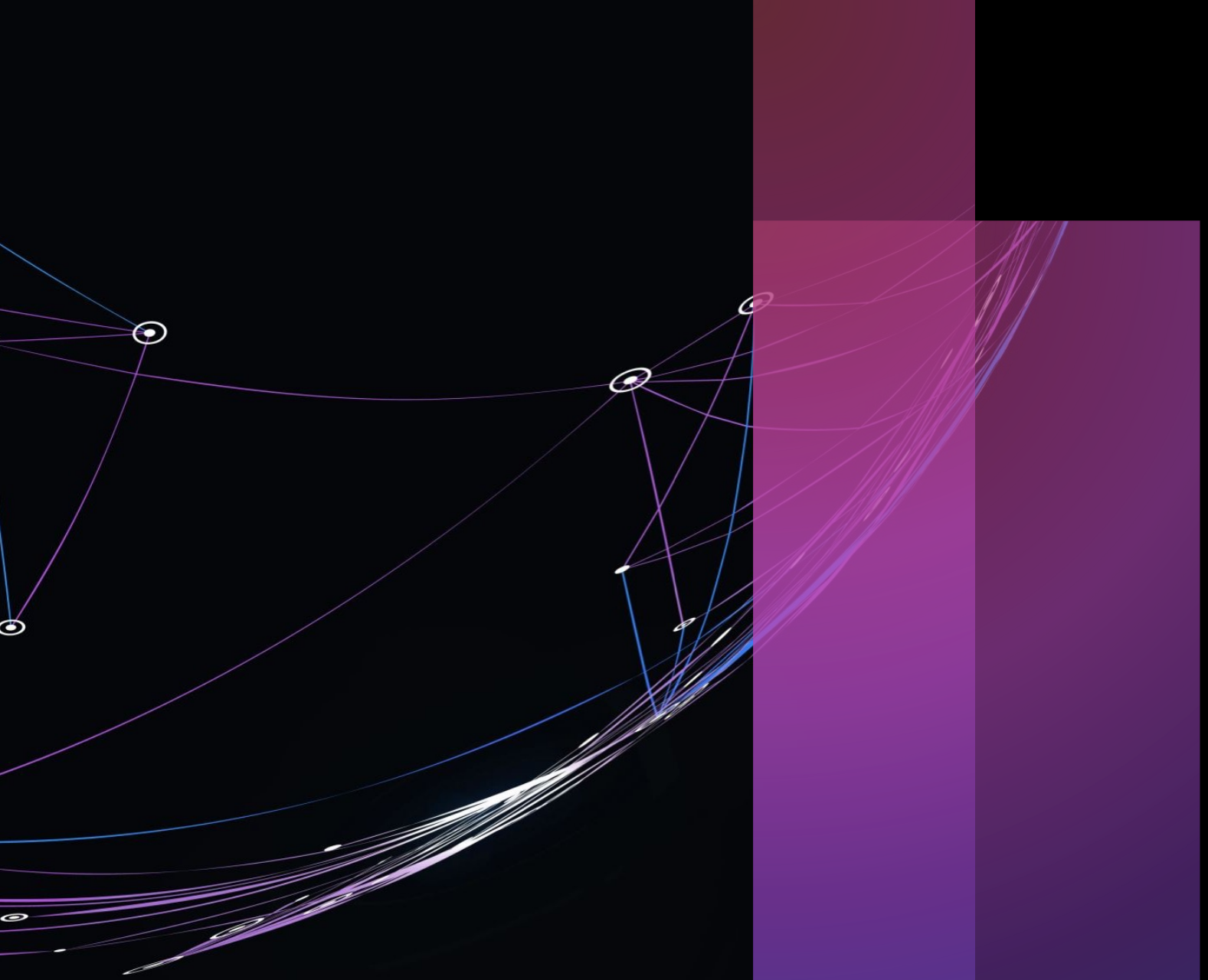
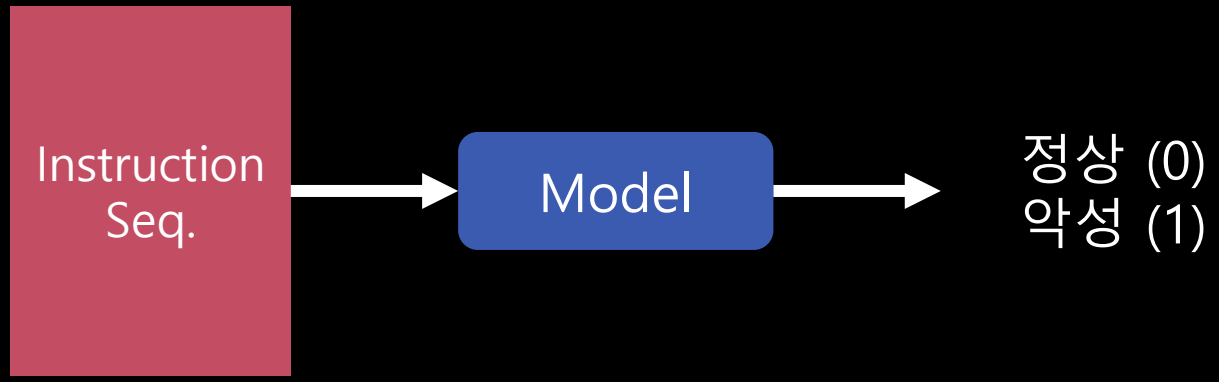


AI-based Malware Detection Final

박준영
정상윤

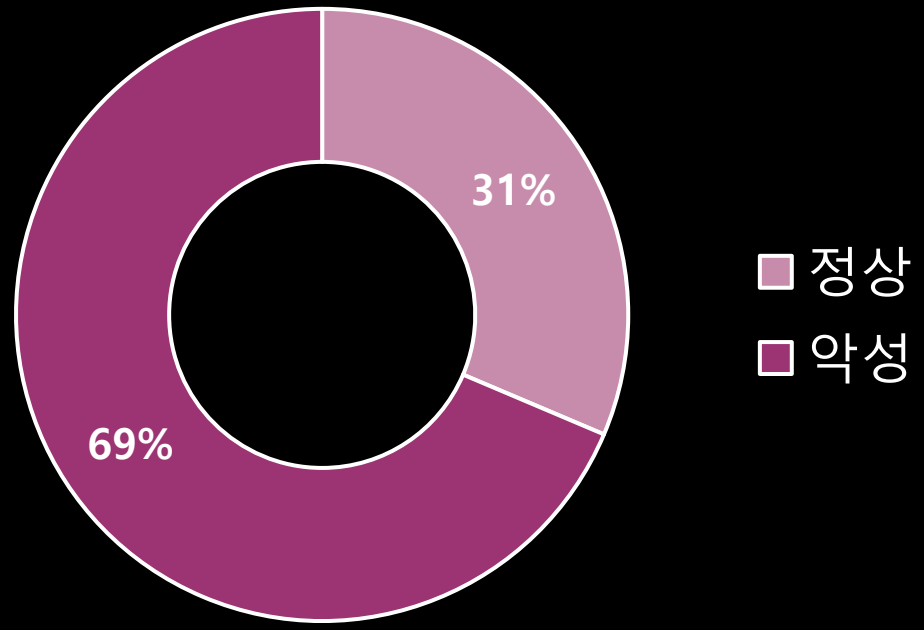
Introduction

- Malware detection
  - 프로그램을 동적 분석하여 얻은 instruction sequence를 바탕으로 악성코드 탐지



Introduction

- KISA-CISC 2017 Malware 2nd dataset 사용
  - 총 6,558개의 프로그램
  - 정상파일: 2,059 개
  - 악성파일: 4,500 개



- Instruction sequence
  - 실행된 명령어의 종류만 기술되어 있음.
  - 명령어 operand, memory addr. 등 존재하지 않음
  - 총 명령어 종류: 1,290개

Introduction

- 두 가지 방식을 사용하여 악성 코드를 탐지한다.
  - 통계적 방법 (statistical method)
    : 프로그램의 구조적 특징을 통계적으로 분석
  - 딥러닝 이용 (deep learning method)
    : 프로그램의 명령어 실행 순서 및 구성을 이용해 분석

Methods: data cleaning & preprocess

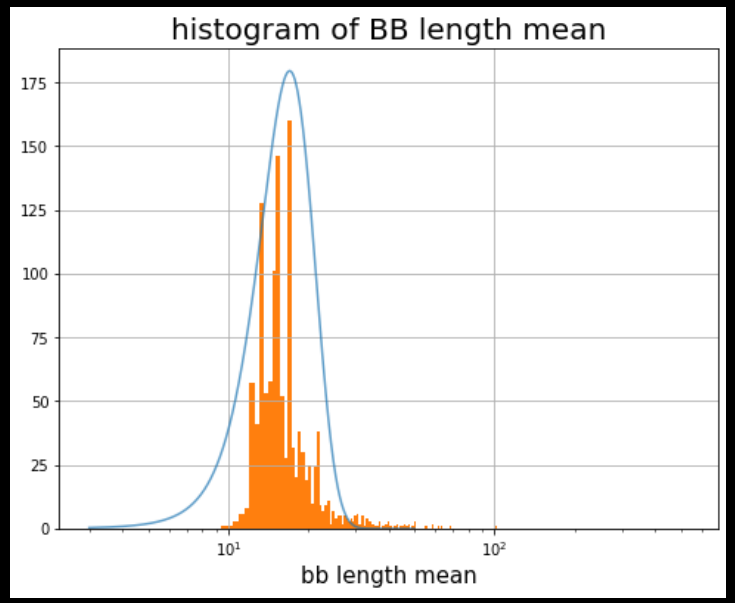
- instruction sequence의 길이가 100이하일 경우, 실행이 제대로 안된 경우일 가능성 높음.

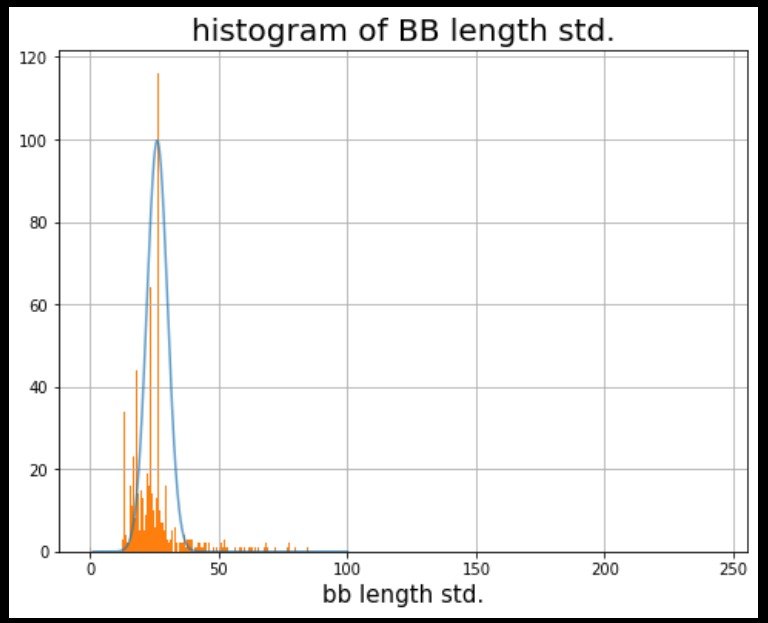
→ instruction seq. 길이가 100보다 큰 프로그램만 분석함.

- Instruction sequence를 다음 instruction을 delimiter로 basic block으로 나눔.
  - Jump instruction
  - Call instruction
  - Return instruction

Methods: statistical method

- Idea: Basic block의 길이 분포가 정규분포를 이루고
  + malware는 out of distribution에 속할 것이라고 가정





Methods: statistical method

- 평균과 표준편차를 바탕으로 해당 프로그램이 normal 파일일 확률을 계산

$$zscore(x;\mu,\sigma) = \frac{x-\mu}{\sigma}$$

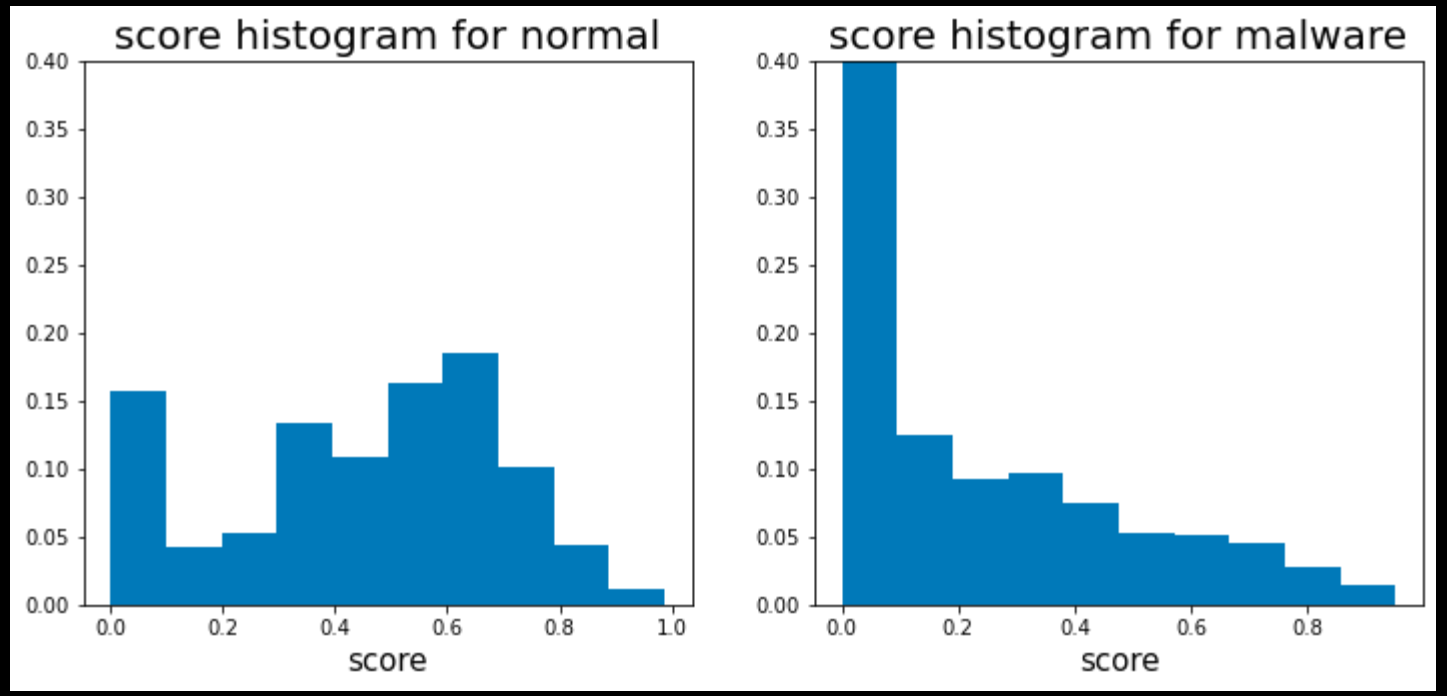
$$score(x;\mu,\sigma) = 2 \times \left(1 - \Phi(|zscore|)\right)$$

최종 score는

$$total(\mu_p,\sigma_p) = score(\mu_p;\mu_\mu,\sigma_\mu)^{w_1} \times score(\sigma_p;\mu_\sigma,\sigma_\sigma)^{w_2}$$

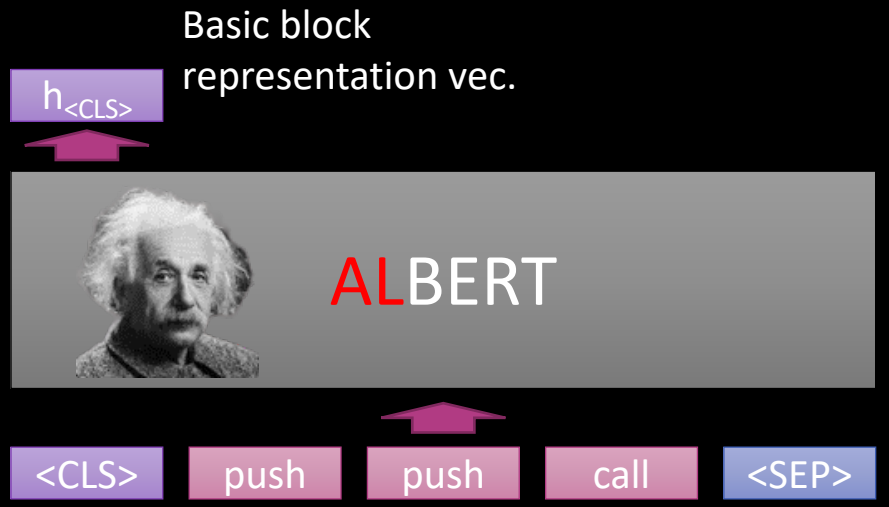
Methods: statistical method

- 상관계수를 가장 크게 하는 w_1과 w_2를 구함.
  - $w_1^* \approx 0.332$, $w_2^* \approx 0.444$



Methods: deep learning method

- Basic Block Encoder
  - ALBERT 구조(2019)를 사용하여 각 basic block의 임베딩 구함.
    - Model parameter: 약 8M
    - B개의 basic block을 가지는 프로그램의 표현 = $\mathbb{R}^{B\times 768}$
  - Masked Language Model task를 이용해 학습



<Basic Block Encoder>

Methods: deep learning method

- Basic Block Encoder
  - 개별 basic block만 보고서는 malware와 normal 파일을 구분하기 어렵다.
  - 한 프로그램을 이루는 여러 basic block을 엮어서 분석 해야 한다.

▶ Basic block 임베딩을 이용해 program 임베딩을 만들자 ◀

Methods: deep learning method

- Program Encoder
  - 모든 basic block이 동등한 중요도를 가지고 있지 않다고 가정
    → basic block을 weighted sum 하여 program 임베딩 계산

$$Emb_{program} = \sum_{i} w_i Emb_{bb_i}$$

Methods: deep learning method

- Program Encoder
  - 좋은 weight를 사용하면 같은 분류기에서 정확도가 높을 것이다.

  → Optimization method를 이용해 최적의 weight를 구하자.

  - 하지만… basic block 관련 데이터셋은 약 150GB…
  - 분류기까지 학습하여 weight의 평가 지표를 구하는데 10분 걸림
    - CPU: Intel i9-10940X
    - RAM: DDR4 32GB×4
    - SSD: Seagate BarraCuda Q5 ZP2000CV30001

Methods: deep learning method

- Program Encoder
  - 멀티 코어를 활용하여 최적화 하는 방안을 고안함.
  - 목표: 정확도를 최대화 하는 정규 분포를 찾는다.

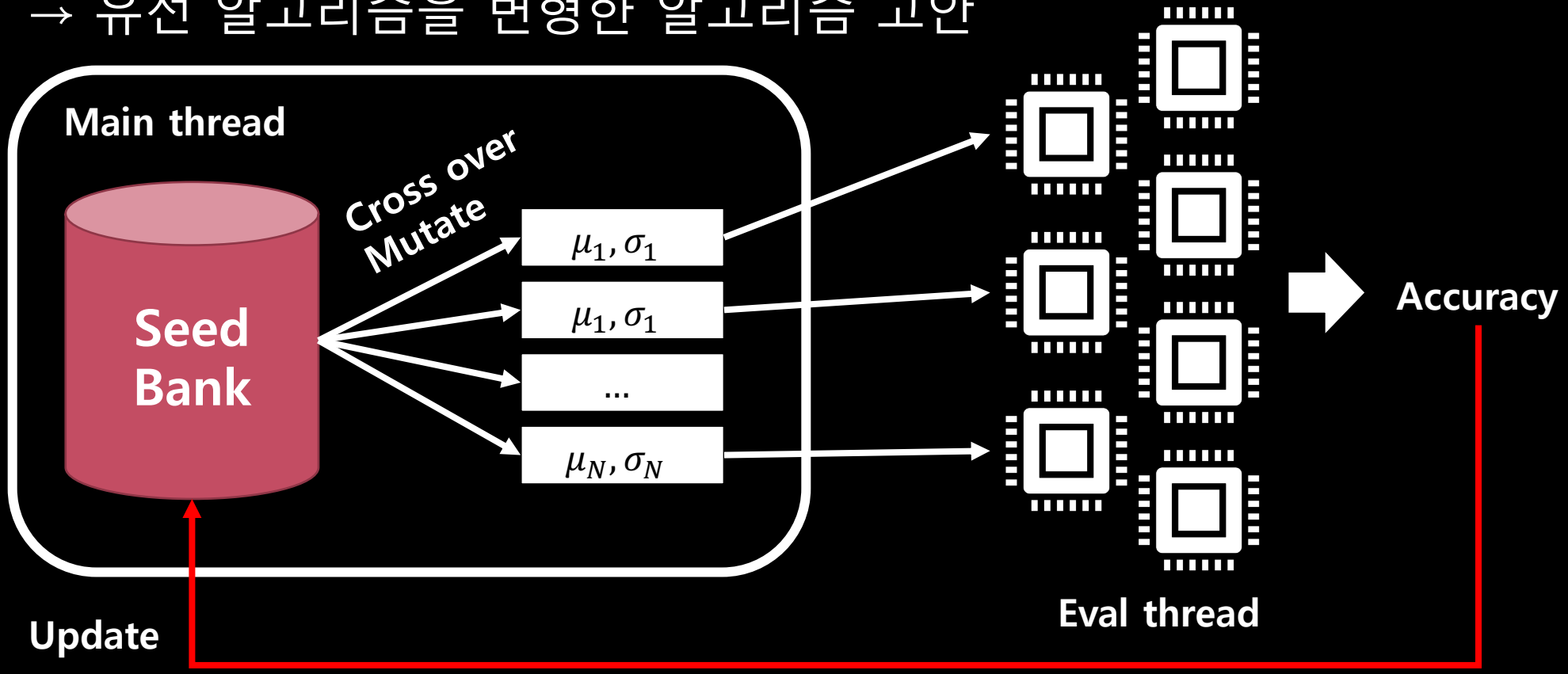
$$w \sim N(\mu, \sigma^2)$$

구해야하는 변수 두 개

Methods: deep learning method

- Program Encoder
  - 멀티 코어를 활용하여 최적화 하는 방안을 고안함.

  → 유전 알고리즘을 변형한 알고리즘 고안



Methods: deep learning method

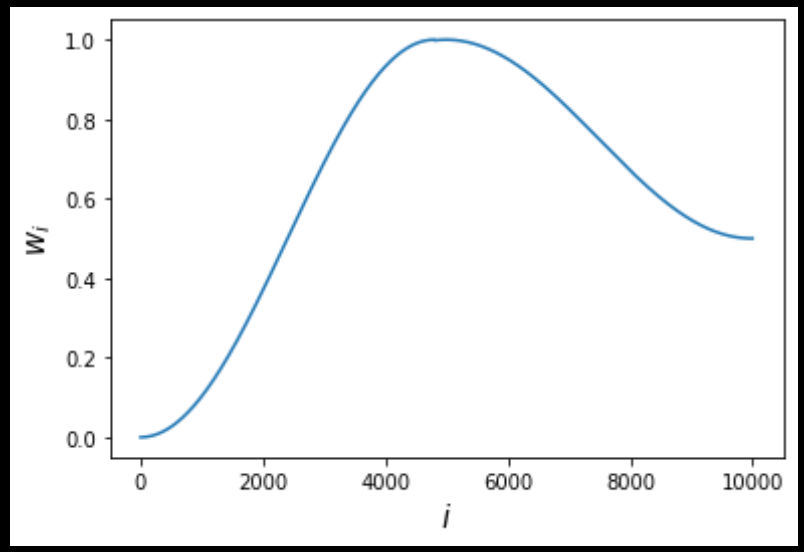
- Program Encoder
  - Cross over
    - 두 임의의 유전자 (μ_1, σ_1), (μ_2, σ_2)에 대하여
    - 새로운 유전자 $(w_1 \cdot \mu_1 + (1 - w_1) \cdot \mu_2,\ w_2 \cdot \sigma_1 + (1 - w_2) \cdot \sigma_2)$
      - 단, w_1, w_2는 random weight
  - Mutate
    - 임의의 유전자 (μ, σ)에 대하여
    - 새로운 유전자 $(\mu + w_1, \sigma + w_2)$
      - 단, w_1, w_2는 random weight

Methods: deep learning method

- Program Encoder
  - 하지만 정규분포는 수치적으로 문제가 있음.
    - 프로그램이 basic block이 많지 않을 경우 weight가 모두 0이 될 수 있음.
    - 일부 basic block에 지나치게 큰 weight가 부여되는 문제가 종종 발생.

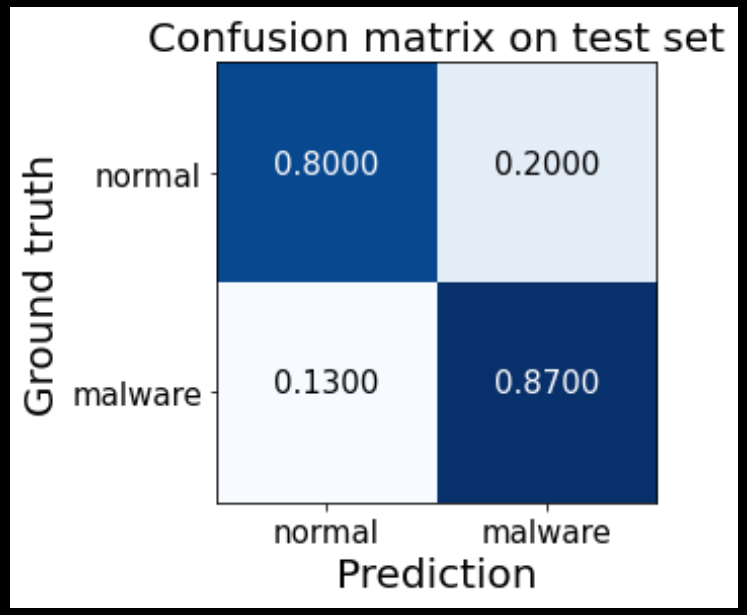
→ 정규 분포 비슷하게 생긴 부드러운 함수를 만듦

$$w_i = \begin{cases} \dfrac{1-\cos\left(\frac{\pi i}{0.48 \cdot N}\right)}{2}, & i < 0.48 \cdot N \\ \dfrac{3-\cos\left(\frac{0.52 \cdot \pi i}{0.48 \cdot 0.5}\right)}{4}, & i \geq 0.48 \cdot N \end{cases}$$



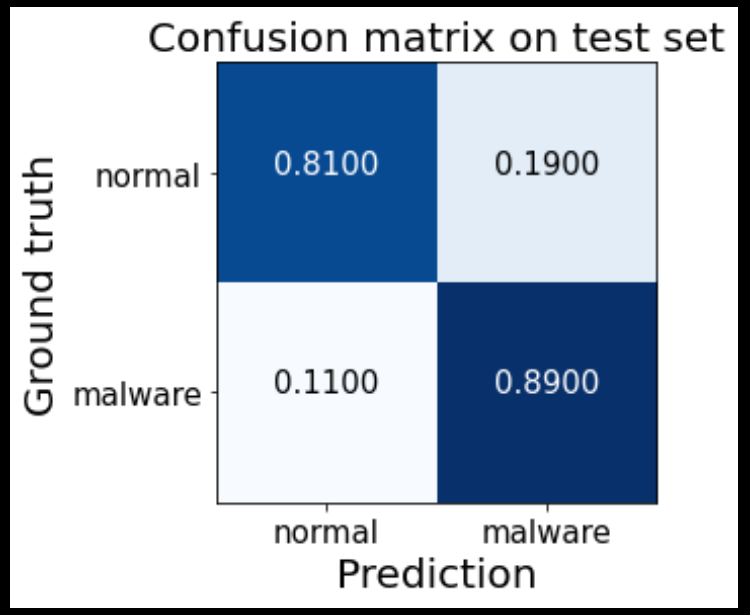
Methods: deep learning method

- Program Encoder + SVM classifier
  - Test accuracy: 83.5% (기존: 79%)



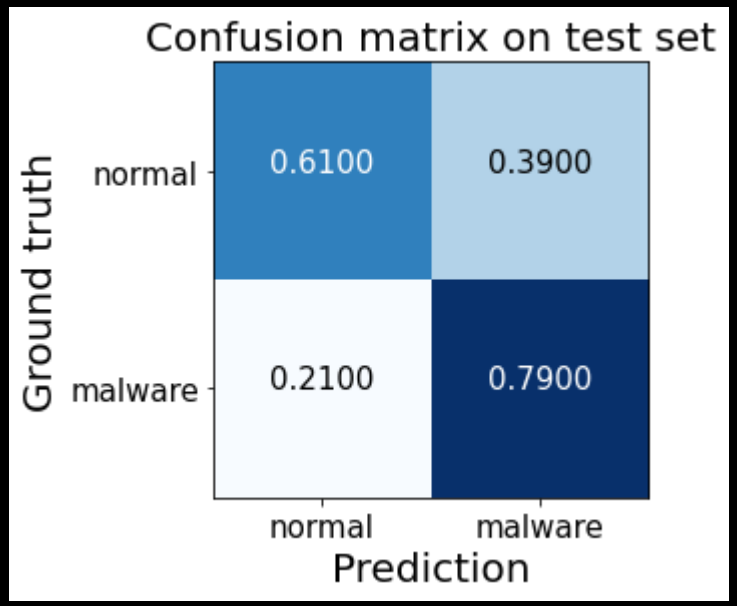
Methods: deep learning method

- Program Encoder + XGBoost
  - Test accuracy: 85%
  - Test F1-score: 0.856



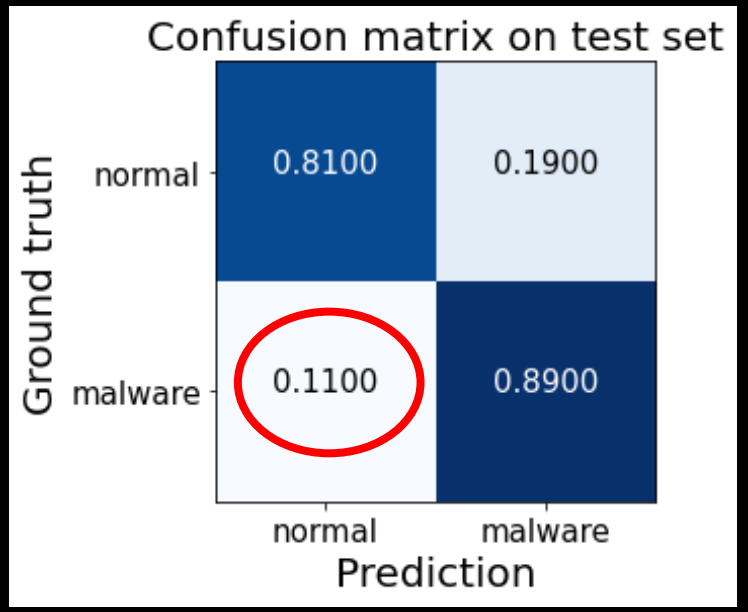
Methods: ensemble

- 통계적 기법과 딥러닝 기법을 앙상블
  - 바로 앙상블 하면 유의미한 성능 하락을 보임.
    - Test accuracy: 70%



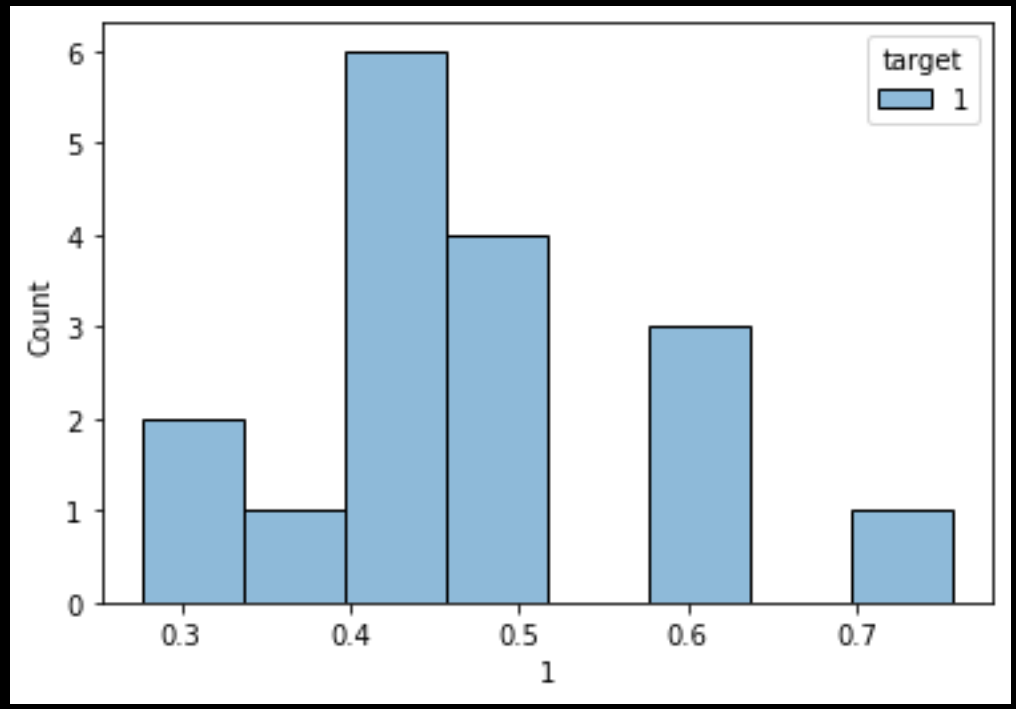
Methods: ensemble

- 앙상블의 목적을 분명히 해보자.
  - 딥러닝 기반 모델은 malware를 normal로 표기하는 비율이 높다.



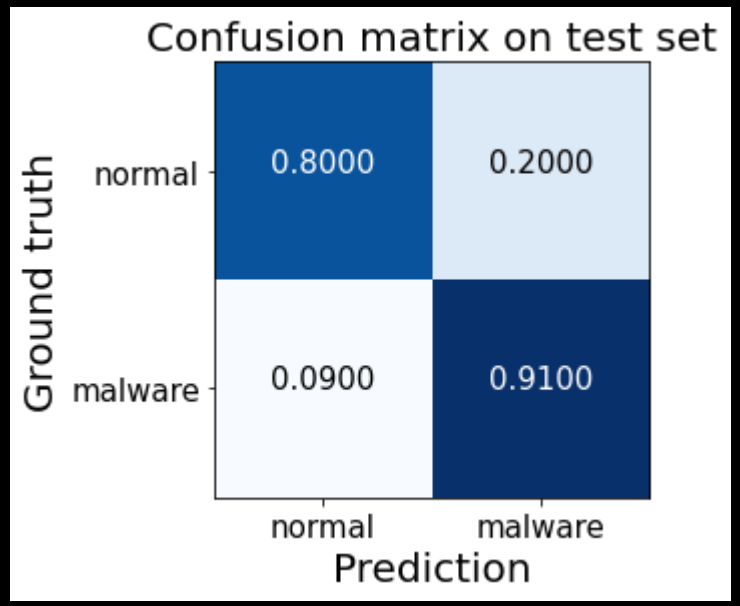
Methods: ensemble

- 앙상블의 목적을 분명히 해보자.
  - Normal로 오분류 된 일부 케이스를 보면 confidence가 낮다.
    → Confidence가 낮은 케이스만 따로 통계적 기법을 적용!



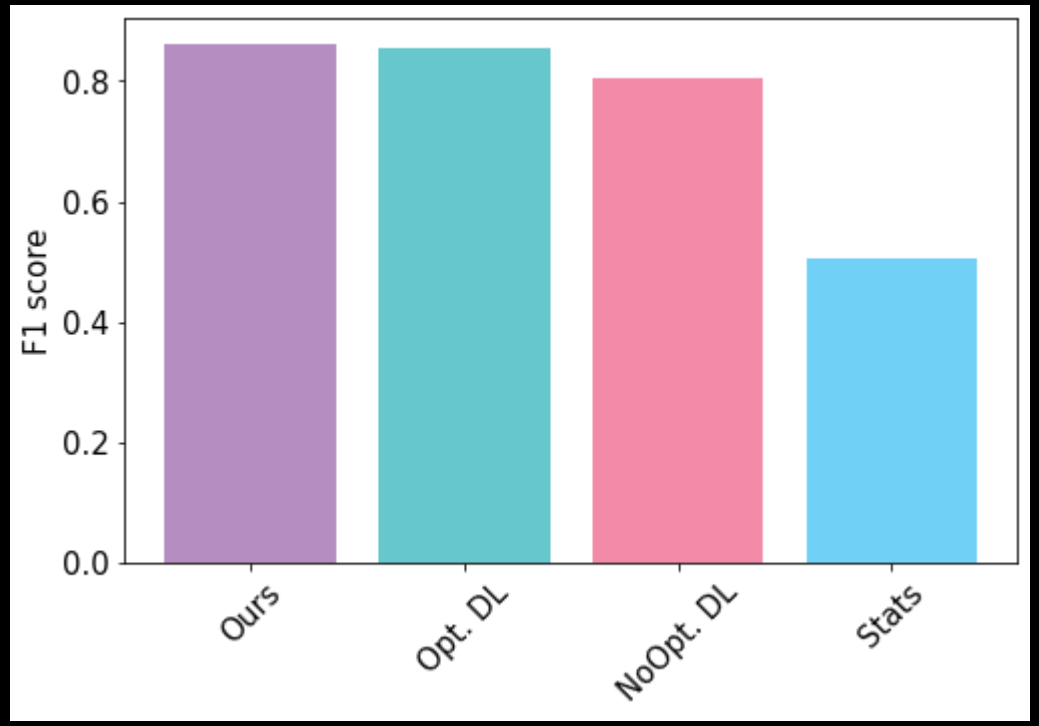
Results

- 최종 성능
  - Test accuracy: 85.5%
  - Test F1-score: 0.863



Results

- Ablation Study



Discussion

- 모든 프로그램이 동일한 부분이 중요할 리가 없다.
  - 일부 프로그램은 초반에 악성 행위를 할 수도 있고,
  - 일부 프로그램은 후반에 악성 행위를 할 수도 있다.

  → 분명한 한계

- 나중에 기회가 되면 GNN을 적용해보고 싶다..!

Thank You