Reinforcement Learning for Hyperparameter Tuning in Deep Learning-based Side-channel Analysis

https://youtu.be/CmgYLyUMEwM

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

강화학습

부채널분석과 딥러닝

논문 내용

강화학습

- **강화학습**

  상태를 지속적으로 탐험해가며 미래에 주어지는 보상(reward)값을 활용하여 agent가 최적의 수를 행하도록 하는 학습법

  지도/비지도 학습은 데이터의 특징을 찾는 것이지만 강화학습은 어떻게 행동할 지를 가르침

- **가치 기반과 정책 기반 존재**

  → 가치 기반은 최대 가치를 얻도록 하는 것이며 이후에 정책이 될 수 있고,

  정책 기반은 애초에 높은 가치를 얻기 위한 정책(어떠한 행동을 취할지에 대한 확률 분포)을 가르침

- **Markov Decision Process(MDP)와 가치 함수**를 기반으로 함

  → 그러나 현실에 존재하는 대부분의 문제들은 MDP로 해결하기 어려움

  (상태나 행동 공간이 적고, 미리 상태 전이 확률 및 보상을 알고 있어야 함 → model-based)

- **Q-learning이 주로 사용된다고 함**

  → Model-free 방식 : 위와 같은 요소를 알 수 없으므로 가치 함수의 값을 예측하면서 학습

- 업데이트 주기에 따라 **Monte-Carlo**와 **Temporal-difference learning**으로도 나뉨

강화학습

- **가치 함수**

  누적 보상을 추정하기 위해 상태 가치 함수와 행동 가치 함수를 적용 → 최대 보상을 얻도록 함

- **상태 가치 함수 (V)**

  현재 상태에서 얼마나 더 보상을 받을 수 있을지

- **행동 가치 함수 (Q)**

  현재 상태에서 이러한 행동을 했을 때 얼마나 더 보상을 받을 수 있을지

- **Monte-carlo**

  : 전체 에피소드가 끝나야 보상을 알 수 있음

- **Temporal-difference learning**

  : 매 time step마다 가치 함수 갱신 → 에피소드 중에 다음 상태의 가치 함수로 현재 상태의 가치 함수를 갱신

강화학습

- **Exploitation**

  : 현재 알고 있는 가장 최적의 행동 선택 (실제로 최적은 아닐 수 있으나 알고 있는 선에서 최적) → greedy

- **Exploration**

  : 다양한 경험을 쌓기 위해 랜덤으로 아무 행동을 선택 (Exploitation만 하다 보면 실제 최적 행동을 선택하지 못 할 수 있으므로 이를 보완하기 위해 가끔 수행)

- **행동 정책** : 학습데이터를 발생시키기 위한 정책 → 탐험성 유지 위한 정책
- **목표 정책** : 학습 대상 정책 → 최적의 수를 찾기 위해 개선해야 할 대상이 되는 정책
- **On-policy**

  행동 정책과 목표 정책이 동일 → 학습이 쉽지만 local optimum에 빠질 수 있음

- **Off-policy**

  행동 정책과 목표 정책이 다름 → 학습 데이터가 되는 경험은 행동 정책에 의해 생성되고, 이것을 가지고 목표 정책을 평가

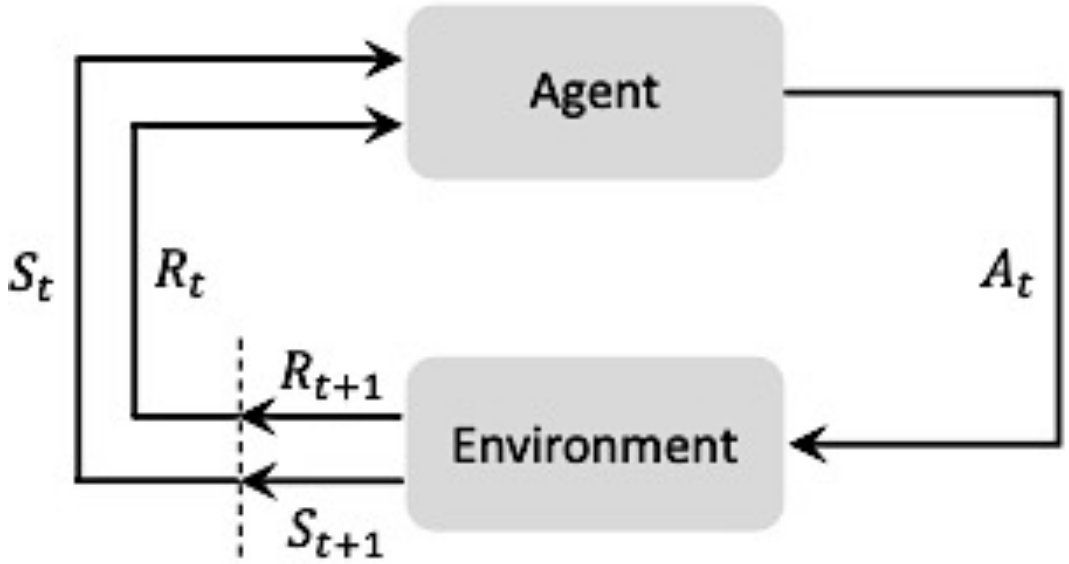
Q-learning

- **Q-learning**

  - 대표적인 가치 기반 + Off-policy temporal-diffenece learning 알고리즘

  - state s에서 행동 a를 취할 때의 보상을 측정하고 상호작용하며 저장된 Q-value를 갱신 → 전체 보상을 극대화 하기 위함

  - 탐험을 통해 점차 학습해나감



강화학습

- 강화학습 과정

For each (s, a) initialize the table entry $\hat{Q}(s, a)$ to zero

Repeat (for each episode)

Observe the initial state s

Repeat (for each step of episode)

Select an action a from state s (e.g. ε-greedy) and execute it

Receive immediate reward r

Observe the new state s'

Update the table entry for $\hat{Q}(s, a)$ as follows:

$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$

$s \leftarrow s'$

Until s is terminal (goal state)

Exploration (탐험)

Q-learning

- **Q-value 갱신 함수**

Discount
: 현재 보상이 미래 보상보다 얼마나 더 중요한지를 의미
(단기 및 장기 보상의 가치)

Q-learning rate
: 얼마나 빨리 새 정보를 학습하는가

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

S_t에서 A_t를 취했을 때 현재 기대 보상

S_{t+1}에서의 모든 action a에 대한 기대 보상 중 최대값

- 대부분은 Q-value 추정 값을 lookup table로 저장 **(Tabular Q-learning)**

	State 0	State 1
Action 0	Q(S0, A0)	Q(S1, A0)
Action 1	Q(S0, A1)	Q(S1, A1)

부채널 분석과 딥러닝

- 현재, 부채널 분석과 딥러닝에 관한 연구들이 다수 진행
- **키 값을 알아내기 위해 신경망을 구성하고 공격을 수행하거나, 이를 위한 전처리 과정에 딥러닝 적용,**
  또는 신경망 자체에 대한 부채널 분석을 수행 등
- 부채널 공격 수행의 경우, 더 높은 성능을 얻기 위해 **하이퍼 파라미터 튜닝이 중요**
- 뒤에 나올 논문 또한 **강화학습을 통해 부채널 분석을 위한 신경망의 하이퍼 파라미터 튜닝**을 수행한 연구 결과
- 하이퍼파라미터 튜닝을 하지 중요하게 여기지 않거나,
  랜덤 및 그리드 서치 등의 알고리즘을 사용하여 최적화한 경우도 다수 존재
- **하이퍼 파라미터 최적화를 중요하게 여긴 연구도 다수 존재**
  - 학습해야할 파라미터(가중치 및 바이어스)의 수에 관련된 요소를 줄이기 위한 하이퍼 파라미터 튜닝
  - 동일 성능 달성 but 더 작은 모델 사이즈를 위한 최적화
  - CNN의 필터 수, 커널 사이즈, 스트라이드, 완전연결층의 유닛 수 등
  - 최적 하이퍼 파라미터를 찾기 위한 최적화 알고리즘 사용
- 강화학습은 아직 SCA 분야에서 잘 연구되지 않음

논문 내용

1. **SCA에 강화학습을 사용**
2. **Q-learning을 적용하였으며, 새로운 reward function 고안**
3. **부채널 공격 성능이 좋으며 적은 파라미터 달성**

Reinforcement Learning for Hyperparameter Tuning in Deep Learning-based Side-channel Analysis

Jorai Rijsdijk[1], Lichao Wu[1], Guilherme Perin[1] and Stjepan Picek[1]

Delft University of Technology, The Netherlands

논문 내용

- **MetaQNN**

  고성능 CNN 구조를 생성해내기 위해 강화학습을 사용한 meta-modeling 알고리즘

  Q-learning을 활용하여 레이어나 하이퍼파라미터를 설정

  accuracy를 reward로 사용

  즉, 하이퍼파라미터 선택에 따른 보상으로 정확도를 사용하여, 정확도가 높아지는 (최대 보상) 경우를 선택하도록 학습

- **Meta QNN in SCA**

  그러나, 정확도가 SCA에서는 적절한 평가 지표가 아니며, MetaQNN은 시간 소모 및 계산 비용이 너무 큼

  → SCA를 위한 신경망 생성에 MetaQNN을 사용하기 위해 새로운 reward function을 고안

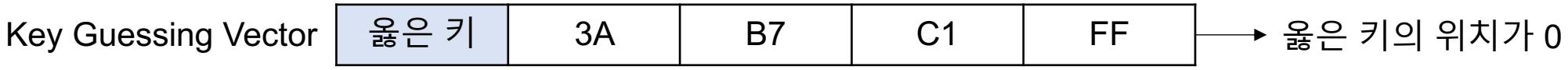
논문 내용

- **Reward Function**

  → 기존의 MetaQNN의 평가 지표(정확도)는 SCA에 부적합

  → Guessing Entropy(GE)와 검증 데이터셋에 대한 정확도 + 3가지 메트릭 포함

  1. $\boldsymbol{GE}$: SCA의 일반적 평가지표, 모든 키 추측 중에서 옳은 키의 평균 키 순위 위치 → 엔트로피가 낮을수록 키 복구가 정확



  2. **검증 데이터셋에 대한 정확도(α)** : 높은 검증 정확도는 Q_{tGE} 가 낮음을 의미하므로 reward에 포함

  3. **3가지 메트릭**

GE가 0에 수렴하는데 필요한 trace 수

$$t' = \frac{t_{max} - min(t_{max}, \overline{Q}_{t_{GE}})}{t_{max}}$$

고정된 수의 attack set

고정된 수의 attack set에서 GE를 0으로 만들기 위해 필요한 trace의 백분율

$$GE'_{10} = \frac{128 - min(GE_{10}, 128)}{128}$$

$$GE'_{50} = \frac{128 - min(GE_{50}, 128)}{128}$$

Attack trace의 10%/50%를 사용했을 때의 GE

논문 내용

- **Reward Function**

  이러한 메트릭들을 통해 GE가 0에 수렴하는 것이 실패할 경우에도 적절하고, 공격 성능까지 더 잘 추정할 수 있는 reward function을 만든다고 주장

$$R = \frac{t' + GE'_{10} + 0.5 \times GE'_{50} + 0.5 \times a}{3}$$

더 적은 파형으로 비밀키를 찾기 위해 t'와 $GE'10$에 더 높은 가중치 줌
(두 값이 클수록 더 적은 파형을 사용한 것이므로)

- 이를 기반으로 **더 적은 파라미터를 갖는 모델을 위한 reward function**

  p'로 인해 평균 reward의 기준선이 높아짐 (실험 결과)

$$p' = \frac{max(0,\ p_{max} - p)}{p_{max}}, \quad R' = \frac{t' + GE'_{10} + 0.5 \times GE'_{50} + 0.5 \times a + p'}{4}$$

- 실제 신경망의 파라미터 수(p)
- 가질 수 있는 파라미터의 최대값(p_{max}) ; 설정 가능

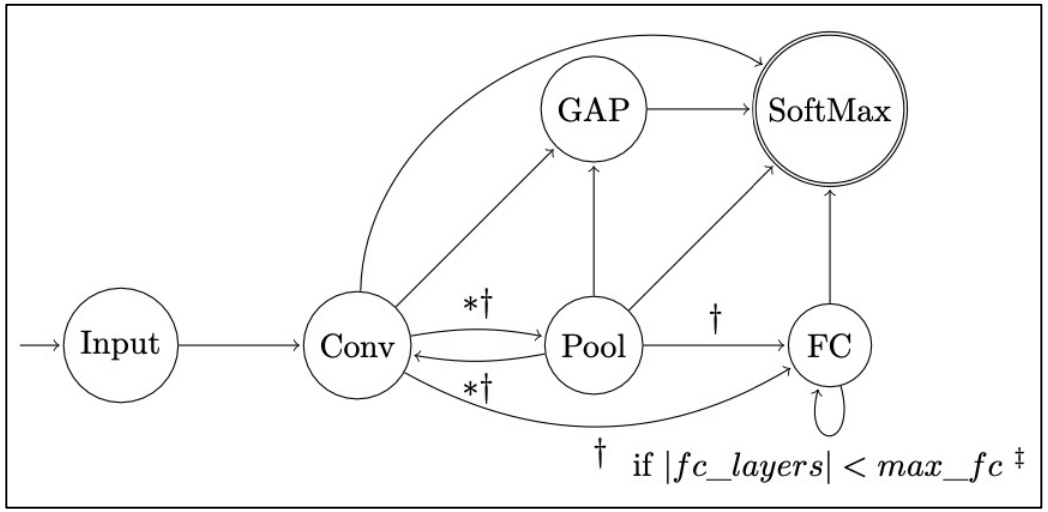
- 이와 같이 설계된 reward function을 통해 R을 계산하며, Q-learning에 적용됨

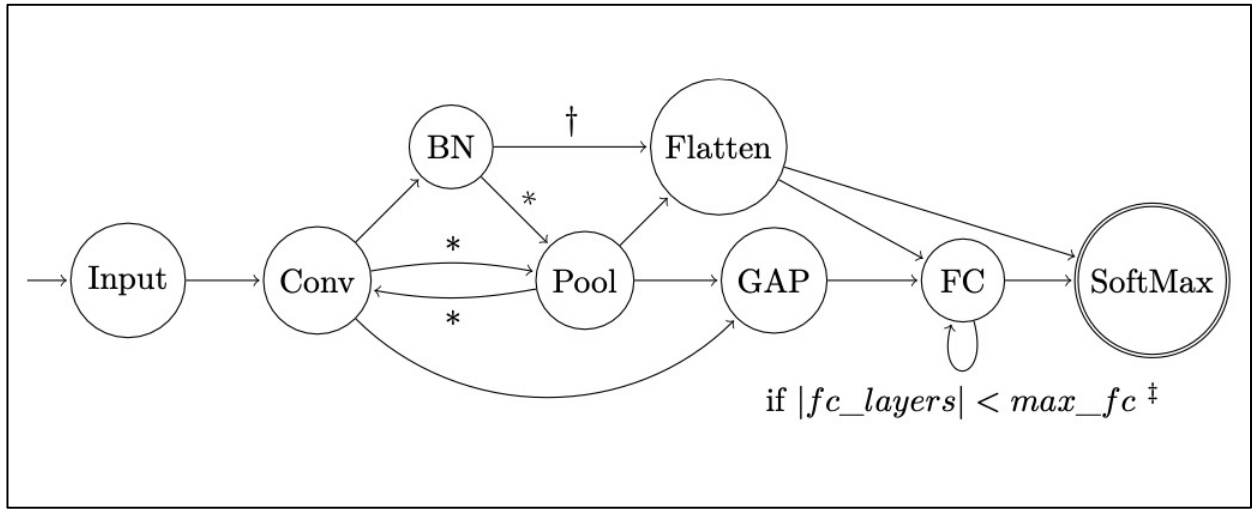
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

* q-learning rate는 실험에 의한 최적 값인 $1/t^{0.85}$ 사용

논문 내용

- 기존 MetaQNN(왼), 제안 기법(오)





Maximum Total Layers	14
Maximum Fully Connected Layers	3
Fully Connected Layer Size	[2, 4, 10, 15, 20, 30]
Convolutional Padding Type	SAME
Convolutional Layer Depth	[2, 4, 8, 16, 32, 64, 128]
Convolutional Layer Kernel Size	[1, 2, 3, 25, 50, 75, 100]
Convolutional Layer Stride	1
Pooling Layer Filter Size	[2, 4, 7, 25, 50, 75, 100]
Pooling Layer Stride	[2, 4, 7, 25, 50, 75, 100]
SoftMax Initializer	Glorot Uniform
Initializer for other layers	He Uniform
Activation function	SeLU

1. 네트워크 수렴을 빠르고 안정화 시키기 위해 Batch Normalization 추가
2. SCA에 주로 쓰이는 VGG와 유사한 구조 적용
3. 기존은 이전 단계에서 Global Average Pooling(GAP)와 Softmax를 선택했지만, 제안 모델은 다름
4. 현재 상태에서의 피처맵 표현 크기보다 layer의 크기가 더 클 경우, *로 가지 못하고 십자가로 감

논문 내용

- **하이퍼파라미터 튜닝을 적절히 하는지 성능 평가를 위해 두 가지의 leakage model 사용**

  1. Hamming Weight leakage model : 중간 값 1바이트의 HW를 label로 사용하는 모델 (즉, 9개의 class)
  2. Identity leakage model : 중간 값 1바이트를 label로 사용하는 모델 (0~255, 총 256개의 class)

- **Datasets**

  1. ASCAD

  1) fixed key, 700 features, 45000 / 5000 / 10000

  2) random key, 1400 features, 200000 / 100000

  2. CHES CTF

  Training : fixed key, 45000

  Validation, Test : different key, 2500/5000

  2200 features

논문 내용

- **실험 결과 중 일부..**

  - 평균 보상과 이동 평균 보상을 비교.

  - Epsilon이 1이면 랜덤, 0.1이면 10%만 랜덤 액션하여 신경망 생성

  - Epsilon당 평균 보상과 이동 평균 보상은

    Epsilon이 감소함에 따라 증가할 경우 제대로 학습됨을 의미

    (갈수록 랜덤이 아니기 때문에)

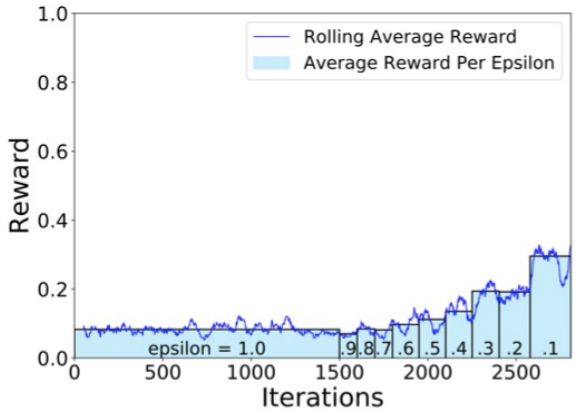
  → 즉, Epsilon이 감소함에 따라 agent가 발전

    처음엔 랜덤이다가 보상을 높이는 쪽으로 가게 됨

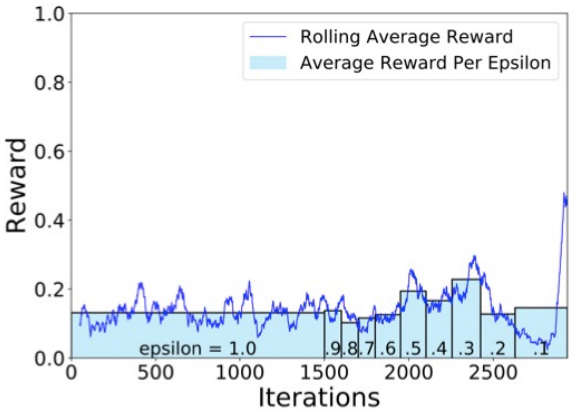
  → 그래프도 보면, epsilon이 적어질수록 보상이 증가

- 더 적은 파라미터를 위한 reward function 사용한 경우(RS),

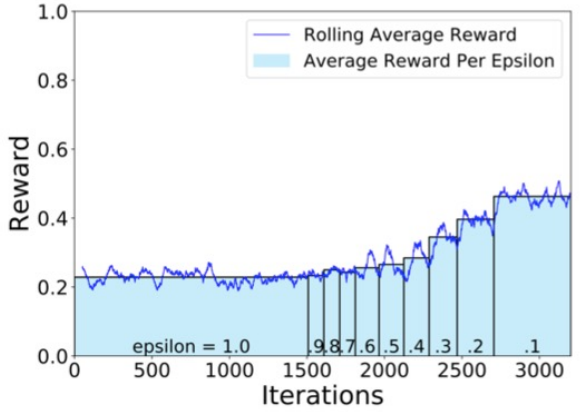
  보상의 평균 기준선이 더 높음을 알 수 있음 ($+p$')



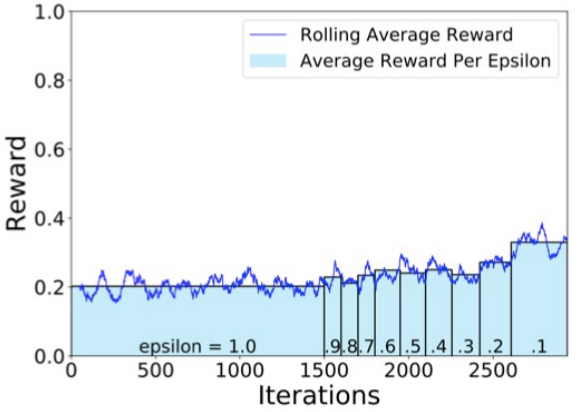
(a) CNN ASCAD Fixed Key HW Model



(b) CNN ASCAD Fixed Key ID Model



(c) CNN ASCAD Fixed Key HW Model (RS)



(d) CNN ASCAD Fixed Key ID Model (RS)

감사합니다.