Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis

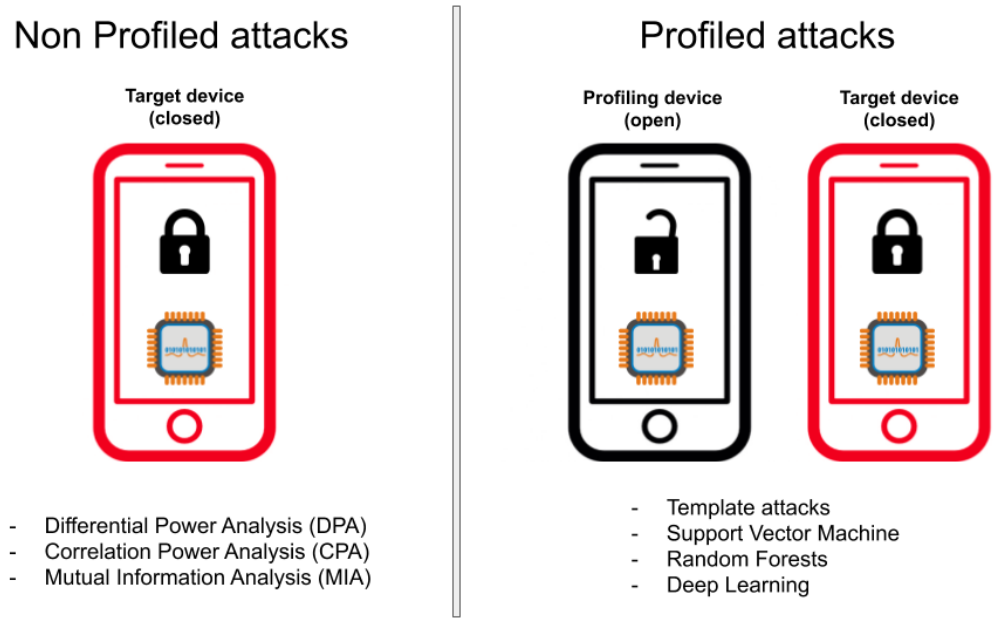
Benjamin Timon

August 28, 2019

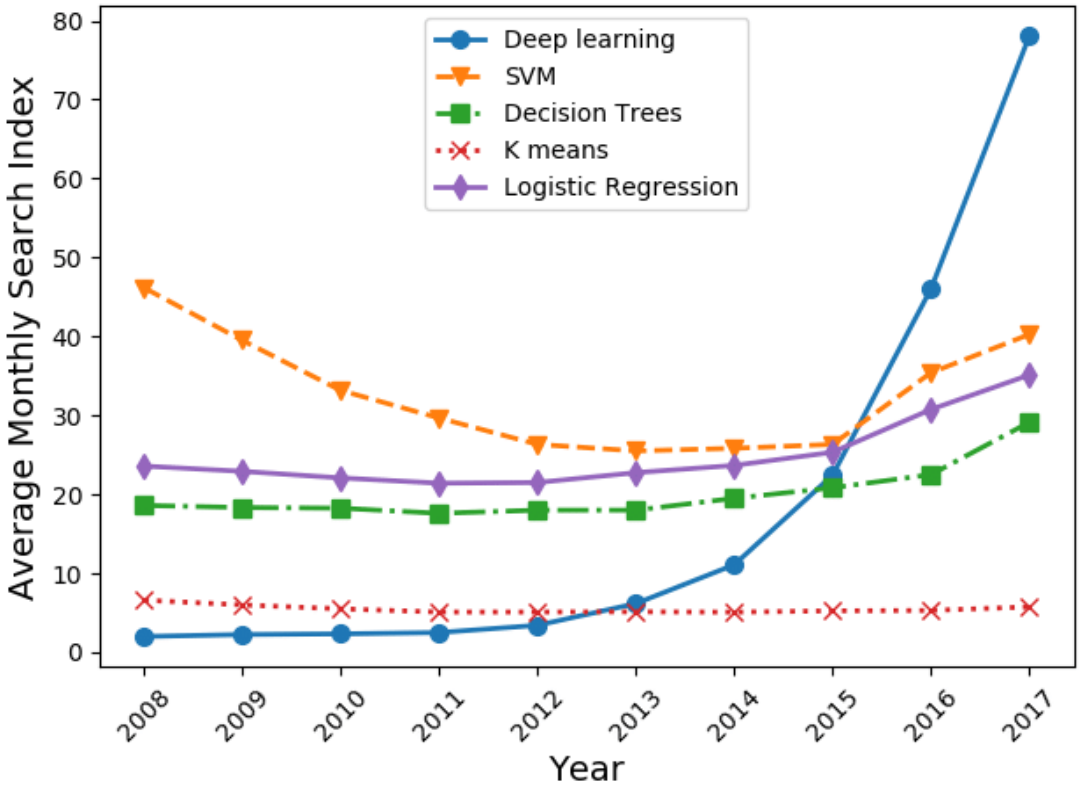
Python notebook presentation for CHES 2019

Introduction & Motivation

Profiled vs Non-Profiled attacks



Machine Learning trend



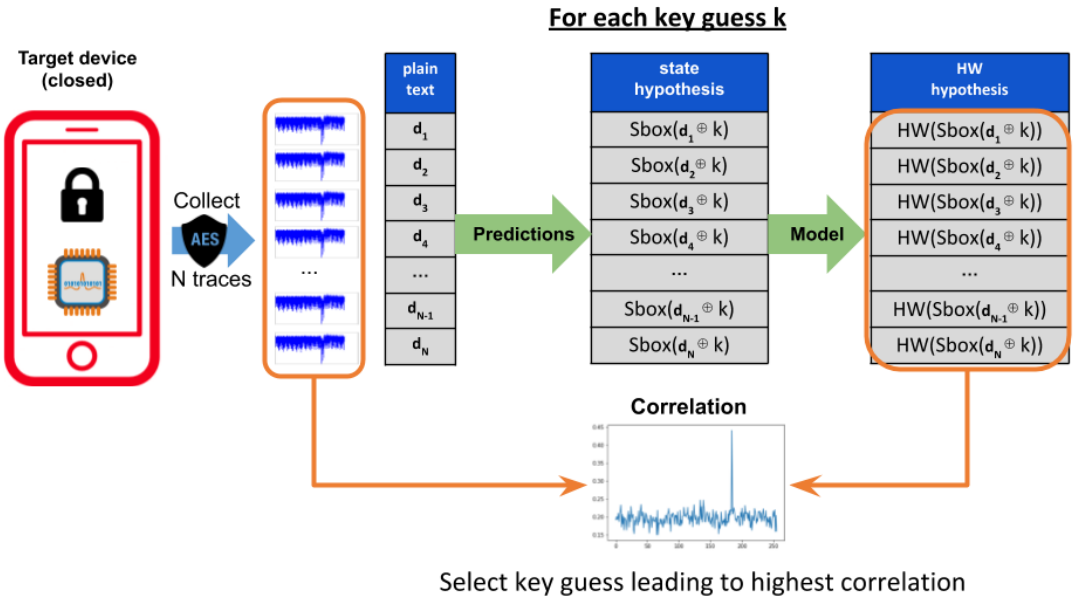
- Recently, many publications on Deep Learning for Side-Channel attacks
- Publications show clear interest of DL for SCA
- Most of time, DL outperforms other techniques
- Can adapt the network architecture to the challenge
    - For instance CNN for de-synchronized traces

This research

How can we use Deep Learning for Non-Profiled Side-Channel attacks?

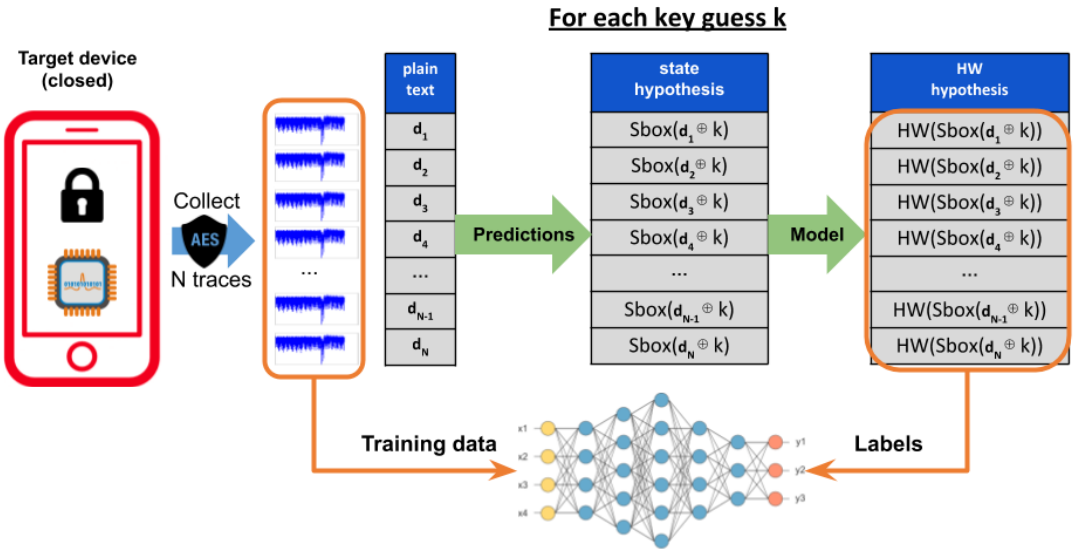
Differential Deep Learning Analysis (DDLA)

Correlation Power Analysis



Select key guess leading to highest correlation

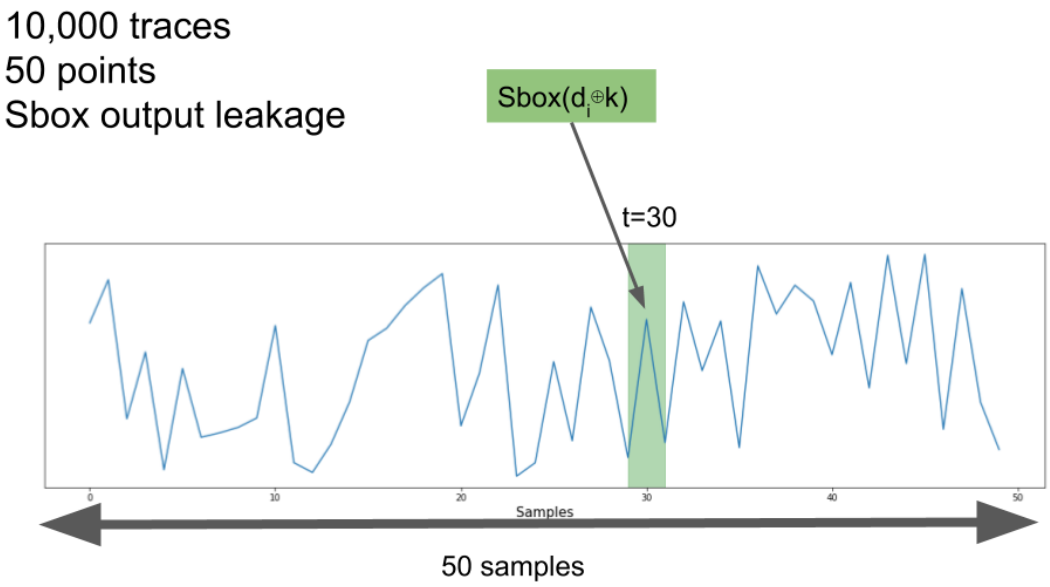
Follow similar strategy for DDLA



Select key guess leading to the better training

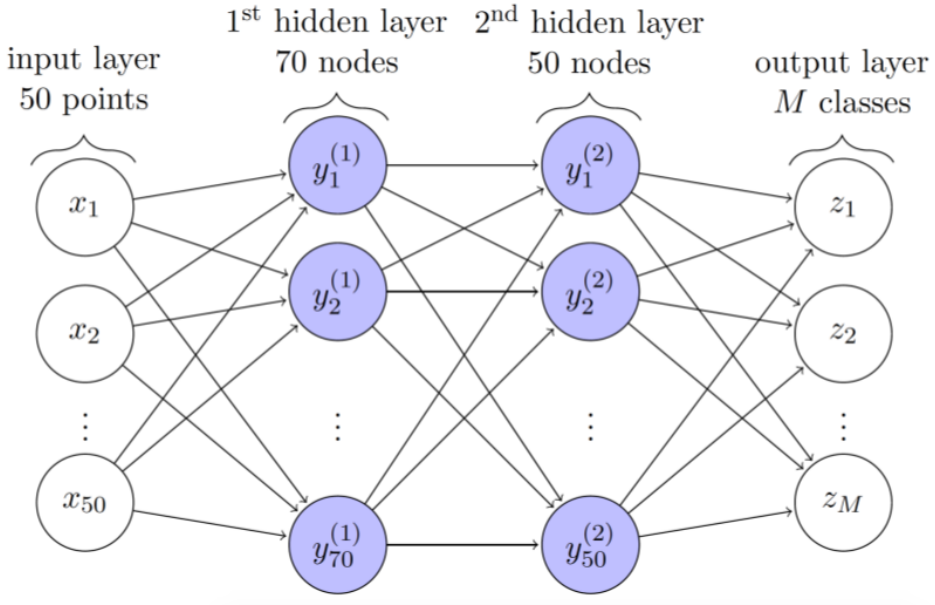
Demonstration: attack with accuracy and loss

Generate simulation traces




```
In [1]: from demo import gen_simu_data

        # Generate simulation traces for demonstration
        data_training = gen_simu_data()
```

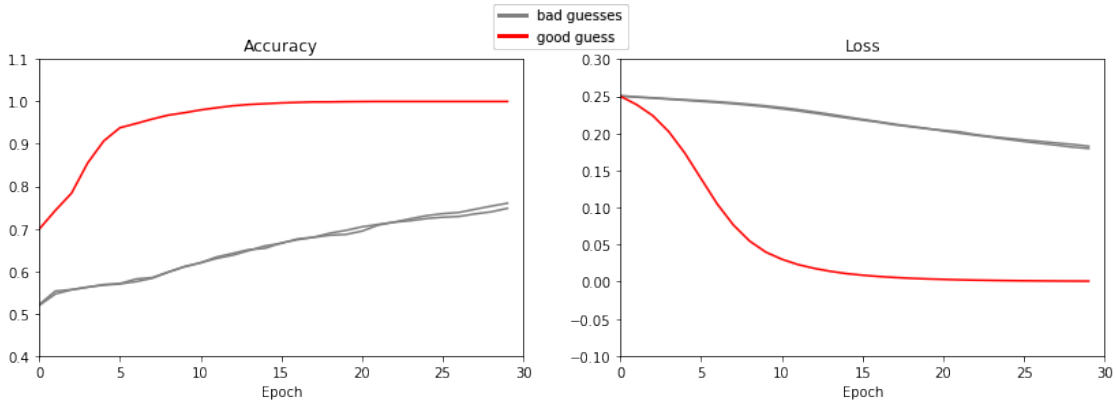
Network



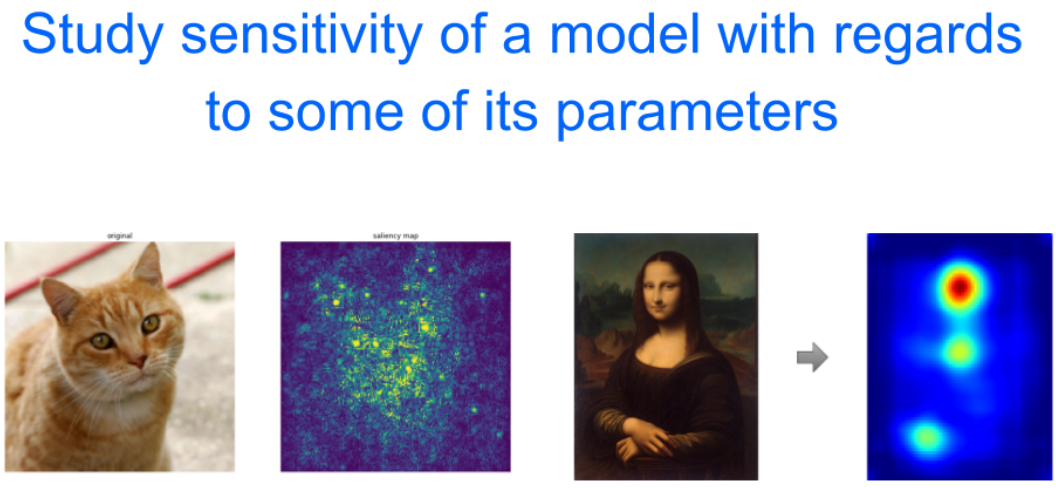
Observe loss and accuracy during training

```
In [2]: from demo import demo_ddla_acc_loss

        ddla = demo_ddla_acc_loss(data_training)
        ddla.run(n_epochs=30)
        ddla.fig
```

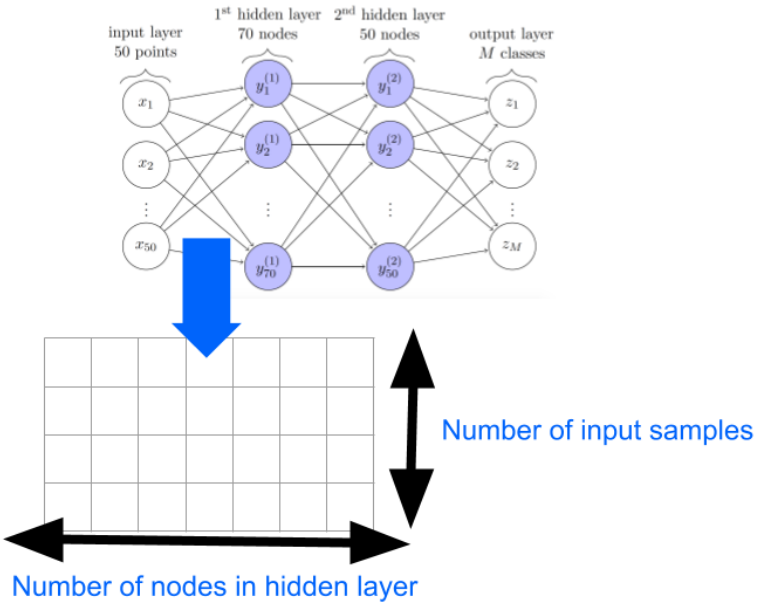
Out[2]:



Sensitivity Analysis

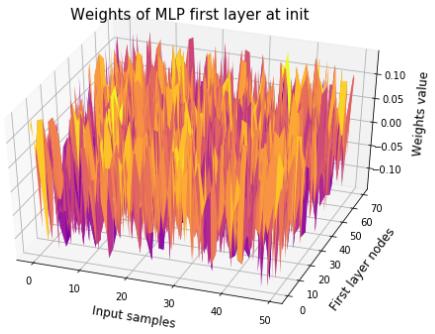


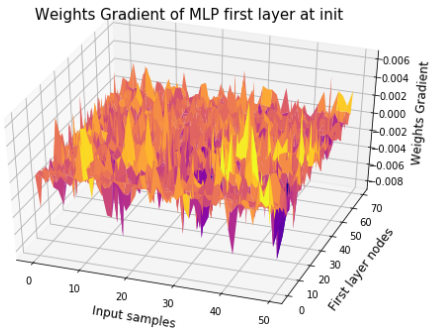
Demonstration: Observe first layer gradient




```
In [3]: from plots import plot_weights_and_grad_3d

        plot_weights_and_grad_3d(data_training)
```

Out[3]:

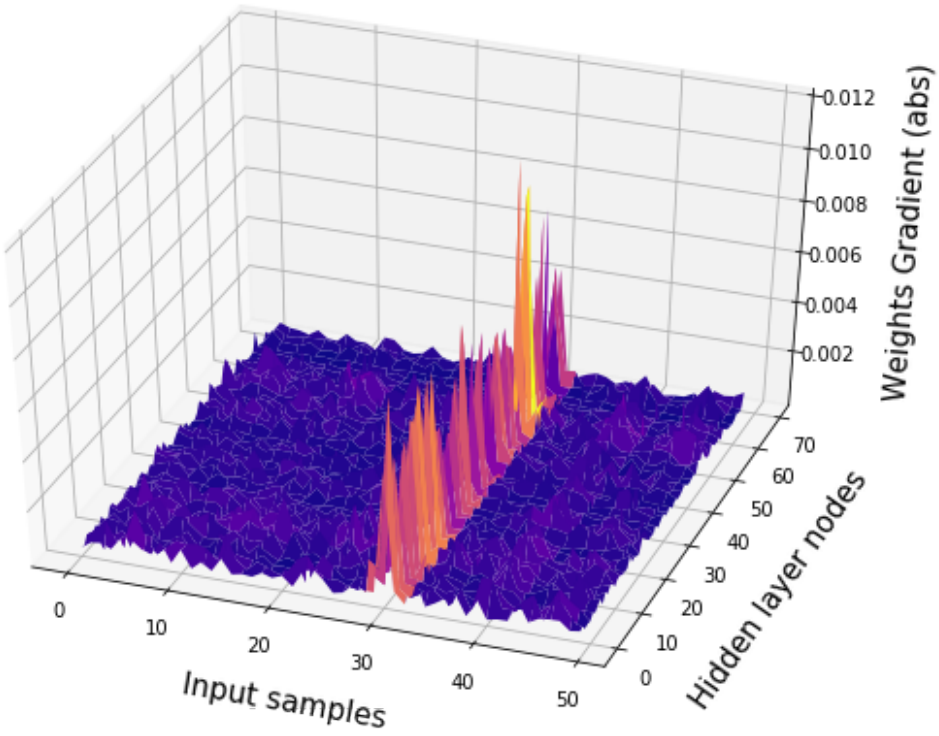


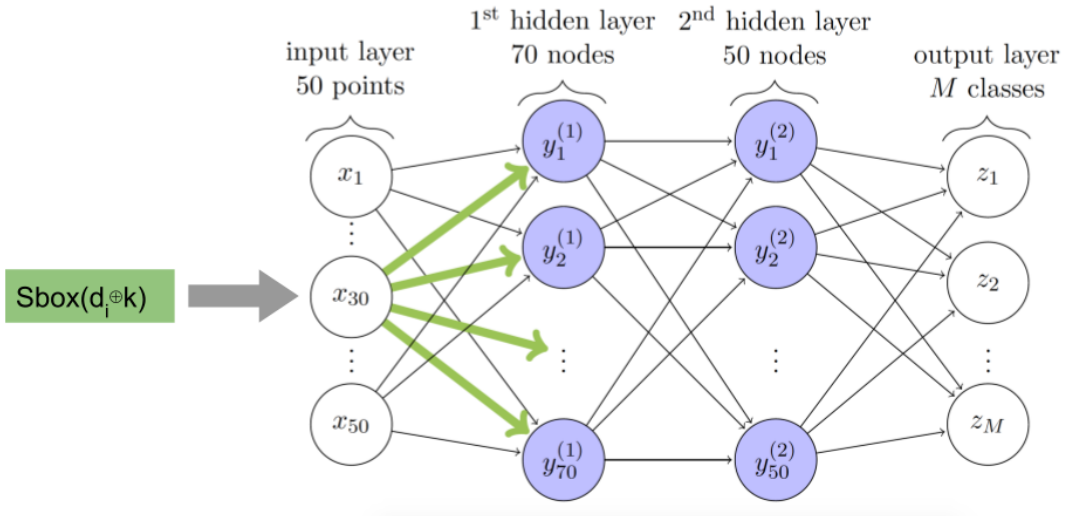


Observe first layer gradient during training

```
In [4]:  from demo import demo_ddla_gradient

         ddla = demo_ddla_gradient(data_training)
         ddla.run(15)
         ddla.fig
```

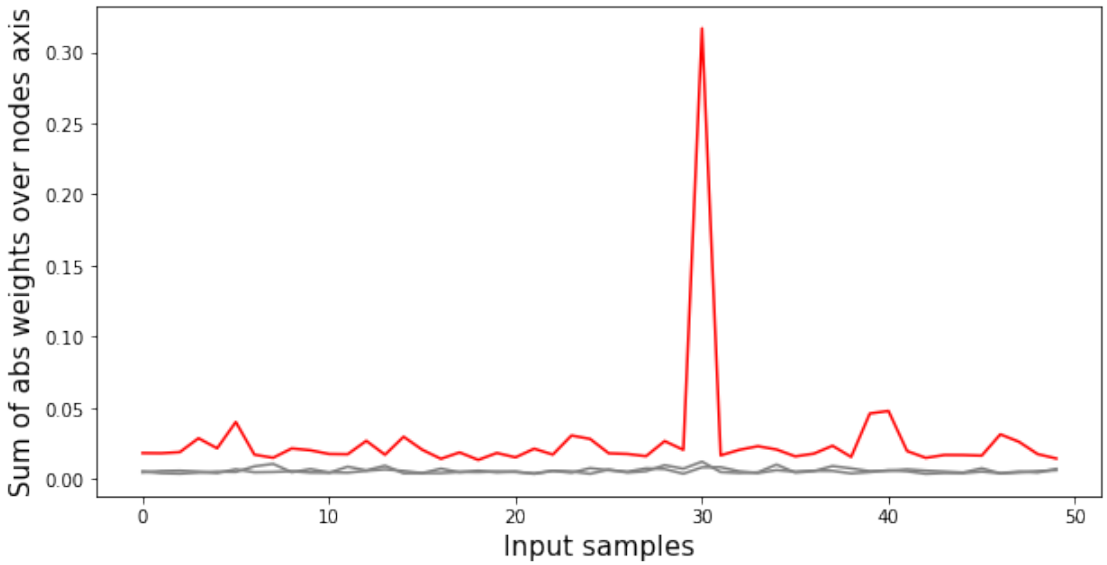
Out[4]:



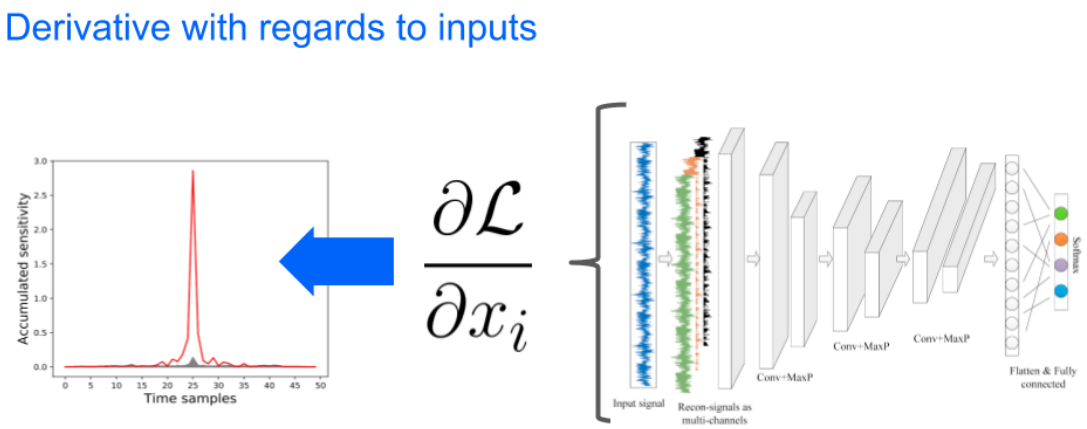


In [5]:
```
from plots import plot_weights_2d

plot_weights_2d(ddla)
```

Out[5]:

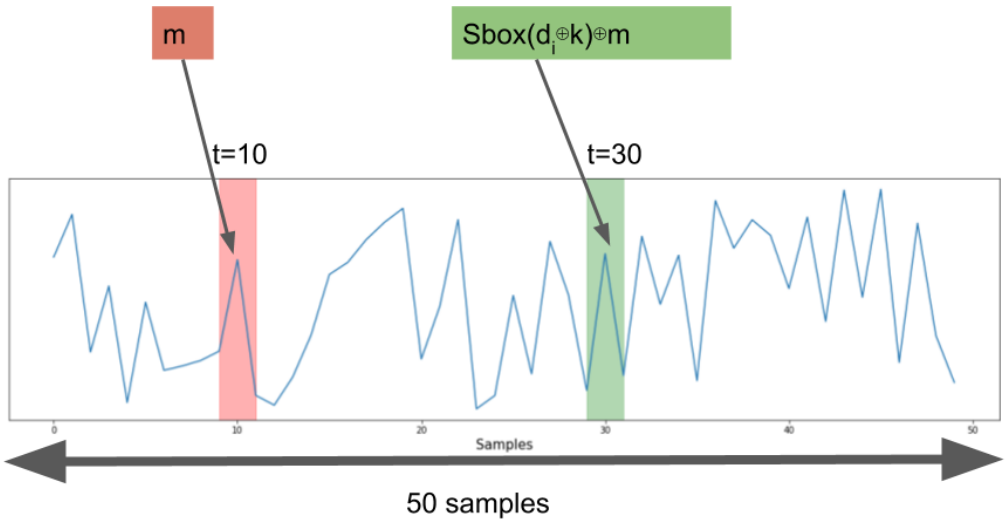


Derivatives with regards to the inputs



Masked implementations

Generate masked simulation traces

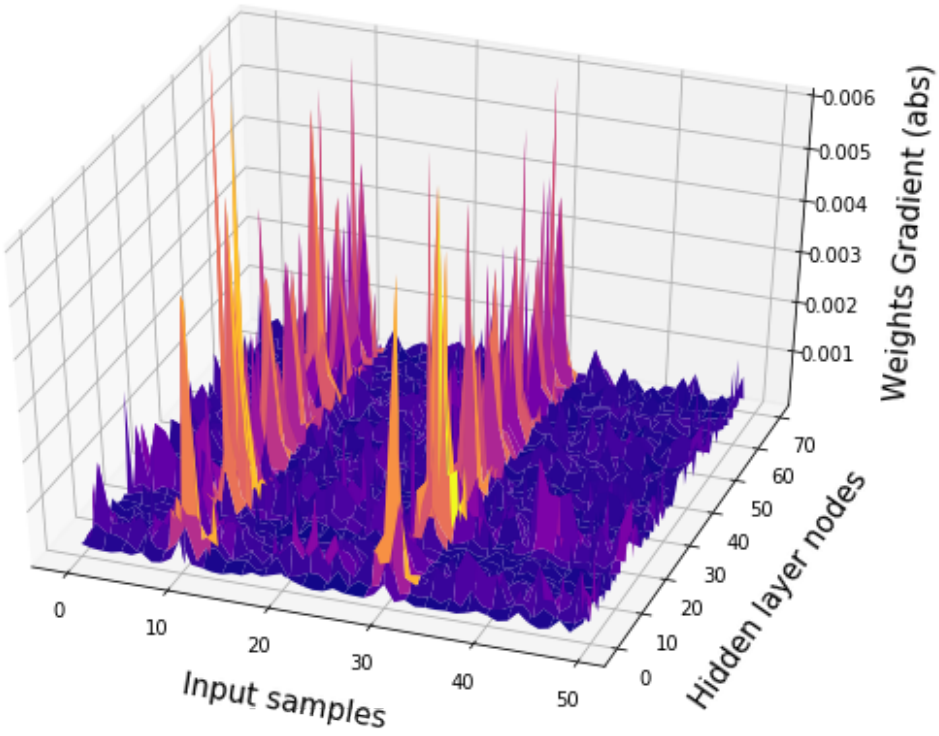


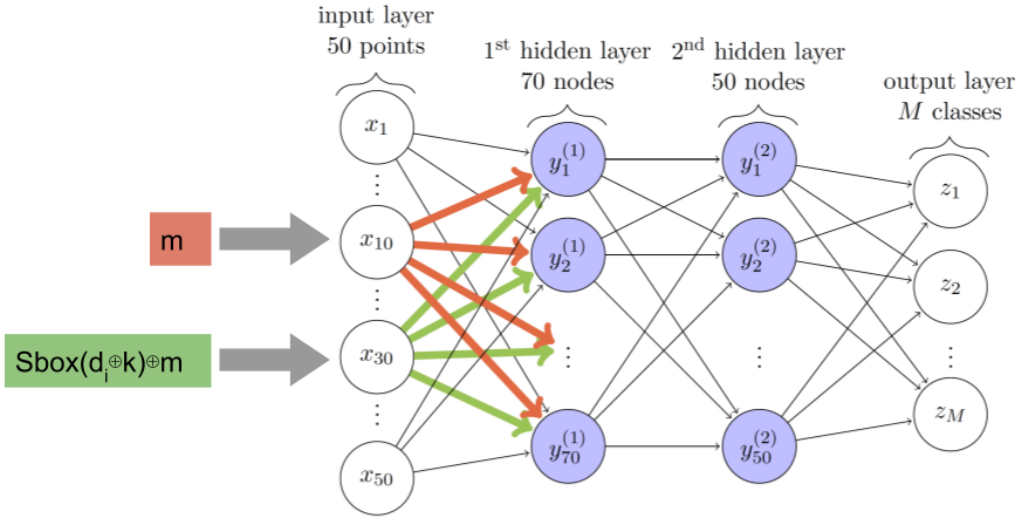

```
In [6]:  from demo import gen_simu_masked_data

         # Generate masked simulation traces for demonstration
         data_training = gen_simu_masked_data()
```

Demonstration: high-order DDLA

```
In [7]:  from demo import demo_ddla_high_order

         ddla = demo_ddla_high_order(data_training)
         ddla.run(20)
         ddla.fig
```

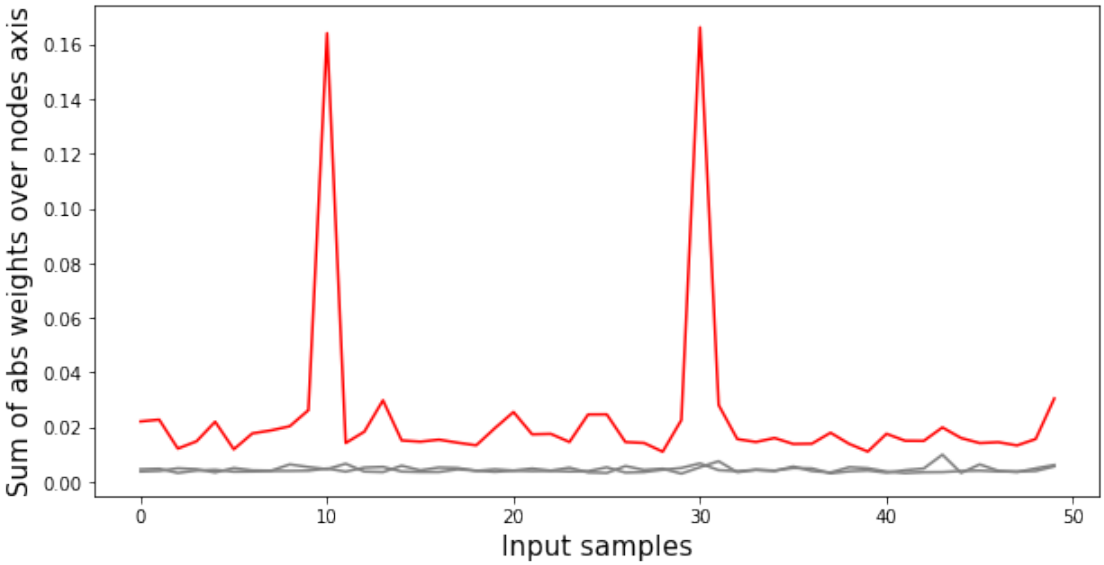
```
Out[7]:
```






```
In [8]:  from plots import plot_weights_2d

         plot_weights_2d(ddla)
```

Out[8]:



Masked implementations

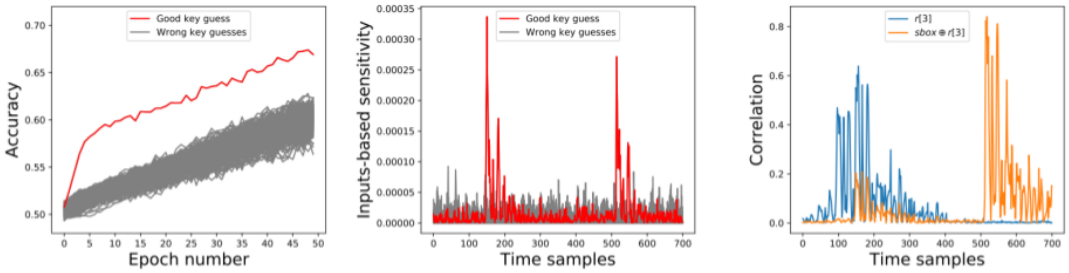
- Attack works on masked implementation
- Same attack process for first and high order attack
- No preprocessing needed
- Sensitivity analysis reveals Sbox and mask locations in the trace

→ interesting alternative for attacks in black box for with no details about the implementation (number of masks etc)

Results on masked implementations

- ChipWhisperer: masked implementations with 1 and 2 masks

- ASCAD masked implementation



Conclusion

Two contributions introduced in the paper:

- Use Deep Learning and Neural Networks to perform Non-Profiled Attacks
  - Leverage the power of DL and Neural Networks for Non-Profiled attacks
  - Same attack process to target non-protected and masked implementations
  - Works with CNN against de-synchronized traces

- Introducing Sensitivity Analysis for Side-Channel to locate leakage areas in the traces while using neural networks
  - Reveals intermediate values and masks leakage areas
  - Applicable to any neural network architecture
  - Applicable to Profiled DL training as well

Thank you