

*Deep Learning*을 사용한 차분공격

2019.11.01

한양대학교 수학과
유대훈, 주영진

C@mp Lab.

Contents

1

(Multiple) Differential Cryptanalysis

2

Deep Learning 개요

3

Deep Learning을 사용한 차분공격

Reference

- 1.** Aron Gohr, “Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning”, CRYPTO 2019
- 2.** C. Blondeau and B. Gerard, “Multiple Differential Cryptanalysis: Theory and Practice”, FSE 2011
- 3.** K. He et al., “Deep Residual Learning for Image Recognition”, CVPR 2016

Notation

- PS, KS, CS : 각각 Plaintext, Key, Ciphertext의 Space
- \oplus : Exclusive or 연산
- \boxplus : Modulo Addition 연산
- \lll, \ggg : cyclic rotation 연산
- $Enc_k(A)$: 평문 A 를 키 k 로 암호화한 값
 - k 를 알고 있거나 임의의 k 에 대해 표기할 때는 k 를 생략
- $\Pr[X]$: 사건 X 가 발생할 확률
- $a \xleftarrow[s]{} A$: a 를 집합 A 에서 랜덤하게 선택

Contribution

- 1. Calculate the predicted difference distribution of Speck32/64 with one specific input difference for up to 8 round**
- 2. Make a powerful distinguisher by using Machine Learning**
- 3. Develop a selective key search policy based on Bayesian optimization**
- 4. Apply “Few-shot learning” on cryptographic problems(find good input difference).**

(Multiple) Differential Cryptanalysis

❖ Block Cipher

- 고정된 길이의 평문과 비밀키를 입력으로 하여 동일한 길이의 암호문을 생성
- $F: \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n$
- 고정된 키에 대해서 F_k 로 표기함

❖ Pseudo Random Function

- $\text{Func}_n := \{\text{all function } F \text{ s.t. } F: \{0,1\}^n \rightarrow \{0,1\}^n\}$
- F_k (uniform한 key k)가 Func_n 에서 uniform random하게 선택한 임의의 함수 f 와 구분 불가능(indistinguishable)할 때, F 를 Pseudo random function이라고 한다.

Def) Let $F: \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$. F is a pseudo random function if for all PPT distinguisher D , there is a negligible function negl s.t.

$$|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^f(\cdot)(1^n) = 1]| \leq \text{negl}(n)$$

where the first prob is taken over unif choice of $k \in \{0,1\}^n$ and the second prob is taken over unif choice of $f \in \text{Func}_n$.

(Multiple) Differential Cryptanalysis

❖ 차분 분석(Differential Cryptanalysis)

- 선택 평문 공격(Chosen Plaintext Attack)
 - 공격자는 원하는 평문에 대한 암호문을 얻을 수 있는 상황
- 차분 특성을 통해 랜덤 함수와 암호 알고리즘을 구분함으로써 키 복구를 하는 공격이다.
 - n 비트 평문: $P_1, P_2 (= P_1 \oplus \Delta P)$
 - n 비트 암호문: $C_1 (= Enc(P_1)), C_2 (= Enc(P_2))$
 - Let $\Delta C = C_1 \oplus C_2$
 - $p = \Pr[C_1 \oplus C_2 = \Delta C | P_1 \oplus P_2 = \Delta P]$ notation) $\Pr[\Delta P \rightarrow \Delta C]$
 - Enc 함수가 랜덤 함수인 경우 $p = 1/2^n$
 - 암호 알고리즘의 경우 $p > 1/2^n$ 인 경우가 존재함
 - 이러한 특징을 통해 랜덤 함수와 암호알고리즘을 구분
 - 이때, $(\Delta P, \Delta C)$ 를 암호알고리즘의 차분특성이라 함

(Multiple) Differential Cryptanalysis

❖ 차분 분석(Differential Cryptanalysis)

- 용어 정리
 - 차분(difference) : 어떤 두 값을 Xor 한 값
 - 차분(differential) : 입력 차분, 출력 차분으로 구성 ($\Delta P, \Delta C$)
 - 차분 특성(differential characteristic) : 입력 차분, 출력 차분 뿐만 아니라 중간 과정의 모든 차분(difference)을 포함 ($\Delta P, \Delta P_1, \dots, \Delta P_{r-1}, \Delta C$)
- Q) 차분에 대한 확률은 어떻게 구할 것인가?
 - Ans) 차분(differential)에 대한 정확한 확률은 구하기 어렵다.
→ 해당 차분을 만족하는 차분특성의 최대확률(차분 확률의 lower bound)을 구하여 어느정도 가늠
- Q) 차분 특성을 어떻게 찾을 것인가?
 - Ans) 여러가지 방법이 존재함. 최근 연구에서는 MILP, SAT problem등의 Optimization 문제 해결을 통해 차분 특성을 찾는 방법이 제시됨.

(Multiple) Differential Cryptanalysis

❖ 차분의 전파

- SPN 구조의 블록암호
 - 라운드 함수가 Key Xor, S-box, Permutation으로 구성됨
 - Key Xor은 차분을 변화시키지 않음
 - Permutation은 차분을 확률 1로 변화시킴
 - Permutation 뿐만 아니라 xor에 대하여 선형으로 정의되는 모든 확산 계층은 모두 비슷하게 확률 1로 차분을 변화시킨다.
 - S-box에 대해서는 확률적으로 차분을 변화시킴
- ARX 구조의 블록암호
 - 라운드 함수가 Key Xor, Word Xor, Modulo Addition, Rotation으로 구성됨
 - SPN구조와 마찬가지로 Xor에 대해 선형으로 정의되는 계층은 확률 1로 차분을 변화시킴
 - Modulo Addition에 대해서는 확률적으로 차분을 변화시킴

(Multiple) Differential Cryptanalysis

❖ S-box에서의 차분

▪ Differential Distribution Table(DDT)

- S-box에서 입력 차분에 대해 출력 차분이 몇 번 나오는지를 나타낸 표
- $(\Delta i, \Delta o)$ 번째 원소는 $|\{x \in \{0,1\}^m : S(x) \oplus S(x \oplus \Delta i) = \Delta o\}|$
- Ex. Present 암호 알고리즘의 DDT c.f.) m비트 입력 S-box

[16, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 4, 0, 0, 0, 4, 0, 4, 0, 0, 0, 4, 0, 0]
[0, 0, 0, 2, 0, 4, 2, 0, 0, 0, 2, 0, 2, 2, 2, 0]
[0, 2, 0, 2, 2, 0, 4, 2, 0, 0, 2, 2, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 4, 2, 2, 0, 2, 2, 0, 2, 0, 2, 0]
[0, 2, 0, 0, 2, 0, 0, 0, 0, 2, 2, 2, 4, 2, 0, 0]
[0, 0, 2, 0, 0, 0, 2, 0, 2, 0, 0, 4, 2, 0, 0, 4]
[0, 4, 2, 0, 0, 0, 2, 0, 2, 0, 0, 0, 2, 0, 0, 4]
[0, 0, 0, 2, 0, 0, 0, 2, 0, 2, 0, 4, 0, 2, 0, 4]
[0, 0, 2, 0, 4, 0, 2, 0, 2, 0, 0, 0, 2, 0, 4, 0]
[0, 0, 2, 2, 0, 4, 0, 0, 2, 0, 2, 0, 0, 2, 2, 0]
[0, 2, 0, 0, 2, 0, 0, 4, 2, 2, 2, 0, 2, 0, 0, 0]
[0, 0, 2, 0, 0, 4, 0, 2, 2, 2, 0, 0, 0, 2, 0, 0]
[0, 2, 4, 2, 2, 0, 0, 2, 0, 0, 2, 2, 0, 0, 0, 0]
[0, 0, 2, 2, 0, 0, 2, 2, 2, 0, 0, 0, 2, 2, 0, 0]
[0, 4, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 4, 4, 0]

- DDT로 특정 입력 차분에 대해 특정 출력 차분이 나올 확률 계산 가능
 - Ex. Present 암호의 S-box의 입력 차분 0x1이 출력 차분 0x3이 나올 확률은 $\frac{4}{16}$

(Multiple) Differential Cryptanalysis

❖ Modulo Addition에서의 차분

- S-box 연산에서는 한 개의 입력에 대해 하나의 출력
- Modulo Addition 연산은 두개의 입력에 대해 하나의 출력
- $x dp^+(\alpha, \beta \rightarrow \gamma) := 2^{-2m} |\{(x, y) : ((x \oplus \alpha) \boxplus (y \oplus \beta)) \oplus (x \boxplus y) = \gamma\}|$
(i.e. 입력 차분이 α, β 일때 출력 차분 γ 가 나올 확률) c.f.) m비트 word 단위 Modulo Addition

Theorem

$$x dp^+(\alpha, \beta \rightarrow \gamma) = 2^{-\sum_{i=0}^{m-2} \neg eq(\alpha_i, \beta_i, \gamma_i)}$$

$$\text{where } eq(\alpha_i, \beta_i, \gamma_i) = \begin{cases} 1 & \text{if } \alpha_i = \beta_i = \gamma_i \\ 0 & \text{otherwise} \end{cases}$$

- 위 Thm을 사용하여 확률을 계산
($\alpha_i, \beta_i, \gamma_i$ 의 각 비트 값이 다른 것의 개수를 세면 된다.)

(Multiple) Differential Cryptanalysis

❖ 차분 확률 계산

- 차분 $(\Delta i, \Delta o)$ 가 만족할 확률 $p = \Pr[\Delta i \rightarrow \Delta o]$ 구하는 방법
 - 입력 차분 Δi
 - 첫번째 라운드 입력 차분 Δi 이 출력 차분 Δi_1 이 될 확률 p_1
 - 두번째 라운드 입력 차분 Δi_1 이 출력 차분 Δi_2 이 될 확률 p_2
 - ...
 - 마지막 라운드 입력 차분 Δi_{r-1} 이 출력 차분 Δo 이 될 확률 p_{r-1}
 - $p \geq \prod_{i=1}^{r-1} p_i$ c.f. 각 라운드에서 차분이 전파되는 사건이 독립이라는 가정!
- 각 라운드의 차분 $(\Delta i, \Delta o)$ 이 만족할 확률 구하는 방법
 - 라운드 함수의 각 부분에서 차분이 변화되는 확률을 구하여 곱함

(Multiple) Differential Cryptanalysis

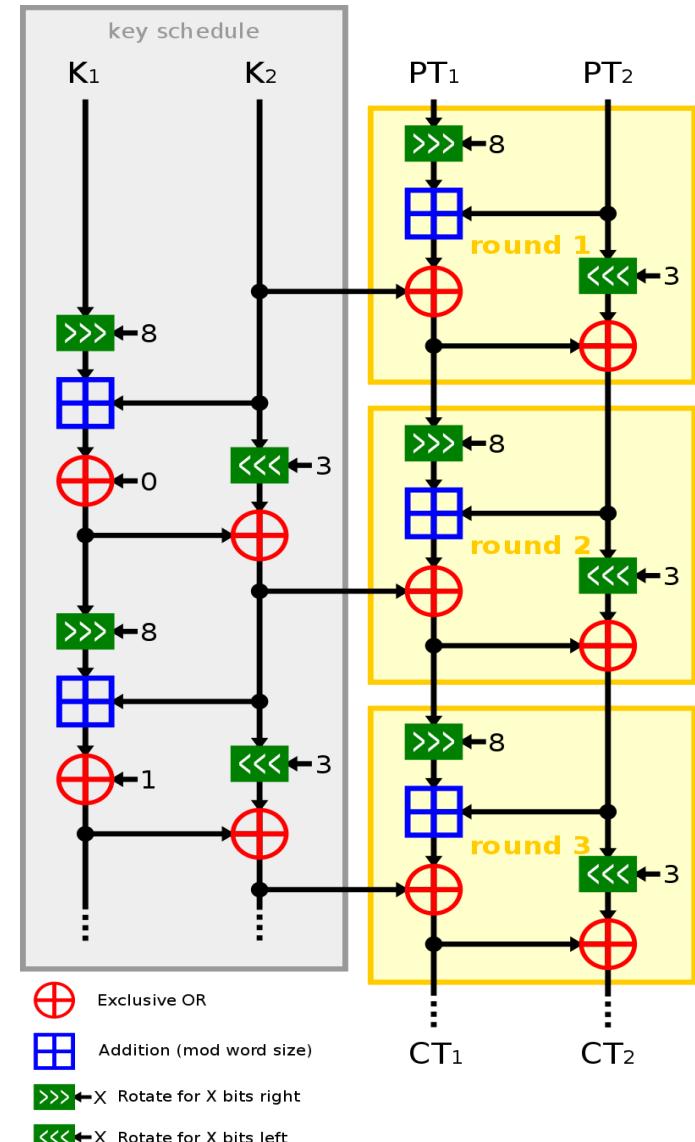
❖ 블록암호 Speck

- ARX(Addition, Rotation, Xor)구조 암호
- $2n$ 비트 평문 블록을 mn 비트 키를 사용하여 암호화 (Speck $2n/mn$ 으로 표기)
- 각 라운드마다 n 비트 라운드 키가 Xor 됨

<Speck32/64의 파라미터>

Parameters	Value
Block	32
Key size	64
Word size	16
Key words	4
α	7
β	2
rounds	22

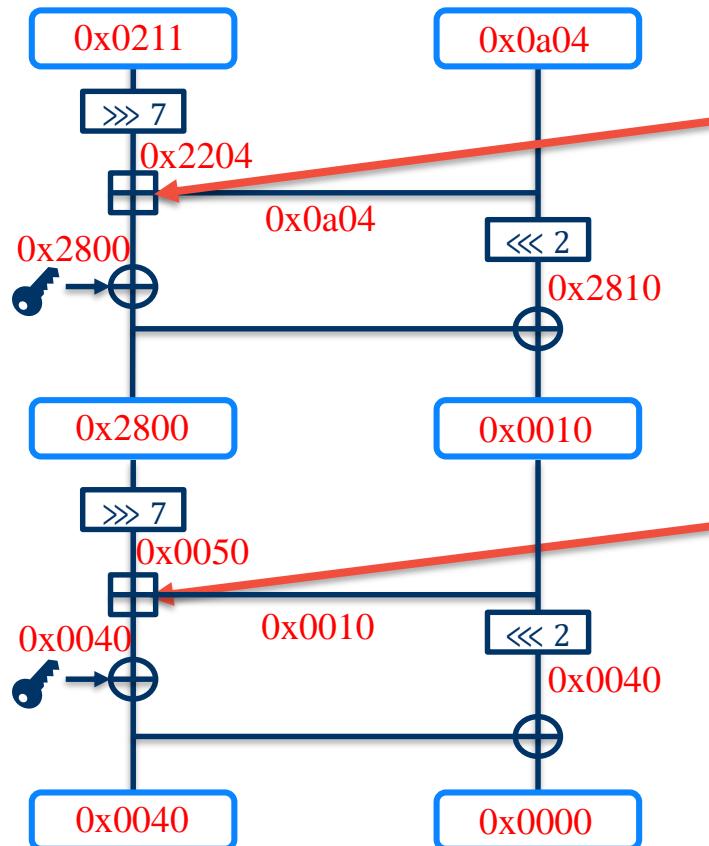
* α, β 는 cyclic rotation 연산에 대한 파라미터



(Multiple) Differential Cryptanalysis

❖ 간단한 예제

- ARX구조인 Speck 암호에 대해 다룰 것이므로 Speck32/64에 대한 예를 사용
- 예제의 차분은 높은 확률을 갖도록 하기 위해 찾은 값(방법은 다양함)



$$p = 2^{-\sum_{i=0}^{14} \neg eq(0x2204, 0xa04, 0x2800)}$$

0x2204 = 0010 0010 0000 0100

0xa04 = 0000 1010 0000 0100

0x2800 = 0010 1000 0000 0000

$$\Rightarrow p = 2^{-4}$$

$$p = 2^{-\sum_{i=0}^{14} \neg eq(0x0050, 0x0010, 0x0040)}$$

0x0050 = 0000 0000 0101 0000

0x0010 = 0000 0000 0001 0000

0x0040 = 0000 0000 0100 0000

$$\Rightarrow p = 2^{-2}$$

$$\therefore \Pr[(0x0211_0a04) \rightarrow (0x0040_0000)] = 2^{-6}$$

(Multiple) Differential Cryptanalysis

❖ 키 복구

- 과정
 - 높은 확률을 갖는 r 라운드에 대한 차분 $(\Delta i, \Delta o)$ 를 찾음
 - $r + 1$ 라운드에 적용되는 라운드 키를 추측함
 - 추측한 키로 암호문 쌍들을 부분 복호화 함
 - 부분 복호화로 얻은 값들의 차분이 Δo 인지 확인
 - Δo 인 경우 추측한 키를 옳은 키 후보로 카운트
 - 가장 많이 카운트 된 키를 옳은 키로 추측함
- 데이터 복잡도
 - 공격 과정에 필요한 (평문, 암호문)의 개수
 - 적절히 작은 상수 c 에 대해 $N = c/p$ 개 정도의 평문 쌍과 그에 해당하는 암호문 쌍이 있으면 공격 가능
 - 전체 평문의 개수 2^n 을 넘지 않아야 함

(Multiple) Differential Cryptanalysis

❖ 키 복구

① $\Delta = 0x0211_0a04$



② C_1, C_2

- ④ $C'_1 \oplus C'_2 = 0x0040_0000$ 이면,
이때 사용된 키의 카운트 1 증가
- 추측하는 라운드 키가 m 비트 일 때
wrong key : $N/2^m$ 카운트
right key : $c + N/2^m$ 카운트
- ⑤ 가장 많이 카운트 된 키를 마지막 라운드 키로 추측함.

* 암호화에 사용된 키가 아닌 다른 키로 암호화
되는 함수는 랜덤 함수라고 가정함.

(Multiple) Differential Cryptanalysis

❖ Multiple Differential Cryptanalysis(MDC)

- Notation

- $\Delta_0 := \{\delta_0 : \exists \delta_r \text{ s.t. } (\delta_0, \delta_r) \in \Delta\}$
- $\Delta_r^{(i)} := \{\delta_r : (\delta_0^{(i)}, \delta_r) \in \Delta\}$
- 블록 암호 E 의 라운드 함수를 F 로 표기함.

- DC의 일종으로 여러 개의 차분을 한번에 고려한 공격
- 차분 집합 $\Delta = \{(\delta_0^{(0)}, \delta_r^{(0)}), \dots, (\delta_0^{(n_1)}, \delta_r^{(n_2)})\}$ 의 모든 차분을 동시에 고려해서 키를 복구함

(Multiple) Differential Cryptanalysis

❖ Multiple Differential Cryptanalysis(MDC)

Algorithm. Multiple Differential Cryptanalysis

Input : N chosen plaintext/ciphertext (x_i, y_i) s.t. $y_i = E_{k^*}(x_i)$

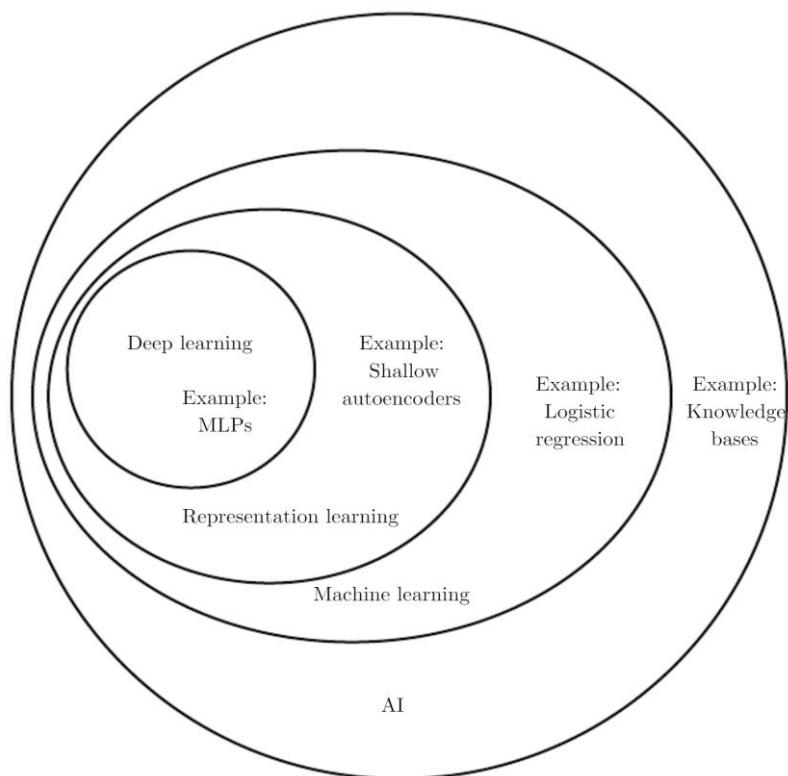
Output : The Key used to encipher the samples

1. Set a table D of 2^{n_k} counter to 0
2. For each $\delta_0^{(i)} \in \Delta_0$
 1. for (x_a, x_b) s.t. $x_b = x_a \oplus \delta_0^{(i)}$
 1. if $y_a \oplus y_b \in \Delta_{r+1}^{(i)}$
 1. for k
 1. compute $\delta = F_k^{-1}(y_a) \oplus F_k^{-1}(y_b)$
 2. if $\delta \in \Delta_r^{(i)}$, then $D[k] = D[k] + 1$
 3. Generate a list L of the l candidate with the highest value of $D[k]$
 4. For $k \in L$
 1. for possible master key K corresponding to k
 1. if $E_K(x) = y = E_{K^*}(x)$, then return K

머신러닝

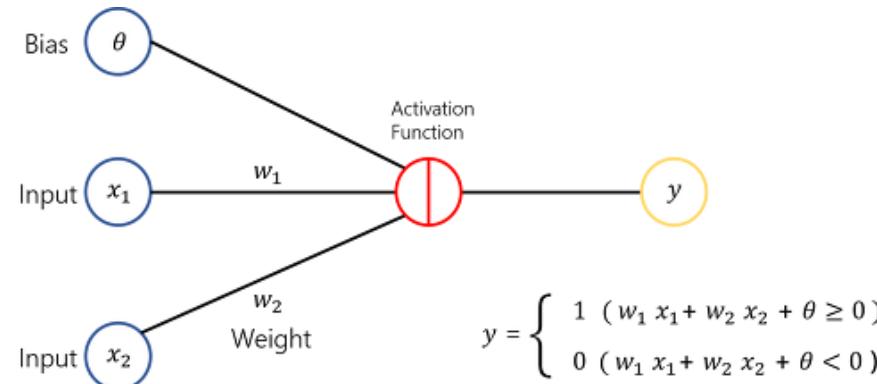
- ❖ A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E .

- *Mitchell, T. (1997). Machine Learning.* -

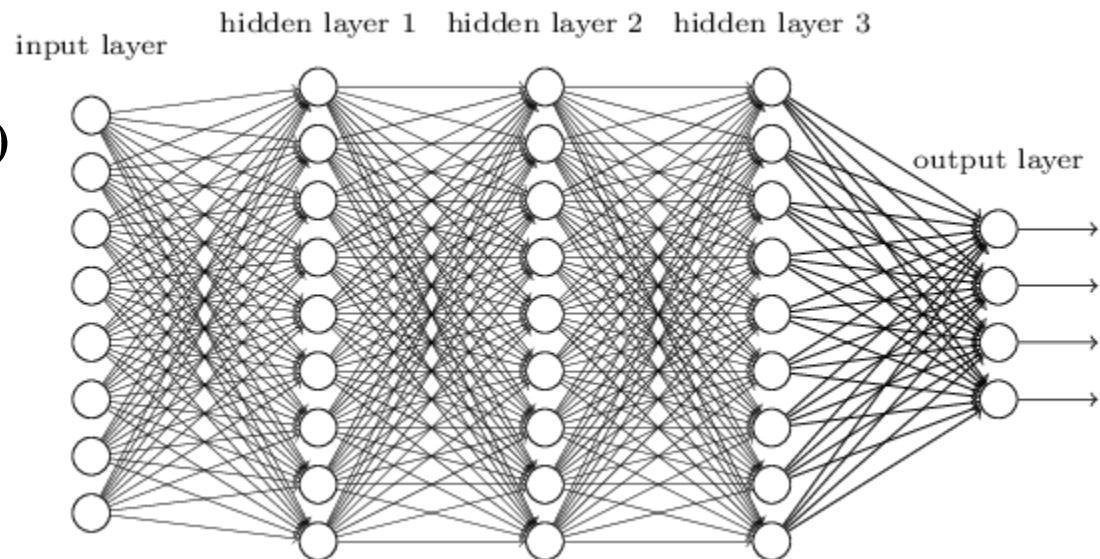


Deep Neural Network

❖ Perceptron

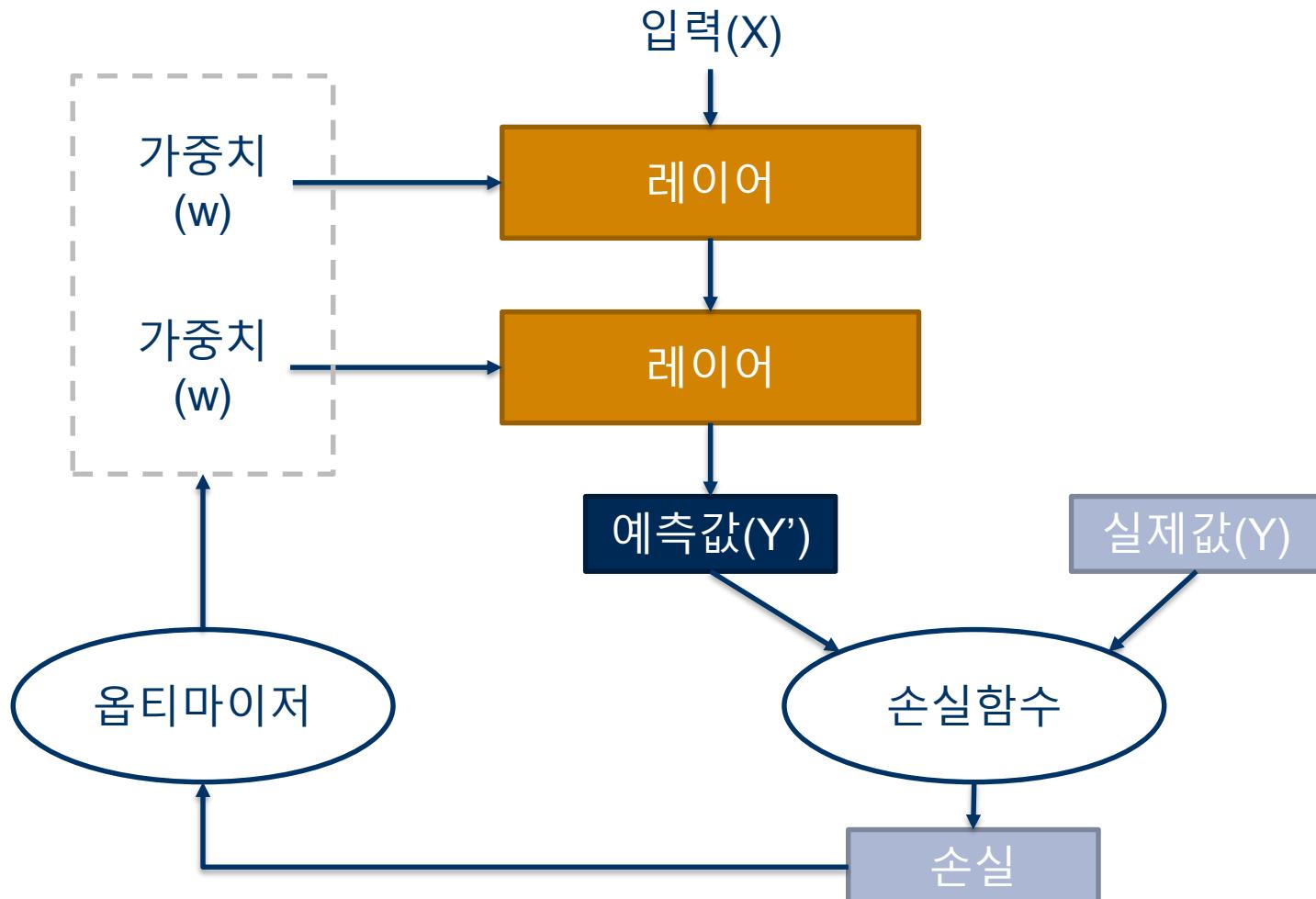


❖ DNN (Deep Neural Network)



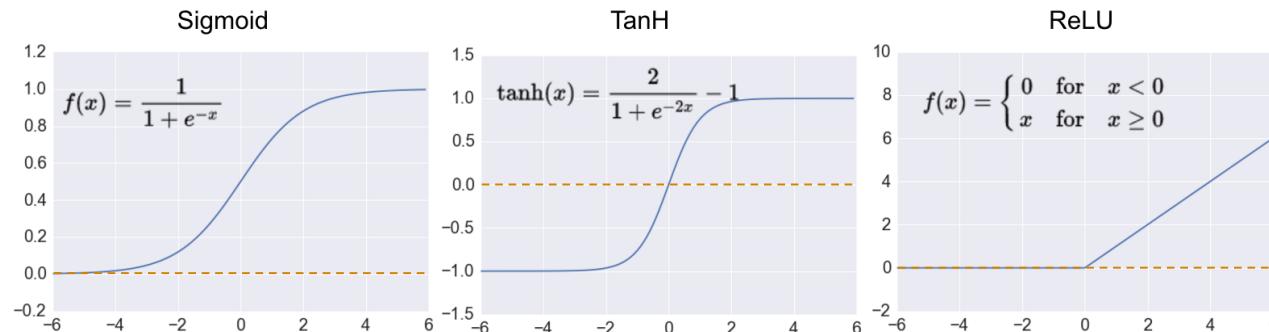
Deep learning

❖ 네트워크 학습을 위한 역전파(Backpropagation) 과정



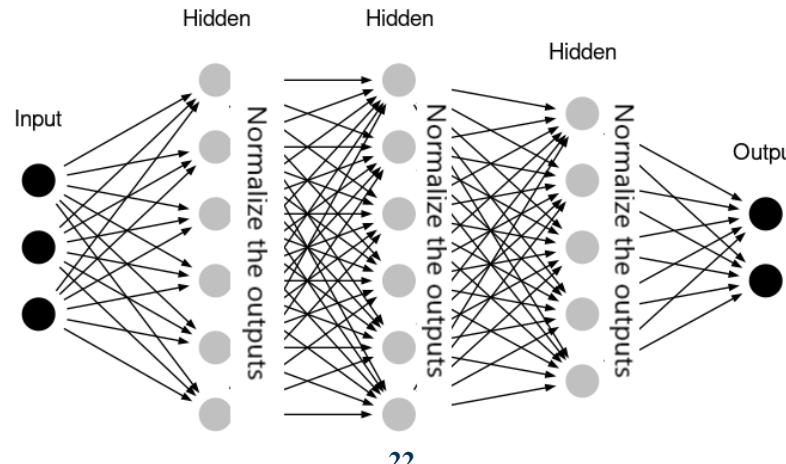
Deep Learning

❖ Activation Function



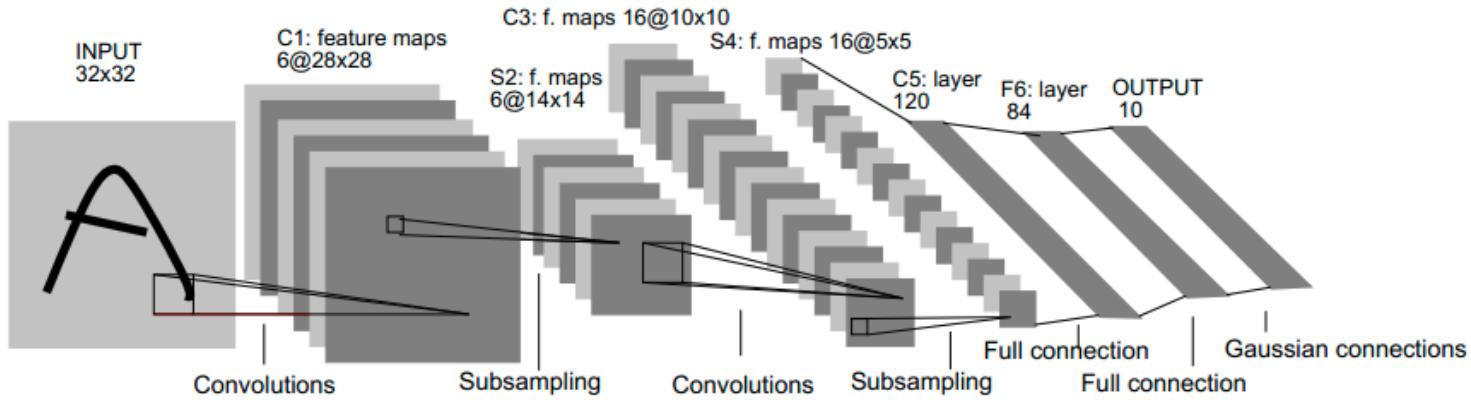
❖ Batch Normalization

- Gradient Vanishing / Gradient Exploding을 막기 위한 기법
- 각 레이어의 출력을 정규화하여 다음 레이어의 입력으로 전달



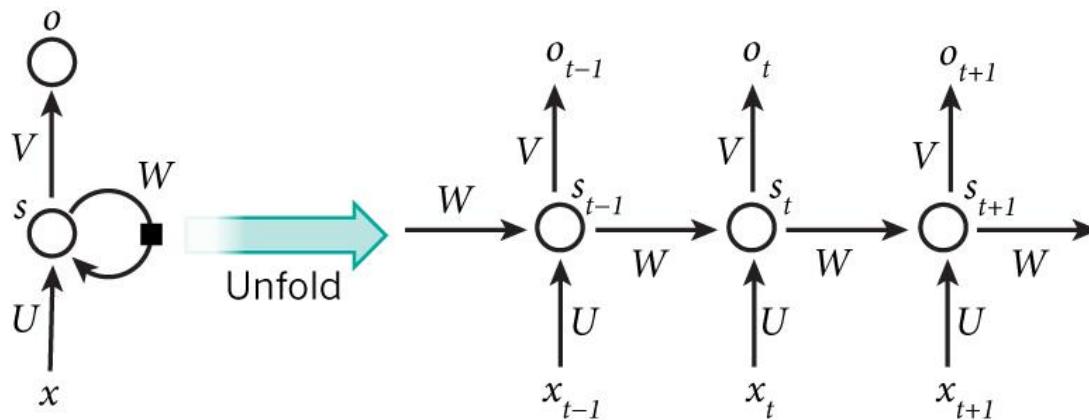
대표적 딥러닝 모델

❖ CNN(Convolutional Neural Network)



LeNet 5, Yann Lecun, 1998

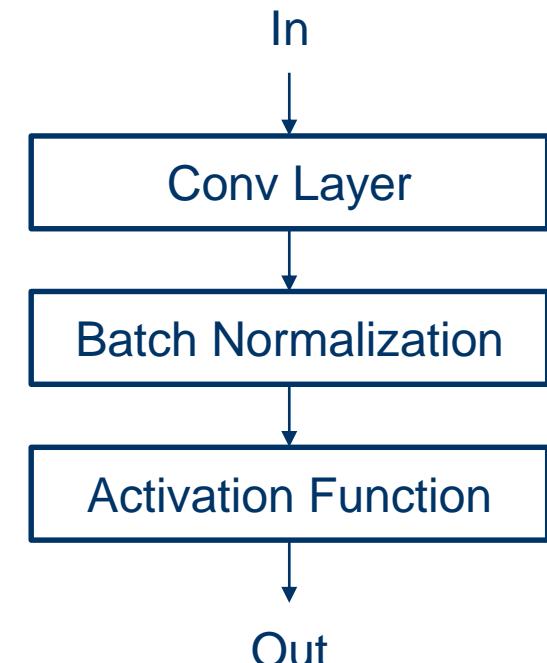
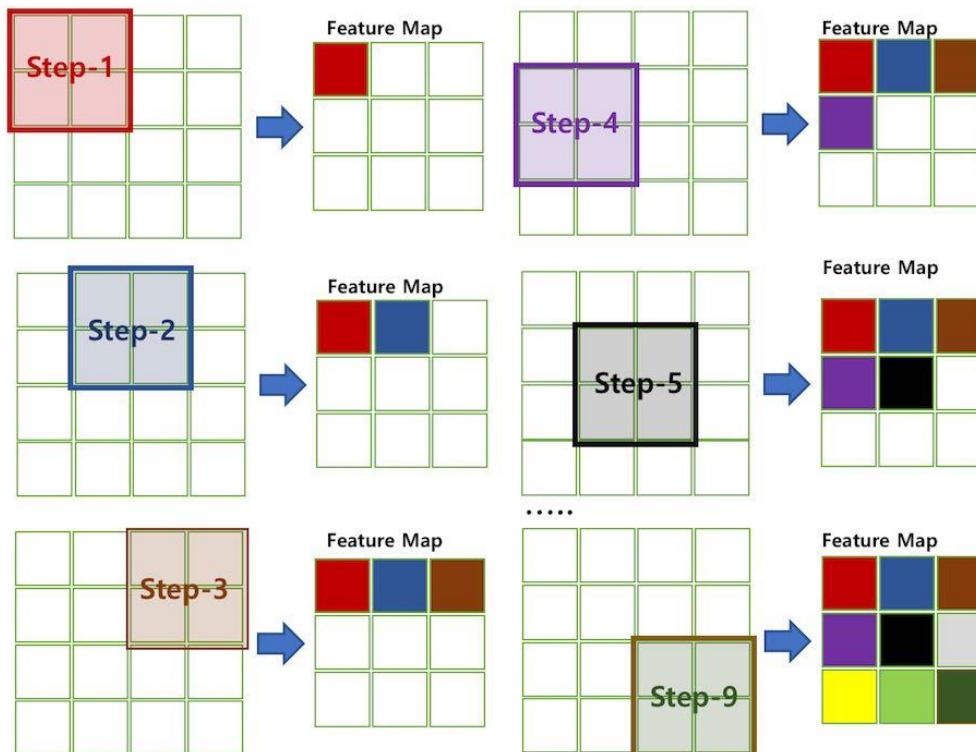
❖ RNN(Recurrent neural networks)



Basic Convolutional Neural Network

❖ Convolution Layer

- 레이어를 완전연결하지 않고 local input 만을 연결하여 단순화
- 학습가능한 필터를 이용하여 이미지 내 특징 추출

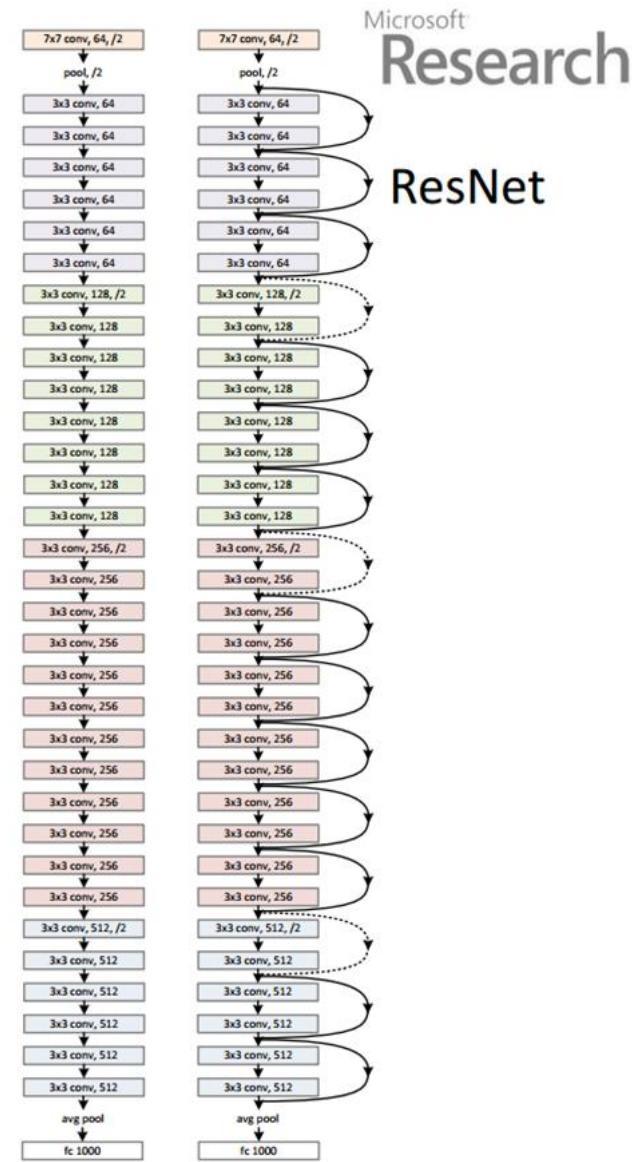
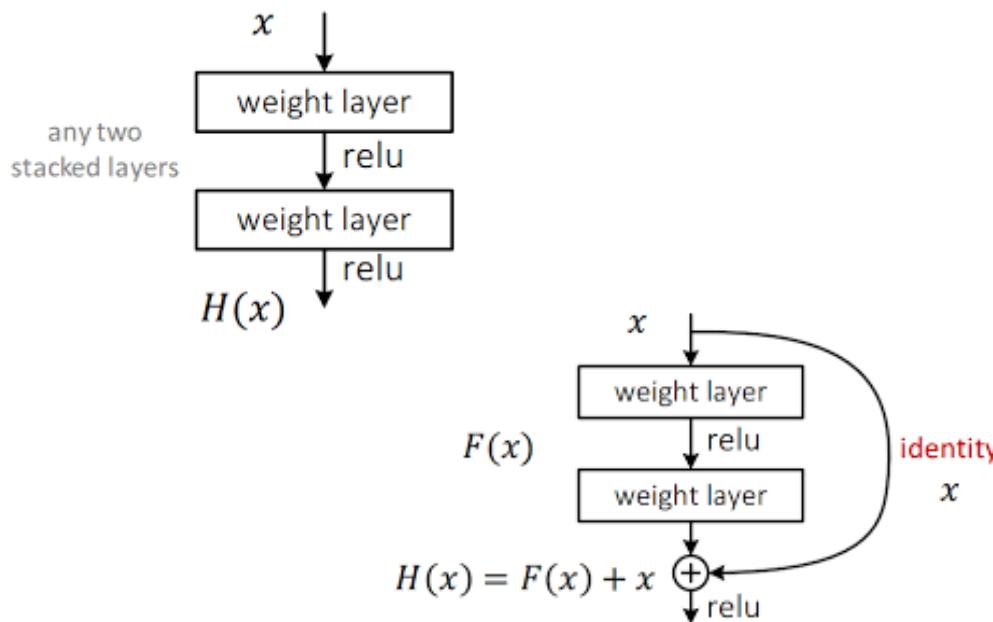


Basic convolution block

Resnet

- ❖ Gradient Vanishing 현상을 회피하기 위해 기존의 Network에 skip connection 추가

- K. He. 등이 2015년 제안
 - Gradient Vanishing : 네트워크의 깊이가 깊어지면, 학습이 어려워지는 현상



Deep Learning을 사용한 차분 공격

❖ DL 차분 공격 개요

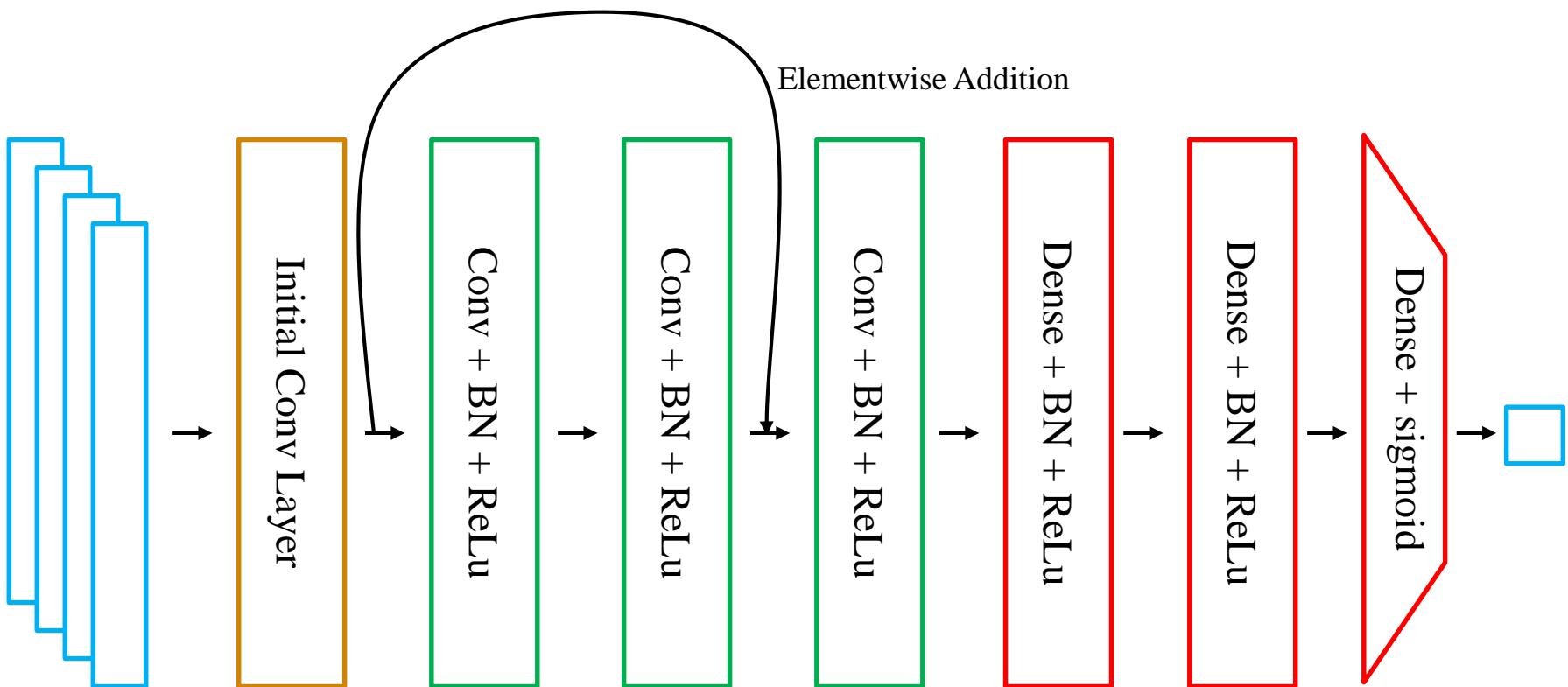
- 공격 대상 알고리즘 : Speck 32/64
- Deep Learning을 통해 암호문과 랜덤 함수를 구분하는 Distinguisher를 생성
 - Neural Distinguisher 또는 ND라고 표기
 - 딥러닝 모델이 입력 차분이 1개인 MDC를 학습한 것처럼 보임
- ResNet 모델을 사용함
 - 실험 결과 Residual Block이 1개인 모델과 10개인 모델의 결과 차이가 크지 않음
 - DNN 모델에 비해서는 좋은 결과를 얻을 수 있음.
- Deep Learning 모델을 통해 11라운드 Speck32/64에 대한 키 복구
 - 7라운드 ND를 사용
 - Neutral bit를 사용하여 초기 2라운드 확장
 - 1라운드를 추가로 확장
 - 첫 라운드 키 연산 뒤에 non-linear 연산(S-box 연산)이 없음
 - 이에 따라 한 라운드 후의 차분이 특정 차분이 되게 하도록 평문 선택 가능
 - 위 과정을 통해 10라운드 Distinguisher 생성
 - 11라운드 라운드 키를 복구함
 - Bayesian Optimization이라는 딥러닝 기술을 사용

Deep Learning을 사용한 차분 공격

❖ Neural Distinguisher 모델 생성

- ResNet 모델 사용

- Speck32/64에 대해서는 Residual Block의 개수가 정확도에 영향이 없음
- 따라서 1개의 Residual block을 갖는 ResNet 생성



Deep Learning을 사용한 차분 공격

❖ Neural Distinguisher 학습

- 학습 데이터 생성

1. For $i = 1$ to N

$$P_i \xleftarrow[s]{\delta} \text{Plaintext Space}$$

$\widehat{P}_i = P_i \oplus \Delta P$ (Speck 32/64에 적용하는 경우 $\Delta P = 0x0040_0000$ 사용)

2. $Y \xleftarrow[s]{\delta} \mathbb{Z}_2^N$ (Y 의 i 번째 성분을 Y_i 로 표기)

3. For $i = 1$ to N

If $Y_i = 0$

$$(C_i, \widehat{C}_i) \xleftarrow[s]{\delta} \text{Ciphertext Space}^2$$

Else if $Y_i = 1$

$$k \xleftarrow[s]{\delta} \text{Key Space}$$

$$(C_i, \widehat{C}_i) = \left(Enc_k(P_i), Enc_k(\widehat{P}_i) \right)$$

4. Let D be a length N vector s.t. $D_i = (P_i, \widehat{P}_i, C_i, \widehat{C}_i)$

- D, Y 를 학습 데이터로 사용함

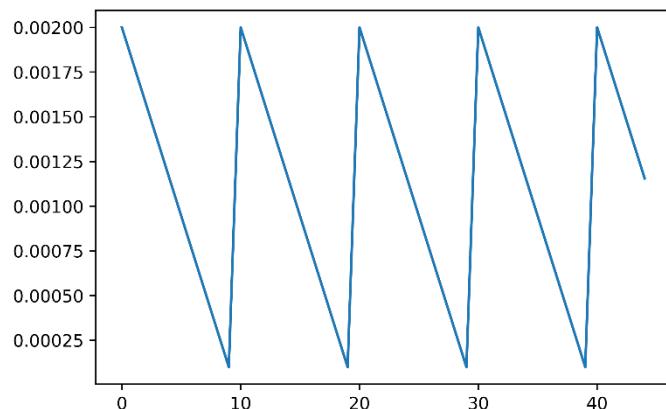
- 모델의 입력 : D
- 모델의 출력 : Y

Deep Learning을 사용한 차분 공격

❖ Neural Distinguisher 학습

▪ 학습 파라미터

- Epochs : 200
- Batch size : 5000
- Train data : 10^7 개
- Validation data : 10^6 개
- L2 regularization 사용($c = 10^{-5}$)
- Cyclic learning rate 사용
 - i 번째 epoch에서 learning rate를 l_i 라 할 때, $l_i = \alpha + \frac{(n-i) \bmod n+1}{n}(\beta - \alpha)$
 - Speck32/64에 대한 모델에서는 $n = 9, \alpha = 10^{-4}, \beta = 2 \cdot 10^{-3}$ 으로 사용



Deep Learning을 사용한 차분 공격

❖ Neural Distinguisher 학습

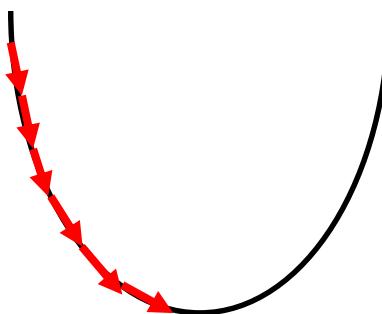
- Batch Size
 - 일반적으로 Batch Size와 학습 속도는 비례, 성능은 반비례한다.
(Batch \downarrow 속도 \downarrow 성능 \uparrow / Batch \uparrow 속도 \uparrow 성능 \downarrow)
 - 최근 연구에서 BN을 쓰는 경우 batch size가 지나치게 작으면 학습이 불안정해져서 성능이 낮아진다는 결과가 있음
 - 적절한 batch size를 여러 번의 실험을 통해 찾아야 함
- Regularization
 - Regularization은 Overfitting을 방지하는 기법
 - L1 Regularization
 - 작은 가중치는 0이 되도록 함
 - 일반적으로 의미 없는 값을 제거하는 데 목적이 있음
 - L2 Regularization
 - 특정 가중치가 비이상적으로 커져서 학습에 영향을 주는 것을 방지

Deep Learning을 사용한 차분 공격

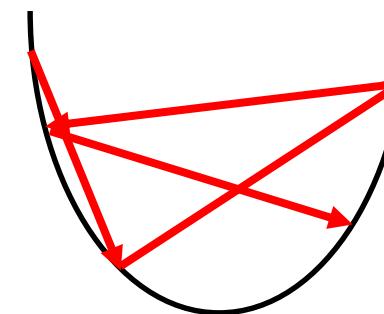
❖ Neural Distinguisher 학습

▪ Cyclic Learning Rate

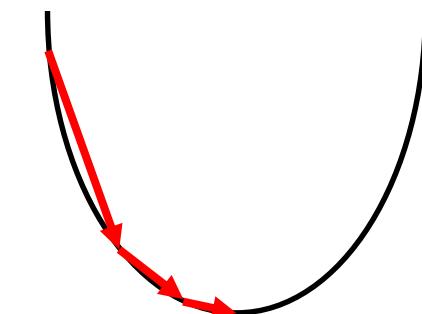
- Learning rate는 Optimization 문제 해결 과정 중 한 스텝에서 다음 스텝까지의 차이라고 이해 가능
- Cyclic learning rate는 Learning rate 값을 주기적으로 변화시키며 학습



lr 이 너무 작은 경우



lr_0 이 너무 큰 경우



lr 을 변경시켜 가면서 사용한 경우

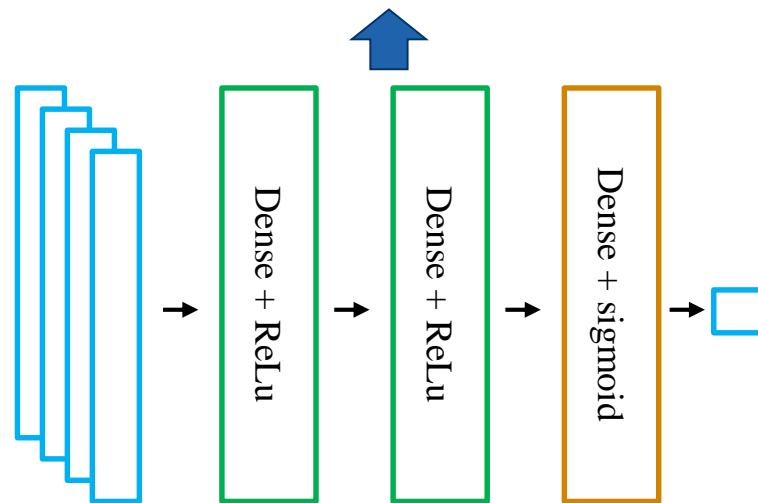
Deep Learning을 사용한 차분 공격

❖ Neural Distinguisher 학습

▪ 학습 결과

- 10^6 개 Sample로 Accuracy 측정

	DNN	ResNet 1	ResNet 10
5 round	0.882	0.926	0.927
6 round	0.691	0.784	0.787
7 round	0.521	0.608	0.610



Deep Learning을 사용한 차분 공격

❖ Neural Distinguisher를 사용한 키 복구

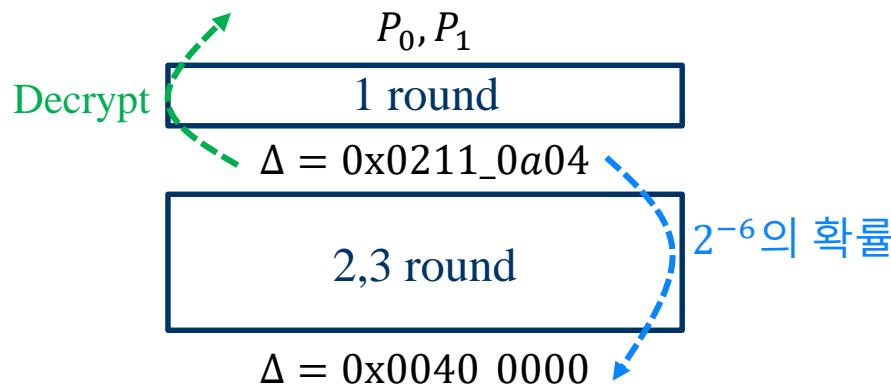
- 7 라운드 ND를 사용해서 10라운드 Distinguisher 생성
- 10라운드 Distinguisher를 사용하여 11라운드로 축소된 Speck32/64의 11번째 라운드 키 복구

1. 1 round 확장

- 첫 라운드 키 연산 이후에 Non-linear 연산이 없음
- 한 라운드 이후의 차분이 특정 차분이 되도록 생성 가능
 - 1 라운드 이후의 값이 특정 차분을 만족하도록 생성
 - 임의의 키(0을 사용해도 됨)를 사용해서 1라운드 복호화

2. 2 round 확장

- 2라운드에 대해 $\Pr[(0x0211_0a04) \rightarrow (0x0040_0000)] = 2^{-6}$ 임을 사용

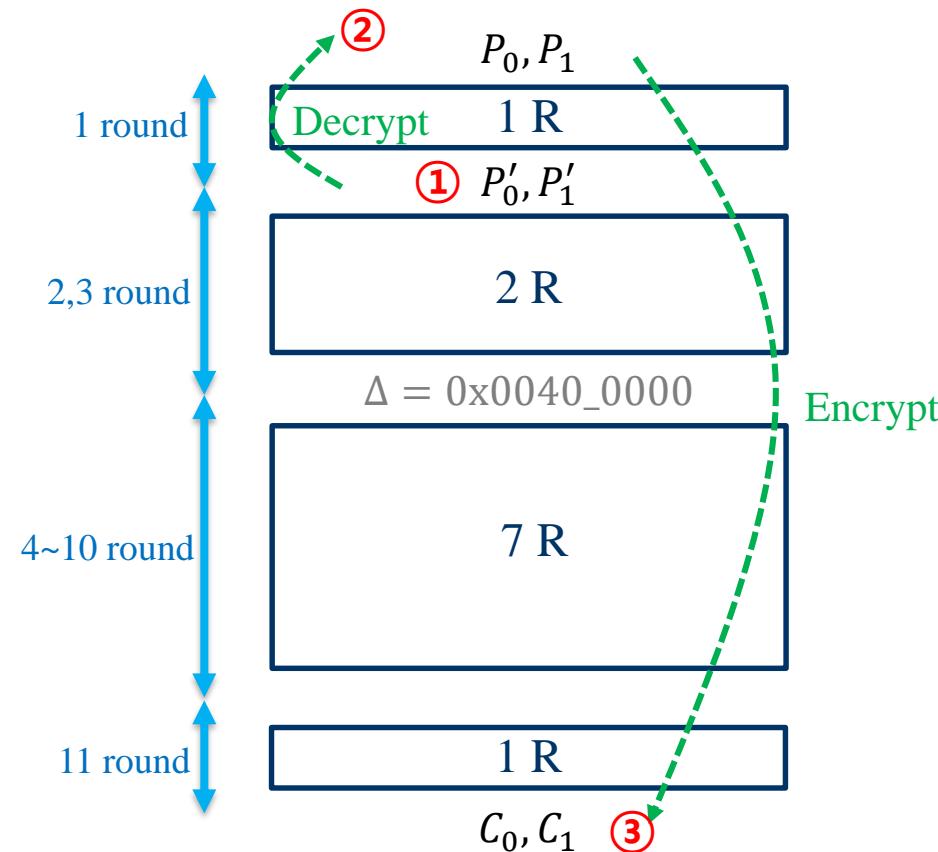


Deep Learning을 사용한 차분 공격

❖ Neural Distinguisher를 사용한 키 복구

- 데이터 생성

- $P'_0 \xleftarrow[s]{\sim} \{0,1\}^{32}, P'_1 = P'_0 \oplus \Delta P$ ($\Delta P = 0x0211_0a04$)
- 임의의 라운드키로(0을 사용) Decrypt 1 round $\Rightarrow P_0, P_1$
- Encrypt 11 round $\Rightarrow C_0, C_1$



Deep Learning을 사용한 차분 공격

❖ Neural Distinguisher를 사용한 키 복구

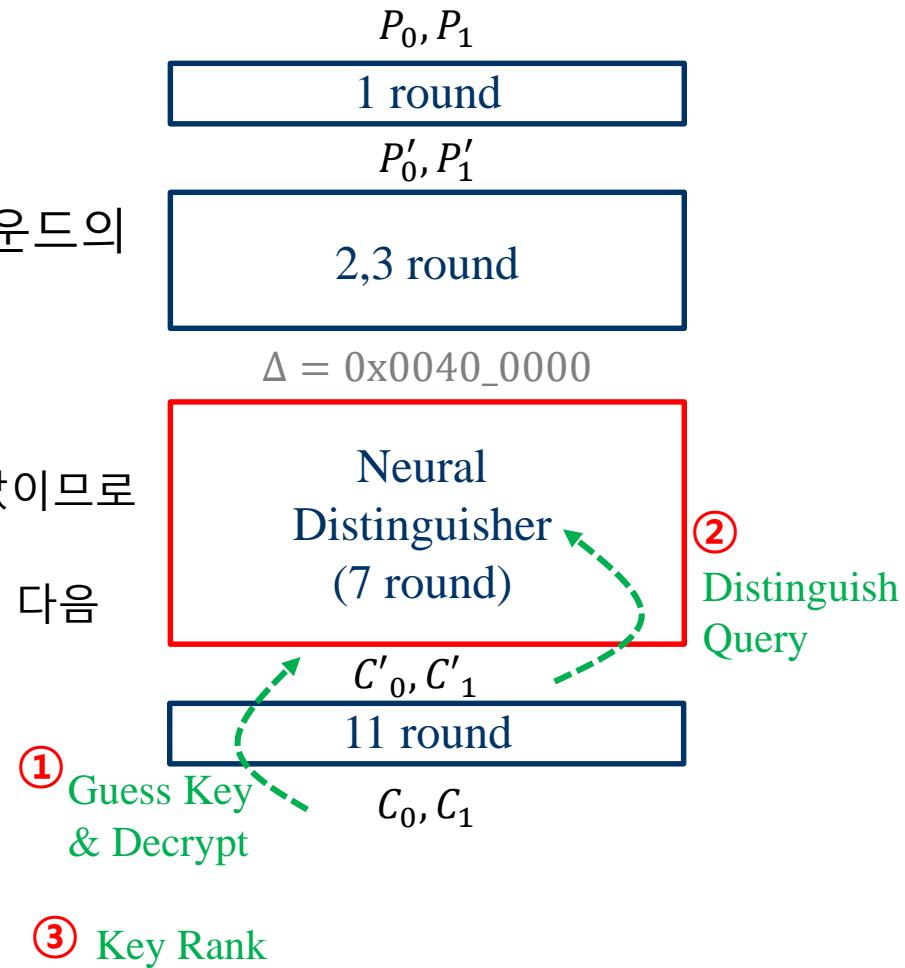
▪ 키 복구

1. Key를 추측해서 1라운드 복호화
2. 7 라운드 ND에 query
3. Key값에 대한 rank value 계산
4. 가장 높은 값을 갖는 Key를 11라운드의 라운드 키로 추측함.

▪ 키 랭크

- ND의 결과값이 Sigmoid 함수의 결과값이므로 0에서 1 사이의 값
- $-\infty$ 에서 ∞ 까지의 값을 갖게 하기 위해 다음 함수로 rank value 계산

$$v = \sum_{i=1}^N \log_2 \left(\frac{z_i}{1 - z_i} \right)$$



Deep Learning을 사용한 차분 공격

❖ Neural Distinguisher를 사용한 키 복구

▪ Bayesian Key Search

- 적절한 파라미터(Batch size, learning rate 등등)을 선택하는 것은 어려움
- 이때 Random Search, Grid Search, Bayesian Search 등이 사용됨
- Deep Learning을 사용한 차분공격에서는 파라미터를 찾는 것이 아니라 키를 찾는데 Bayesian search라는 기법을 사용함
- 현재까지 조사된 값들을 바탕으로 목적 함수를 최적화 하는 다음 후보 값을 확률적 추정 결과를 통해 찾음
- 확률적 추정 결과를 얻기 위해 미리 학습된 자료 필요
 - 실험 결과 right key와 hamming distance가 작을 수록 rank value가 크다는 사실을 확인
 - hamming distance에 따른 rank value 값의 평균과 표준편차를 미리 계산 (hamming distance가 h 일 때 평균 μ_h , 표준편차 σ_h 로 표기)
 - 키 후보 k_1, \dots, k_n 에 대해서 실제 사용된 키, k_i 에 대한 rank value의 평균 m_{k_i}

$$\sum_{i=1}^n \frac{(m_{k_i} - \mu_{k_i \oplus k})^2}{\sigma_{k_i \oplus k}^2}$$

가 작게 나오는 k 를 다음 후보 키로 사용함.

Deep Learning을 사용한 차분 공격

❖ 결과

1. Bayesian Key Search 사용하지 않은 경우

방법	Computational	Data
Differential	2^{46}	2^{13}
ND	2^{38}	$2^{13.2}$

- ND의 경우 Computational complexity는 SIMD로 구현된 speck을 계산하는 시간과 비교함
- ND의 경우 실험적으로 100번의 시도 중 99번 키 복구 성공

2. Bayesian Key Search 사용한 경우

- 단일 Thread에서 500초
- 100개의 암호문에 대해 실험적으로 52.1% 키 복구 성공