kernel Function*

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d \quad \leftrightarrow \quad \Phi: \mathbb{R}^D \rightarrow \mathcal{F} \subset \mathbb{R}^{\binom{D+d-1}{d}} \text{ all } d\text{th-degree monomials}$$

## KDA - the training

"Fisher Discriminant Analysis with Kernels" ([SM99])

Between-class (inter-class) Covariance Matrix

### LDA

$$\blacktriangleright \mathbf{S_B} = \sum_{z \in \mathcal{Z}} N_z (\vec{\mu}_s - \vec{x})(\vec{\mu}_s - \vec{x})^\top$$

### KDA

$$\blacktriangleright \mathbf{M} = \sum_{z \in \mathcal{Z}} N_z (\vec{M}_s - \vec{M}_T)(\vec{M}_s - \vec{M}_T)^\top$$



---

<sup>1</sup>  $\vec{M}_s$  and  $\vec{M}_T$  are two  $N$ -sized column vectors whose entries are given by:

$$\vec{M}_z[j] = \frac{1}{N_z} \sum_{i:z_i=z}^{N_z} K(\mathbf{x}_j^{z_j}, \mathbf{x}_i^{z_i}), \quad \vec{M}_T[j] = \frac{1}{N} \sum_{i=1}^N K(\mathbf{x}_j^{z_j}, \mathbf{x}_i^{z_i}).$$

<sup>2</sup>  $I$  is a  $N_z \times N_z$  identity matrix,  $I_{N_z}$  is a  $N_z \times N_z$  matrix with all entries equal to  $\frac{1}{N_z}$  and  $K_z$  is the  $N \times N_z$  sub-matrix of  $K = (K(\mathbf{x}_i^{z_i}, \mathbf{x}_j^{z_j}))_{i=1, \dots, N}^{j=1, \dots, N}$  storing only columns indexed by the indices  $i$  such that  $z_i = z$

## KDA - the training

"Fisher Discriminant Analysis with Kernels" ([SM99])

Within-class (intra-class) Covariance Matrix

### LDA

- $\mathbf{S}_B = \sum_{z \in \mathcal{Z}} N_z (\vec{\mu}_s - \bar{\vec{x}})(\vec{\mu}_s - \bar{\vec{x}})^T$
- $\mathbf{S}_W = \sum_{z \in \mathcal{Z}} \sum_{i=1}^{N_z} (\mathbf{x}_i^z - \vec{\mu}_s)(\mathbf{x}_i^z - \vec{\mu}_s)^T$

### KDA

- $\mathbf{M} = \sum_{z \in \mathcal{Z}} N_z (\vec{M}_s - \vec{M}_T)(\vec{M}_s - \vec{M}_T)^T$ <sup>1</sup>
- $\mathbf{N} = \sum_{z \in \mathcal{Z}} \mathbf{K}_z (\mathbf{I} - \mathbf{I}_{N_z}) \mathbf{K}_z^T$ <sup>2</sup>
- 

---

<sup>1</sup>  $\vec{M}_s$  and  $\vec{M}_T$  are two  $N$ -sized column vectors whose entries are given by:

$$\vec{M}_z[j] = \frac{1}{N_z} \sum_{i:z_i=z}^{N_z} K(\mathbf{x}_j^{z_j}, \mathbf{x}_i^{z_i}), \quad \vec{M}_T[j] = \frac{1}{N} \sum_{i=1}^N K(\mathbf{x}_j^{z_j}, \mathbf{x}_i^{z_i}).$$

<sup>2</sup>  $\mathbf{I}$  is a  $N_z \times N_z$  identity matrix,  $\mathbf{I}_{N_z}$  is a  $N_z \times N_z$  matrix with all entries equal to  $\frac{1}{N_z}$  and  $\mathbf{K}_z$  is the  $N \times N_z$  sub-matrix of  $\mathbf{K} = (K(\mathbf{x}_i^{z_i}, \mathbf{x}_j^{z_j}))_{i=1, \dots, N} \atop j=1, \dots, N$  storing only columns indexed by the indices  $i$  such that  $z_i = z$

## KDA - the training

"Fisher Discriminant Analysis with Kernels" ([SM99])

Eigenvector problem

Computational Complexity  $O(D^3)$

### LDA

- ▶  $\mathbf{S}_B = \sum_{z \in \mathcal{Z}} N_z (\vec{\mu}_s - \bar{\vec{x}})(\vec{\mu}_s - \bar{\vec{x}})^\top$
- ▶  $\mathbf{S}_W = \sum_{z \in \mathcal{Z}} \sum_{i=1}^{N_z} (\mathbf{x}_i^z - \vec{\mu}_s)(\mathbf{x}_i^z - \vec{\mu}_s)^\top$
- ▶  $\vec{\alpha}_i$  eigenvectors of  $\mathbf{S}_W^{-1} \mathbf{S}_B$   $[D \times D]$

Computational Complexity  $O(N^3)$

### KDA

- ▶  $\mathbf{M} = \sum_{z \in \mathcal{Z}} N_z (\vec{M}_s - \vec{M}_T)(\vec{M}_s - \vec{M}_T)^\top$ <sup>1</sup>
- ▶  $\mathbf{N} = \sum_{z \in \mathcal{Z}} \mathbf{K}_z (\mathbf{I} - \mathbf{I}_{N_z}) \mathbf{K}_z^\top$ <sup>2</sup>
- ▶  $\vec{\nu}_i$  eigenvectors of  $\mathbf{N}^{-1} \mathbf{M}$   $[N \times N]$
- ▶

---

<sup>1</sup>  $\vec{M}_s$  and  $\vec{M}_T$  are two  $N$ -sized column vectors whose entries are given by:

$$\vec{M}_z[j] = \frac{1}{N_z} \sum_{i:z_i=z}^{N_z} K(\mathbf{x}_j^{z_j}, \mathbf{x}_i^{z_i}), \quad \vec{M}_T[j] = \frac{1}{N} \sum_{i=1}^N K(\mathbf{x}_j^{z_j}, \mathbf{x}_i^{z_i}).$$

<sup>2</sup>  $\mathbf{I}$  is a  $N_z \times N_z$  identity matrix,  $\mathbf{I}_{N_z}$  is a  $N_z \times N_z$  matrix with all entries equal to  $\frac{1}{N_z}$  and  $\mathbf{K}_z$  is the  $N \times N_z$  sub-matrix of  $\mathbf{K} = (K(\mathbf{x}_i^{z_i}, \mathbf{x}_j^{z_j}))_{i=1, \dots, N} \atop j=1, \dots, N$  storing only columns indexed by the indices  $i$  such that  $z_i = z$

## KDA - the training

"Fisher Discriminant Analysis with Kernels" ([SM99])

New trace projection

Computational Complexity  $O(D^3)$

### LDA

- ▶  $\mathbf{S}_B = \sum_{z \in \mathcal{Z}} N_z (\vec{\mu}_s - \bar{\vec{x}})(\vec{\mu}_s - \bar{\vec{x}})^\top$
- ▶  $\mathbf{S}_W = \sum_{z \in \mathcal{Z}} \sum_{i=1}^{N_z} (\mathbf{x}_i^z - \vec{\mu}_s)(\mathbf{x}_i^z - \vec{\mu}_s)^\top$
- ▶  $\vec{\alpha}_i$  eigenvectors of  $\mathbf{S}_W^{-1} \mathbf{S}_B$   $[D \times D]$
- ▶  $\epsilon_\ell^{LDA}(\vec{x}) = \sum_{i=1}^D \vec{\alpha}_\ell[i] \vec{x}[i]$

Computational Complexity  $O(N^3)$

### KDA

- ▶  $\mathbf{M} = \sum_{z \in \mathcal{Z}} N_z (\vec{M}_s - \vec{M}_T)(\vec{M}_s - \vec{M}_T)^\top$ <sup>1</sup>
- ▶  $\mathbf{N} = \sum_{z \in \mathcal{Z}} \mathbf{K}_z (\mathbf{I} - \mathbf{I}_{N_z}) \mathbf{K}_z^\top$ <sup>2</sup>
- ▶  $\vec{\nu}_i$  eigenvectors of  $\mathbf{N}^{-1} \mathbf{M}$   $[N \times N]$
- ▶  $\epsilon_\ell^{KDA}(\vec{x}) = \sum_{i=1}^N \vec{\nu}_\ell[i] K(\mathbf{x}_i^{z_i}, \mathbf{x})$

<sup>1</sup>  $\vec{M}_s$  and  $\vec{M}_T$  are two  $N$ -sized column vectors whose entries are given by:

$$\vec{M}_z[j] = \frac{1}{N_z} \sum_{i:z_i=z}^{N_z} K(\mathbf{x}_j^{z_j}, \mathbf{x}_i^{z_i}), \quad \vec{M}_T[j] = \frac{1}{N} \sum_{i=1}^N K(\mathbf{x}_j^{z_j}, \mathbf{x}_i^{z_i}).$$

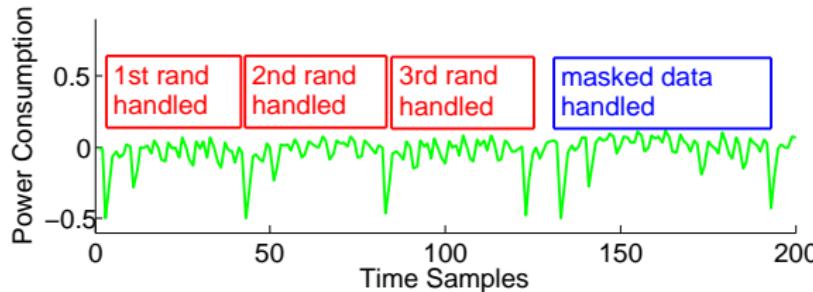
<sup>2</sup>  $\mathbf{I}$  is a  $N_z \times N_z$  identity matrix,  $\mathbf{I}_{N_z}$  is a  $N_z \times N_z$  matrix with all entries equal to  $\frac{1}{N_z}$  and  $\mathbf{K}_z$  is the  $N \times N_z$  sub-matrix of  $\mathbf{K} = (K(\mathbf{x}_i^{z_i}, \mathbf{x}_j^{z_j}))_{i=1, \dots, N} \atop j=1, \dots, N$  storing only columns indexed by the indices  $i$  such that  $z_i = z$

## Experimental setup

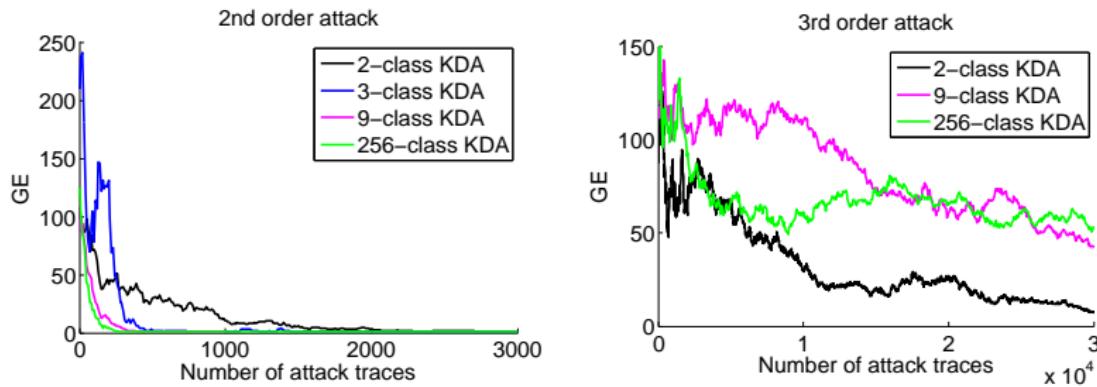
Target device and acquisitions:

- ▶ 8-bit AVR microprocessor Atmega328P
- ▶ power-consumption acquired via the ChipWhisperer [OC14] platform
- ▶  $D = 200$ , 4 clock-cycles are selected
- ▶ 9.000 KDA training traces

Sensitive variable:  $Z = \text{Sbox}_{\text{AES}}(P \oplus K^*)$

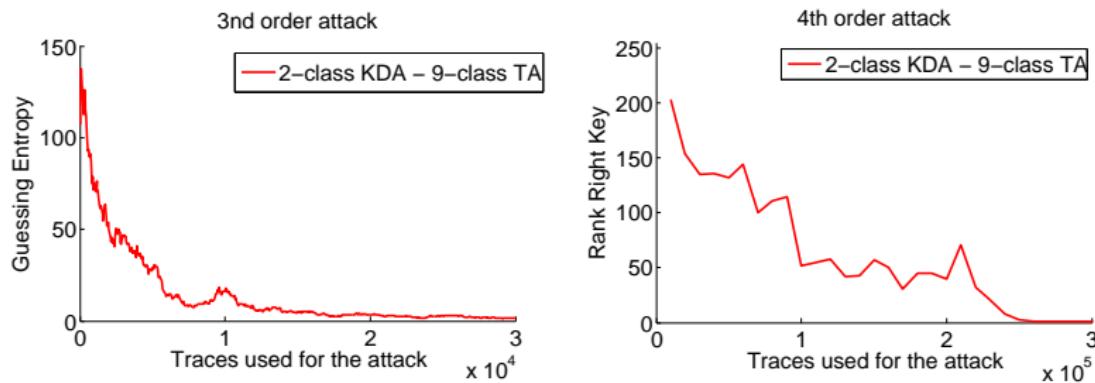


## Second and third order



GE = Guessing Entropy (mean rank of the right key candidate)

## Third and Fourth Order



- $d = 2 \rightarrow \mathcal{F} = \mathbb{R}^{\binom{200+2-1}{2}} \Rightarrow 20.100$  implicit coefficients
- $d = 3 \rightarrow \mathcal{F} = \mathbb{R}^{\binom{200+3-1}{3}} \Rightarrow 1.353.400$  implicit coefficients
- $d = 4 \rightarrow \mathcal{F} = \mathbb{R}^{\binom{200+4-1}{4}} \Rightarrow 68.685.050$  implicit coefficients

Same time of execution of the KDA algorithm!

## Conclusions on KDA

### Strong points

- ▶ KDA with  $d$ -th degree polynomial kernel function is suitable to attack  $(d - 1)$ th-order masking
- ▶ KDA computational complexity is independent from the order  $d$
- ▶ Tested and effective on a real case, positively compared to PP

	2nd order	3-rd order	4th order
KDA	✓	✓	✓
PP	✓	✗	✗

### Limits and drawbacks

- ▶ Memory-based  $[\epsilon_{\ell}^{\text{KDA}}(\vec{x}) = \sum_{i=1}^N \vec{\nu}_{\ell}[i] \mathcal{K}(x_i^{z_i}, \mathbf{x})]$

## Conclusions on KDA

### Strong points

- ▶ KDA with  $d$ -th degree polynomial kernel function is suitable to attack  $(d - 1)$ th-order masking
- ▶ KDA computational complexity is independent from the order  $d$
- ▶ Tested and effective on a real case, positively compared to PP

	2nd order	3-rd order	4th order
KDA	✓	✓	✓
PP	✓	✗	✗

### Limits and drawbacks

- ▶ Memory-based [ $\epsilon_\ell^{\text{KDA}}(\vec{x}) = \sum_{i=1}^N \vec{v}_\ell[i] K(x_i^{z_i}, \vec{x})]$ ] +  $O(N^3)$  complexity → Non-scalability to big training set

## Conclusions on KDA

### Strong points

- ▶ KDA with  $d$ -th degree polynomial kernel function is suitable to attack  $(d - 1)$ th-order masking
- ▶ KDA computational complexity is independent from the order  $d$
- ▶ Tested and effective on a real case, positively compared to PP

	2nd order	3-rd order	4th order
KDA	✓	✓	✓
PP	✓	✗	✗

### Limits and drawbacks

- ▶ Memory-based  $[\epsilon_{\ell}^{\text{KDA}}(\vec{x}) = \sum_{i=1}^N \vec{v}_{\ell}[i] \mathbf{K}(x_i^{z_i}, \vec{x})] + O(N^3)$  complexity → Non-scalability to big training set
- ▶ Regularization hyper-parameter  $\mu$ :  $\mathbf{N} = \sum_{z \in \mathcal{Z}} \mathbf{K}_z (\mathbf{I} - \mathbf{I}_{N_z}) \mathbf{K}_z^T + \mu \mathbf{I}$

## Conclusions on KDA

### Strong points

- ▶ KDA with  $d$ -th degree polynomial kernel function is suitable to attack  $(d - 1)$ th-order masking
- ▶ KDA computational complexity is independent from the order  $d$
- ▶ Tested and effective on a real case, positively compared to PP

	2nd order	3-rd order	4th order
KDA	✓	✓	✓
PP	✓	✗	✗

### Limits and drawbacks

- ▶ Memory-based  $[\epsilon_{\ell}^{\text{KDA}}(\vec{x}) = \sum_{i=1}^N \vec{\nu}_{\ell}[i] K(x_i^{z_i}, \vec{x})] + O(N^3)$  complexity → Non-scalability to big training set
- ▶ Regularization hyper-parameter  $\mu$ :  $N = \sum_{z \in \mathcal{Z}} K_z(I - I_{N_z})K_z^T + \mu I$
- ▶ Does not allow the localisation of Pol

## Contents

1. Context
2. State of the Art, Objectives, Contributions
3. Kernel Discriminant Analysis against Masking
  - 3.1 Kernel Discriminant Analysis
  - 3.2 Experimental Results
4. Deep Learning against Misalignment
  - 4.1 Data Augmentation
  - 4.2 Experimental Results
5. Conclusions

## Motivations

### Profiling phase

- ▶ manage de-synchronization problem  $[\mathcal{D}_{\text{train}} \longrightarrow \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D]$
- ▶ mandatory dimensionality reduction  $[\mathcal{D}_{\text{train}} \longrightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C]$
- ▶ estimate
  - ▶  $P_{\epsilon(\rho(\vec{x}))} | Z=z$ ,  $P_{\epsilon(\rho(\vec{x}))}$ ,  $p_Z$  (generative model)
    - ▶ Gaussian hypothesis (**Template Attack**)[CRR03]
  - ▶  $p_Z | \epsilon(\rho(\vec{x}))$  (discriminative model)

Many independent preprocessing steps and assumptions

## Motivations

### Profiling phase

### DEEP LEARNING

- ▶ manage de-synchronization problem  $[\mathcal{D}_{\text{train}} \longrightarrow \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D]$
- ▶ mandatory dimensionality reduction  $[\mathcal{D}_{\text{train}} \longrightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C]$
- ▶ estimate
  - ▶  $P_{\epsilon(\rho(\vec{x}))} | Z=z$ ,  $P_{\epsilon(\rho(\vec{x}))}$ ,  $p_Z$  (generative model)
    - ▶ Gaussian hypothesis (Template Attack)[CRR03]
  - ▶  $p_Z | \vec{x}$  (discriminative model)  
by means of a neural network  $\hat{p}(\vec{x}, W) \approx p_{Z | \vec{x}=\vec{x}}$

Many independent preprocessing steps and assumptions  
↔ integrated and agnostic approach

## Multi-Layer Perceptron

In SCA litterature [MHM13; MZ13; MMT15; MDM16]

### Multi-Layer Perceptron (MLP)

$$\hat{p}(\vec{x}, W) = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \dots \circ \lambda_1(\vec{x}) = \vec{y} \approx p_{Z \mid \vec{X}=\vec{x}}$$

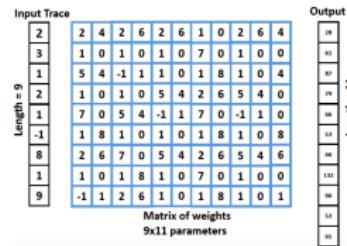
## Multi-Layer Perceptron

In SCA litterature [MHM13; MZ13; MMT15; MDM16]

### Multi-Layer Perceptron (MLP)

$$\hat{p}(\vec{x}, W) = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\vec{x}) = \vec{y} \approx p_Z \mid \vec{x} = \vec{x}$$

$\lambda_i$  linear functions (linear combinations of time samples) depending on some **trainable weights**  $W$



## Multi-Layer Perceptron

In SCA litterature [MHM13; MZ13; MMT15; MDM16]

### Multi-Layer Perceptron (MLP)

$$\hat{pdf}(\vec{x}, W) = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\vec{x}) = \vec{y} \approx p_{Z | \vec{X}=\vec{x}}$$

$\lambda_i$  linear functions (linear combinations of time samples) depending on some **trainable weights**  $W$

$\sigma_i$  non-linear *activation* functions

Input Trace		Matrix of weights 9x11 parameters											Output	
2		2	4	2	6	2	6	1	0	2	6	4	28	
3		1	0	1	0	1	0	7	0	1	0	0	63	
1		5	4	-1	1	1	0	1	8	1	0	4	87	
2		1	0	1	0	5	4	2	6	5	4	0	79	
1		7	0	5	4	-1	1	7	0	-1	1	0	66	
-1		1	8	1	0	1	0	1	8	1	0	8	53	
8		2	6	7	0	5	4	2	6	5	4	6	66	
1		1	0	1	8	1	0	7	0	1	0	0	121	
9		-1	1	2	6	1	0	1	8	1	0	1	66	

## Multi-Layer Perceptron

In SCA litterature [MHM13; MZ13; MMT15; MDM16]

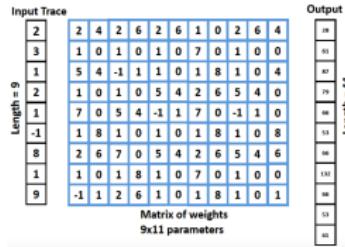
## Multi-Layer Perceptron (MLP)

$$\hat{pdf}(\vec{x}, W) = \textcolor{red}{s} \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \dots \circ \lambda_1(\vec{x}) = \vec{y} \approx p_{Z|X=\vec{x}}$$

$\lambda_i$  linear functions (linear combinations of time samples) depending on some **trainable weights**  $W$

$\sigma_i$  non-linear activation functions

**s** normalizing softmax function



## Multi-Layer Perceptron

In SCA litterature [MHM13; MZ13; MMT15; MDM16]

### Multi-Layer Perceptron (MLP)

$$\hat{pdf}(\vec{x}, W) = \textcolor{red}{s} \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\vec{x}) = \vec{y} \approx p_{Z|X=\vec{x}}$$

$\lambda_i$  linear functions (linear combinations of time samples) depending on some **trainable weights**  $W$

$\sigma_i$  non-linear *activation* functions

$\textcolor{red}{s}$  normalizing *softmax* function

Architecture hyper-parameters

Input Trace		Matrix of weights 9x11 parameters											Output	
2		2	4	2	6	2	6	1	0	2	6	4		28
3		1	0	1	0	1	0	7	0	1	0	0		63
1		5	4	-1	1	1	0	1	8	1	0	4		87
2		1	0	1	0	5	4	2	6	5	4	0		66
1		7	0	5	4	-1	1	7	0	-1	1	0		53
-1		1	8	1	0	1	0	1	8	1	0	8		66
8		2	6	7	0	5	4	2	6	5	4	6		121
1		1	0	1	8	1	0	7	0	1	0	0		66
9		-1	1	2	6	1	0	1	8	1	0	1		53

# Multi-Layer Perceptron

In SCA litterature [MHM13; MZ13; MMT15; MDM16]

## Multi-Layer Perceptron (MLP)

$$\hat{pdf}(\vec{x}, W) = \textcolor{red}{s} \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \dots \circ \lambda_1(\vec{x}) = \vec{y} \approx p_{Z|X=\vec{x}}$$

$\lambda_i$  linear functions (linear combinations of time samples) depending on some **trainable weights**  $W$

$\sigma_i$  non-linear activation functions

**s** normalizing softmax function

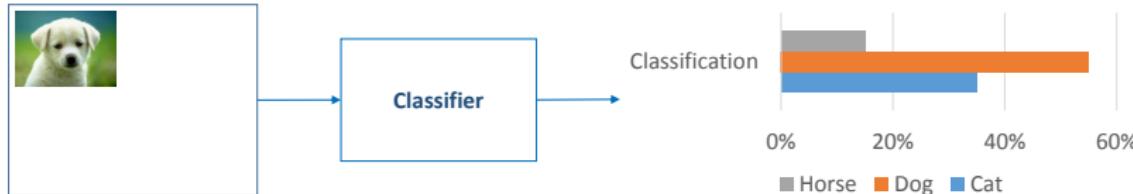
## Architecture hyper-parameters

## Universal approximation theorem

Input Trace		Matrix of weights 9x11 parameters									Output	
2		2	4	2	6	2	6	1	0	2	6	4
3		1	0	1	0	1	0	7	0	1	0	0
1		5	4	-1	1	1	0	1	8	1	0	4
2		1	0	1	0	5	4	2	6	5	4	0
1		7	0	5	4	-1	1	7	0	-1	1	0
-1		1	8	0	1	0	1	8	1	0	8	
8		2	6	7	0	5	4	2	6	5	4	6
1		1	0	1	8	1	0	7	0	1	0	0
9		-1	1	2	6	1	0	1	8	1	0	1

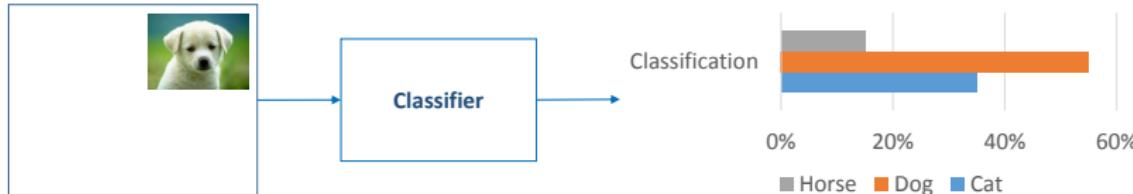
## Convolutional Neural Networks

### Translation-Invariance



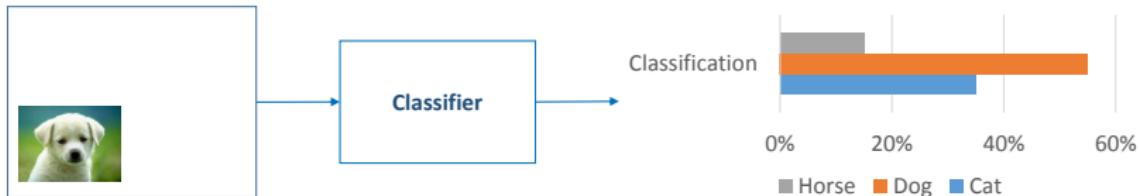
## Convolutional Neural Networks

### Translation-Invariance



## Convolutional Neural Networks

### Translation-Invariance



## Convolutional Neural Networks

### Translation-Invariance



## Convolutional Neural Networks

### Translation-Invariance



## Convolutional Neural Networks

### Translation-Invariance



## Convolutional Neural Networks

## Translation-Invariance

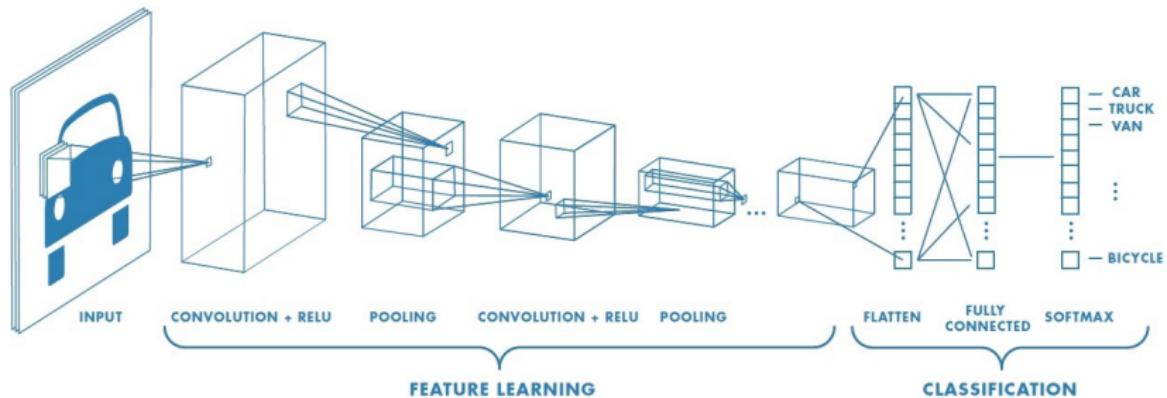
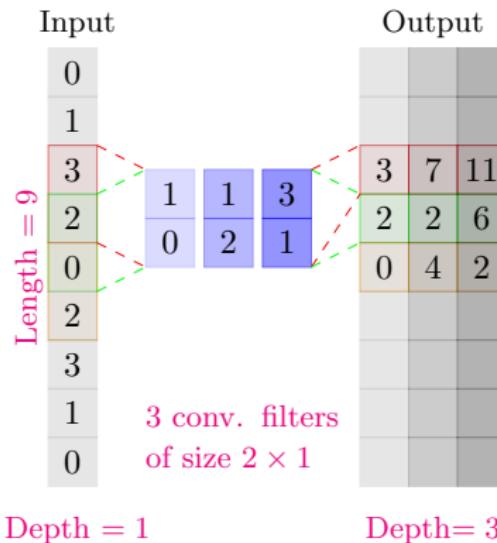
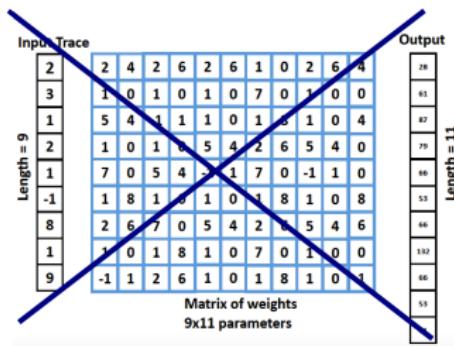


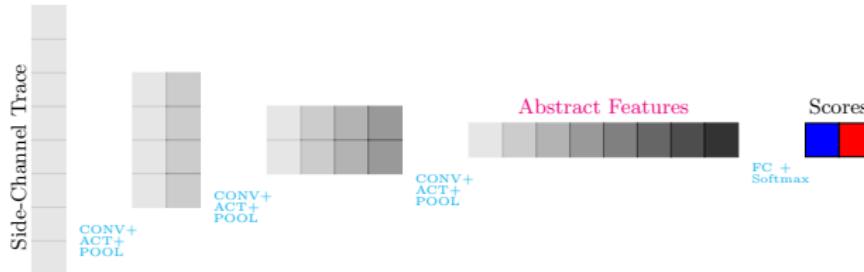
Figure: Source: medium.com

## Convolutional Layers



## A kind of CNN architecture

Temporal Features



Architecture inspired by AlexNet [KSH12], VGG [SZ14], ResNet [He+16] design rules:

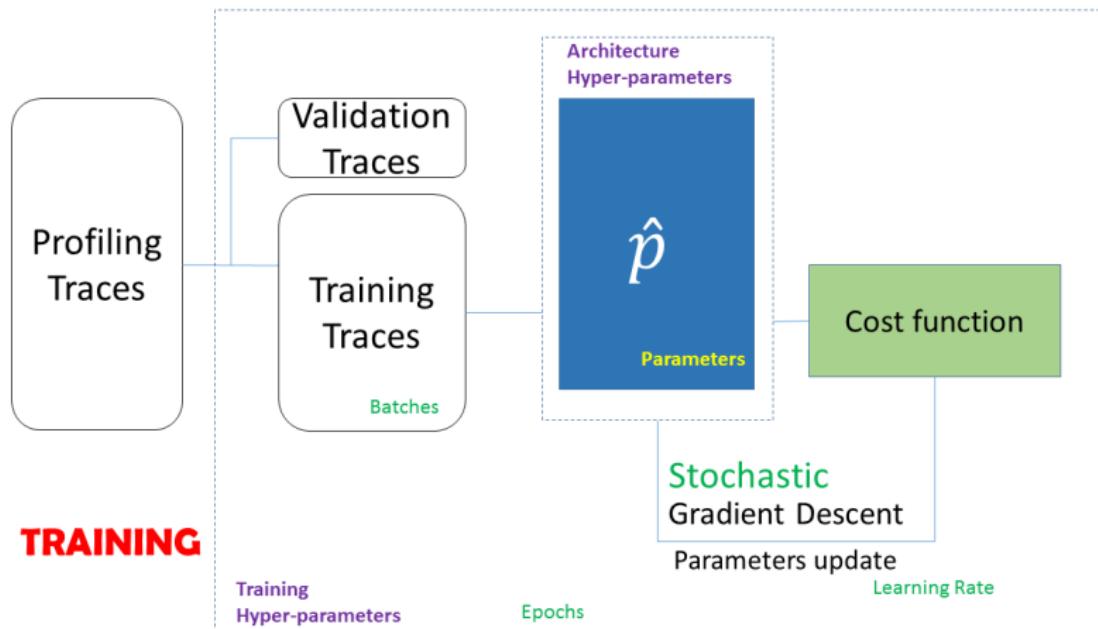
- ▶ Reduce temporal features to only one
- ▶ Maintain time complexity of each layer (one-half pooling when number of feature maps are doubled)

### Model used in our experiments

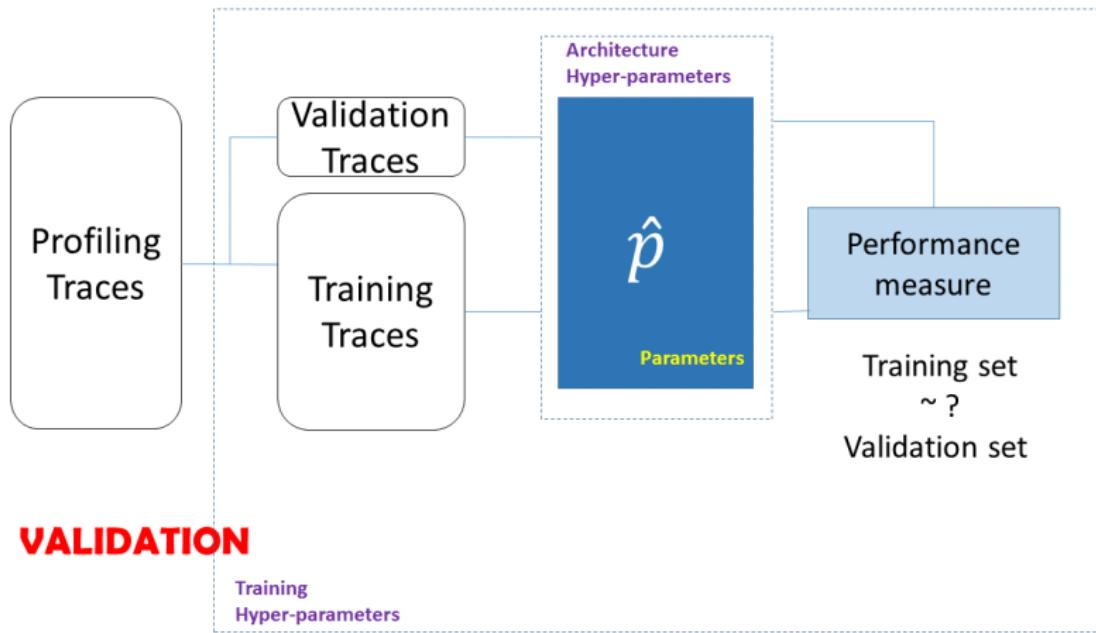
- ▶ 4 Conv + Pool layers
- ▶ tanh activations
- ▶ batch normalisation [IS15]
- ▶ 1 *fully connected layer* + softmax

## Training and Validation

## Training and Validation



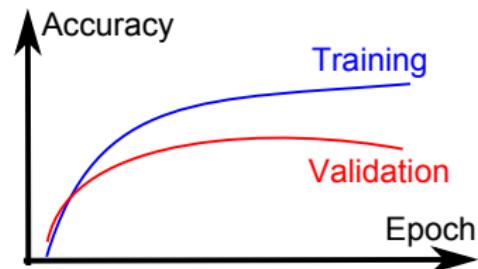
## Training and Validation



## Overfitting

Evaluate and compare training and validation accuracy

Learn by heart (**OVERRFITTING**)

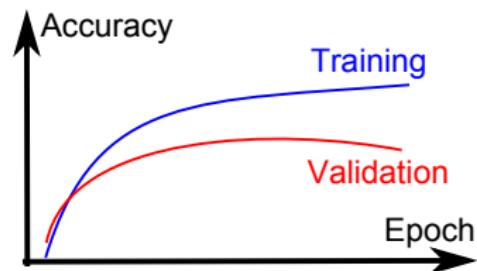
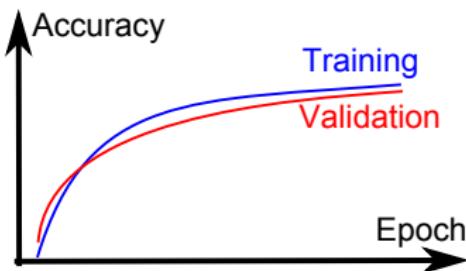


## Overfitting

Evaluate and compare training and validation accuracy

Understand significant features

Learn by heart (**OVERFITTING**)



## Overfitting

Evaluate and compare training and validation accuracy

Learn by heart (**OVERRFITTING**)

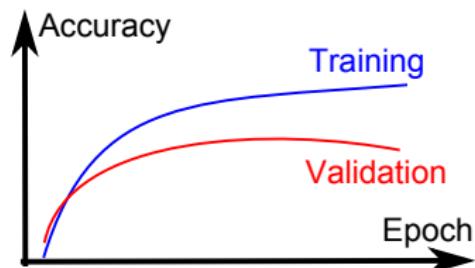
Why?

Too complex model

Not enough training data

Solution?

Data augmentation



## Data Augmentation

### Data Augmentation

Artificially generate new training data by deforming those previously acquired,  
Applying transformations that preserve the label  $Z$

### Countermeasure Emulation Idea

Emulate the effects of misaligning countermeasures to generate new traces

#### SHIFTING

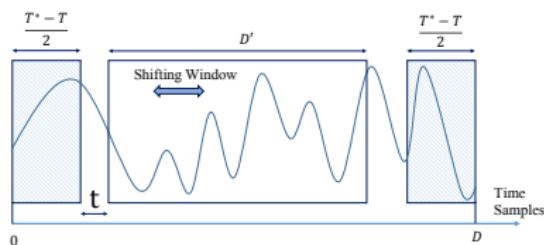


Figure:  $SH_T$

#### ADD-REMOVE

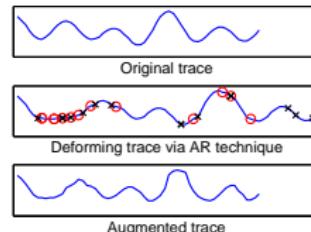


Figure:  $AR_R$

Parameter  $T$ : # of possible positions

Parameter  $R$ : # of added and removed points

Data Augmentation techniques are applied online during training phase.

## Experimental Results

- ▶ Random delays (software countermeasure)
- ▶ Artificial Jitter (simulated hardware countermeasure)
- ▶ Real Jitter (hardware countermeasure)

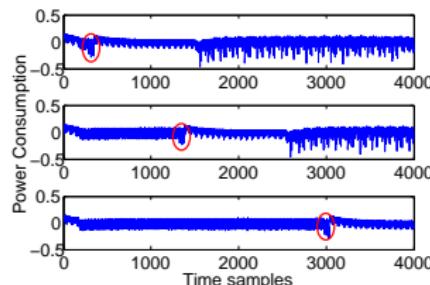
Keras 1.2.1 library with Tensorflow backend [Cho+15] (open source, today 2.2.4)

## Experimental Results

- ▶ Random delays (software countermeasure)
- ▶ Artificial Jitter (simulated hardware countermeasure)
- ▶ Real Jitter (hardware countermeasure)

Keras 1.2.1 library with Tensorflow backend [Cho+15] (open source, today 2.2.4)

## Random delays



(a) One leaking operation

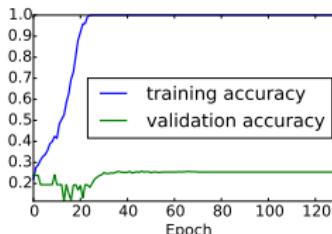
### Setup

- ▶ Target Chip: Atmega328P
- ▶ Target Variable:  $Z = \text{HW}(\text{Sbox}(P \oplus K))$
- ▶ Acquisition: through ChipWhisperer[OC14] platform,  $\approx 4,000$  time samples
- ▶ Countermeasure: Random Delays - insertion of  $r$  *nop* operations,  $r \in [0, 127]$  uniform random
- ▶ 1,000 training traces

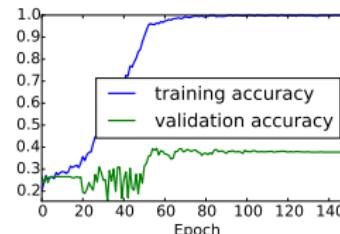
## Random delays

Data augmentation vs overfitting

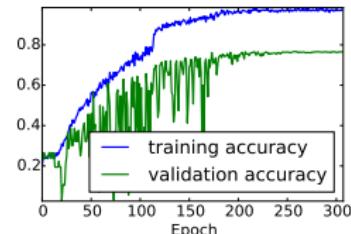
### Training



$SH_0$



$SH_{100}$

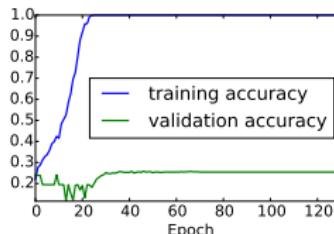


$SH_{500}$

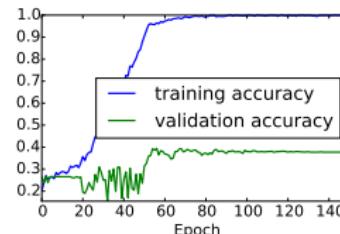
## Random delays

Data augmentation vs overfitting

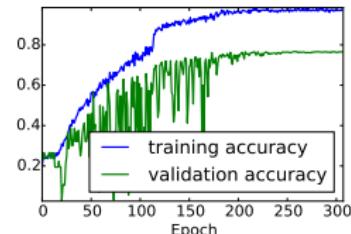
### Training



$SH_0$



$SH_{100}$



$SH_{500}$

### Attack

		$SH_0$	$SH_{100}$	$SH_{500}$		
Accuracy	$N^*$	27.0%	> 1,000	31.8%	101	78% 7

Table:  $N^*$  = number of attack traces to have GE = 1.

## Conclusions about CNN

- ▶ CNNs provide an integrated approach to construct a discriminative model from misaligned data

## Conclusions about CNN

- ▶ CNNs provide an integrated approach to construct a discriminative model from misaligned data
- ▶ CNN models may have high capacity and require plenty of data to be trained

## Conclusions about CNN

- ▶ CNNs provide an integrated approach to construct a discriminative model from misaligned data
- ▶ CNN models may have high capacity and require plenty of data to be trained
- ▶ Side-Channel-adapted Data Augmentation techniques

## Conclusions about CNN

- ▶ CNNs provide an integrated approach to construct a discriminative model from misaligned data
- ▶ CNN models may have high capacity and require plenty of data to be trained
- ▶ Side-Channel-adapted Data Augmentation techniques
- ▶ Effectiveness/efficiency of the CNN+Data Augmentation approach experimentally verified

## Contents

1. Context
2. State of the Art, Objectives, Contributions
3. Kernel Discriminant Analysis against Masking
  - 3.1 Kernel Discriminant Analysis
  - 3.2 Experimental Results
4. Deep Learning against Misalignment
  - 4.1 Data Augmentation
  - 4.2 Experimental Results
5. Conclusions

## Conclusions

- ▶ A wide part of Side-Channel literature consider leakages localised in small and known portions of signal
- ▶ In practical context, curse of dimensionality affects the potential optimality of profiling attacks
- ▶ In many domains Machine Learning solutions are used to tackle it
- ▶ Profiling attacks  $\approx$  classification task
- ▶ Generative model approach:
  - ▶ Classification-oriented techniques for dimensionality reduction
  - ▶ LDA and KDA generalization to tackle masking countermeasure
- ▶ Discriminative model approach:
  - ▶ Neural Networks, big data scalability
  - ▶ CNN to integrate resynchronization in a unique model construction process

## Conclusions

- ▶ A wide part of Side-Channel literature consider leakages localised in small and known portions of signal
- ▶ In practical context, curse of dimensionality affects the potential optimality of profiling attacks
- ▶ In many domains Machine Learning solutions are used to tackle it
- ▶ Profiling attacks  $\approx$  classification task
- ▶ Generative model approach:
  - ▶ Classification-oriented techniques for dimensionality reduction
  - ▶ LDA and KDA generalization to tackle masking countermeasure
- ▶ Discriminative model approach:
  - ▶ Neural Networks, big data scalability
  - ▶ CNN to integrate resynchronization in a unique model construction process

## Today and in the future

- ▶ ASCAD database and SCA/DL community
- ▶ From CNN to Pol, visualizing techniques
- ▶ Advanced-attack-oriented machine learning task (instead of multiple classification)
- ▶ Collision attacks  $\approx$  verification task (siamese network)

## Conclusions

- ▶ A wide part of Side-Channel literature consider leakages localised in small and known portions of signal
- ▶ In practical context, curse of dimensionality affects the potential optimality of profiling attacks
- ▶ In many domains Machine Learning solutions are used to tackle it
- ▶ Profiling attacks  $\approx$  classification task
- ▶ Generative model approach:
  - ▶ Classification-oriented techniques for dimensionality reduction
  - ▶ LDA and KDA generalization to tackle masking countermeasure
- ▶ Discriminative model approach:
  - ▶ Neural Networks, big data scalability
  - ▶ CNN to integrate resynchronization in a unique model construction process

## Today and in the future

- ▶ ASCAD database and SCA/DL community
- ▶ From CNN to Pol, visualizing techniques
- ▶ Advanced-attack-oriented machine learning task (instead of multiple classification)
- ▶ Collision attacks  $\approx$  verification task (siamese network) **Thank You!**

## References |

- [Arc+06] C. Archambeau et al. “Template Attacks in Principal Subspaces”. English. In: *Cryptographic Hardware and Embedded Systems - CHES 2006*. Ed. by Louis Goubin and Mitsuru Matsui. Vol. 4249. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 1–14. ISBN: 978-3-540-46559-1. DOI: 10.1007/11894063\_1. URL: [http://dx.doi.org/10.1007/11894063\\_1](http://dx.doi.org/10.1007/11894063_1).
- [BDP10] Martin Bär, Hermann Drexler, and Jürgen Pulkus. “Improved template attacks”. In: *COSADE2010* (2010).
- [BHW12] Lejla Batina, Jip Hogenboom, and Jasper G.J. van Woudenberg. “Getting More from PCA: First Results of Using Principal Component Analysis for Extensive Power Analysis”. English. In: *Topics in Cryptology CT-RSA 2012*. Ed. by Orr Dunkelman. Vol. 7178. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 383–397. ISBN: 978-3-642-27953-9. DOI: 10.1007/978-3-642-27954-6\_24. URL: [http://dx.doi.org/10.1007/978-3-642-27954-6\\_24](http://dx.doi.org/10.1007/978-3-642-27954-6_24).

## References II

- [Bat+11] Lejla Batina et al. "Mutual information analysis: a comprehensive study". In: *Journal of Cryptology* 24.2 (2011), pp. 269–291.
- [Bha+14] Shivam Bhasin et al. "Side-channel leakage and trace compression using normalized inter-class variance". In: *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy*. ACM. 2014, p. 7.
- [BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. "Correlation power analysis with a leakage model". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2004, pp. 16–29.
- [Bru+15] Nicolas Bruneau et al. "Less is more". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2015, pp. 22–41.

## References III

- [CRR03] Suresh Chari, JosyulaR. Rao, and Pankaj Rohatgi. "Template Attacks". English. In: *Cryptographic Hardware and Embedded Systems - CHES 2002*. Ed. by Burton S. Kaliski, Cetin K. Koc, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 13–28. ISBN: 978-3-540-00409-7. DOI: 10.1007/3-540-36400-5\_3. URL: [http://dx.doi.org/10.1007/3-540-36400-5\\_3](http://dx.doi.org/10.1007/3-540-36400-5_3).
- [Cho+15] François Chollet et al. *Keras*. <https://github.com/fchollet/keras>. 2015.
- [CK14] Omar Choudary and Markus G Kuhn. "Efficient template attacks". In: *Smart Card Research and Advanced Applications*. Springer, 2014, pp. 253–270.
- [Dur+15] François Durvaux et al. "Efficient selection of time samples for higher-order DPA with projection pursuits". In: *Constructive Side-Channel Analysis and Secure Design*. Springer, 2015, pp. 34–50.

## References IV

- [GLRP06] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. “Templates vs. stochastic methods”. In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2006, pp. 15–29.
- [Gie+08] Benedikt Gierlichs et al. “Mutual information analysis”. In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2008, pp. 426–442.
- [He+16] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

## References V

- [HZ12] Annelie Heuser and Michael Zohner. “Intelligent Machine Homicide”. English. In: *Constructive Side-Channel Analysis and Secure Design*. Ed. by Werner Schindler and SorinA. Huss. Vol. 7275. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 249–264. ISBN: 978-3-642-29911-7. DOI: 10.1007/978-3-642-29912-4\_18. URL: [http://dx.doi.org/10.1007/978-3-642-29912-4\\_18](http://dx.doi.org/10.1007/978-3-642-29912-4_18).
- [Hos+11] Gabriel Hospodar et al. “Machine learning in side-channel analysis: a first study”. English. In: *Journal of Cryptographic Engineering* 1.4 (2011), pp. 293–302. ISSN: 2190-8508. DOI: 10.1007/s13389-011-0023-x. URL: <http://dx.doi.org/10.1007/s13389-011-0023-x>.
- [IS15] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *CoRR* abs/1502.03167 (2015). URL: <http://arxiv.org/abs/1502.03167>.

## References VI

- [KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. "Differential power analysis". In: *Annual International Cryptology Conference*. Springer. 1999, pp. 388–397.
- [KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. 2012, pp. 1106–1114.
- [LBM15] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. "A machine learning approach against a masked AES". In: *Journal of Cryptographic Engineering* 5.2 (2015), pp. 123–139.
- [LBM14] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. "Power analysis attack: an approach based on machine learning". In: *International Journal of Applied Cryptography* 3.2 (2014), pp. 97–115.

## References VII

- [MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards.* Vol. 31. Springer Science & Business Media, 2008.
- [MDM16] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. “Profiling power analysis attack based on MLP in DPA contest V4. 2”. In: *Telecommunications and Signal Processing (TSP), 2016 39th International Conference on*. IEEE. 2016, pp. 223–226.
- [MHM13] Zdenek Martinasek, Jan Hajny, and Lukas Malina. “Optimization of power analysis using neural network”. In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2013, pp. 94–107.
- [MMT15] Zdenek Martinasek, Lukas Malina, and Krisztina Trasy. “Profiling power analysis attack based on multi-layer perceptron network”. In: *Computational Problems in Science and Engineering*. Springer, 2015, pp. 317–339.

## References VIII

- [MZ13] Zdenek Martinasek and Vaclav Zeman. "Innovative method of the power analysis". In: *Radioengineering* 22.2 (2013), pp. 586–594.
- [OC14] Colin O'Flynn and Zhizhang David Chen. "ChipWhisperer: An open-source platform for hardware embedded security research". In: *Constructive Side-Channel Analysis and Secure Design*. Springer, 2014, pp. 243–260.
- [Osw+06] Elisabeth Oswald et al. "Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers". English. In: *Topics in Cryptology CT-RSA 2006*. Ed. by David Pointcheval. Vol. 3860. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 192–207. ISBN: 978-3-540-31033-4. DOI: 10.1007/11605805\_13. URL: [http://dx.doi.org/10.1007/11605805\\_13](http://dx.doi.org/10.1007/11605805_13).

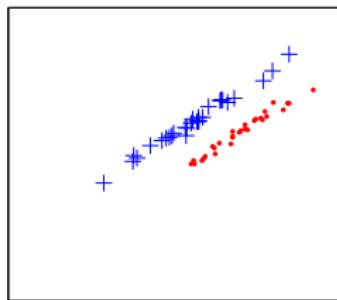
## References IX

- [SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. “A stochastic model for differential side channel cryptanalysis”. In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2005, pp. 30–46.
- [SM99] Bernhard Schölkopf and Klaus-Robert Müller. “Fisher discriminant analysis with kernels”. In: *Neural networks for signal processing IX* 1 (1999), p. 1.
- [SZ14] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

## References X

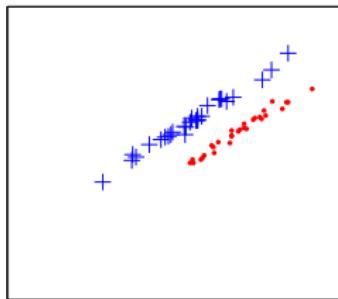
- [SA08] François-Xavier Standaert and Cedric Archambeau. "Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages". English. In: *Cryptographic Hardware and Embedded Systems CHES 2008*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 411–425. ISBN: 978-3-540-85052-6. DOI: 10.1007/978-3-540-85053-3\_26. URL: [http://dx.doi.org/10.1007/978-3-540-85053-3\\_26](http://dx.doi.org/10.1007/978-3-540-85053-3_26).
- [TM97] Mitchell T. M. *Machine Learning*. McGraw-Hill, New York, 1997.

## LDA: an optimal binary linear classifier



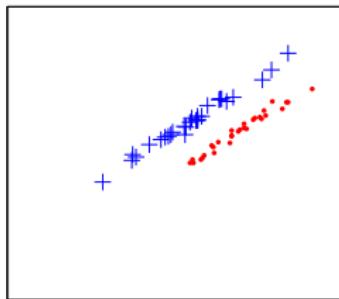
- ▶ Classify data  $\vec{x}$  into 2 classes  $\mathcal{Z} = \{s_1, s_2\}$

## LDA: an optimal binary linear classifier



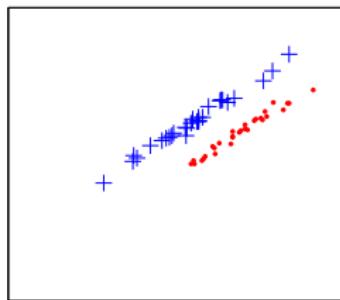
- ▶ Classify data  $\vec{x}$  into 2 classes  $\mathcal{Z} = \{s_1, s_2\}$
- ▶ Generative model:  $p_{\vec{X} | Z=s_j}(\vec{x})$ ,  $p_Z(s_j)$  and  $p_{\vec{X}}(\vec{x})$

## LDA: an optimal binary linear classifier



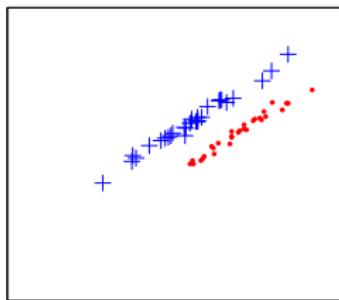
- ▶ Classify data  $\vec{x}$  into 2 classes  $\mathcal{Z} = \{s_1, s_2\}$
- ▶ Generative model:  $p_{\vec{X} | Z=s_j}(\vec{x})$ ,  $p_Z(s_j)$  and  $p_{\vec{X}}(\vec{x})$
- ▶ Posterior probabilities (via Bayes' theorem), then classify through the *log-likelihood ratio*:  $a = \log \left[ \frac{\Pr(s_1 | \vec{x})}{\Pr(s_2 | \vec{x})} \right]$   
(boundary surface  $a = 0$ )

## LDA: an optimal binary linear classifier



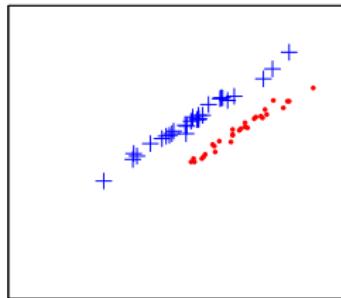
- ▶ Classify data  $\vec{x}$  into 2 classes  $\mathcal{Z} = \{s_1, s_2\}$
- ▶ Generative model:  $p_{\vec{X} | Z=s_j}(\vec{x})$ ,  $p_Z(s_j)$  and  $p_{\vec{X}}(\vec{x})$
- ▶ Posterior probabilities (via Bayes' theorem), then classify through the *log-likelihood ratio*:  $a = \log \left[ \frac{\Pr(s_1 | \vec{x})}{\Pr(s_2 | \vec{x})} \right]$  (boundary surface  $a = 0$ )
- ▶ Two assumptions about class-conditional densities:
  - ▶ Gaussian distributions with parameters  $\mu_j, \Sigma_j$
  - ▶ Homoscedasticity:  $\Sigma_j = \Sigma$  for all  $j$

## LDA: an optimal binary linear classifier



- ▶ Classify data  $\vec{x}$  into 2 classes  $\mathcal{Z} = \{s_1, s_2\}$
- ▶ Generative model:  $p_{\vec{X} | Z=s_j}(\vec{x})$ ,  $p_Z(s_j)$  and  $p_{\vec{X}}(\vec{x})$
- ▶ Posterior probabilities (via Bayes' theorem), then classify through the *log-likelihood ratio*:  $a = \log \left[ \frac{\Pr(s_1 | \vec{x})}{\Pr(s_2 | \vec{x})} \right]$  (boundary surface  $a = 0$ )
- ▶ Two assumptions about class-conditional densities:
  - ▶ Gaussian distributions with parameters  $\mu_j, \Sigma_j$
  - ▶ Homoscedasticity:  $\Sigma_j = \Sigma$  for all  $j$
- ▶  $\Rightarrow a = \vec{w}^T \vec{x} + w_0$  (linear decision boundary,  $\vec{w}$  and  $w_0$  functions of  $\Sigma, \mu_j$ )

## LDA: an optimal binary linear classifier

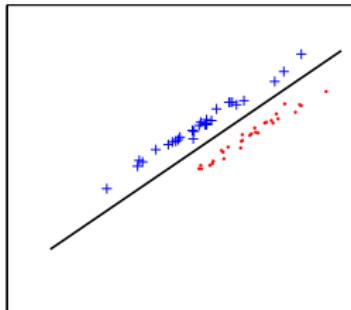


- ▶ Classify data  $\vec{x}$  into 2 classes  $\mathcal{Z} = \{s_1, s_2\}$
- ▶ Generative model:  $p_{\vec{X} | Z=s_j}(\vec{x})$ ,  $p_Z(s_j)$  and  $p_{\vec{X}}(\vec{x})$
- ▶ Posterior probabilities (via Bayes' theorem), then classify through the *log-likelihood ratio*:  $a = \log \left[ \frac{\Pr(s_1 | \vec{x})}{\Pr(s_2 | \vec{x})} \right]$  (boundary surface  $a = 0$ )
- ▶ Two assumptions about class-conditional densities:
  - ▶ Gaussian distributions with parameters  $\mu_j, \Sigma_j$
  - ▶ Homoscedasticity:  $\Sigma_j = \Sigma$  for all  $j$
- ▶  $\Rightarrow a = \vec{w}^\top \vec{x} + w_0$  (linear decision boundary,  $\vec{w}$  and  $w_0$  functions of  $\Sigma, \mu_j$ )

### Generalised linear discriminative model

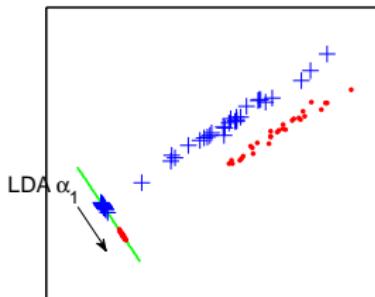
$$\Pr(s_1 | \vec{x}) = \sigma(\vec{w}^\top \vec{x} + w_0) \text{ , where } \sigma(a) = \frac{1}{1 + e^{-a}} \text{ logistic sigmoid} \quad (2)$$

## LDA and Fisher Criterion



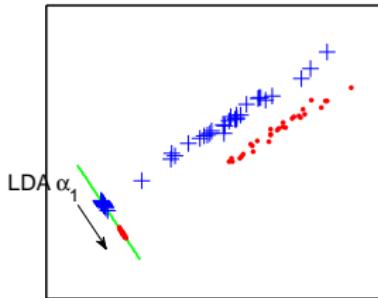
- ▶ LDA: linear decision boundary  
 $a = \vec{w}^T \vec{x} + w_0$

## LDA and Fisher Criterion



- ▶ LDA: linear decision boundary  
 $a = \vec{w}^T \vec{x} + w_0$
- ▶ Equivalently, project data onto  $\vec{w}^T \vec{x}$  (orthogonally to the decision boundary), than classify by a real threshold (optimally  $w_0$ ).

## LDA and Fisher Criterion

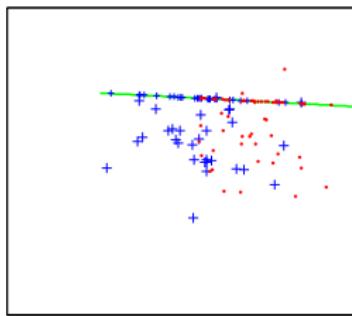


- ▶ LDA: linear decision boundary  
 $a = \vec{w}^T \vec{x} + w_0$
- ▶ Equivalently, project data onto  $\vec{w}^T \vec{x}$  (orthogonally to the decision boundary), than classify by a real threshold (optimally  $w_0$ ).
- ▶ Two assumptions about class-conditional densities:
  - ▶ Gaussian distributions with parameters  $\mu_j, \Sigma_j$
  - ▶ Homoscedasticity:  $\Sigma_j = \Sigma$  for all  $j$

### Fact, abuse and preference for the dimensionality reduction formulation

- ▶ When LDA assumptions are met, the solution  $\vec{\alpha}_1$  of the Fisher's criterion is orthogonal to  $\vec{w}$ .
- ▶ assumption not required
- ▶ naturally multi-class
- ▶ continuation in the next slide: dimensionality reduction for template attack

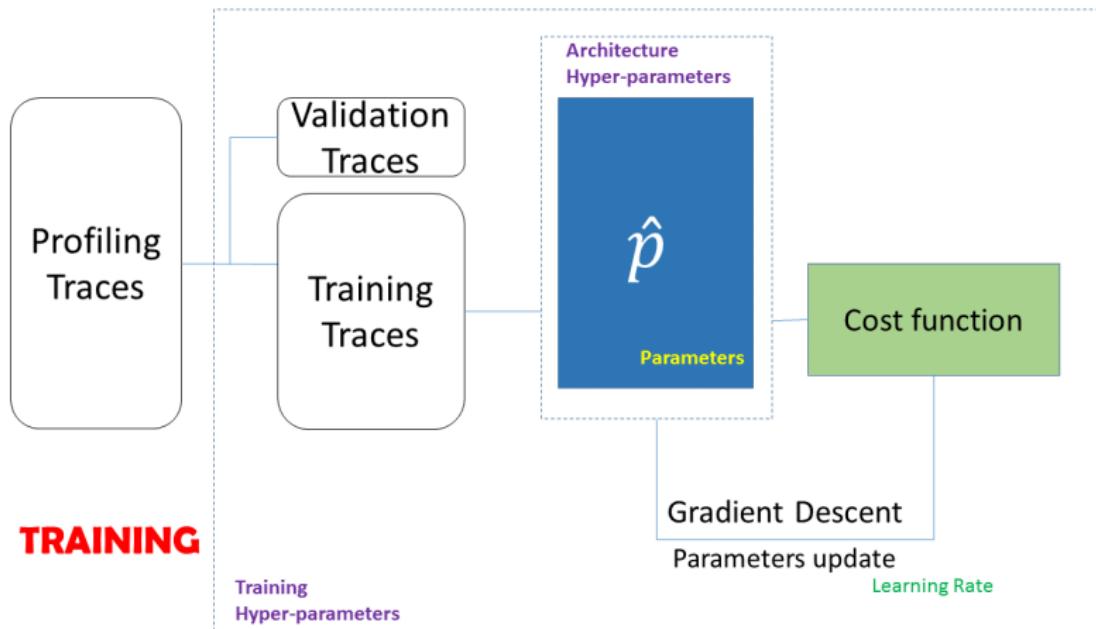
## Linear separability



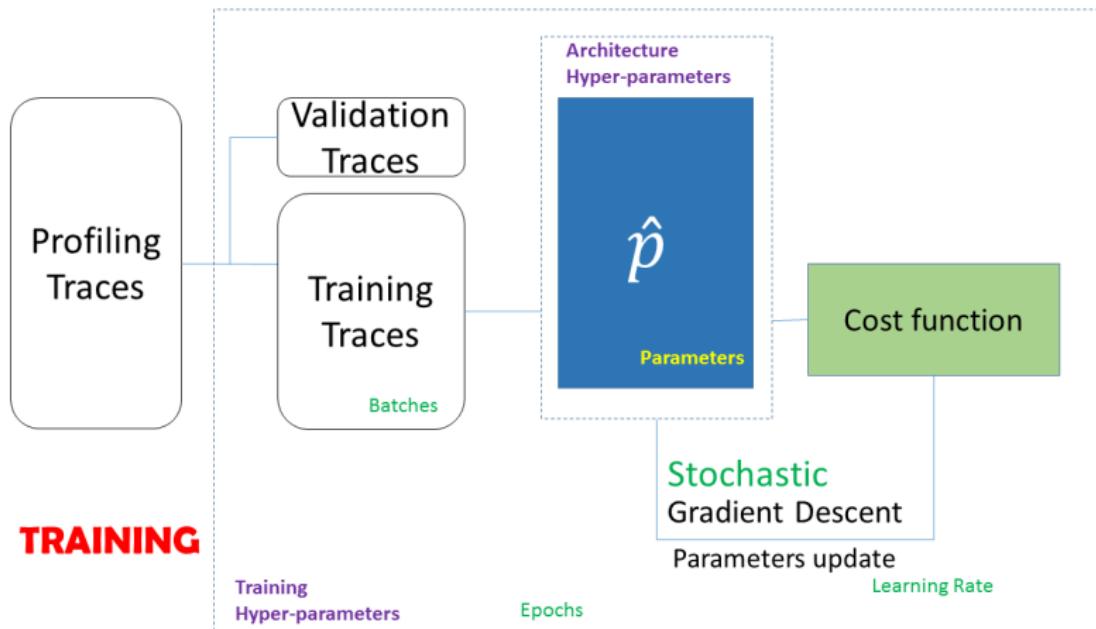
LDA: linear decision boundary  $a = \vec{w}^\top \vec{x} + w_0$  ( $\vec{w} = \Sigma^{-1}(\mu_1 - \mu_2)$ )

What if  $\mu_1 = \mu_2$ ?

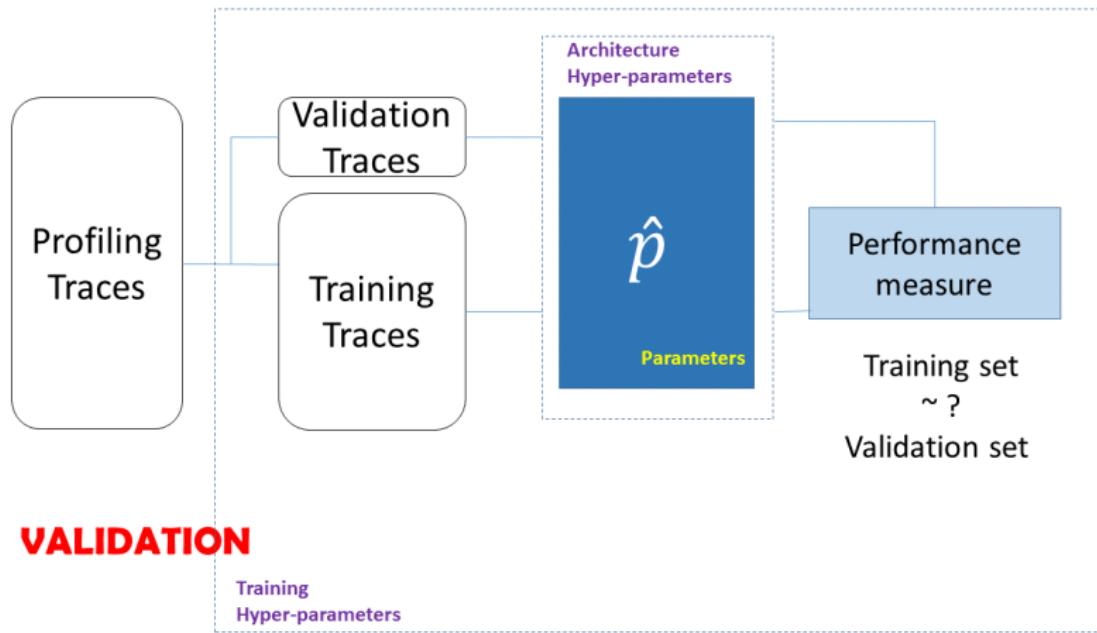
## Training-Validation-Test



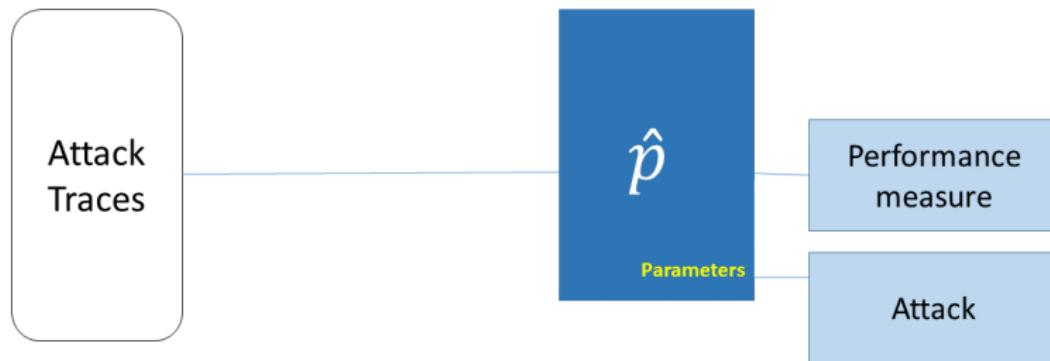
## Training-Validation-Test



## Training-Validation-Test

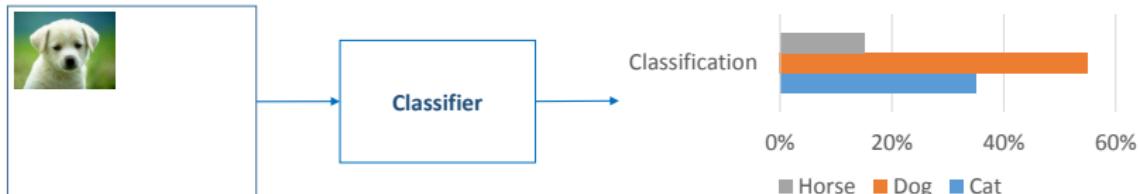


## Training-Validation-Test

**TEST**

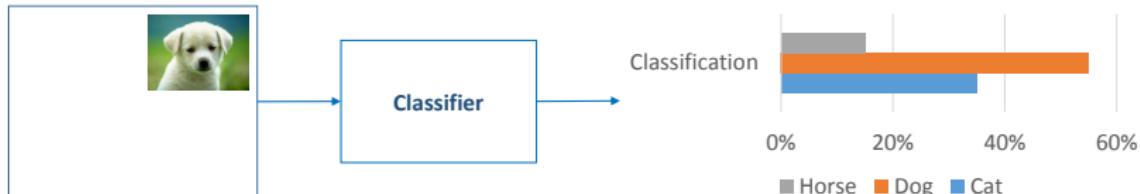
## Convolutional Neural Networks

### Translation-invariance



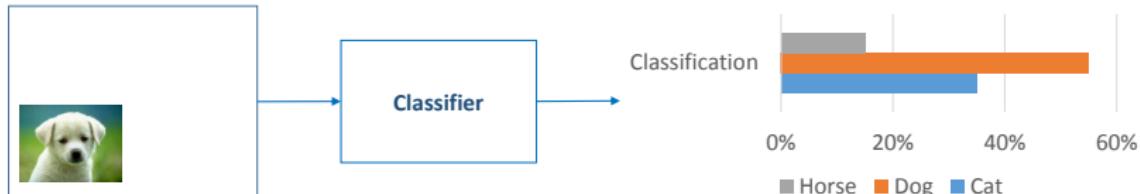
## Convolutional Neural Networks

### Translation-invariance



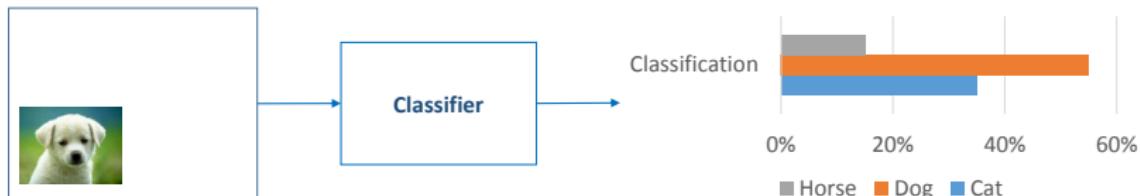
## Convolutional Neural Networks

### Translation-invariance



## Convolutional Neural Networks

### Translation-invariance



It is important to explicit the data translation-invariance

## Convolutional Neural Networks

### Translation-invariance



It is important to explicit the data translation-invariance

## Convolutional Neural Networks

### Translation-invariance



It is important to explicit the data translation-invariance

## Convolutional Neural Networks

### Translation-invariance



It is important to explicit the data translation-invariance

## Convolutional Neural Networks

### Translation-invariance



It is important to explicit the data translation-invariance

## Convolutional Neural Networks

### Translation-invariance



It is important to explicit the data translation-invariance  
Convolutional Neural Networks: share weights across space

## Convolutional Neural Networks

### Translation-invariance



It is important to explicit the data translation-invariance  
Convolutional Neural Networks: share weights across space

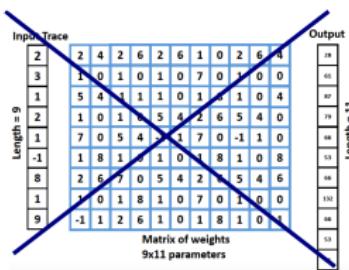


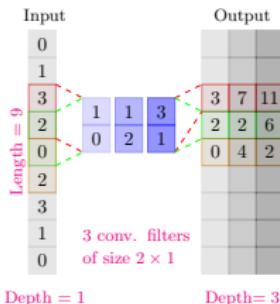
Figure: Linear layer of a CNN. <https://127.0.0.1:5007/> | Full Example | 57/42

## Convolutional Neural Networks

## Translation-invariance



It is important to explicit the data translation-invariance  
Convolutional Neural Networks: share weights across space



**Figure:** Linear layer in a ConvNet (*Convolutional*)

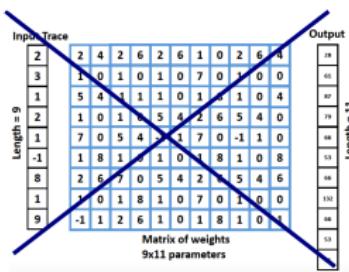


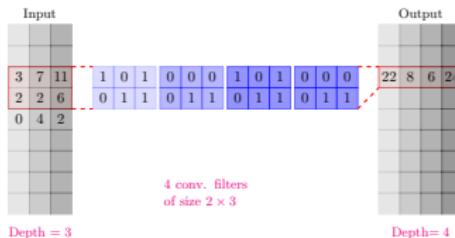
Figure: Linear layer in an MLP (Fully Connected) | 57 / 42

## Convolutional Neural Networks

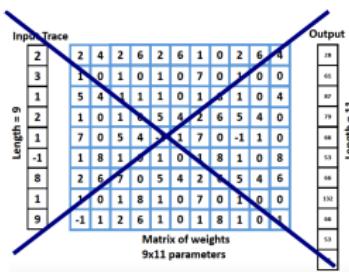
### Translation-invariance



It is important to explicit the data translation-invariance  
 Convolutional Neural Networks: share weights across space



**Figure:** Linear layer in a ConvNet (*Convolutional Layer*)



**Figure:** Linear layer in a ConvNet (*Matrix of weights*)

## Convolutional Neural Networks

### Translation-invariance



It is important to explicit the data translation-invariance  
Convolutional Neural Networks: share weights across space

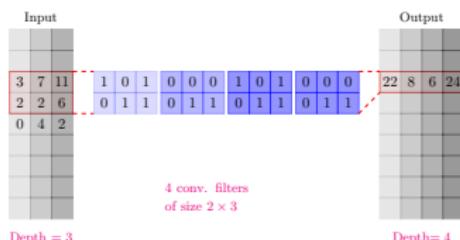


Figure: Linear layer in a ConvNet (*Convolutional Layer*)

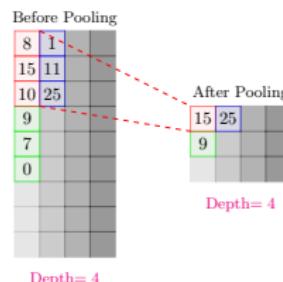


Figure: Max Pooling Layer

## Cost function - Cross-entropy

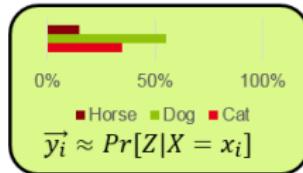
- ▶ batch of training data  $(\vec{x}_i, z_i)_{i \in I}$ , outputs of the current model  $(\vec{y}_i)_{i \in I}$
- ▶ labels  $z_i = s_j$  are *one-hot encoded*:  $\vec{z}_i = \vec{s}_j = (0, \dots, 0, \underbrace{1}_j, 0, \dots, 0)$

## Loss function

$$\mathcal{L} = -\frac{1}{|I|} \sum_{i \in I} \sum_{t=1}^{|Z|} \vec{z}_i[t] \log \vec{y}_i[t] \quad (3)$$

## Maximum-*a-posteriori* or Cross-entropy

- ▶  $\vec{y}_i \approx \Pr[Z \mid \vec{X} = \vec{x}_i]$



## Cost function - Cross-entropy

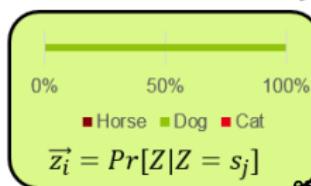
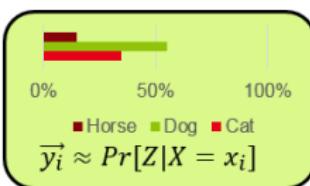
- ▶ batch of training data  $(\vec{x}_i, z_i)_{i \in I}$ , outputs of the current model  $(\vec{y}_i)_{i \in I}$
- ▶ labels  $z_i = s_j$  are *one-hot encoded*:  $\vec{z}_i = \vec{s}_j = (0, \dots, 0, \underbrace{1}_j, 0, \dots, 0)$

## Loss function

$$\mathcal{L} = -\frac{1}{|I|} \sum_{i \in I} \sum_{t=1}^{|Z|} \vec{z}_i[t] \log \vec{y}_i[t] \quad (3)$$

## Maximum-a-posteriori or Cross-entropy

- ▶  $\vec{y}_i \approx \Pr[Z | \vec{X} = \vec{x}_i]$
- ▶  $\vec{z}_i \approx \Pr[Z | Z = \vec{s}_j]$
- ▶  $\mathbb{H}(\vec{z}_i, \vec{y}_i) = \mathbb{H}(\vec{z}_i) + D_{KL}(\vec{z}_i || \vec{y}_i) = \mathbb{E}_{\vec{z}_i}[-\log \vec{y}_i] = -\sum_{t=1}^{|Z|} \vec{z}_i[t] \log \vec{y}_i[t]$



## Capacity-Overfitting-Regularization

### Regression example

Performance metric: Mean Square Error (MSE)

MSE\_train=44.228280, MSE\_test=330.984916

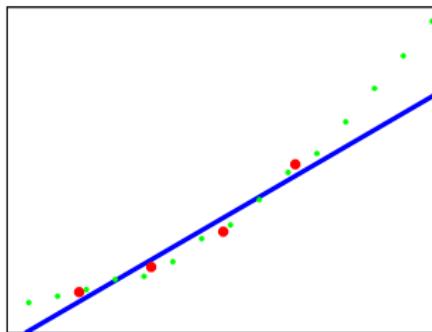


Figure: Linear regression → underfitting

## Capacity-Overfitting-Regularization

### Regression example

Performance metric: Mean Square Error (MSE)

MSE\_train=44.228280, MSE\_test=330.984916

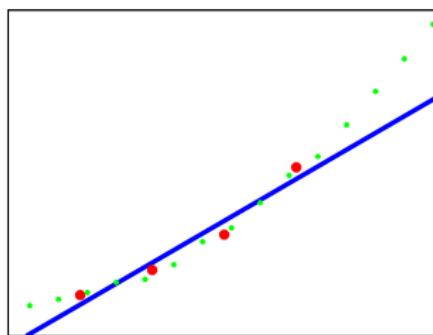


Figure: Linear regression → underfitting

MSE\_train=2.243097, MSE\_test=61.891672

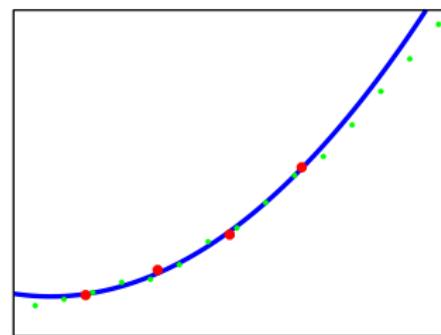


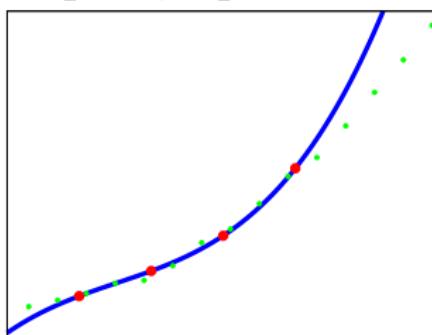
Figure: Quadratic regression → fits

## Capacity-Overfitting-Regularization

### Regression example

Performance metric: Mean Square Error (MSE)

MSE\_train=0, MSE\_test=970.081580



MSE\_train=2.243097, MSE\_test=61.891672

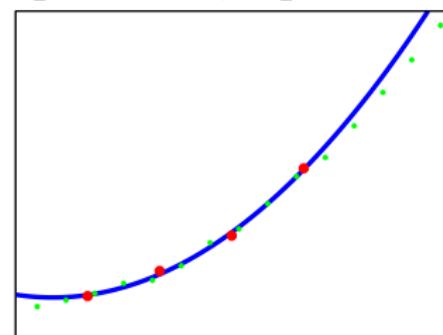


Figure: Cubic regression → overfitting

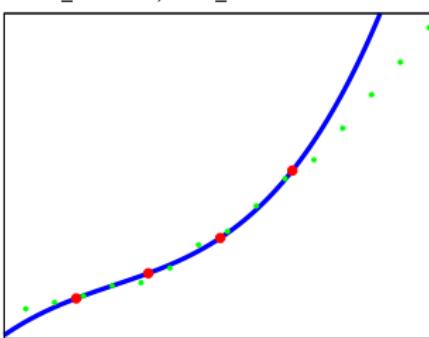
Figure: Quadratic regression → fits

## Capacity-Overfitting-Regularization

### Regression example

Performance metric: Mean Square Error (MSE)

MSE\_train=0, MSE\_test=970.081580



MSE\_train=3.040333, MSE\_test=58.377719

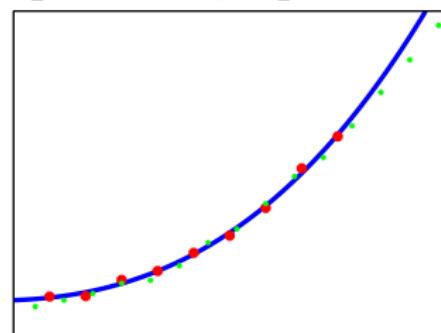


Figure: Cubic regression → overfitting

Figure: Cubic regression with more training data

## Capacity-Overfitting-Regularization

### Regression example

Performance metric: Mean Square Error (MSE)

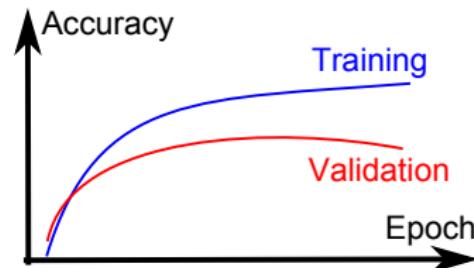
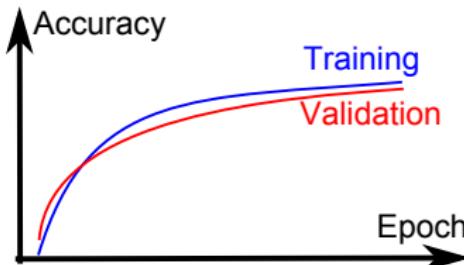
### Classification via Neural Network

Performance measure: Accuracy (Classification rate)

Evaluate and compare training and validation accuracy

Understand significant features

Learn by heart (**OVERTFITTING**)



## Capacity-Overfitting-Regularization

### Regression example

Performance metric: Mean Square Error (MSE)

### Classification via Neural Network

Performance measure: Accuracy (Classification rate)

Evaluate and compare training and validation accuracy

Learn by heart (**OVERTFITTING**)

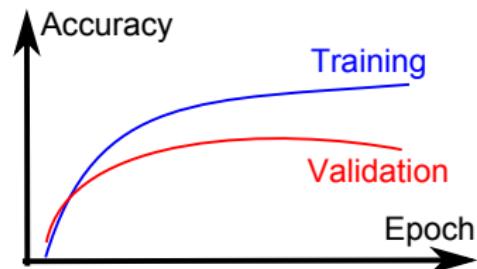
Why?

Too complex model

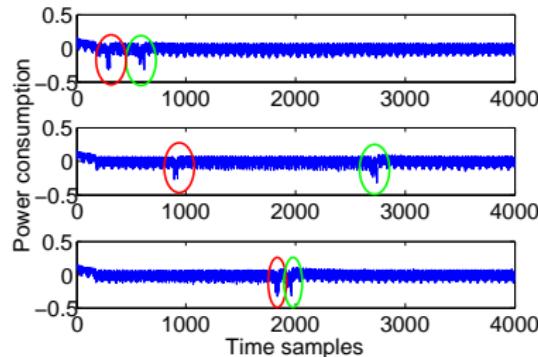
Not enough training data

Solution?

Data augmentation



## Random Delays - Two Leaking Operations

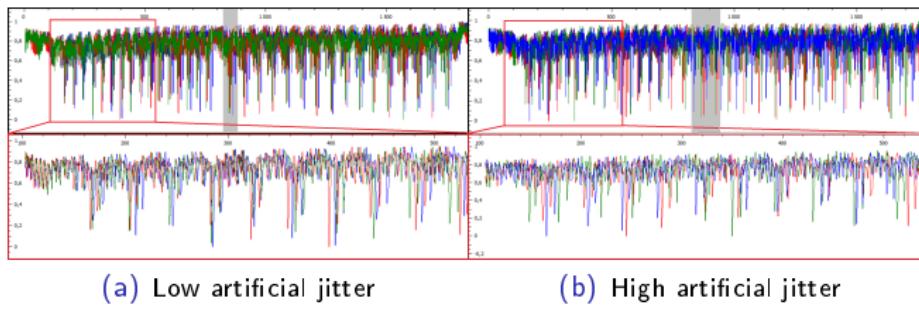


### Two leaking operations

First operation - Test acc: 76.8%,  $N^* = 7$

Second operation - Test acc: 82.5%,  $N^* = 6$

## Artificial Jitter



## Target

- ▶ Target Variable:  $Z = \text{HW}(\text{Sbox}(P \oplus K))$
- ▶  $\approx 2000$  time samples
- ▶ Countermeasure: artificial signal treatment simulating clock jitter
- ▶ 10000 training traces

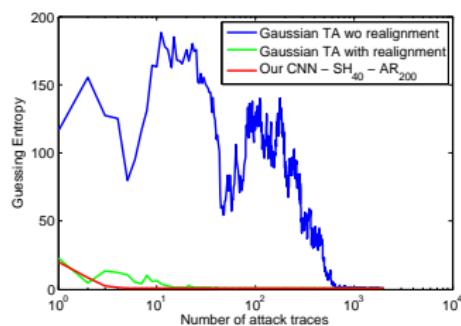
## Artificial Jitter (2)

*Low\_jitter*

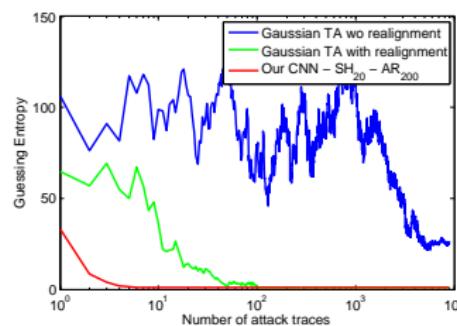
Acc	$N^*$	$SH_0$	$SH_{20}$	$SH_{40}$	
AR <sub>0</sub>		57.4%	14	82.5%	6
AR <sub>100</sub>		86.0%	6	87.0%	5
AR <sub>200</sub>		86.6%	6	85.7%	6

*High\_jitter*

Acc	$N^*$	$SH_0$	$SH_{20}$	$SH_{40}$	
AR <sub>0</sub>		40.6%	35	51.1%	9
AR <sub>100</sub>		50.2%	15	72.4%	11
AR <sub>200</sub>		64.0%	11	75.5%	8



(c) Low Jitter



(d) High Jitter

## Artificial Jitter

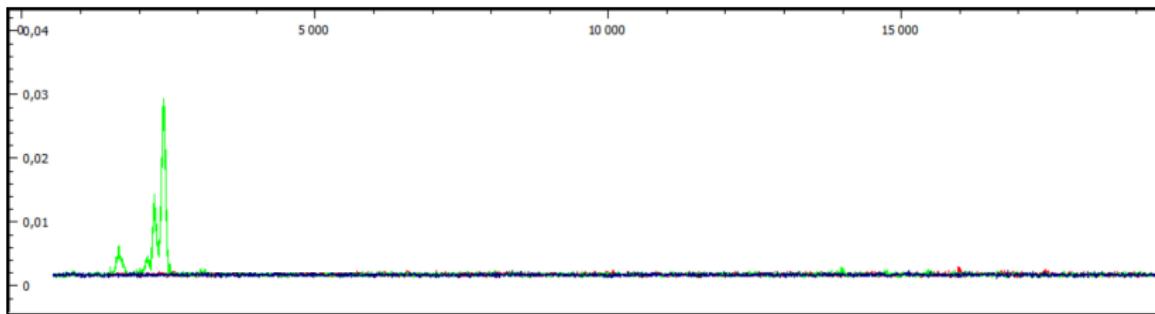
<i>DS_low_jitter</i>		SH <sub>0</sub>		SH <sub>20</sub>		SH <sub>40</sub>		SH <sub>200</sub>	
<i>a</i>	<i>b</i>								
<i>c</i>	<i>d</i>								
AR <sub>0</sub>	100.0%	68.7%	99.8%	86.1%	98.9%	84.1%			
	57.4%	14	82.5%	6	83.6%	6			
AR <sub>100</sub>	87.7%	88.2%	82.4%	88.4%	81.9%	89.6%			
	86.0%	6	87.0%	5	87.5%	6			
AR <sub>200</sub>	83.2%	88.6%	81.4%	86.9%	80.6%	88.9%			
	86.6%	6	85.7%	6	87.7%	5			
AR <sub>500</sub>							85.0%	88.6%	
							86.2%	5	
<i>DS_high_jitter</i>		SH <sub>0</sub>		SH <sub>20</sub>		SH <sub>40</sub>		SH <sub>200</sub>	
<i>a</i>	<i>b</i>								
<i>c</i>	<i>d</i>								
AR <sub>0</sub>	100%	45.0%	100%	60.0%	98.5%	67.6%			
	40.6%	35	51.1%	9	62.4%	11			
AR <sub>100</sub>	90.4%	57.3%	76.6%	73.6%	78.5%	76.4%			
	50.2%	15	72.4%	11	73.5%	9			
AR <sub>200</sub>	83.1%	67.7%	82.0%	77.1%	82.6%	77.0%			
	64.0%	11	75.5%	8	74.4%	8			
AR <sub>500</sub>							83.6%	73.4%	
							68.2%	11	

## Real Jitter (1)

### Target

- ▶ AES hardware implementation
- ▶ strong jitter effect
- ▶ Target Variable:  $Z = \text{Sbox}(P \oplus K)$
- ▶ 2,500 selected time samples
- ▶ 99,000 training traces

SNR first Sbox

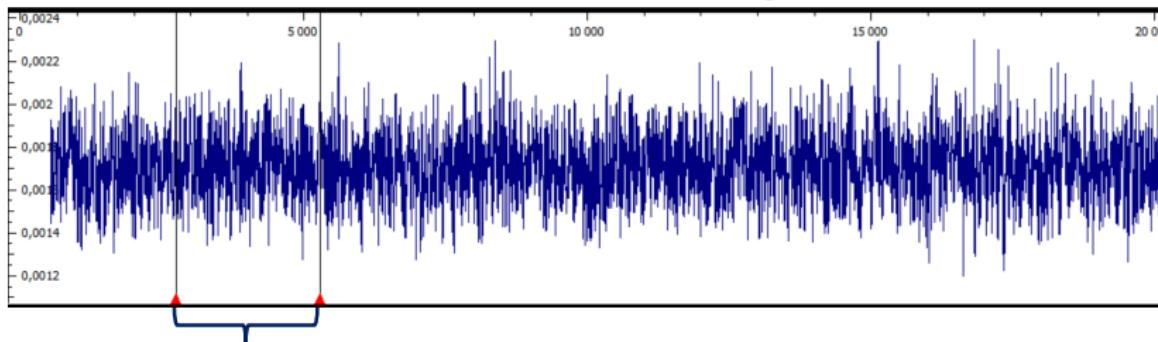


## Real Jitter (1)

### Target

- ▶ AES hardware implementation
- ▶ strong jitter effect
- ▶ Target Variable:  $Z = \text{Sbox}(P \oplus K)$
- ▶ 2,500 selected time samples
- ▶ 99,000 training traces

SNR second Sbox without realignment



Entry region for CNN (2,500 pts)

## Real Jitter (2)

		$SH_0 AR_0$	$SH_{10} AR_{100}$	$SH_{20} AR_{200}$		
Acc	$N^*$	1.2%	137	1.3%	89	1.8%
						54

## Real Jitter (2)

		SH <sub>0</sub> AR <sub>0</sub>	SH <sub>10</sub> AR <sub>100</sub>	SH <sub>20</sub> AR <sub>200</sub>
Acc	N*	1.2%	137	1.3%
			89	1.8% 54

