Deep Stacking Ensemble Learning applied to Profiling Side-Channel Attacks

Dorian LLAVATA[1,2], Eleonora CAGLI[1], Rémi EYRAUD[2], Vincent GROSSO[2], Lilian BOSSUET[2]

[1] Univ. Grenoble Alpes, F-38000, Grenoble, France, CEA, LETI, MINATEC Campus, F-38054 Grenoble, France.
{dorian.llavata,eleonora.cagli}@cea.fr

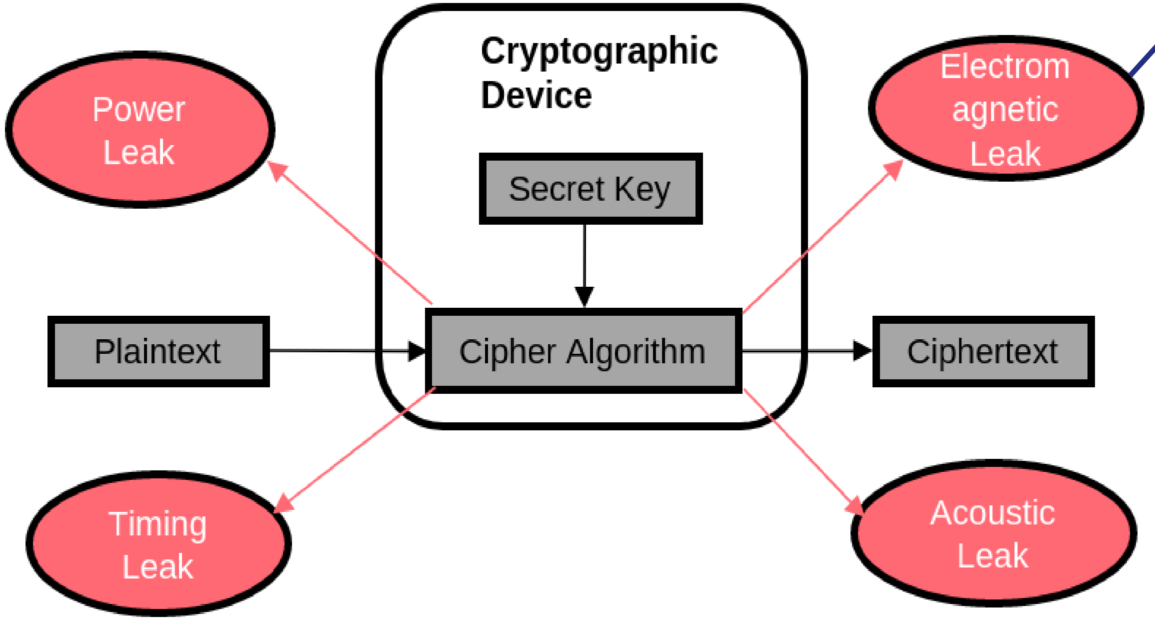
[2] Univ. Jean Monnet Saint-Etienne, CNRS, Institut d'Optique Graduate School, Lab. Hubert Curien UMR 5516, F-42023, SAINT-ETIENNE, France.
{remi.eyraud,vincent.grosso,lilian.bossuet}@univ-st-etienne.fr

1. Context

Side-Channel Attacks and Ensemble Learning

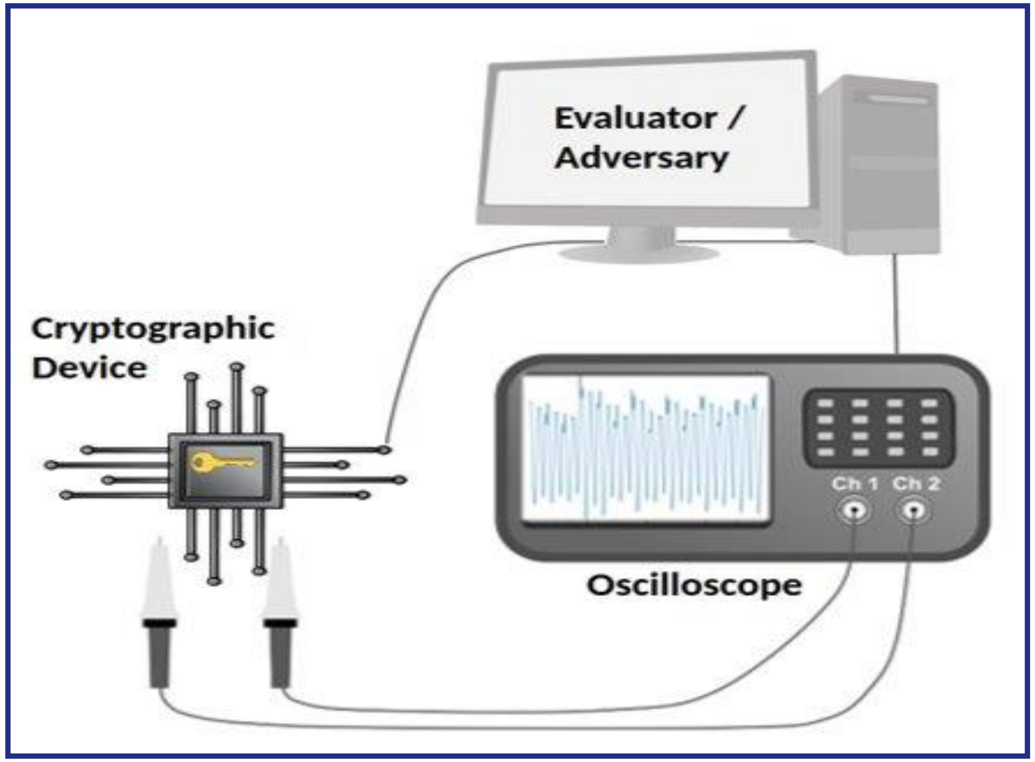
What is Side-Channel Attacks (SCA)?

Physical attacks by observation



Acquisition of leakage traces

E.g: collecting EM traces



Objective: find the secret by exploiting physical leaks

Deep Learning for Profiling SCA

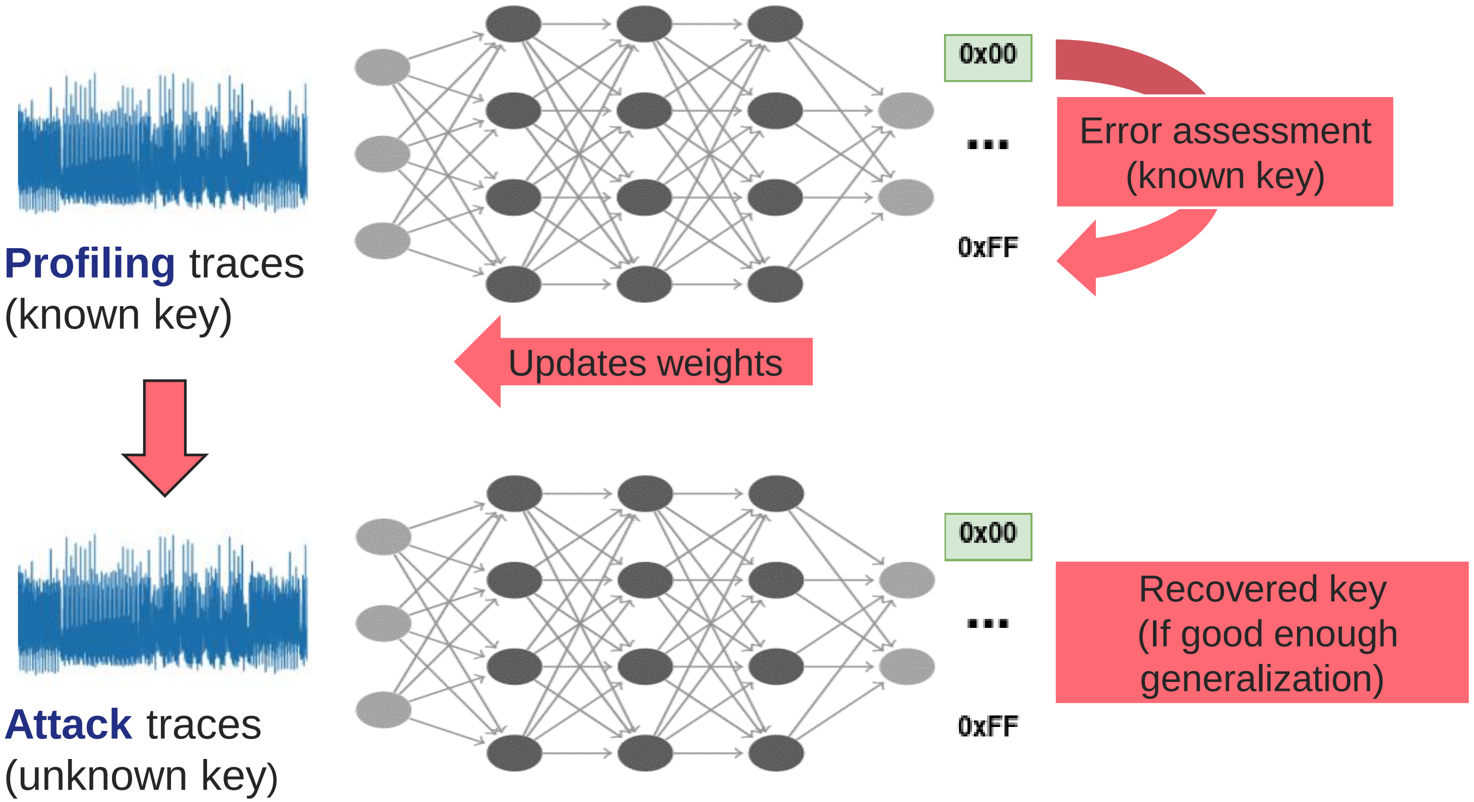
Profiling attack :

1) Profiling phase : Characterization of the leakage on a clone of the target (in a supervised manner)
2) Attack phase : Using the profiled leakage model to attack the real target device

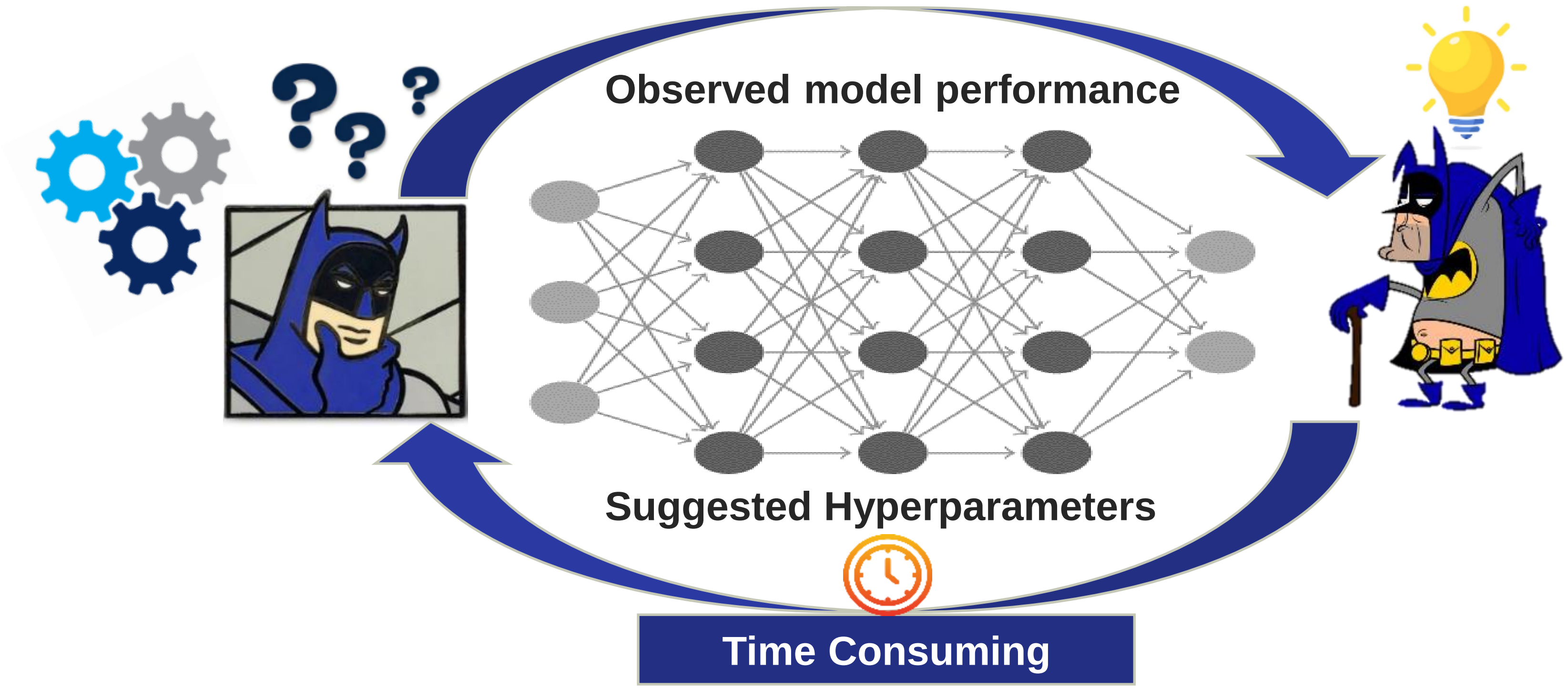
Deep learning is nowadays widely used to perform profiling SCA

Training
Profiling on the clone device

Inference
Attack on the target device



Hyperparameterization effort



A wide range of hyperparameters to set :

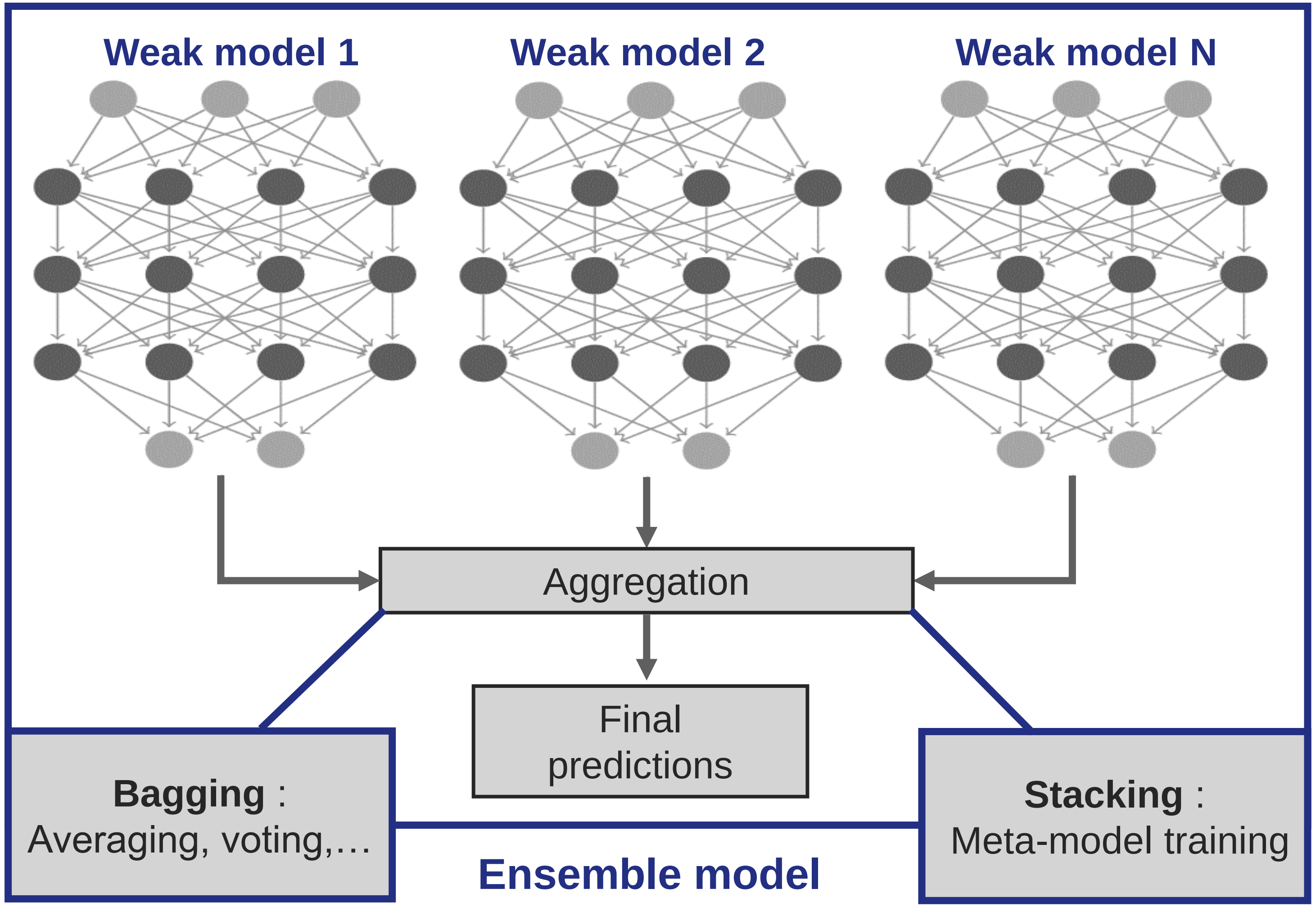
Related to architecture design : number of layers, type of layers, number of neurons, activation function,…

Related to training process **:** batch size, loss function, optimizer, number of epochs,…

➢**The more complex the architecture, the greater the hyperparameterization effort required**

Ensemble Learning

Reduce hyperparameterization effort / improve generalization



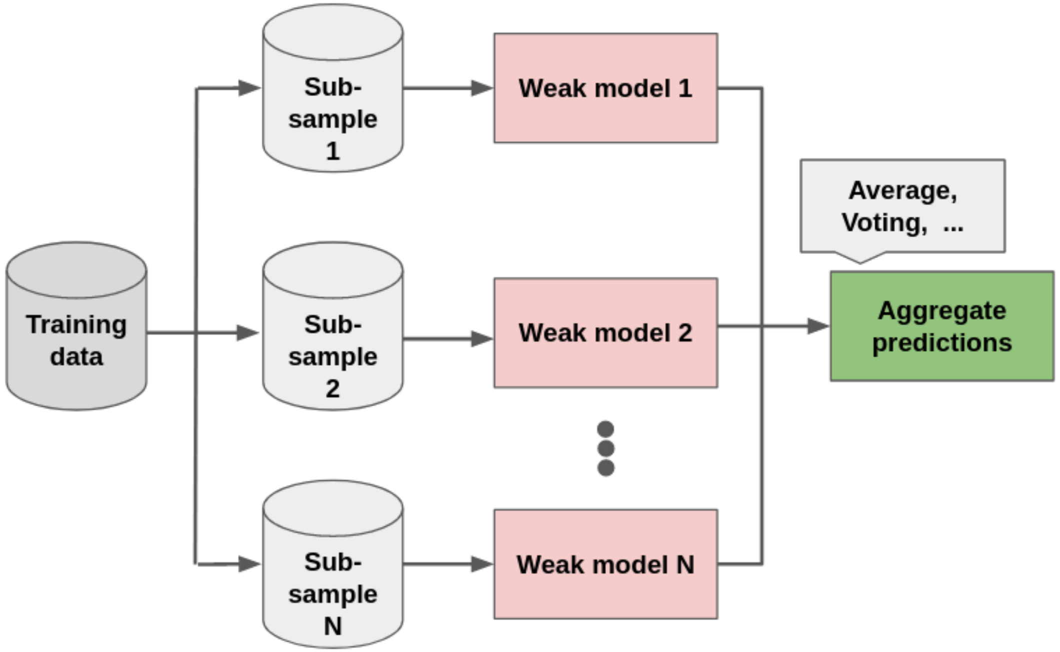
how to aggregate the predictions?

Deep Bagging Ensemble has already been explored in the SCA context [PCP20], [ZBHV21]

Bagging aggregation

Differences between traditional Bagging and SCA Bagging

Traditional Bagging



SCA Bagging Perin *et al* [PCP20]

- Training set not sub-sampled
- Aggregation :

$$e_k = \sum_{m=1}^{W} \sum_{i=1}^{Q} log(F[f(k,p_i), t_i]_m)$$

Where W is the number of weak models,
Q is the number of attack traces,
$f(.)$ is the sensitive operation,
$F[f(k,p_i),t_i]_m$ denotes the $f(k,p_i)$-th compotent of output of the model m, given the trace t_i as input.

Bagging limitations :

Each weak model contributes equally to the ensemble prediction, independently of their performance

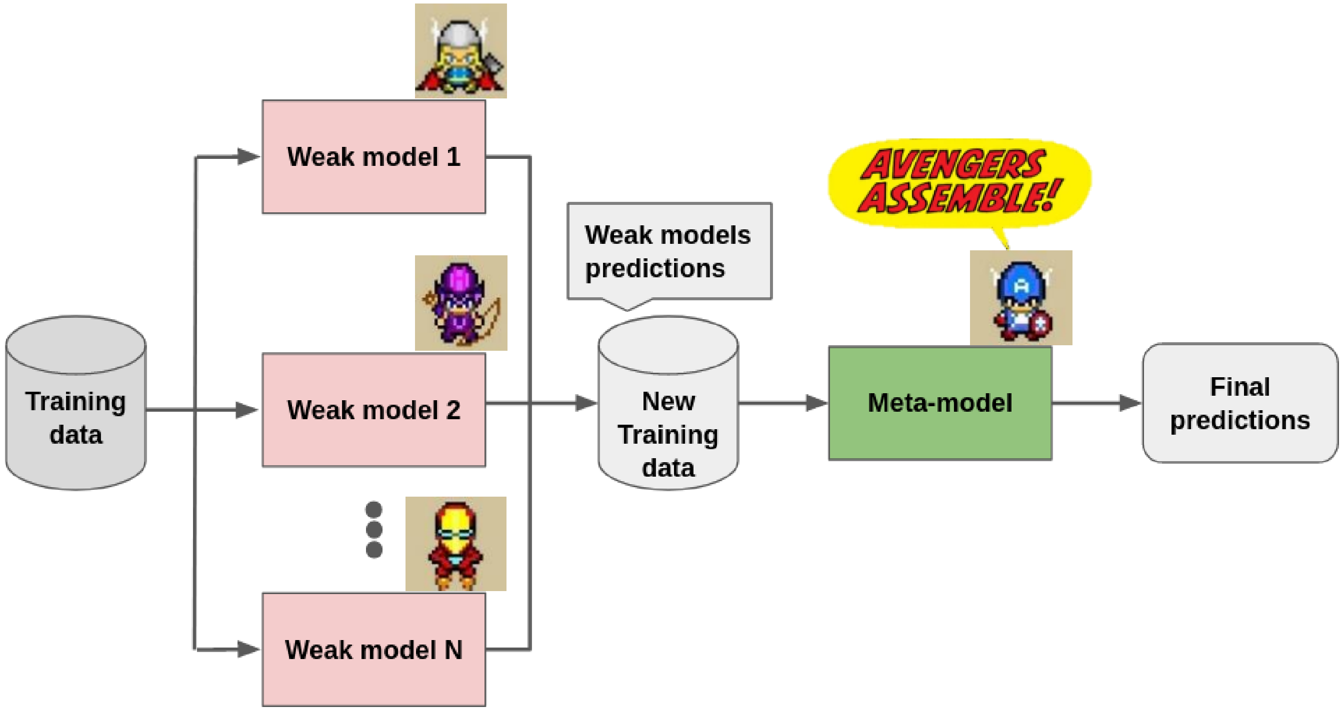
➢ Potential problem in the presence of significant performance gaps between weak models

Need for diversity among weak models

➢ Potential problem if lack of diversity between weak models

Stacking aggregation

Learn the best way to combine predictions



The meta-model takes the leadership

- ➢ Ensemble success depends on the ability of the meta-model to learn how to combine predictions

What data should be used to train the meta-model?

Good practice :

- ➢Train weaks models on TRAIN
- ➢Train meta-model on weak models VAL predictions
- ➢Meta-model inference on weak models TEST predictions

Due to the lack of validation data, we used the training data, considering the risk of overfitting

2. Experimental results

Dataset and Metric

Dataset	Train	Val	Test	Features	Contermeasure
AES HD	40,000	10,000	25,000	1250	Only high noise
ASCADF 0d	40,000	10,000	10,000	700	1st order masking
ASCADV 0d	160,000	40,000	100,000	1400	1st order masking
ASCADV 100d	160,000	40,000	100,000	1400	1st order masking + desynchronization

ASCADv1 : masked AES-128 implementation [https://github.com/ANSSI-FR/ASCAD]

- ASCADF : fixed key for training traces
- ASCADV : variable keys for training traces
- ASCADV 100d : Add 100 desynchronization samples

AES HD : unprotected (but very noisy) AES-128 hardware implementation [https://github.com/AESHD/AES_HD_Dataset]

METRIC :

Na = nb of attack traces required to get a constant Guessing Entropy to 1

Experiments settings

Weak models 1

Train 10 neural networks with random architectures (MLPs and CNNs)

- We target directly the Sbox output value as sensitive variable (256 possible values)
- Ordered weak models by their attack performance (Na)

Meta-model 2

Train 30 random meta-models for each ensemble size (2-10)

Choice of meta-model: MLP with random architecture

Hyperparameter	min	max	step
Number of layers	2	8	1
Number of neurons	100	1000	100
Activation	Relu, Elu, Selu, Gelu, Tanh		
Epoch	Early stopping : Val loss Patience 20		

Random hyper-parameters

Early stop to avoid overfitting

Comparison 3

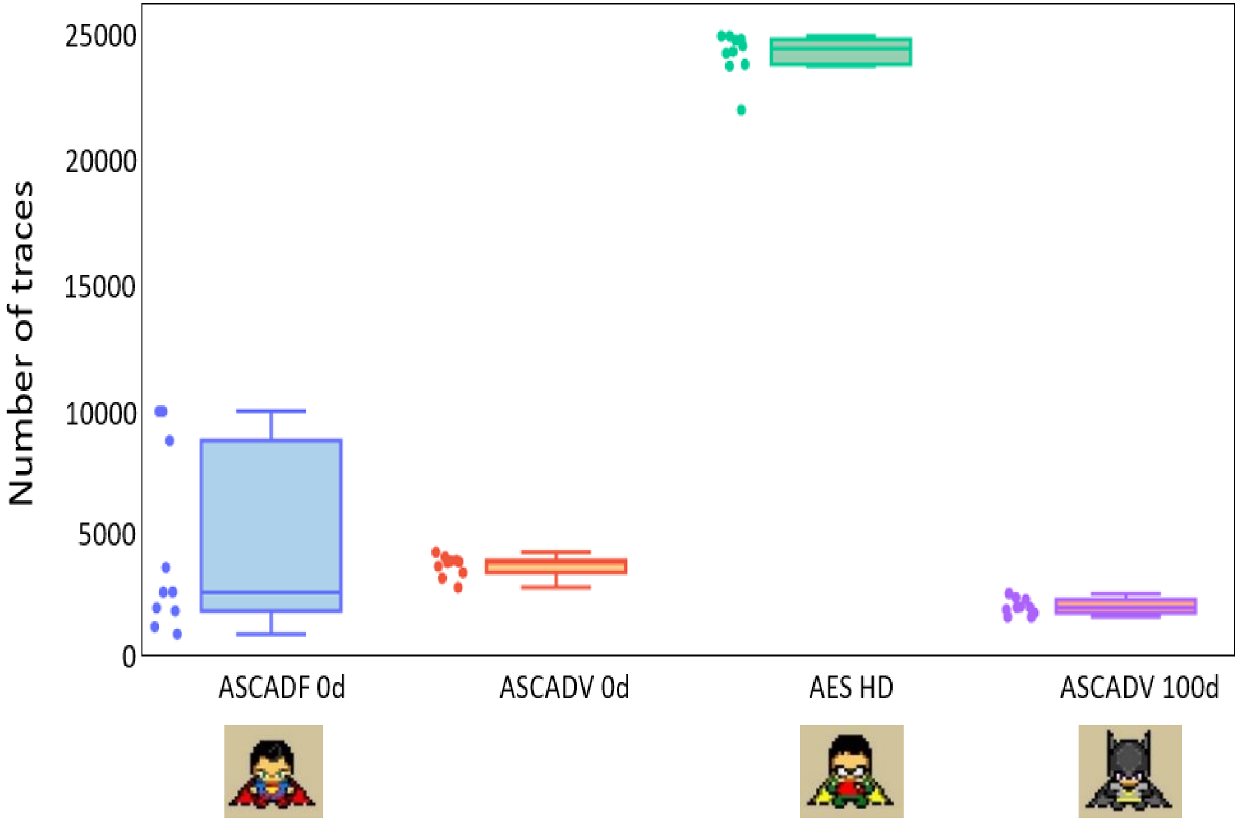
Check attack performance

- Check the robustness of Stacking
- Comparison with Bagging Ensemble and best single weak model

Ensemble configuration

Excellent Ensemble = accurate weak models and complementary errors

Our weak models :



3 different kinds of problem encountered :

ASCADF 0d : performance gap

ASCADV 100d : lack of diversity

AES HD : very poorly weak models

Results on ASCADF 0d

Weak models with significant performance gap

Na values are estimated considering only successful meta-models
The best result is highlighted by a green cell

Size of Ensemble	Nb success (Na <1109)	Na			Improvement in number of traces
		Min	Max	Mean	
2	30/30	371	853	576	66.54%
3	23/30	368	1098	696	66.81%
4	24/30	203	1064	680	81.69%
5	23/30	342	1062	674	69.16%
6	14/30	452	1043	588	59.24%
7	13/30	450	1070	604	59.42%
8	18/30	357	1086	666	67.80%
9	17/30	377	814	589	66.00%
10	15/30	427	989	631	61.49%

Best improvement (best meta-model)

Stacking improved overall attacks performance by more than 59%

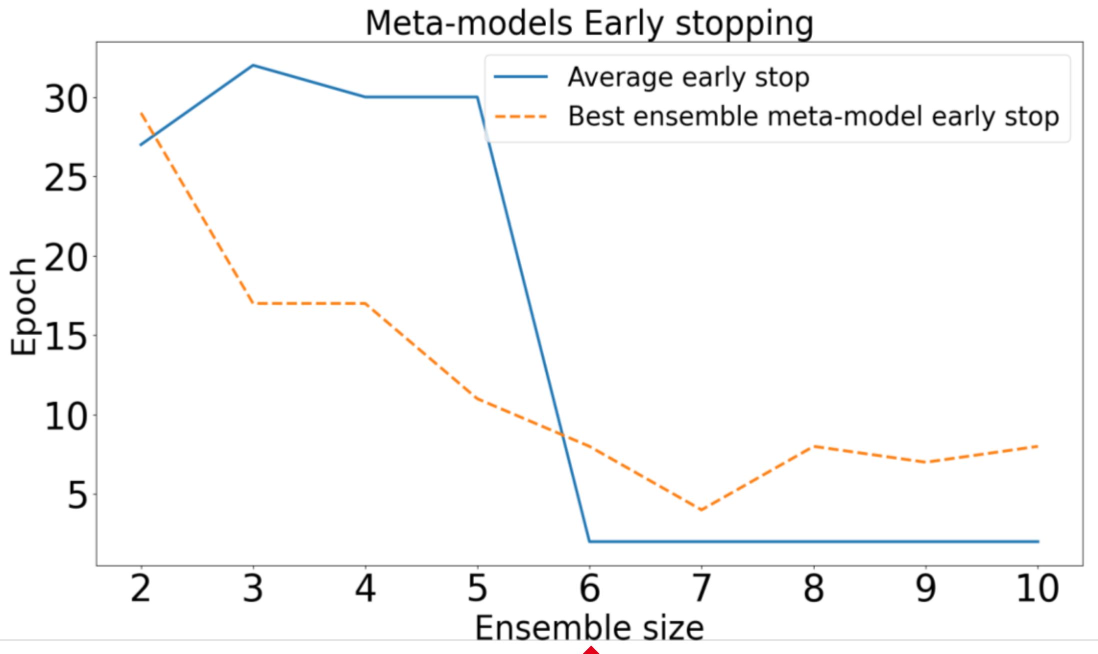
Nb meta-models Na < best weak model Na

By increasing the ensemble size, the number of successful meta-models decreases

Results on ASCADF 0d

Weak models with significant performance gap

Size of Ensemble	Nb success (Na <1109)	Na Min	Na Max	Na Mean	Improvement in number of traces
2	30/30	371	853	576	66.54%
3	23/30	368	1098	696	66.81%
4	24/30	203	1064	680	81.69%
5	23/30	342	1062	674	69.16%
6	14/30	452	1043	588	59.24%
7	13/30	450	1070	604	59.42%
8	18/30	357	1086	666	67.80%
9	17/30	377	814	589	66.00%
10	15/30	427	989	631	61.49%

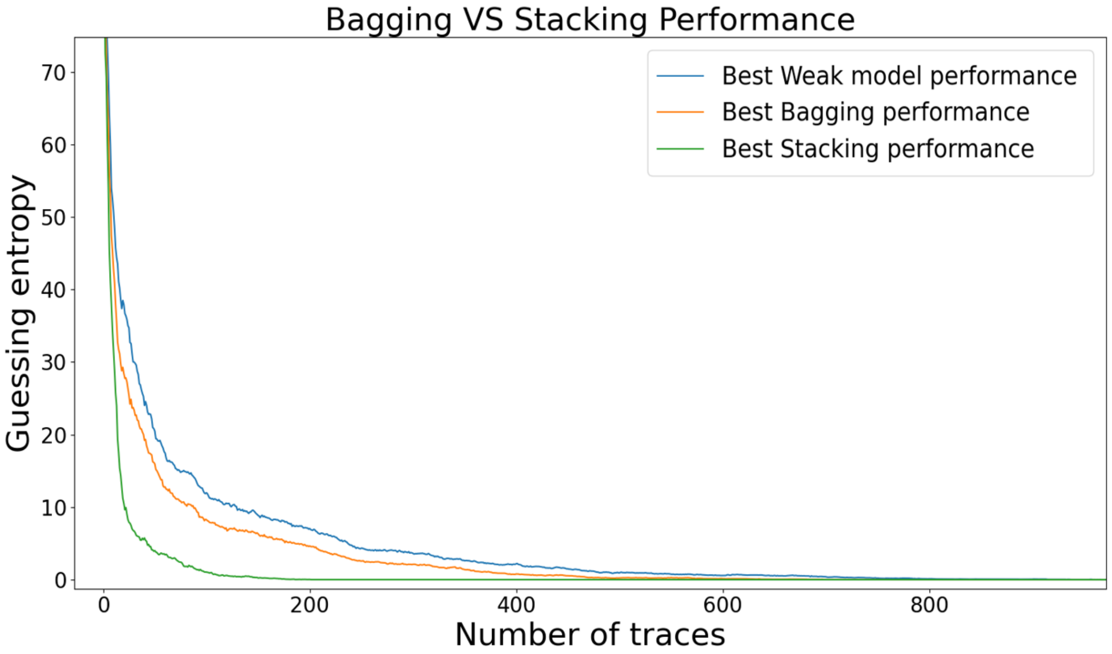


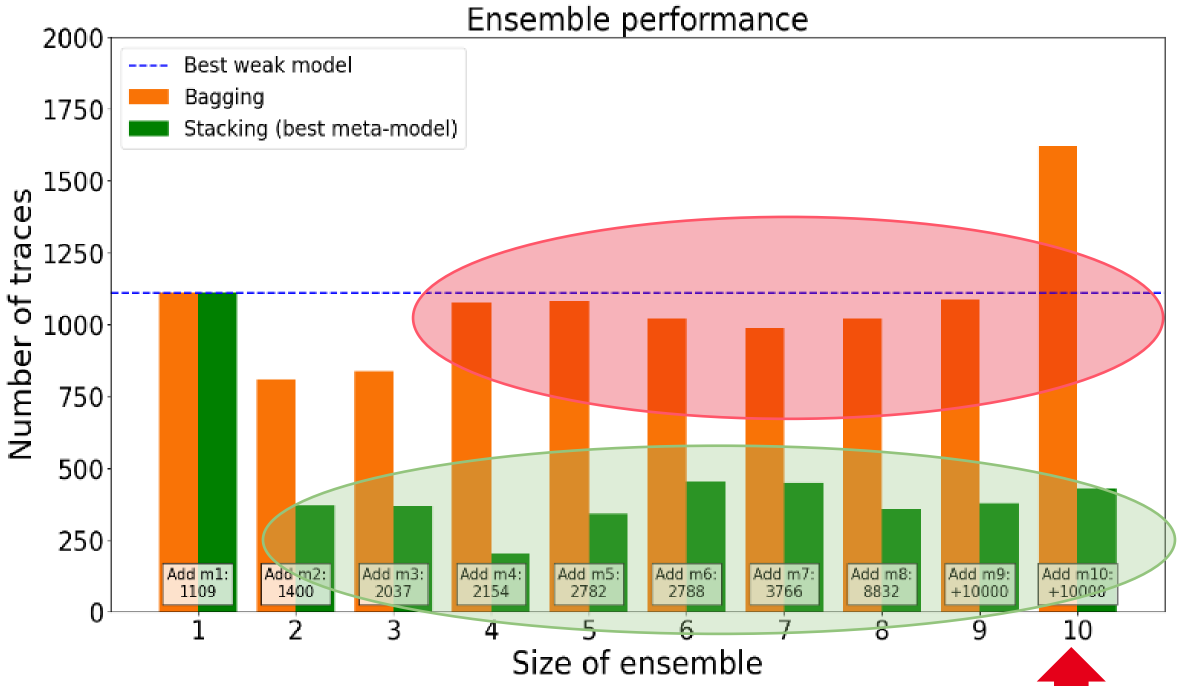
Addition of weak models make the meta-model learning task easier
→ overfit quickly without learning relevant information

By increasing the ensemble size, the number of successful meta-models decreases

Results on ASCADF 0d

Weak models with significant performance gap





Stacking converges faster and obtain higher attack performance than Bagging

Bagging KO / Stacking robust

Bagging strongly impacted by performance gap

Stacking less impacted since the meta-model learn the relevance of each weak model

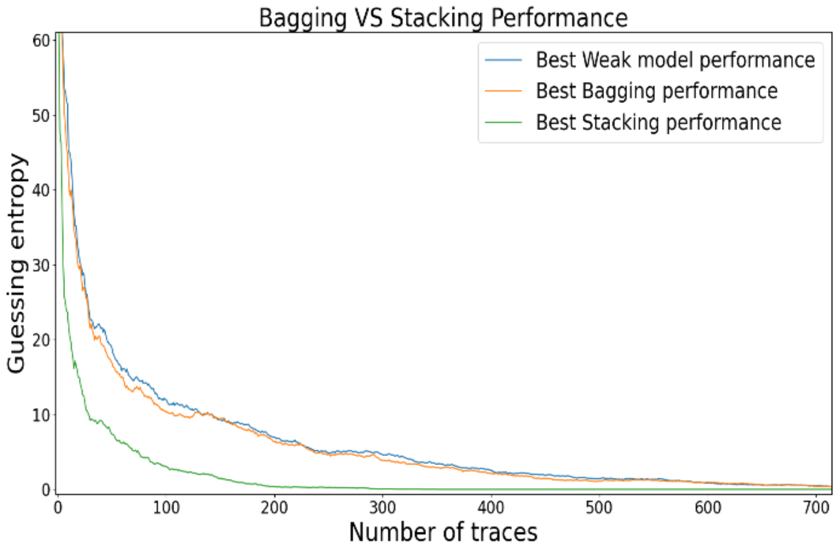
Results on ASCADV 100d

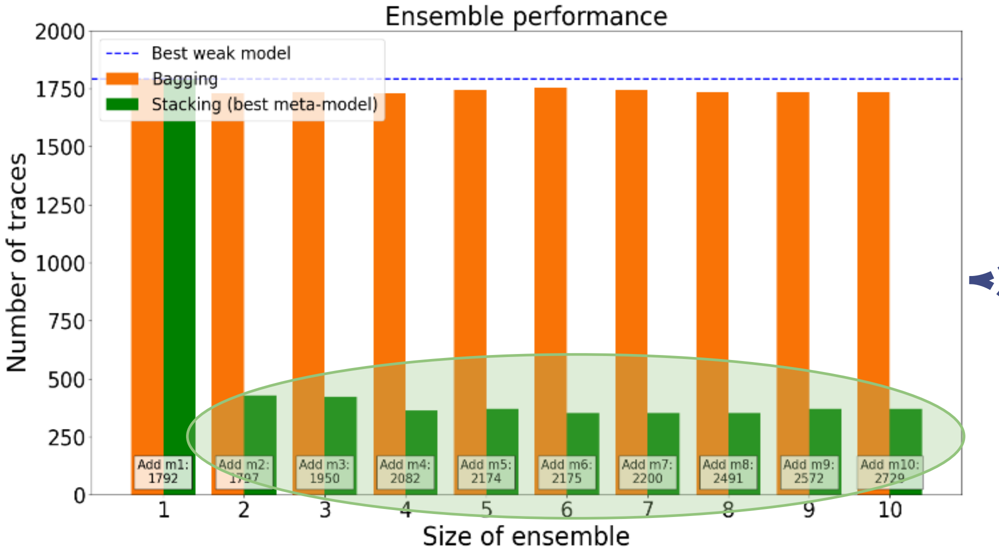
Weak models with lack of diversity

Size of Ensemble	Nb success (Na <1792)	Na			Improvement in number of traces
		Min	Max	Mean	
2	30/30	429	1172	808	76.06%
3	30/30	423	1256	735	76.39%
4	30/30	362	1160	763	79.79%
5	30/30	369	1141	711	79.40%
6	30/30	352	1070	700	80.35%
7	30/30	351	1130	742	80.41%
8	30/30	351	1333	741	80.41%
9	30/30	369	1097	737	79.40%
10	30/30	369	1137	717	79.40%

Stacking improved overall attack performance by more than 76%

Meta-model training more robust :
adding weak models did not decrease the number of successful meta-models





Bagging KO

Stacking works

Stacking robust : the meta-model is able to learn from small variations between the weak models predictions

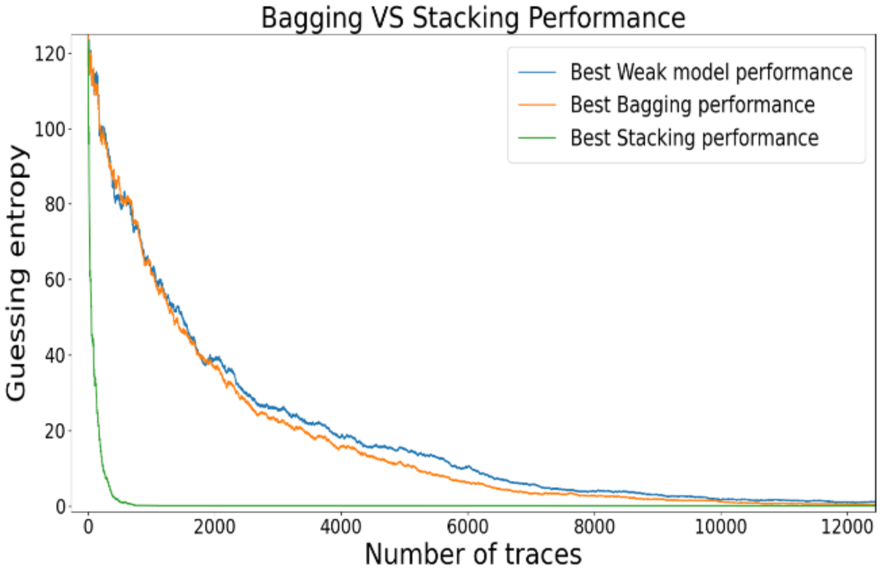
Results on AES HD

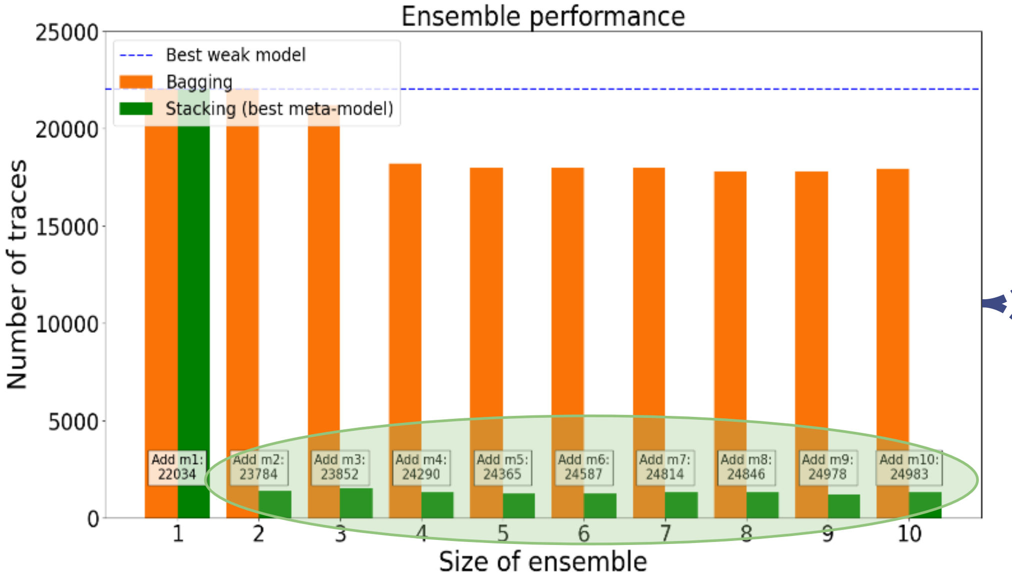
Weak models with very poor performance

Size of Ensemble	Nb success (Na <22034)	Na Min	Na Max	Na Mean	Improvement in number of traces
2	25/30	1365	4179	2212	93.80%
3	27/30	1507	20542	2704	93.16%
4	28/30	1324	11394	2286	93.99%
5	28/30	1251	8014	2038	94.32%
6	27/30	1253	9641	1988	94.31%
7	29/30	1324	12604	2377	93.99%
8	26/30	1315	8947	1962	94.03%
9	27/30	1220	4556	1865	94.46%
10	27/30	1318	9092	2106	94.01%

Stacking improved overall attack performance by more than 93%

Meta-model training more robust :
adding weak models did not decrease the number of successful meta-models





Bagging works well

Stacking works better

Stacking results in better weak models combination and much greater improvement in attack performance

Stacking VS Bagging

Stacking outperforms Bagging in our experiments

Dataset	Best weak model	Bagging improvement in number of traces	Stacking (best meta-model) improvement in number of traces
AES HD	22034	17798 (20%)	1220 (94%)
ASCADF 0d	1109	709 (28%)	203 (81%)
ASCADV 0d	2973	2194 (26%)	582 (80%)
ASCADV 100d	1792	1730 (3%)	351 (80%)

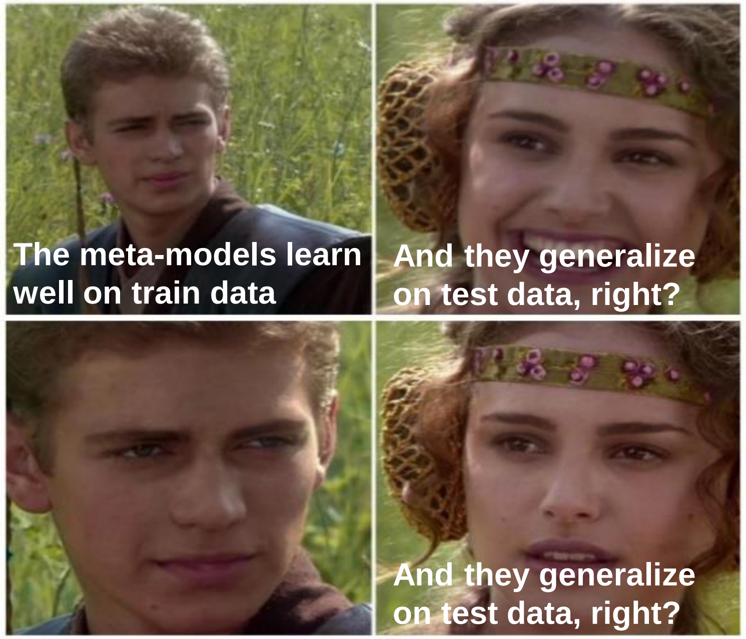
Significant gain in attack performance across all datasets

- **Less impacted by the individual performance of weak models**
  The meta-model learn the relevance of each weak model

- **Less impacted by the lack of diversity**
  The meta-model is able to learn from small variations in predictions

- **More flexible aggregation**
  No need for the evaluator to select weak models

Generalizable Meta-model



Stacking prone to overfitting :

- We observed that the ensemble model proved often to be too complex for the problem
- **2-layer** meta-models always generalize and improve attack performance

Hyperparameter	Architecture 1	Architecture 2
Number of layers	**2**	**2**
Number of neurons	600	300
Activation	elu	tanh
Epoch	Early stopping : Val loss Patience 20	
Learning Rate	0.0001	
Mini Batch	100	
Optimizer	RMSprop	

➡ **No need for complex meta-model**

Comparison with state-of-the-art

Less hyperparameterization

Dataset	Reference	Hyperparameterization method	Na
ASCADF 0d	Arch. in [3]	-	1146
	Arch. in [22]	Reinforcement learning.	202
	Our best Meta-model	4 random weak models with Na between [1109-2154]	203
ASCADV 0d	Arch. in [3]	-	1275
	Arch. in [22]	Reinforcement learning.	490
	Our best Meta-model	5 random weak models with Na between [2973-3970]	582
ASCADV 100d	Arch. 1 in [23]	-	3333
	Arch. 2 in [23]	Regularization technique.	347
	Our best Meta-model	7 random weak models with Na between [1792-2200]	351
AES HD	Arch. in [15]	-	25000
	Arch. in [27]	Visualization tools	1050
	Our best Meta-model	9 random weak models with Na between [22034-24983]	1220

similar performance to an efficient architecture obtained by RL learning

similar performance to an efficient architecture obtained by human effort

Suitable approach to limit the need for the evaluator to perform a fine hyperparameterization

Take-Away Messages

Reduce hyperparameterization effort with Ensemble
Not completely replace the hyperparameters search → relax

We extends the previous works which used Bagging Ensemble in SCA context [PCP20]

- ➢ Highlights some of the limitations of the Bagging aggregation
- ➢ Stacking better performance and flexible solution to address Bagging limitations
  - ✓ Less impacted by weak models performance
  - ✓ Less impacted by lack of diversity
  - ✓ Limited ensemble size is enough to build strong ensemble model

Stacking suitable to relieve the security evaluator from performing a fine hyperparameterization

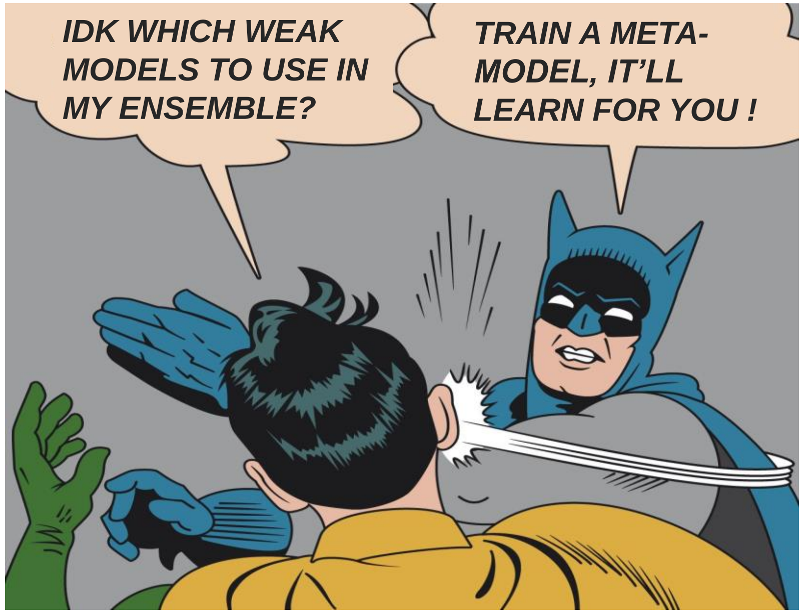
As a counterpart :

- ➢ Ensemble success depend on the meta-model training
- ➢ Prone to overfitting (promoted by the use of TRAIN data to train the meta-model, if possible use other data)

Generalizable Meta-model

- ➢ No need to consider complex meta-model (avoid overfitting)
- ➢ Further simplify the meta-model could be beneficial

Future works : Boosting ensemble in SCA context



Thanks

Questions?

CEA-Leti, Grenoble, France
cea-leti.com
dorian.llavata@cea.fr

LABORATOIRE HUBERT CURIEN
UMR • CNRS • 5516 • SAINT-ETIENNE

INNOVATION DÉFENSE LAB

INSTITUT CARNOT CEA LETI