Journal of The Institute of Electronics and Information Engineers Vol.59, NO.7, July 2022 https://doi.org/10.5573/ieie.2022.59.7.3
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2022-59-7-1

소프트웨어 기반 임베디드 시스템용 적대적 공격 검출 시스템

(Software-based Adversarial Attack Detection System for Embedded Systems)

주 상 현*, 김 인 모*, 김 명 선**

(Sanghyun Joo, Inmo Kim, and Myungsun Kim©)

요 약

최근 DNN의 추론을 잘못된 분류가 되도록 유도하는 것을 목적으로 하는 적대적 공격은 그 종류가 매우 다양해지고 더욱 정교해지고 있다. 이러한 영향으로 DNN 모델들은 더욱 적대적 공격에 쉽게 노출되고 있다. 로봇이나 자율주행 시스템과 같은 임베디드 시스템도 예외가 아니고 적대적 공격으로 인한 오분류의 결과는 치명적인 결과로 이어질 수 있다. 하지만 임베디드 시스템은 DNN 연산 장치의 성능과 메모리 용량이 제한적이어서 적대적 공격을 빠른 속도로 판별하는 것이 매우 어렵다. 이를 극복하고자 전용 하드웨어를 개발하면 개발 비용 상승과 공격 방식과 타겟 DNN 모델의 변경에 유연하게 대처하는 것은 어렵다. 이러한 문제를 해결하기 위해서 본 연구에서는 임베디드 시스템을 위한 소프트웨어 기반 적대적 공격 검출 기법을 제안한다. 이 기법은 검출에 필요한 추가적인 메모리를 최소화하고자 검출 대상 은닉층을 선별하여 사용한다. 또한 타겟 DNN이 추론을 수행할 때 병렬적으로 검출을 진행하고 이 둘 간의 실행시간 격차를 최소화한다. 실험 결과 제안된 기법 적용 전 대비 실행시간의 차이는 최대 99.6% 감소하였고 적대적 공격 탐지 정확도를 유지하면서 메모리 사용량은 최대 83.9% 감소하였다.

Abstract

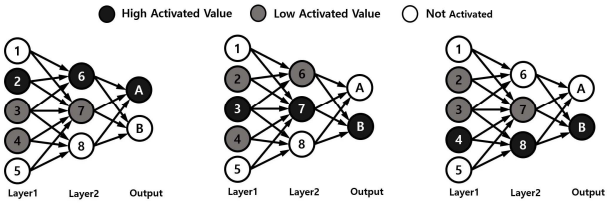
Recently, adversarial attacks aimed at inducing DNN reasoning to be misclassified are becoming more diverse and more sophisticated. Due to this trend, DNN models are more easily exposed to adversarial attacks. Embedded systems such as robots and driverless car systems are no exception, and the result of misclassification due to adversarial attacks can lead to fatal consequences. However, embedded systems have limited performance in DNN computational units and memory capacity, making it very difficult to detect adversarial attacks in a limited time. To overcome this, developing a dedicated hardware unit cause high development costs, and also the hardware is not as flexible as to cope with various kinds of attacks and any change in target DNN model. To solve this problem, in this study, we proposes a software-based adversarial attack detection mechanism for embedded systems. The mechanism selects and utilizes several hidden layers to be detected to minimize the additional memory required for adversarial detection. Additionally, when the target DNN performs inference, the adversarial detection process is in parallel and minimizes the time gap between the two. As experiments show, the execution time difference compared to before applying the proposed mechanism decreased by up to 99.6% and the memory usage decreased by up to 83.9% while maintaining attack detection accuracy.

Keywords : Adversarial attack, Adversarial detection, Embedded systems, DNN inference, NIC

Ⅰ. 서 론

최근 DNN은 지속적 발전으로 로봇, 자율주행 자동차, 스마트폰과 같이 임베디드 시스템 환경에서 활발히 활용된다. 그러나 DNN은 오분류를 초래하는 적대적 공격에 굉장히 취약하다는 것이 밝혀졌고[1] 이에 다양한 적대적 공격 방법이 발달 되어 DNN이 적대적 공격에 더욱 노출되었다. 적대적 공격의 종류가 늘어남에 따라

*학생회원, 한성대학교 IT융합공학과(Department of IT Convergence Engineering, Hansung University)
**정회원, 한성대학교 AI응용학과(Department of Applied Artificial Intelligence, Hansung University)
© Corresponding Author(E-mail : kmsjames@hansung.ac.kr)
※ 본 연구는 한성대학교 교내학술연구비 지원과제임.
Received ; April 11, 2022 Revised ; May 2, 2022 Accepted ; May 3, 2022



(a) inference benign data (b) inference adversarial data (c) inference other adversarial data

그림 1. 추론할 때 활성화되는 뉴런과 각 뉴런의 활성화 값
Fig. 1. Neurons activated in inference and the activation value of each neuron.

적대적 공격을 탐지하고 방어하는 방법론과 시스템이 발맞춰 개발되었고 특히 적대적 공격 탐지는 우수한 성능을 보이는 여러 기술이 선보여졌다. 일반적으로 적대적 공격 탐지 시스템은 DNN의 추론과 병렬적으로 동시에 실행된다[2]. 하지만 임베디드 시스템 환경은 서버 컴퓨팅 환경 대비 메모리, 프로세서 성능 등의 제약이 있어 DNN 추론과 적대적 공격 탐지를 동시에 실행하기에 불리한 조건이다. 예시로 실시간성이 보장되어야 하는 자율주행 자동차 시스템에 적대적 공격이 가해지는 것을 방지하기 위해 적대적 공격 탐지 시스템을 접목했다면 한 개의 입력 데이터에 대한 DNN 추론 종료 시간과 적대적 공격 탐지 종료 시간의 차이가 작거나 적대적 공격 탐지가 먼저 종료되어야 유의미하다. 하지만 여러 제약 사항이 있는 임베디드 시스템에선 서버 컴퓨팅 환경에 비해 종료 시간 차이가 커질 수밖에 없다.

본 논문에서는 여러 적대적 공격 탐지 시스템 중 성능이 좋은 적대적 공격 탐지 시스템을 활용하여 실질적으로 임베디드 시스템에서 적대적 공격 탐지 시스템을 활용할 때를 위해 DNN 추론 종료 시간과 적대적 공격 탐지 종료 시간의 차이를 소프트웨어 기반 솔루션을 통해 감소시키고자 한다.

Ⅱ. 적대적 공격 탐지 시스템

1. NIC (Neural Network Invariant Checking)

적대적 공격이란 영상에 사람의 눈으로 확인할 수 없는 잡음(perturbation)을 추가하여 DNN이 잘못된 분류를 하도록 적대적 예제(Adversarial example)를 생성하는 것이다. 적대적 공격 탐지란 타겟 DNN에서 입력 데이터가 추론될 때 해당 데이터가 적대적 예제인지 아닌지 판별하는 것을 의미한다. 적대적 공격의 종류가 늘어난 만큼 적대적 공격 탐지 또한 여러 방법이 존재한다. 본 논문은 Neural Network Invariant Checking[3](이하 NIC)를 활용하는 탐지 시스템을 타겟으로 한다.

그림 1은 NIC를 활용한 탐지 기법을 소개한다. 은닉층1($Layer1$)과 은닉층2($Layer2$), A와 B로 분류하는 출력층($Output$)을 가지는 DNN에 정상 데이터가 입력되었을 때(그림 1의 (a))와 각기 다른 적대적 예제가 입력되었을 때(그림 1의 (b), (c))를 나타낸다. 그림 1의 (a)는 정상 데이터가 A로 알맞게 분류되었지만 그림 1의 (b), (c)에서는 적대적 예제가 B로 오분류되었다. NIC에서 DNN의 은닉층 활성화 함수가 어떤 뉴런에 0이 아닌 값을 전달하면 해당 뉴런을 활성화 뉴런이라 하고 활성화 뉴런의 값을 활성화 값이라고 한다. 그림 1의 각 원은 은닉층의 뉴런을 의미하고 색이 있는 원을 활성화 뉴런이라 하며 원의 색이 진할수록 뉴런의 활성화 값이 큰 것이다.

NIC는 타겟 DNN이 정상 데이터를 추론했을 때 각 은닉층의 활성화 값들을 해당 은닉층의 Value Invariants(이하 VI)라 하고 각각 연속된 두 은닉층의 활성화 뉴런들을 해당 은닉층들의 Provenance Invariants(이하 PI)라고 한다. 그림 1의 (a)에서 활성화 뉴런 2, 3, 4의 활성화 값들이 은닉층1의 VI, 활성화 뉴런 6, 7의 활성화 값들이 은닉층2의 VI이며 활성화 뉴런 2, 3, 4, 6, 7들이 은닉층1, 은닉층2의 PI이다.

NIC에서는 입력 데이터를 타겟 DNN에 추론했을 때 각 은닉층의 활성화 값들이 해당 은닉층 VI와 다르거나 연속된 두 은닉층 각 쌍의 활성화 뉴런들이 해당 은닉층 PI와 다른 경우 적대적 공격을 받은 데이터로 판단한다. 그림 1의 (b)에서 사용된 입력 데이터는 각 은닉층의 활성화 값들이 해당 은닉층의 VI와 다르다. 이를 VI를 위반했다고 표현하고 이때의 입력 데이터를 적대적 예제로 판단한다. 그림 1의 (c)에서 사용된 입력 데이터는 은닉층1과 은닉층2에서의 활성화 뉴런들이 은닉층1과 은닉층2의 PI와 다르다. 이를 PI를 위반했다고 표현하고 이때의 입력 데이터 또한 적대적 예제로 판단한다.

그림 1의 DNN을 통해 각 은닉층의 VI와 각각 연속된 두 개의 은닉층들의 PI를 구하는 방법은 다음과 같다. 은닉층1과 은닉층2 각각의 VI는 타겟 DNN이 정상 데이터를 추론할 때 각 은닉층의 출력 벡터이다. 은닉층1과 은닉층2의 PI를 구하기 위해서는 먼저 Derived Model(이하 DM)이라는 DNN이 두 개 필요하다. (a)에서 은닉층1의 DM은 입력층부터 은닉층1 뒤에 새로운 소프트맥스 계층을 이어 붙인 DNN이고 은닉층2의 DM은 입력층부터 은닉층2 뒤에 새로운 소프트맥스 계층을

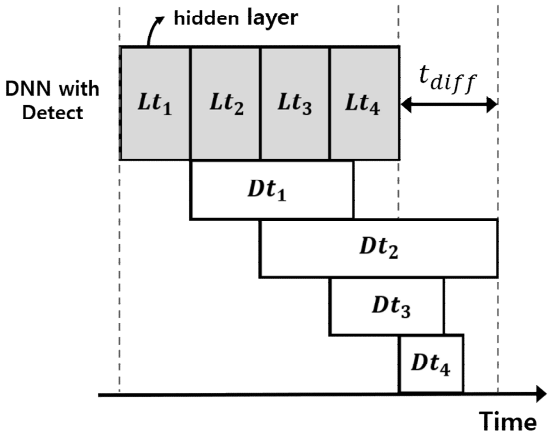


그림 2. NIC를 적용한 타겟 DNN의 추론 소요시간과 적대적 공격 탐지 소요시간의 차이

Fig. 2. Difference between the inference time of the target DNN and adversarial attack detection time when NIC is applied.

이어 붙인 DNN이다. DM의 각 소프트맥스 계층의 출력 차원은 타겟 DNN의 소프트맥스 출력 차원과 같다. 두 DM의 출력 벡터를 컨캣(concatenate)하면 은닉층1과 은닉층2의 PI가 된다.

NIC는 One-Class SVM[4](이하 OCSVM)을 사용하여 모든 VI와 PI를 훈련한다. 각각의 VI와 PI는 정상 데이터로부터만 구할 수 있으므로 훈련 데이터로는 정상 데이터만을 사용한다. 타겟 DNN에 훈련 데이터를 추론하여 각 은닉층의 출력을 구해 이것으로 은닉층마다 VI와 PI를 구한 후 OCSVM으로 훈련한다. 즉, 은닉층마다 VI 탐지용 OCSVM(이하 VI-SVM)과 PI 탐지용 OCSVM(이하 PI-SVM)이 존재한다.

Ⅲ. 임베디드 NIC의 문제점

자율주행 자동차와 같은 임베디드 시스템은 서버 컴퓨팅 환경과 달리 낮은 성능의 프로세서와 GPU를 사용하고 메모리 크기 또한 상대적으로 작다. 본 장에서는 임베디드 시스템에서 NIC를 적용한 타겟 DNN을 추론할 때 드러나는 NIC에 대한 문제점을 설명하고자 한다.

1. 타겟 DNN 추론과 NIC 탐지의 소요시간 차이

그림 2는 멀티 프로세스 환경에서 NIC를 4개의 은닉층으로 이루어진 타겟 DNN에 적용하여 타겟 DNN의 추론 소요시간과 적대적 공격 탐지 소요시간을 나타낸다. 타겟 DNN의 각 은닉층 추론 시간은 $Lt_1 \sim Lt_4$로, 각 은닉층 추론 후 탐지 소요시간은 $Dt_1 \sim Dt_4$ 로 나타낸다. t_{diff}는 적대적 공격 탐지가 완료될 때까지의 소요시간과 타겟 DNN 추론 소요시간의 차이이며 가장 늦게 완료되는 탐지 소요시간이 늘어날수록 t_{diff}가 증가한다.

$Dt_1 \sim Dt_4$ 각각은 VI-SVM 또는 PI-SVM 모델의 크기가 클수록 늘어난다. 그림 2에서는 각 은닉층의 탐지 소요시간 중 Dt_2가 가장 기므로 두 번째 은닉층의 VI-SVM 또는 PI-SVM의 크기가 가장 큰 모델을 가진다.

NIC를 적용한 타겟 DNN 추론에서 t_{diff}발생은 필연적이다. 타겟 DNN은 추론이 끝난 후 t_{diff}동안 입력 데이터에 대한 분류 결과를 신뢰할 수 없다. 따라서 t_{diff}는 작을수록 좋다. 그러나 임베디드 시스템상에서 그림 2와 같이 타겟 DNN에 NIC를 적용하여 실행하면 서버상에서 실행했을 때 대비 각각의 적대적 공격 탐지 소요시간이 증가하므로 t_{diff}가 증가할 수밖에 없는 문제점이 발생한다.

2. NIC의 메모리 요구량

OCSVM 모델을 저장할 때 서포트 벡터를 저장하고 서포트 벡터의 차원은 OCSVM 훈련 데이터의 차원과 같기 때문에 OCSVM 모델의 크기는 훈련 데이터의 차원이 커지거나 서포트 벡터의 개수가 늘어날수록 커진다. 타겟 DNN에서 모든 PI-SVM들은 훈련 데이터의 차원이 같으므로 PI-SVM 모델의 크기에 영향을 주는 것은 서포트 벡터의 개수이다. 반면 VI-SVM의 훈련 데이터는 타겟 DNN의 각 은닉층 출력 벡터이므로 타겟 DNN 각각의 VI-SVM 모델의 크기는 훈련 데이터의 차원과 서포트 벡터의 개수에 영향을 받는다. PI-SVM들의 훈련 데이터 차원은 타겟 DNN의 훈련 데이터 셋에 의해 정해지고 대부분의 VI-SVM들의 훈련 데이터 차원에 비해 월등히 작으므로 PI-SVM 모델들의 크기는 대부분의 VI-SVM 모델들의 크기보다 작다. 예시로 본 논문의 실험에서 타겟 DNN 중 하나로 사용된 MobileNetV1의 PI-SVM 모델 크기가 가장 큰 것은 792KB지만 VI-SVM 모델 크기가 가장 큰 것은 2.4GB이다.

NIC는 타겟 DNN의 입력 데이터가 VI와 PI를 위반하는지 각 은닉층에서 확인하는 것으로 적대적 공격 탐지를 수행한다. 그러므로 타겟 DNN의 은닉층 개수만큼 VI-SVM, DM, 그리고 PI-SVM이 필요하다. 만일 타겟 DNN의 총은닉층 개수가 50개라고 가정하면 NIC를 이용한 적대적 공격 탐지에 필요한 OCSVM과 DNN이 총

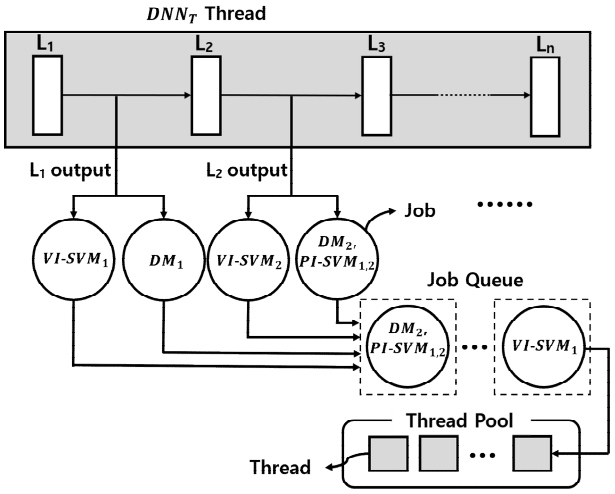


그림 3. 임베디드 NIC 시스템 실행 구조
Fig. 3. Workflow of embedded NIC system.

149개(VI-SVM 50개, DM 50개, PI-SVM 49개)가 필요하다. 또한 50개의 VI-SVM 중 1GB 이상의 큰 크기를 가지는 VI-SVM이 여러 개 존재할 수 있다. NIC의 이러한 OCSVM 기반 탐지 기법은 수 GB의 메모리가 필요하여 임베디드 시스템에 적합하지 않을 수 있다.

Ⅳ. 문제해결 방법 및 설계

임베디드 시스템에서 t_{diff}를 줄이는 방법의 하나로 전용 하드웨어를 탑재하여 탐지 성능을 극대화하는 방법이 있으나 비용 측면의 부담과 같은 이유로 실질적인 방법이라고 할 수 없다. 따라서 본 논문에서는 t_{diff}를 최소로 하기 위한 소프트웨어 기반 임베디드 NIC 시스템을 제안한다.

1. 임베디드 NIC 시스템 설계

그림 3은 본 논문에서 제안하는 임베디드 NIC 시스템 구조를 나타낸다. 기본적으로 단일 프로세스를 사용하는 멀티 스레드이다. 은닉층 n개로 이루어진 타겟 DNN을 $DNN_T = \{L_1, L_2, L_3 \dots L_n\}$으로 표기하고 모든 은닉층에서 이뤄지는 적대적 공격 탐지 과정 중 L_1, L_2, L_3에서의 적대적 공격 탐지 과정을 예시로 설명한다. DNN_T에 NIC를 적용하여 적대적 공격 탐지를 하면 L_1의 출력 벡터를 입력으로 받는 VI-SVM $VI-SVM_1$ 과 L_2의 출력 벡터를 입력으로 받는 VI-SVM $VI-SVM_2$이 존재한다. DM은 L_1과 소프트맥스 계층으로 구성된 DM_1과 L_2과 소프트맥스 계층으로 구성된 DM_2가 있고 DM_1과 DM_2의 출력을 컨캣한 벡터를 입력으로 받는 PI-SVM $PI-SVM_{1,2}$가 있다.

DNN_T는 한 개의 스레드에서 추론되며 적대적 공격 탐지의 실행은 스레드 풀의 스레드에서 실행된다. 잡큐(Job Queue)는 스레드 풀의 스레드에 잡을 전송하는 선입선출 방식의 대기 큐이다. 잡큐에서 스레드 풀의 스레드로 전송되는 잡(Job)은 NIC에 필요한 각각의 VI-SVM 추론 과정, DM 추론 과정, 그리고 PI-SVM 추론 과정을 의미한다. DM 추론 과정과 PI-SVM 추론 과정은 의존성이 존재하므로 하나의 잡으로 설계하여 한 개의 스레드에서 실행되도록 설계하였다. 잡큐에 잡이 있고 스레드 풀에 있는 스레드들이 모두 잡을 할당받은 상태라면 잡이 끝나는 스레드가 발생하기 전까지 잡큐에서 잡을 내보내지 않는다.

GPU의 병렬성을 극대화하기 위해 본 논문에서는 타겟 DNN 추론, VI-SVM 추론 그리고 DM 추론 시 CUDA에서 제공하는 CUDA 스트림 API를 사용하였다. 스트림 API를 사용하지 않을 경우 디폴트 스트림만 사용하게 되어 GPU의 사용 가능한 남는 SM을 활용하지 못하게 되지만 멀티 스트림 사용 시 남는 SM들이 각 스트림에 있는 GPU 작업을 동시에 실행할 수 있게 되어 GPU 병렬성을 향상할 수 있다.

그림 3의 동작은 다음과 같다. DNN_T의 추론이 시작되고 L_1의 출력 벡터가 L_2로 입력되기 전에 $VI-SVM_1$잡과 DM_1잡을 생성한다. 생성된 각 잡들은 잡큐를 통한 후 스레드 풀의 스레드에서 DNN_T 추론과 병렬적으로 실행되고 L_1의 출력 벡터가 L_2로 입력되어 L_2의 출력 벡터를 얻게 되면 $VI-SVM_2$잡과 $DM_2, PI-SVM_{1,2}$잡을 생성한 후 실행한다. $DM_2, PI-SVM_{1,2}$잡에서는 DM_2의 출력 벡터를 얻은 후 DM_1의 출력 벡터와 컨캣하여 $PI-SVM_{1,2}$의 추론을 실행한다. 이와 같은 동작을 DNN_T의 은닉층마다 반복한다. DNN_T에 입력된 데이터가 각 잡에서 하나라도 적대적 예제라고 판별이 되면 입력 데이터는 적대적 예제라고 판단한다.

2. VI, PI 위반 탐지 과정의 최소화

NIC는 타겟 DNN의 모든 은닉층에서 VI와 PI를 위반하는지 확인하는 것을 통해 타겟 DNN의 입력 데이터가 정상 데이터인지 적대적 예제인지를 판별한다. 정

상 데이터는 모든 은닉층에서 VI와 PI를 위반하지 않아야 하며 이를 탐지하는 정확도를 각각 VI 탐지율, PI 탐지율이라 칭하고 두 개의 탐지율은 정상 데이터 탐지율을 의미한다. 반면 적대적 예제는 VI 또는 PI를 위반하며 각각의 위반 탐지 정확도를 VI 위반 탐지율과 PI 위반 탐지율이라고 한다. 두 개의 위반 탐지율은 적대적 공격 탐지율이다. 적대적 예제들을 NIC를 적용한 타겟 DNN에서 추론해보면 특정 은닉층에서는 많은 적대적 예제들을 탐지하고 특정 은닉층에서는 적대적 예제들을 거의 탐지하지 못하거나 모든 적대적 예제들을 탐지하지 못한다. 타겟 DNN의 은닉층 중에 VI 위반 탐지율이나 PI 위반 탐지율이 낮은 은닉층들은 적대적 공격 탐지를 제대로 수행하지 못하기 때문에, 본 논문에서는 3장에서 설명한 메모리 이슈로 인하여 이들에 대한 OCSVM 및 DM 과정을 생략한다. 결과적으로 적대적 공격 탐지율이 높은 일부 은닉층을 선별하여 해당 은닉층에서만 VI 또는 PI의 위반 여부를 진행했다. 본 논문에서는 실험을 통해 이를 확인했고 5장에서 이 실험에 대해 자세히 설명한다.

제안한 기법으로 기존 NIC의 적대적 공격 탐지율을 유지한 채 NIC에서는 모든 은닉층에서 필요했던 VI-SVM, DM 그리고 PI-SVM의 개수를 크게 줄여 메모리가 제한적인 임베디드 시스템에서도 문제없이 실행할 수 있도록 하였다. 적대적 공격 탐지를 수행하는 일부 은닉층 선별에 관한 내용은 다음 장에서 실험을 통하여 자세히 설명하도록 한다.

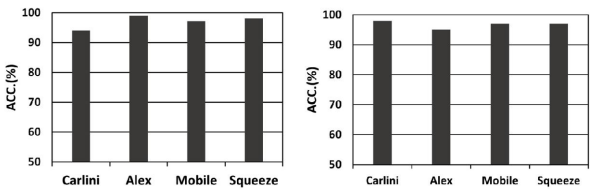
V. 실 험

본 장에서는 본 논문에서 제안하는 임베디드 NIC의 효용성을 검증한다. 먼저 실험환경을 설명하고 실험에 사용된 각 모델의 VI와 PI의 위반 검출을 수행하는 특정 은닉층을 선별한 근거를 실험 결과와 함께 설명한다. 마지막으로 구체적인 성능 평가 결과를 제시한다.

1. 실험환경

본 실험은 NVIDA(사)의 Jetson AGX Xavier(이하 AGX-Xavier)[5]를 타겟 임베디드 시스템으로 사용하였다. 표 1은 사용한 AGX-Xavier의 하드웨어와 소프트웨어 사양을 나타낸다.

실험에는 Carlini model[6], AlexNet[7], MobileNetV1[8], SqueezeNet[9] 총 4종류의 DNN 모델을 타겟 DNN으로 사용하였다. 실험부에서 표기의 간소화를 위하여 4종류의 DNN 모델을 각각 Carlini, Alex, Mobile, Squeeze로 표기한다. 각 타겟 DNN 학습에 사용되는 데이터 셋은 CIFAR-10[10] 데이터 셋을 사용하였고, 적대적 공격 예제는 CIFAR-10 데이터 셋에 FGSM[11] 공격을 NIC와 같은 방식으로 가해 100개를 생성하였다.



(a) VI (b) PI

그림 4. 각 타겟 DNN의 모든 은닉층의 VI, PI 탐지율 중 최소값

Fig. 4. Minimum of VI and PI detection rates in all hidden layers of each target DNN.

표 1. Jetson AGX Xavier 상세정보

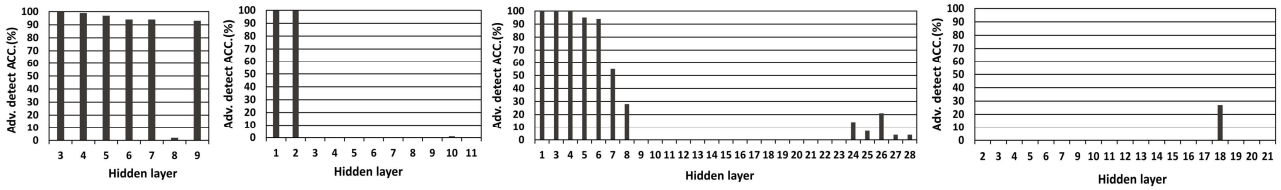
Table 1. Detailed information of Jetson AGX Xavier.

GPU	512-core Volta GPU with Tensor Cores
CPU	8-core ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3
Memory	32GB 256-Bit LPDDR4x
Jetpack	ver 4.2
OS	Linux kernel version 4.9.140
CUDA	CUDA 10.0.166

2. 타겟 DNN별 각 은닉층에서의 VI, PI 탐지율 및 위반 탐지율

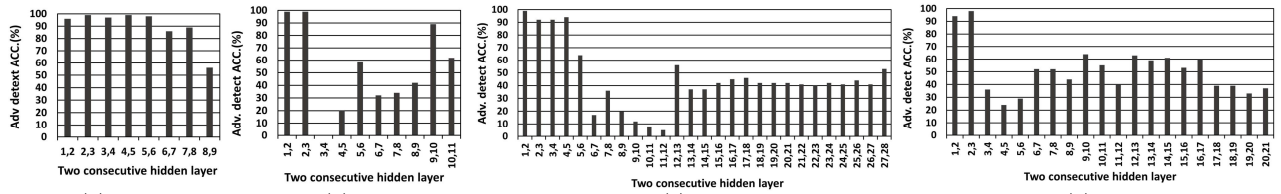
본 절에서는 4종류의 타겟 DNN에서 적대적 공격 탐지 은닉층을 선별한 기준을 실험 결과를 통해 설명한다. 그림 4는 50,000장의 정상 데이터에 대해 각 타겟 DNN의 모든 은닉층 중 최소 VI 탐지율(a)과 최소 PI 탐지율(b)을 각각 나타낸다. 타겟 DNN별 모든 은닉층에서 최소 95% 이상의 VI, PI 탐지율을 나타내는 것을 확인할 수 있다. 타겟 DNN의 종류, 은닉층의 위치와 상관없이 정상 데이터 탐지율은 항상 높으므로 적대적 공격 탐지 은닉층을 결정하기 위하여 각 은닉층의 VI, PI 위반 탐지율만을 고려하였다.

그림 5는 4종류의 타겟 DNN의 모든 은닉층에서 VI를 위반한 적대적 예제의 탐지 정확도를 나타낸다. 각 타겟 DNN별 은닉층 개수는 Carlini 9개, Alex 11개, Mobile 28개, Squeeze 21개이고 x축은 각 타겟 DNN의 입력 계층부터의 은닉층 순서를 의미한다. 그래프에 표기가 되지 않은 은닉층은 차원이 너무 큰 관계로 본 실



(a) Carlini (b) Alex (c) Mobile (d) Squeeze

그림 5. 타겟 DNN의 각 은닉층에서 적대적 예제의 VI 위반 탐지율
Fig. 5. VI violation detection rate of adversarial samples in each hidden layer of target DNNs.



(a) Carlini (b) Alex (c) Mobile (d) Squeeze

그림 6. 타겟 DNN의 각 은닉층에서 적대적 예제의 PI 위반 탐지율
Fig. 6. PI violation detection rate of adversarial samples in each hidden layer of target DNNs.

험 환경에서 OCSVM 학습을 진행하지 못한 은닉층이다. 그림 6은 4종류의 타겟 DNN의 모든 연속된 2개의 은닉층에서 PI를 위반한 적대적 예제의 탐지 정확도를 나타낸다.

그림 5를 통해서 알 수 있듯이 Carlini, Alex, Mobile은 각각 (3번), (1번, 2번), (1번, 3번, 4번) 은닉층에서 100%의 정확도로 VI 위반을 감지할 수 있다. 이들 은닉층 중에서 VI 탐지율 또한 가장 높은 은닉층만을 선택한 결과로 Carlini는 3번, Alex는 2번, Mobile은 3번 은닉층으로 선정하였다.

Squeeze의 경우 VI 위반 탐지율을 확인한 결과 적대적 공격 예제에 대한 탐지율이 가장 높은 은닉층이 27%로 매우 낮은 정확도를 나타냈다. 하지만 PI 위반 탐지율 그래프에서는 탐지율이 98%인 2, 3번 은닉층이 존재하였다. 따라서 Squeeze는 VI-SVM을 수행하지 않고 PI-SVM만 수행하기로 했다. 이때 2% 부족한 PI 위반 탐지율를 만회하고자 다음으로 높은 PI 위반 탐지율을 기록한 1, 2번 은닉층을 PI-SVM 대상으로 추가하여 1～3번의 연속된 3개의 은닉층에서 PI 위반에 대한 탐지를 수행하도록 하였다.

3. 임베디드 NIC 시스템 성능 평가

본 절에서는 임베디드 NIC 시스템을 멀티 프로세스 환경에서 타겟 DNN과 NIC가 병렬 실행되도록 설계된 시스템(이하 멀티 프로세스 NIC 시스템)과 비교하여 성능을 평가한다. 그림 7은 멀티 프로세스 NIC 시스템 설계 구조이다. 그림 5와 동일하게 n개의 은닉층으로 구성된 타겟 DNN은 DNN_T로 표기했고 L_1, L_2, L_3에서의 적대적 공격 탐지 과정만 자세히 나타냈다. DNN_T는 한 개의 프로세스에서 추론되고 각 은닉층의 VI-SVM도 서로 다른 프로세스에서 추론이 실행되며 그림 7에서 $VI_1\,Proc.$, $VI_2\,Proc.$는 각각 L_1과 L_2의 VI-SVM을 추론하는 프로세스다. DM 추론과 PI-SVM 추론은 같은 프로세스에서 실행되며 $DM_1\,Proc.$는 L_1의 DM을 추론하는 프로세스, $DM_2, PI_{1,2}\,Proc.$는 L_2의 DM 추론과 PI-SVM 추론을 실행하는 프로세스다. 프로세스 간 데이터 통신을 하기 위해 파이썬에서 제공하는 Queue API[12]를 사용하였고 $VI-Queue$는 DNN_T 각 은닉층의 출력 벡터를 VI-SVM 추론을 실행하는 프로세스들에게 전송하기 위한 큐이며 $PI-Queue$는 각 은닉층의 출력 벡터를 DM과 PI-SVM 추론을 실행하는 프로세스들에게 전송하기 위한 큐이다. DNN_T 추론 실행 후 프로세스 생성 오버헤드가 발생하지 않도록 DNN_T 추론 전에 적대적 공격 탐지에 필요한 프로세스들을 미리 생성한다.

L_1, L_2, L_3 을 기준으로 멀티 프로세스 NIC 시스템 동작을 그림 7의 ①～④으로 설명하면 다음과 같다. ① DNN_T의 추론이 시작되고 L_1의 출력 벡터가 L_2로 입력되기 전에 $VI-Queue$와 $PI-Queue$에 전달되고 DNN_T 프로세스에서 미리 생성된 $VI_1\,Proc.$과 $DM_1\,Proc.$로 신호를 보낸다. ②그 후 L_1의 출력 벡터

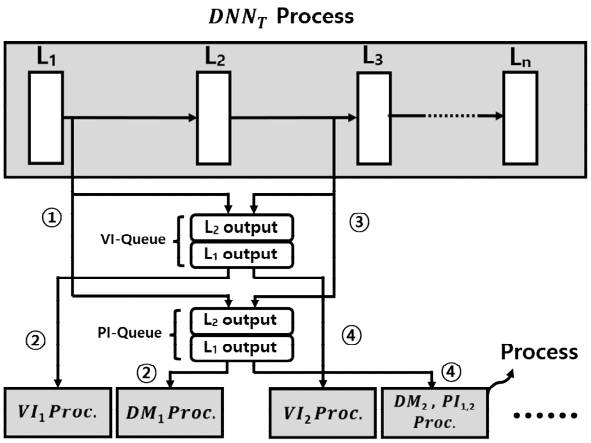


그림 7. 멀티 프로세스 NIC 시스템의 실행 구조
Fig. 7. Workflow of multi-process NIC system.

는 L_2에 입력되게 되고 신호를 받은 $VI_1 Proc.$과 $DM_1 Proc.$는 각각 $VI-Queue$과 $PI-Queue$로부터 L_1의 출력 벡터를 가져와서 VI-SVM 추론과 DM 추론을 실행하고 이때 DNN_T의 추론은 계속 진행되고 있다. ③L_2의 출력 벡터가 L_3로 입력되기 전에 L_2의 출력 벡터를 각각의 큐에 전달한다. ④DNN_T 프로세스로부터 신호를 받은 $VI_2 Proc.$와 $DM_2, PI_{1,2} Proc.$는 각각의 OCSVM과 DM 추론을 실행한다.

그림 8～9에서 Sol.은 임베디드 NIC 시스템의 결과를 나타내고 Multi-Process는 멀티 프로세스 NIC 시스템의 것을 나타낸다. 그림 8은 임베디드 NIC 시스템과 멀티 프로세스 NIC 시스템의 전체 소요시간을 나타낸다. 모든 타겟 DNN에 대하여 멀티 프로세스 NIC 시스템 대비 임베디드 NIC 시스템에서 전체 소요시간이 매우 감소함을 알 수 있다. Carlini는 93.3%, Alex는 94.3%, Mobile은 91.7%, Squeeze는 95.2% 소요시간이 감소하였다.

그림 9는 임베디드 NIC 시스템과 멀티 프로세스 NIC 시스템에서의 t_{diff}를 나타낸다. t_{diff}가 임베디드 NIC 시스템에서 많이 감소한 것을 확인할 수 있다. Carlini는 96.7%, Alex는 96.8%, Mobile은 99.6%, Squeeze는 99.3%만큼 t_{diff}가 감소했다. 4 종류의 타겟 DNN에서 t_{diff}가 감소했지만 Mobile과 Squeeze에서의 감소가 더 뚜렷했다. Carlini와 Alex의 총은닉층 개수는 각각 9개와 11개이고 Mobile과 Squeeze의 총은닉층 개수는 각각 28개와 21개이다. 따라서 타겟 DNN의 추론 시간이 Carlini와 Alex가 Mobile과 Squeeze에 비에 빨리 종료되는 반면 적대적 공격 탐지 종료 시간은 4종류

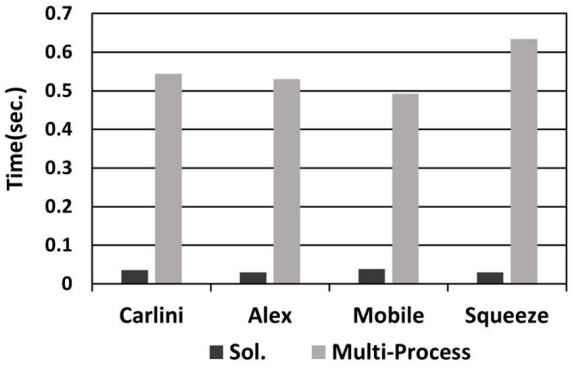


그림 8. NIC 적용 시 각 타겟 DNN의 전체 수행 시간 비교
Fig. 8. Comparing the overall execution time for each target DNN when applying NIC.

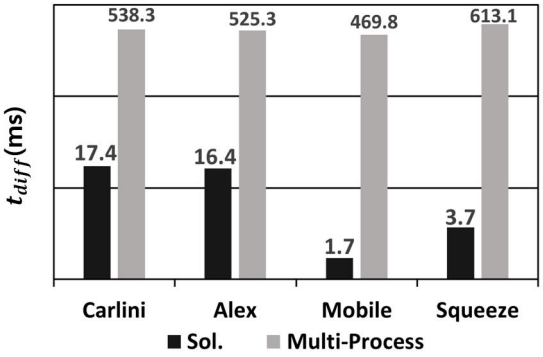


그림 9. NIC 적용 시 각 타겟 DNN의 t_{diff} 비교
Fig. 9. Comparing t_{diff} for each DNN when applying NIC.

의 타겟 DNN 간 차이가 미미하여 Mobile과 Squeeze처럼 은닉층이 깊은 DNN에서 t_{diff}의 감소율이 높은 것을 알 수 있다.

4. 최대 메모리 사용량

그림 10은 4개의 타겟 DNN을 각각 본 논문에서 제안한 소프트웨어 환경에서 적대적 공격 탐지율이 높은 특정 은닉층에서만 VI 또는 PI 위반 탐지 확인을 했을 때(Sol.)와 모든 은닉층에서 VI 또는 PI 위반 탐지 확인을 했을 때(NIC)의 최대 메모리 사용량을 나타낸다. 최대 메모리 사용량 측정은 NVIDIA(사)에서 제공하는 tegrastats 유틸리티[13]를 사용하여 진행했다. 각각의 메모리를 측정하기 전에 tegrastats 유틸리티를 실행하여 이미 사용되고 있는 메모리 사용량을 측정한 후 NIC가 적용된 시스템을 실행하여 0.01초마다 메모리 사용량을 측정하였다. 그 후 최대 메모리 사용량과 이미 사용되고 있던 메모리 사용량의 차이를 통해 타겟 DNN별 최

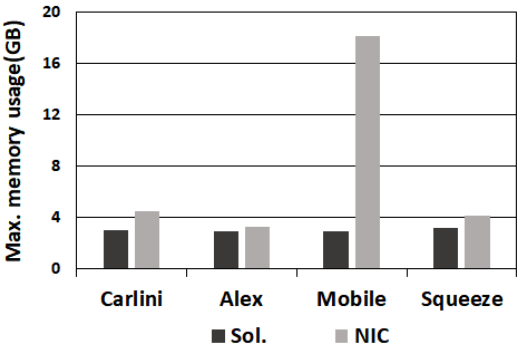


그림 10. 적대적 공격 탐지를 실행하는 은닉층 수에 따른 각 타겟 DNN의 최대 메모리 사용량 비교

Fig. 10. Comparing the maximum memory usage of each target DNN depending on the number of hidden layers executing adversarial attack detection.

대 메모리 사용량을 구했다. 또한 본 논문에서는 텍스트 형식으로 저장되던 OCSVM을 바이너리 형식으로 저장하여 메모리 사용량을 최소화했다.

그림 10에서 Carlini, Alex, Mobile, Squeeze 각각 최대 메모리 사용량이 특정 은닉층에서 적대적 공격 탐지를 했을 때 모든 은닉층에서 적대적 공격 탐지를 했을 때 대비 32.7%, 11.1%, 83.9%, 23.2% 감소했다. 앞에서 말했듯이 최대 메모리 사용량에 가장 큰 영향을 미치는 것은 VI-SVM 모델의 크기이다. Mobile의 VI-SVM 중 가장 큰 크기를 가지는 것이 Carlini, Alex, Squeeze 각각의 가장 큰 크기의 VI-SVM보다 10.3배, 65.7배, 14.4배만큼 컸기 때문에 Mobile의 최대 메모리 사용량 차이가 다른 타겟 DNN에 비해 유독 크다.

그림 10의 Mobile의 경우에서 알 수 있듯이 Carlini, Alex, Squeeze와 달리 NIC에서 핵심적으로 사용하는 VI-SVM와 PI-SVM을 적용할 경우 임베디드 환경에 어울리지 않은 매우 큰 최대 메모리 사용량(Mobile의 경우 18GB 이상)을 나타내는 DNN 모델들이 존재한다. 본 논문에서 제안한 특정 은닉층 선별 기법에 근거한 NIC 적용은 Mobile과 비슷하게 많은 은닉층을 가지는 DNN 모델들에도 적용할 수 있는 솔루션을 제공한다.

Jetson Xavier NX[14]의 시스템 메모리 크기는 본 논문에서 사용한 AGX-Xavier보다 작은 8GB이다. 본 논문에서 제안한 특정 은닉층에서만 적대적 공격 탐지를 실행하는 방법을 적용할 경우 Jetson Xavier NX와 같은 작은 메모리를 사용하는 임베디드 시스템에서도 타겟 DNN들에 NIC를 적용할 수 있다.

VI. 결 론

본 논문에서는 임베디드 시스템 환경에서 타겟 DNN의 추론 소요시간과 NIC를 이용한 적대적 공격 탐지 소요시간의 차이를 최대한 줄일 수 있는 임베디드 NIC 시스템을 제안하였다. 또한 메모리 크기가 제한적인 임베디드 시스템 환경에 맞추어 기존 NIC 방식이 모든 은닉층에서 적대적 공격에 대한 탐지를 수행했던 것을 일부 은닉층에서만 실행하여 필요로 한 OCSVM과 DM의 개수를 최대한 줄이면서 적대적 공격 탐지 정확도를 유지하는 방법을 제안하였다. 임베디드 NIC 시스템은 멀티 프로세스 NIC 시스템 대비 전체 소요시간이 평균 93.6% 감소하였다. 최대 메모리 사용량은 타겟 DNN의 모든 은닉층에서 적대적 공격 탐지를 했을 때 대비 평균 37.6% 감소하여 메모리가 부족한 임베디드 시스템에서도 타겟 DNN의 종류에 상관없이 적대적 공격 탐지가 가능함을 알 수 있었다.

REFERENCES

[1] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199. 2013

[2] Wang, Xingbin, et al. "Dnnguard: An elastic heterogeneous dnn accelerator architecture against adversarial attacks." Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. 2020.

[3] Ma, S., & Liu, Y. Nic: Detecting adversarial samples with neural network invariant checking. In Proceedings of the 26th network and distributed system security symposium (NDSS), San Diego, February. 2019.

[4] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. Estimating the support of a high-dimensional distribution. Neural computation, 13(7), 1443-1471. 2001.

[5] https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

[6] Carlini, N., & Wagner, D. Towards evaluating the robustness of neural networks. In 2017 ieee symposium on security and privacy (sp) (pp. 39-57). IEEE. Paris, France, May. 2017.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proceedings of the 26th

Conference on Neural Information Processing Systems, pp. 1097 - 1105, Lake Tahoe, 2012.
[8] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, abs/1704.04861, 2017.
[9] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360, 2016.
[10] Krizhevsky, A., & Hinton, G. Learning multiple layers of features from tiny images. 2009.
[11] Goodfellow, I. J., Shlens, J., & Szegedy, C. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572. 2014.
[12] https://docs.python.org/3.6/library/multiprocsiing.html
[13] https://docs.nvidia.com/drive/drive_os_5.1.6.1L/nvvib_docs/DRIVE_OS_Linux_SDK_Development_Guide/Utilities/util_tegrastats.html
[14] https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit

저 자 소 개

주 상 현(학생회원)
2021년 한성대학교
IT응용시스템공학
학사 졸업.

<주관심분야 : Adversarial Attack Detection, DNN framework, Deep Learning>

김 명 선(정회원)
2000년～2002년 LG전자
액세스네트워크 연구소
주임연구원
2002년～2011년 삼성전자
DMC 연구소 책임연구원
2016년 서울대학교
전기컴퓨터공학부
박사 졸업.
2016년～2019년 삼성전자 SR연구소 수석연구원
2019년～현재 한성대학교 AI응용학과 조교수
<주관심분야 : 인공지능 시스템, Linux kernel, HW/SW Co-desgin, NPU 등>

김 인 모(학생회원)
2021년 한성대학교
IT응용시스템공학
학사 졸업.

<주관심분야 : GPU Programing, DNN framework, Deep Learning>