(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0098317
(43) 공개일자 2025년07월01일

(51) 국제특허분류(Int. Cl.)
G06N 3/09 (2023.01) *G06N 3/045* (2023.01)
G06N 3/047 (2023.01) *G06N 5/04* (2023.01)
(52) CPC특허분류
G06N 3/09 (2023.01)
G06N 3/045 (2023.01)
(21) 출원번호 10-2023-0189484
(22) 출원일자 2023년12월22일
심사청구일자 2023년12월22일

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
백상헌
서울특별시 서초구 고무래로 35, 109동 1004호
김희원
경기도 고양시 덕양구 화중로 222, 204동 507호
윤성연
경기도 의정부시 시민로 27, 415호
(74) 대리인
김등용

전체 청구항 수 : 총 7 항

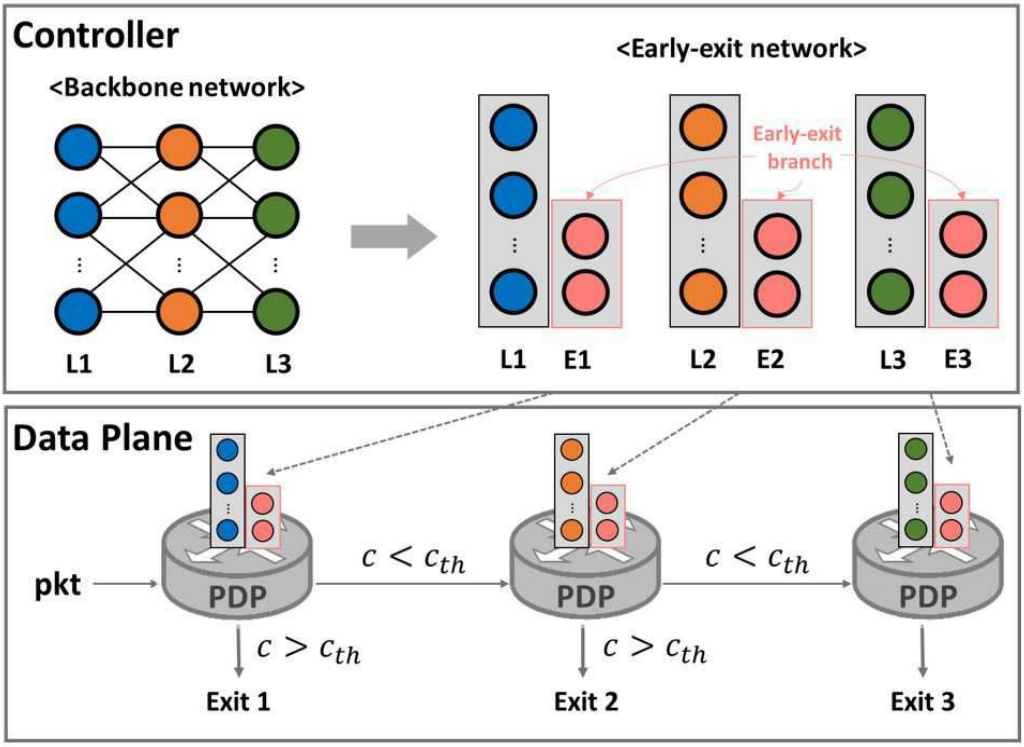
(54) 발명의 명칭 **프로그래밍 가능한 데이터 평면에서의 적응적 인-네트워크 추론 방법**

(57) 요 약

프로그래밍 가능한 데이터 평면에서의 적응적 인-네트워크 추론 방법이 개시된다. 상기 적응적 인-네트워크 추론 방법은 SDN(Software-Defined Network, 소프트웨어 정의 네트워크) 환경에서, 추론 경로를 구성하는 일련의 프로그래밍 가능한 스위치들(programmable switches) 중 적어도 하나에 의해 수행되는 적응적 인-네트워크 추론 방

(뒷면에 계속)

대 표 도 - 도2a



법으로, 상기 스위치들 중 제k(k는 1 보다는 크거나 같고 상기 스위치들의 개수 보다는 작거나 같은 임의의 자연수) 스위치에는 임의의 추론 모델에 포함된 복수의 레이어들 중 제k 레이어와 얼리 엑싯 레이어(early-exit layer)를 포함하는 제k 추론 모델이 설치되고, 상기 스위치들 중 제1 스위치가 제1 패킷을 수신하고, 상기 제1 패킷으로부터 제1 추론 모델의 제1 입력 데이터를 생성하는 단계, 상기 제1 스위치가 상기 제1 입력 데이터를 상기 제1 추론 모델에 입력하여 제1 중간 결과를 획득하는 단계, 상기 제1 스위치가 상기 제1 중간 결과에 기초하여 제1 분류 결과를 획득하는 단계, 및 상기 제1 스위치가 상기 제1 분류 결과에 기초하여, 얼리 엑싯(early-exit) 여부를 판단하는 단계를 포함한다.

(52) CPC특허분류
G06N 3/047 (2023.01)
G06N 5/04 (2023.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711193089
과제번호 2021-0-01810-003
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송혁신인재양성
연구과제명 양자세계를 연결하는 초신뢰 양자인터넷 요소 기술 개발
기 여 율 1/2
과제수행기관명 고려대학교산학협력단
연구기간 2023.01.01 ~ 2023.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711193554
과제번호 2022-0-00531-002
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 방송통신산업기술개발
연구과제명 In-Network Computing 기반 Network for AI 핵심기술 개발
기 여 율 1/2
과제수행기관명 한국과학기술원
연구기간 2023.01.01 ~ 2023.12.31

명 세 서

청구범위

청구항 1

SDN(Software-Defined Network, 소프트웨어 정의 네트워크) 환경에서, 추론 경로를 구성하는 일련의 프로그래밍 가능한 스위치들(programmable switches) 중 적어도 하나에 의해 수행되는 적응적 인-네트워크 추론 방법에 있어서,

상기 스위치들 중 제k(k는 1 보다는 크거나 같고 상기 스위치들의 개수 보다는 작거나 같은 임의의 자연수) 스위치에는 임의의 추론 모델에 포함된 복수의 레이어들 중 제k 레이어와 얼리 엑싯 레이어(early-exit layer)를 포함하는 제k 추론 모델이 설치되고,

상기 스위치들 중 제1 스위치가 제1 패킷을 수신하고, 상기 제1 패킷으로부터 제1 추론 모델의 제1 입력 데이터를 생성하는 단계;

상기 제1 스위치가 상기 제1 입력 데이터를 상기 제1 추론 모델에 입력하여 제1 중간 결과를 획득하는 단계;

상기 제1 스위치가 상기 제1 중간 결과에 기초하여 제1 분류 결과를 획득하는 단계; 및

상기 제1 스위치가 상기 제1 분류 결과에 기초하여, 얼리 엑싯(early-exit) 여부를 판단하는 단계를 포함하는 적응적 인-네트워크 추론 방법.

청구항 2

제1항에 있어서,

상기 임의의 추론 모델은 BNN(Binarized Neural Network, 이진 심층 신경망) 모델이고,

상기 제1 중간 결과는, 상기 제1 레이어에 포함된 뉴런들 각각의 출력에 포함된 "1"의 개수에 대한 활성화 결과를 포함하는,

적응적 인-네트워크 추론 방법.

청구항 3

제2항에 있어서,

상기 제1 분류 결과는, 분류 대상 라벨들 각각 대응하고 제1 얼리 엑싯 레이어에 포함되는 뉴런들 각각의 출력을 나타내는,

적응적 인-네트워크 추론 방법.

청구항 4

제3항에 있어서,

상기 얼리 엑싯 여부를 판단하는 단계는,

상기 제1 분류 결과에 포함되는, 각 라벨들 각각으로 분류될 확률에 기초하여 신뢰도 점수(confidence score)를 산출하고, 산출된 신뢰도 점수에 기초하여 얼리 엑싯 여부를 판단하는,

적응적 인-네트워크 추론 방법.

청구항 5

제4항에 있어서,

상기 얼리 엑싯 여부를 판단하는 단계는, 매치-액션 테이블(Match-Action Table, MAT)을 이용하여 신뢰도 점수를 산출하는,

적응적 인-네트워크 추론 방법.

청구항 6

제5항에 있어서,

상기 얼리 엑싯 여부를 판단하는 단계는, 산출된 신뢰도 점수가 임계치 보다 큰 경우, 얼리 엑싯을 결정하는,

적응적 인-네트워크 추론 방법.

청구항 7

제6항에 있어서,

산출된 신뢰도 점수가 임계치 보다 작은 경우, 상기 제1 중간 결과를 포함하는 제2 패킷을 제2 스위치로 송신하는 단계를 더 포함하는,

적응적 인-네트워크 추론 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 프로그래밍 가능한 데이터 평면에서의 적응적 인-네트워크 추론 방법에 관한 것이다.

배 경 기 술

[0002] 최근 데이터 평면에서 패킷을 빠른 속도로 처리하면서도 다양한 기능을 수행할 수 있는 프로그래밍 가능한 데이터 평면(Programmable Data Plane, PDP) 기술이 발전함에 따라, 데이터 평면 자체에서 패킷 및 플로우 분류, 이상 탐지 등을 수행하는 인-네트워크 인텔리전스(In-network Intelligence, INI) 기술이 개발되고 있다.

[0003] 초기 INI 기술들은 PDP의 메모리 및 계산 능력 한계로 인해 결정 트리(Decision Tree, DT)나 랜덤 포레스트(Random Forest, RF)와 같은 비교적 계산 복잡도가 높지 않은 간단한 모델을 활용하였다. pForest(비특허문헌 1)는 네트워크 트래픽 플로우를 빠르고 정확하게 분류하기 위해서 지도학습으로 미리 학습된 RF 모델을 매치-액션 테이블(Match-Action Table, MAT)을 활용하여 스위치에 올리는 방안을 제안하였다. 더 나아가, IIsy(비특허문헌 2)는 PDP의 자원 제약을 고려하여 서포트 벡터 머신(support vector machine, SVM), 나이브 베이즈 분류(Naive Bayes classifier) 등과 같은 보다 다양한 모델을 데이터 평면에 구현하였다.

[0004] 이러한 INI는 별도의 컨트롤 평면의 개입을 없애거나 최소화하여 서버를 거치는 경우보다 지연 시간을 크게 줄일 수 있다. 또한, 서버로 전달되는 과정에서의 데이터 손실 및 침입 등의 가능성이 사라져 보안적인 측면에도 유리한 면이 있다.

[0005] 한편, 심층 신경망(Deep Neural Network, DNN)은 입력과 가중치의 곱을 모두 더한 뒤 활성화 함수를 거치는 선형 분류기 구조인 다수의 퍼셉트론으로 구성된 네트워크이며, 이와 같은 구조로 뇌의 신경망 구조를 모방하여 뉴런과 뉴런을 연결하는 시냅스 결합의 가중치를 변화시켜 학습하는 기술이다. DNN은 분류, 클러스터링 등 복잡한 문제를 풀기 위한 별도의 복잡한 수학적 모델링 없이도 높은 정확도로 결과를 낼 수 있기 때문에 음성인식, 이미지 인식 등의 다양한 분야에서 널리 활용되고 있으며 최근 통신 네트워크의 구조가 복잡해지고 그 규모가 증가함에 따라 이를 효율적으로 모니터링 및 관리하기 위해서 DNN을 적극적으로 활용한다. 따라서 INI에서도 PDP의 자원 및 계산 능력의 한계를 극복하고 DNN을 직간접적으로 PDP에 구현하는 시도가 이루어진 바 있다.

[0006] P4BNN(비특허문헌 3)은 DNN의 모델 크기를 줄이고 부동 소수점 기반의 복잡한 계산을 단순화하기 위해 이진 심층 신경망(Binarized Neural Network, BNN)을 활용한다. Mousika(비특허문헌 4)는 프로그래밍 가능한 데이터 평면의 메모리 및 연산 기능의 한계를 극복하면서도, 복잡한 모델의 정확한 성능을 제공하기 위한 기법으로 경량화 기법인 지식 증류(Knowledge Distillation, KD)를 사용하여 큰 모델을 스위치에 간접적으로 구현하였다. NNSplit(비특허문헌 5), JointNIDS(비특허문헌 6)는 복잡한 모델을 수직적 또는 수평적으로 분할하여 큰 모델을 다중 스위치에 나누어 배치하고 보다 좋은 성능을 얻는 방식을 적용한 기술들이다. 이러한 기술들은 PDP가 가지는 한계를 극복하고 큰 모델을 분산적으로 배치하여 트래픽 분류나 이상 감지 등의 복잡하고 풀기 어려운 문제를 보다 정확하게 다룰 수 있다.

[0007] 같은 조건에서 보다 좋은 추론 성능을 얻기 위해서는 더 크고 깊은 DNN 모델을 사용하는 것이 일반적이다. 하지만, PDP 같은 경우 다양한 기능을 수행하는 것 뿐만 아니라 빠른 속도로 패킷을 처리하고 전송하는 과정을 최적화 하는 것 역시 중요하기 때문에 INI에 활용할 수 있는 메모리 및 스테이지(파이프라인의 크기) 자원의 한계가 뚜렷하고, 이에 따라 PDP에 올릴 수 있는 모델의 크기 및 복잡도도 제한된다.

[0008] 이에 따라 최근 도 1과 같이, 이러한 큰 모델을 다중 PDP 스위치에 나누어 배치하는 분산적인 솔루션이 제안되고 있다. 예를 들어, 비특허문헌 5는 DNN 모델을 레이어마다 분리하여 각 스위치에 배치함으로써 패킷이 각 스위치를 지날 때 순차적으로 추론이 진행되는 시스템을 구현하였다. 하지만 별도의 지능적인 동작 과정 없이 단순히 큰 모델을 나누어 배치하는 것이 한 패킷의 전반적인 추론 과정에 큰 영향을 주지 못하기 때문에 결과적으로 지연시간 및 처리율 관점에서 의미 있는 성능 향상을 보여주지 못한다.

[0009] 또 다른 예로, 비특허문헌 6은 하나의 DNN 모델을 레이어 별이 아닌 수평적으로 2개의 부 모델로 분리하고 이를 각각 주 스위치와 부 스위치에 올리는 방안을 제안하였다. 해당 기법 같은 경우 주 스위치에서 우선적으로 추론을 진행하고 그 결과의 크기를 확인하여 부 스위치에서 추가적인 추론을 진행할지 여부를 결정하기 때문에 보다 지능적인 추론이 가능하다. 하지만 모델을 수평적으로 분리하는 것은 일반적으로 적응적 추론에서 사용되는 방식이 아닌데, 그 이유는 모델을 수평적으로 나누게 되면 모델의 깊이 대비 한 레이어의 크기가 지나치게 줄어들게 되어 성능이 보장되지 못하며 따라서 모델을 3개 이상으로 나누기 힘들어지기 때문이다. 비특허문헌 6과 같은 경우 이 문제를 해결하기 위해 임의로 중복되는 뉴런을 지정하나 이러한 방식은 전체적인 모델 크기를 키워 추가적인 자원을 소모하게 된다는 단점이 있다.

[0010] 따라서, 본 발명에서는 얼리 엑싯(early-exit, 빠른 종료) 기술을 INI에 적용하여, 1) 큰 모델을 레이어 별로 각 스위치에 나누어 배치하고, 2) 각 스위치(레이어)에서 순차적으로 추론을 진행하여, 만약 충분한 성능이 보장된다면 추론을 종료하는 적응적 인-네트워크 추론 기술을 제안한다.

[0011] 얼리 엑싯(Early-exit)은 모델의 레이어의 원하는 지점에 엑싯(exit, 종료) 여부를 결정하는 분류기를 넣고, 추론 결과를 중간 지점에서 확인하여 신뢰도 점수(confidence score)가 일정 수준 이상 높으면 추론을 멈추고 해당 결과를 사용하는 기법이다(Branchynet(비특허문헌 7)). 하지만, 현재 PDP는 정수 덧셈 및 비트(bit) 연산 등의 간단한 연산만을 지원하기 때문에 신뢰도 점수(confidence score)를 PDP에서 직접 계산하기는 어렵다.

[0012] 또한, 얼리 엑싯 모델을 학습하는 기법은 1) 전체 엑싯 모델을 학습하는 기법과 2) 백본 네트워크와 분리하여 엑싯 네트워크를 학습하는 기법이 있다. 이에 관한 상세한 설명은 비특허문헌 7이 참조될 수 있다. 본 발명에서는, 임의의 기법을 사용하여, 사전에 학습된 모델을 가정한다.

[0013] 이를 극복하기 위해, 본 발명에서는 신뢰도 점수(confidence score)의 정밀도(granularity)를 조정하고 이를 PDP의 대표적인 프로그래밍 언어인 P4(비특허문헌 9) 에서 제공하는 exact 및 range 매치 연산 기반의 MAT로 표현하는 방식을 활용한다. 그리고 이와 같은 방법으로 얻어진 신뢰도 점수(confidence score)를 기반으로 각 스위치에서 얼리 엑싯(early-exit)을 수행할지 여부를 결정하는 방법을 제안한다. 이를 통해 본 발명에서 제안하는 적응적 인-네트워크 추론 시스템을 통해 추론 정확도와 처리 비용 사이의 트레이드 오프를 효과적으로 컨트롤 할 수 있을 것으로 기대한다.

선행기술문헌

특허문헌

[0014] (특허문헌 0001) 대한민국 공개특허 제2020-0023239호 (2020.03.04. 공개)

(특허문헌 0002) 대한민국 공개특허 제2022-0045800호 (2022.04.13. 공개)

(특허문헌 0003) 대한민국 공개특허 제2019-0089628호 (2019.07.31. 공개)

비특허문헌

[0015] (비특허문헌 0001) C. Busse-Grawitz, R. Meier, A. Dietmuller, T. Buhler, and L. Vanbever, "pforest: In-network inference with random forests," arXiv (2019).

(비특허문헌 0002) Changgang Zheng, Zhaoqi Xiong, Thanh T Bui, Siim Kaupmees, Riyad Bensoussane,

Antoine Bernabeu, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman, "IIsy: Practical In-Network Classification," arXiv (2022).

(비특허문헌 0003) Qiaofeng Qin, Konstantinos Poularakis, Kin K. Leung, and Leandros Tassiulas, "Line-Speed and Scalable Intrusion Detection at the Network Edge via Federated Learning," in Proc. IFIP Networking Conference, Virtual conference, June 2020.

(비특허문헌 0004) Guorui Xie, Qing Li, Yutao Dong, Guanglin Duan, Yong Jiang, and Jingpu Duan, "Mousika: Enable General In-Network Intelligence in Programmable Switches by Knowledge Distillation," in Proc. IEEE Conference on Computer Communications (INFOCOM), Virtual conference, May 2022.

(비특허문헌 0005) David Chunhu Li, Muhamad Rizka Maulana, and Li-Der Chou, "NNSplit-SØSupporting the model implementation of large neural networks in a programmable data plane," Computer Networks, Vol. 222 (article number: 109537), February 2023.

(비특허문헌 0006) Thi-Nga Daoand and HyungJune Lee, "JointNIDS: Efficient Joint Traffic Management for On-Device Network Intrusion Detection," IEEE Transactions on Vehicular Technology, Vol. 71, No. 12, pp. 13254-13265, December 2022.

(비특허문헌 0007) Surat Teerapittayanon, Bradley McDanel, and H.T. Kung, "BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks," in Proc. International Conference on Pattern Recognition (ICPR), Cancun, Mexico, December 2016.

(비특허문헌 0008) Elaheh Biglar Beigi, Hossein Hadian Jazi, Natalia Stakhanova, and Ali A. Ghorbani, "Towards effective feature selection in machine learningbased botnet detection approaches," in Proc. IEEE Conference on Communications and Network Security, CA, USA, October 2014.

(비특허문헌 0009) P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese and D. Walker, "P4: Programming Protocol-Independent Packet Processors," SIGCOMM Computer Communication Review(CCR), vol. 44, no. 3, pp. 87-95, 2014.

(비특허문헌 0010) Elaheh Biglar Beigi, Hossein Hadian Jazi, Natalia Stakhanova, and Ali A. Ghorbani, "Towards effective feature selection in machine learning based botnet detection approaches," in Proc. IEEE Conference on Communications and Network Security, CA, USA, October 2014.

발명의 내용

해결하려는 과제

[0016] 본 발명이 이루고자 하는 기술적인 과제는 프로그래밍 가능한 데이터 평면에서의 적응적 인-네트워크 추론 방법을 제공하는 것이다.

과제의 해결 수단

[0017] 본 발명의 일 실시예에 따른 적응적 인-네트워크 추론 방법은, SDN(Software-Defined Network, 소프트웨어 정의 네트워크) 환경에서, 추론 경로를 구성하는 일련의 프로그래밍 가능한 스위치들(programmable switches) 중 적어도 하나에 의해 수행되는 적응적 인-네트워크 추론 방법으로, 상기 스위치들 중 제k(k는 1 보다는 크거나 같고 상기 스위치들의 개수 보다는 작거나 같은 임의의 자연수) 스위치에는 임의의 추론 모델에 포함된 복수의 레이어들 중 제k 레이어와 얼리 엑싯 레이어(early-exit layer)를 포함하는 제k 추론 모델이 설치되고, 상기 스위치들 중 제1 스위치가 제1 패킷을 수신하고, 상기 제1 패킷으로부터 제1 추론 모델의 제1 입력 데이터를 생성하는 단계, 상기 제1 스위치가 상기 제1 입력 데이터를 상기 제1 추론 모델에 입력하여 제1 중간 결과를 획득하는 단계, 상기 제1 스위치가 상기 제1 중간 결과에 기초하여 제1 분류 결과를 획득하는 단계, 및 상기 제1 스위치가 상기 제1 분류 결과에 기초하여, 얼리 엑싯(early-exit) 여부를 판단하는 단계를 포함한다.

발명의 효과

[0018] 본 발명의 실시예에 따른, 프로그래밍 가능한 데이터 평면에서의 적응적 인-네트워크 추론 방법에 의할 경우,

얼리 엑싯(early-exit) 기술을 INI에 적용하여 추론 정확도와 처리 비용 사이의 트레이드 오프를 효과적으로 제어할 수 있는 효과가 있다.

도면의 간단한 설명

[0019] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 상세한 설명이 제공된다.

도 1은 큰 모델의 일부분을 프로그래밍 가능한 스위치에 구현하는 분산 솔루션을 설명하기 위한 도면이다.

도 2a는 본 발명의 일 실시예에 따른 시스템 모델을 도시한다.

도 2b는 추론 추위치에서의 동작 과정을 설명하기 위한 도면이다.

도 3은 MAT를 활용한 신뢰도 점수(confidence score) 계산 과정을 설명하기 위한 도면이다.

도 4는 본 발명의 일 실시예에 따른 적응적 인-네트워크 추론 방법을 설명하기 위한 흐름도이다.

도 5는 신뢰도 점수 임계치에 따른 각 스위치에서의 엑싯 수를 나타내는 그래프를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0020] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태들로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

[0021] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.

[0022] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.

[0023] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0024] 본 명세서에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0025] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0026] 이하, 본 명세서에 첨부된 도면들을 참조하여 본 발명의 실시예들을 상세히 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0027] 도 2a는 본 발명에서 제안하는 시스템 모델을 도시한다. 본 발명은 PDP 기반의 스위치(즉, 프로그래밍 가능한

스위치(programmable switches))와 컨트롤러(controller)로 이루어진 소프트웨어 정의 네트워크(Software-Defined Network, SDN) 환경을 가정한다.

[0028] 컨트롤러는 기존 모델인 백본 네트워크(backbone network, 인공 신경망 기반의 모델로써, 학습이 완료된 추론 모델을 의미할 수 있음)를 각 레이어마다 분리하고, 각각의 레이어에 얼리 엑싯 브랜치(early-exit branch, 얼리 엑싯 레이어 등으로 명명될 수도 있음)를 추가한다. 이때, 얼리 엑싯 브랜치(early-exit branch)는 중간 추론 결과를 바탕으로 엑싯(exit) 가능 여부를 분류해 주는 분류기의 기능을 수행하거나, 중간 추론 결과를 바탕으로 분류 대상 라벨 각각으로 분류될 확률을 도출하는 것으로 이해될 수 있다.

[0029] 또한, 본 발명에서 활용하는 추론 모델은 BNN(Binarized Neual Network, 이진 심층 신경망)을 의미할 수 있으며, 미리 정해진 추론 동작을 수행할 수 있다. 예시적인 추론 동작은 패킷 및/또는 플로우 분류, 이상 탐지 등이 있을 수 있다. BNN은 가중치 및/또는 활성화 함수를 이진 값(예컨대, 0 또는 1)으로 제한하여 동작하는 신경망을 의미한다. 연속적인 가중치를 사용하는 대신 0 또는 1과 같은 이진 값을 사용하여 계산을 단순화할 수 있다. 이에 따라, 추론 모델은 작고 빠르게 실행되며, 메모리 요구 사항을 감소시킬 수 있다.

[0030] 또한, 추론 모델의 각 레이어를 분리하는 것으로 설명하였으나, 실시예에 따라서는 연속되는 적어도 하나의 레이어가 포함되도록 분리되는 것도 가능하다. 이 경우, 연속되는 2 이상의 레이어에 얼리 엑싯 브랜치가 추가된 소규모 모델이 스위치에 설치되는 것도 가능하다.

[0031] 또한, 추론 기능을 수행하는 스위치를 추론 스위치로 명명할 수 있으며, 일련의 스위치들은 추론 경로(inference path)를 구성한다.

[0032] 컨트롤러에 의해 분리된 각 레이어와 얼리 엑싯 브랜치(early-exit branch) 한 쌍은 각 추론 스위치(프로그래밍 가능한 스위치)에 배포되어 설치되고(또는 저장되고), 한 레이어마다 도출된 결과를 바탕으로 엑싯(exit) 가능 여부를 확인할 수 있다. 만약 엑싯(exit)이 가능한 것으로 확인되면 해당 스위치에서 추론 결과를 바로 패킷 헤더에 저장하고 추론을 위한 경로에서 벗어나 패킷의 목적지(destination)를 향해 전송될 수 있다. 반대로 엑싯(exit)이 불가능하다면 계속해서 추론과정을 진행하기 위해 중간 추론 데이터를 패킷 헤더에 추가하여 다음 추론 스위치로 전송해 이후 추론을 이어 나간다. 다음 추론 스위치에서는 앞 스위치에서 추론한 중간 추론 데이터를 바탕으로 다음 추론 과정을 진행하고, 이를 바탕으로 앞서 설명한 단계를 반복한다. 만약 해당 과정이 마지막 추론 스위치까지 진행된다면 마지막 추론 스위치에서 추론 과정을 일괄적으로 종료한 후 목적지로 전송할 수 있다.

[0033] 하나의 추론 스위치에서의 동작 과정을 보다 자세히 서술하면 다음과 같다. 추론 추위치에서의 동작 과정을 도 2b에 도시되어 있다.

[0034] 먼저 패킷이 들어오면 파싱하는 과정을 거친다. 이때, 만약 해당 스위치가 첫 번째 추론 스위치라면 파싱의 결과 중 이더넷, IPv4, TCP 등의 정보를 넣어 추론 과정에 필요한 인풋(추론 모델의 입력을 의미할 수 있음)을 생성한다. 반대로 첫 번째 추론 스위치 이후의 스위치라면 'Inference'라는 커스텀 헤더 안에 저장된 중간 추론 데이터를 인풋으로 사용한다.

[0035] 상술한 과정을 통해 얻어진 인풋(생성된 인풋 또는 추출된 중간 추론 데이터)은 이후 스위치의 BNN(Binarized Neural Network, 이진 심층 신경망) 추론 과정을 거치게 된다. 구체적으로, 인풋을 레이어의 각 뉴런 가중치와 비트별 XNOR(bitwise XNOR) 연산을 통해 비트열을 얻고, 이를 비트카운트(bitcount) 연산을 거쳐 비트열에 포함된 "1"의 개수를 산출할 수 있다. 이후 산출된 "1"의 개수를 활성화 함수(activation function)에 넣어 비트열 길이의 절반 이상의 1이 존재하면 1, 그렇지 않으면 0으로 활성화(activation) 결과를 도출할 수 있다. 이 과정을 각 레이어의 뉴런 만큼 반복하고, 각 뉴런에서의 활성화(activation) 결과는 뉴런의 순서에 맞게 비트열로 축적되며 중간 추론 데이터로 저장된다.

[0036] 이 중간 추론 데이터는 이후 논리적으로 레이어의 바로 뒤에 연결되어 있는 얼리 엑싯 브랜치(early-exit branch)의 인풋 데이터로 활용된다. 얼리 엑싯 브랜치(Early-exit branch)의 추론 과정 역시 앞선 레이어의 추론 과정과 동일하게 브랜치(branch) 내의 뉴런의 가중치로 비트별 XNOR(bitwise XNOR) 및 비트카운트(bitcount) 연산을 거친다. 이때, 활성화(activation)을 하기 전의 비트카운트(bitcount)의 결과는 해당 뉴런이 나타내는 추론 라벨의 정답 확률로 볼 수 있다. 따라서 각 얼리 엑싯 브랜치(early-exit branch)에 속한 뉴런들이 생성하는 결과값들을 바탕으로 해당 레이어의 얼리 엑싯 브랜치(early-exit branch)에서 수행된 추론에 대한 신뢰도에 해당하는 신뢰도 점수(confidence score)를 구해, 정답으로 제시된 라벨이 실제 충분한 정도의 신뢰성을 갖는지 여부를 판단해야 한다. 이를테면, 만약 특정 라벨의 값이 타 라벨의 값에 비해 그 크기가 매우 크다

면 해당 추론은 높은 신뢰도 점수(confidence score)를 가져야 한다. 반대로, 만약 라벨들의 결과값이 어떠한 지배적인 값 없이 비슷한 크기를 갖는다면 해당 추론은 낮은 신뢰도 점수(confidence score)를 갖게 한다. 이와 같이 레이어의 뉴런들이 생성한 결과값을 바탕으로 신뢰도 점수(confidence score)를 계산하는 방법으로 소프트맥스(Softmax) 함수를 사용한다. 이를 수식으로 표현하면 C 가 레이블의 집합이고, y_c는 레이블 c의 결과값이라고 할 때, $\Omega(c) = \exp(y_c)/\sum_{i \in C} \exp(yi)$이다. 하지만, 해당 방식은 PDP 자체로는 수행할 수 없는 지수 계산과 같은 복잡한 연산이 포함되어 있다. 따라서 PDP에서 신뢰도 점수(confidence score)를 얻어내기 위해서는 별도의 방법이 필요하다.

[0037] 도 3은 본 발명이 제안하는 MAT 기반의 신뢰도 점수 계산 방법을 나타낸다. MAT(매치-액션 테이블)의 각 키(key)에는 두 뉴런의 결과값이 사용되는데, 만약 모든 키(key)들의 조합에 대한 신뢰도 점수(confidence score)의 엔트리를 준비한다면 뉴런 가중치 비트열 길이의 제곱에 해당하는 개수의 엔트리를 포함해야 하므로 테이블의 크기가 지나치게 커지게 된다는 문제가 존재한다. 따라서 본 발명에서는 신뢰도 점수(confidence score)의 정밀도(granularity)를 조절하여, 두 키(key) 중 하나의 키(key)의 값을 고정하고(exact match) 다른 키(key)의 값을 범위로 표현이(range match) 가능하다는 점을 활용한다.

[0038] 예를 들어, 도 3은 뉴런 가중치 비트열의 길이가 126bit인 경우의 MAT 일부를 나타낸다. 이때, 만약 두 뉴런의 결과값이 각각 56, 85일 때, 이 중 작은 값인 56을 exact match로 두고 더 큰 값을 85로 둘 경우 도 3과 같이 해당 키(key)들이 신뢰도 점수(confidence score)가 0.7인 엔트리에 속하는 것을 볼 수 있다.

[0039] 신뢰도 점수(confidence score)가 계산된 이후 해당 값이 미리 결정되어 있는 신뢰도 점수 임계치(confidence score threshold)보다 더 큰지 여부를 판단한다. 만약 값이 더 크다면 해당 추론 스위치, 즉 레이어에서의 추론이 충분히 큰 신뢰도를 갖으므로 얼리 엑싯(early-exit)을 수행하고, 만약 그렇지 않다면 중간 추론 데이터를 패킷에 실어 다음 추론 스위치로 포워딩 한다.

[0040] 학습된 모델을 레이어 별로 분할하여 프로그래밍 가능한 스위치에 구현하고, 각 레이어마다 추론 결과를 기반으로 해당 레이어에서 추론을 종료해도 되는지 확인함으로써 PDP 내에서 추론 성능과 처리 비용 간의 트레이드 오프를 효과적으로 통제할 수 있는 본 발명의 동작 과정은 다음과 같이 도 4의 흐름도로 표현될 수 있다. 해당 과정은 발명의 핵심 부분인 데이터 평면, 즉 한 스위치 내의 동작과정을 설명하고 있다.

[0041] 1. 각 스위치에 패킷이 전송되면 헤더를 순서대로 확인하며 어떤 패킷이 들어왔는지 확인한다. 이더타입(Ethertype) 값을 통해 정의된 추론(Inference) 헤더가 있는지 확인하고, 있다면 추가적으로 이 헤더에 저장된 값을, 없다면 일반 IPv4, TCP/UDP 헤더에 저장된 값들 만을 파싱한다.

[0042] 2. 그리고 스위치 ID를 확인하여 첫번째 스위치인지 확인한다. 만약, 첫번째 스위치라면, 앞에서 파싱한 헤더 필드에 저장된 값으로 사용자 정의에 따라 인풋 값을 빌드한다. 반면, 첫번째 스위치가 아니라면, 파싱 과정에서 확인한 추론(Inference) 헤더에 담겨있는 중간 추론 데이터를 인풋으로 한다.

[0043] 3. 인풋을 스위치가 가진 모델 레이어에 넣으면 레이어 내의 각 뉴런들로 연산과정을 수행한다. 먼저, 스위치 레지스터에 이미 학습된 모델의 가중치를 가져와 저장한다.

[0044] 4. 모든 뉴런에 대해서, Output = NN input ⊙ Neuron weight 연산을 수행하고 그 결과로 구성된 Output은 bitcount(Output)을 통해 1의 개수를 세어 결과값을 구한다. 이후 결과값에 활성화(activation)을 적용해 해당 비트카운트(bitcount)의 결과가 과반수 이상이면 1, 그렇지 않으면 0으로 결정한다.

[0045] 5. 해당 추론의 결과는 중간 결과(Intermediate result)에 차례대로 저장해 둔다.

[0046] 6. 레이어 내의 연산이 끝났으므로, 레이어 연산 결과를 통해 엑싯(exit) 여부를 판단하기 위해 해당 레이어 다음에 미리 정의해둔 얼리 엑싯(early-exit) 뉴런을 수행한다. 마찬가지로, EE = NN input ⊙ Neuron weight 연산을 모든 얼리 엑싯(early-exit) 뉴런에서 수행하고, 모든 뉴런의 결과로 구성된 EE는 bitcount(EE)을 통해 1의 개수를 센다.

[0047] 7. 얼리 엑싯(Early-exit) 분류기를 거친 후 활성화(activation)을 거치기 전의 결과값을 기반으로 신뢰도 점수(confidence score)를 계산한다. 구체적으로는 각 경우에 대한 값이 미리 저장되어 있는 엔트리들로 구성된 MAT에 두 결과값을 이그젝트 및 레인지 매치 연산(exact and range match operation)을 적용하여 신뢰도 점수(confidence score)를 구한다.

[0048] 8. 만약 신뢰도 점수(confidence score)의 값이 신뢰도 점수 임계치보다 크면, 현재까지의 추론 결과를 최종 결

과로 하고 다음 스위치로는 정해진 경로에 따라 포워딩한다.

[0049] 9. 반대의 경우 엑싯(exit)이 되지 않기 때문에 현재까지의 추론 연산 결과인 중간 결과(Intermediate result)에 저장된 값을 추론(inference) 헤더에 담아 다음 추론 스위치로 전송한다. 다음 스위치에서는 다음 추론과정을 이어서 수행하고 추론 경로대로 포워딩한다.

[0050] 성능 평가 결과

[0051] 성능 평가는, 각 스위치가 BMv2 소프트웨어 P4 스위치로 구성된 Mininet 기반의 가상 네트워크에서 수행되었다. 각 P4 스위치는 3개의 완전 연결 레이어(fully-connected layer)로 구성된 모델의 각 레이어와 두 개의 얼리 엑싯(Early-exit) 뉴런으로 구성된다. 학습을 위해 ISCX Botnet 2014(비특허문헌 10) 데이터셋을 활용하였으며, 이는 일반적인 패킷과 더불어 Synflood 공격이 담겨 있으며 따라서 이상 감지(Malware detection)를 위해 사용된다. 학습 및 추론을 위한 인풋은 IP 프로토콜 식별자, IP 주소, IP 헤더의 길이, L4 포트, TCP 컨트롤 플래그를 조합한 126비트의 비트열로 구성한다. 10,000개의 패킷이 학습에 사용되었고, 2,000개의 다른 패킷이 테스트를 위해 사용되었다. 신뢰도 점수(Confidence score)를 계산하기 위해서는 소프트맥스(Softmax) 함수를 사용하였고, 정밀도를 0.1로 설정한 후 미리 계산된 엔트리를 P4 rule의 형태로 정의하여 스위치 작동 시 컨트롤 평면에서 넣어주었다. 성능 평가는 스태틱 엑싯(static-exit) 환경과 어댑티브 엑싯(adaptive-exit) 환경으로 나누어 수행하였다. 먼저, 스태틱 엑싯(static-exit) 환경은 패킷이 특정 엑싯(exit) 지점에서 무조건 엑싯(exit)하여 추론을 멈추도록 구성된 환경이다. 그리고, 어댑티브 엑싯(adaptive-exit) 환경은 패킷이 임계치를 넘어야 엑싯(exit)되는 전략에 따라서 엑싯(exit) 여부를 판단하도록 구성된 환경이다.

[0052] 각 환경에 따른 분류 성능은 표 1과 표 2를 통해 확인할 수 있다. 표 1에서는 추론과정이 길어질수록 분류 성능은 좋아지지만, 프로세싱 비용은 증가하기 때문에 정확도와 프로세싱 비용 사이의 트레이드 오프를 확인할 수 있다. 한편, 표 2에서는 신뢰도 점수(confidence score)의 임계치가 증가함에 따라 성능이 좋아짐을 볼 수 있다. 이는 임계치가 높아질수록 더 신뢰도가 높은 추론 결과만 엑싯(exit)이 가능하기 때문이다.

[0053] [표 1]

[0054]

Exit	Recall	Precision	Accuracy	F1-score
1	66.94%	91.20%	63.55%	0.7721
2	97.82%	92.64%	90.83%	0.9516
3	100%	92.37%	92.38%	0.9604

[0055] [표 2]

[0056]

Thresh.	Recall	Precision	Accuracy	F1-score
0.60	67.01%	91.20%	63.55%	0.7726
0.70	67.88%	91.28%	64.34%	0.7786
0.80	97.78%	92.64%	90.78%	0.9514
0.90	99.33%	92.32%	91.75%	0.9570

[0057] 도 5는 신뢰도 점수 임계치에 따른 각 엑싯(exit) 지점에 해당하는 스위치마다의 엑싯(exit) 수를 나타낸다. 도 5를 참조하면, 임계치가 작을 때는 앞쪽 스위치에서 빠르게 엑싯(exit)되는 패킷 수가 많으며 반대로 신뢰도 점수 임계치가 클수록 엑싯(exit)이 더 뒤쪽 스위치에서 되는 경향을 확인할 수 있다. 이는 임계치의 값이 클수록 더 쉽게 엑싯(exit)이 되기 힘들며, 또한 표 1에서의 결과와 같이 본 발명에서 뒤쪽 스위치로 갈수록 추론 성능이 더 좋아지며 이에 따라 신뢰도 점수의 값이 일반적으로 더 커지기 때문이다. 즉, 신뢰도 점수 임계치를 조절하면 전반적으로 패킷들이 엑싯(exit)되는 지점을 변경할 수 있으며, 이에 따라 추론 정확도와 처리 비용 사이의 트레이드 오프를 효과적으로 컨트롤 할 수 있음을 알 수가 있다.

[0058] 한편 표 1, 표 2, 및 도 5의 결과를 종합해 볼 때, 어댑티브 엑싯(adaptive-exit)의 경우 적절한 신뢰도 점수 임계치를 선택한다면 스태틱 엑싯(static-exit)의 경우보다 더 합리적인 최적의 지점(sweet spot)을 찾을 수 있음을 알 수 있다. 예를 들어, 신뢰도 점수 임계치를 0.8로 설정할 경우 추론 정확도의 성능은 90.78%로 스태틱(static)하게 Exit 2에서 엑싯(exit)하는 경우의 90.83%와 거의 유사하나, 도 5의 결과를 확인해보면 많은 비율의 패킷이 Exit 1에서 엑싯(exit)되어 더 적은 처리 비용을 가지는 것을 볼 수가 있다.

[0059] 본 발명은 기본적으로 모델을 레이어 별로 나누어 배치하는 전략을 사용하기 때문에 하나의 PDP 스위치에서 사용할 수 있는 자원의 양이 한정적일 때 이를 극복하고자 활용할 수 있다. 또한, 전송되는 패킷 관점에서 추론 과정을 중간에 빠져나올 수 있기 때문에 본 발명을 활용하면 패킷의 지연(latency) 및 혼잡 감소 측면에서 유리한 측면이 있다.

[0060] 이상에서 설명된 장치는 하드웨어 구성 요소, 소프트웨어 구성 요소, 및/또는 하드웨어 구성 요소 및 소프트웨어 구성 요소의 집합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성 요소는, 예를 들어, 프로세서, 콘트롤러, ALU(Arithmetic Logic Unit), 디지털 신호 프로세서(Digital Signal Processor), 마이크로컴퓨터, FPA(Field Programmable array), PLU(Programmable Logic Unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(Operation System, OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술 분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(Processing Element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(Parallel Processor)와 같은, 다른 처리 구성(Processing Configuration)도 가능하다.

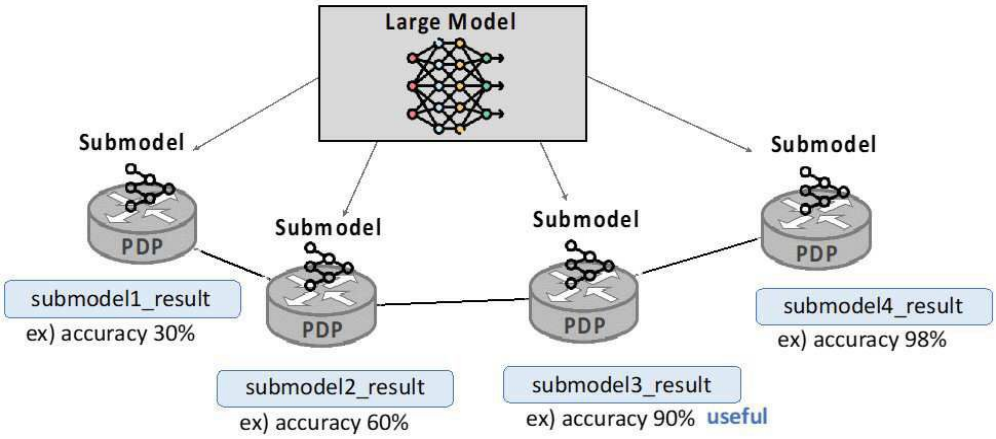
[0061] 소프트웨어는 컴퓨터 프로그램(Computer Program), 코드(Code), 명령(Instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(Collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성 요소(Component), 물리적 장치, 가상 장치(Virtual Equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(Signal Wave)에 영구적으로, 또는 일시적으로 구체화(Embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0062] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM, DVD와 같은 광기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-optical Media), 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

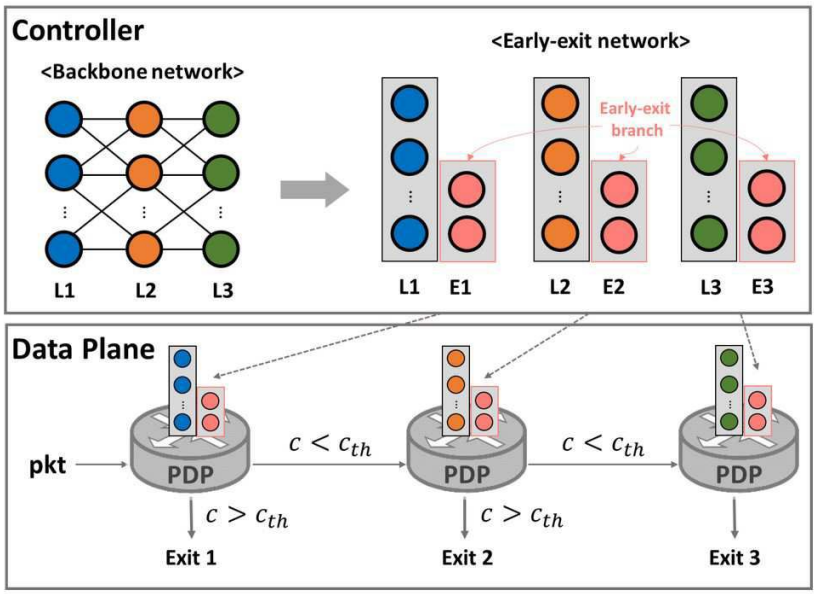
[0063] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성 요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성 요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

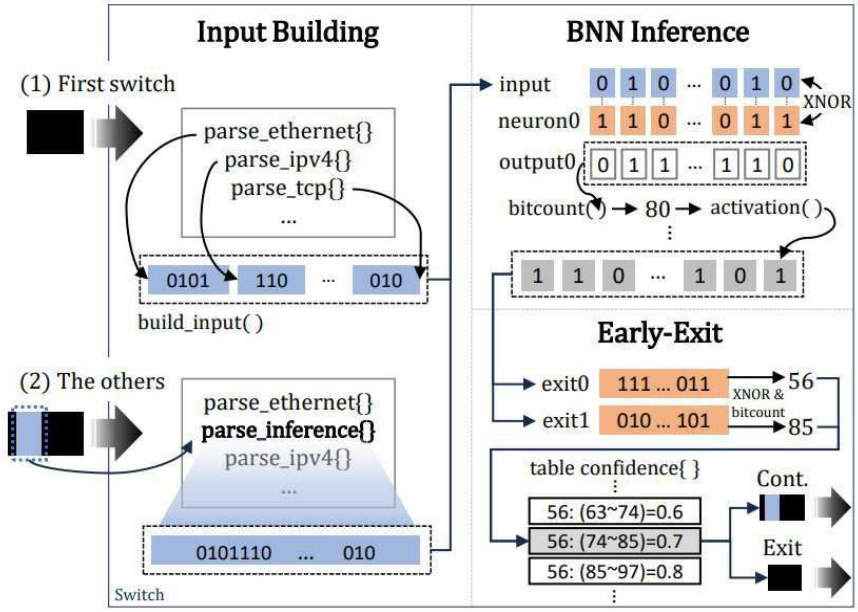
도면1


Large Model
Submodel
Submodel
Submodel
Submodel
PDP
PDP
PDP
PDP
submodel1_result
ex) accuracy 30%
submodel2_result
ex) accuracy 60%
submodel3_result
ex) accuracy 90% useful
submodel4_result
ex) accuracy 98%

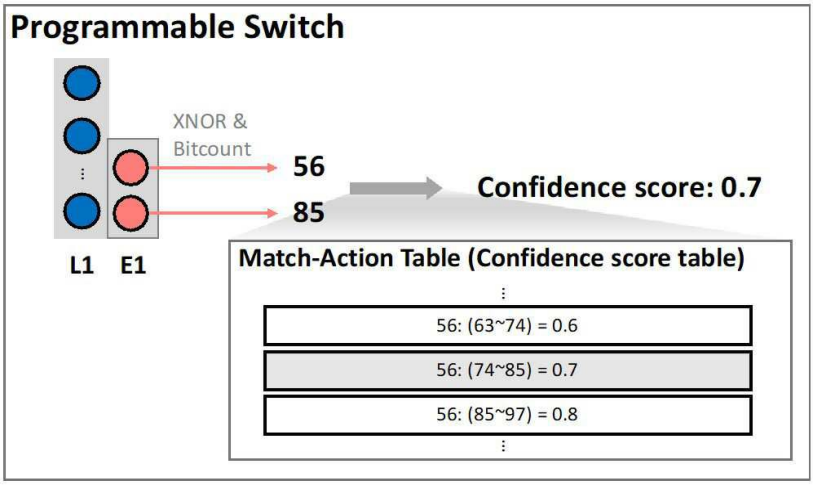
도면2a


Controller
<Backbone network>
<Early-exit network>
Early-exit branch
L1
L2
L3
L1
E1
L2
E2
L3
E3
Data Plane
pkt
PDP
PDP
PDP
$c < c_{th}$
$c < c_{th}$
$c > c_{th}$
$c > c_{th}$
Exit 1
Exit 2
Exit 3

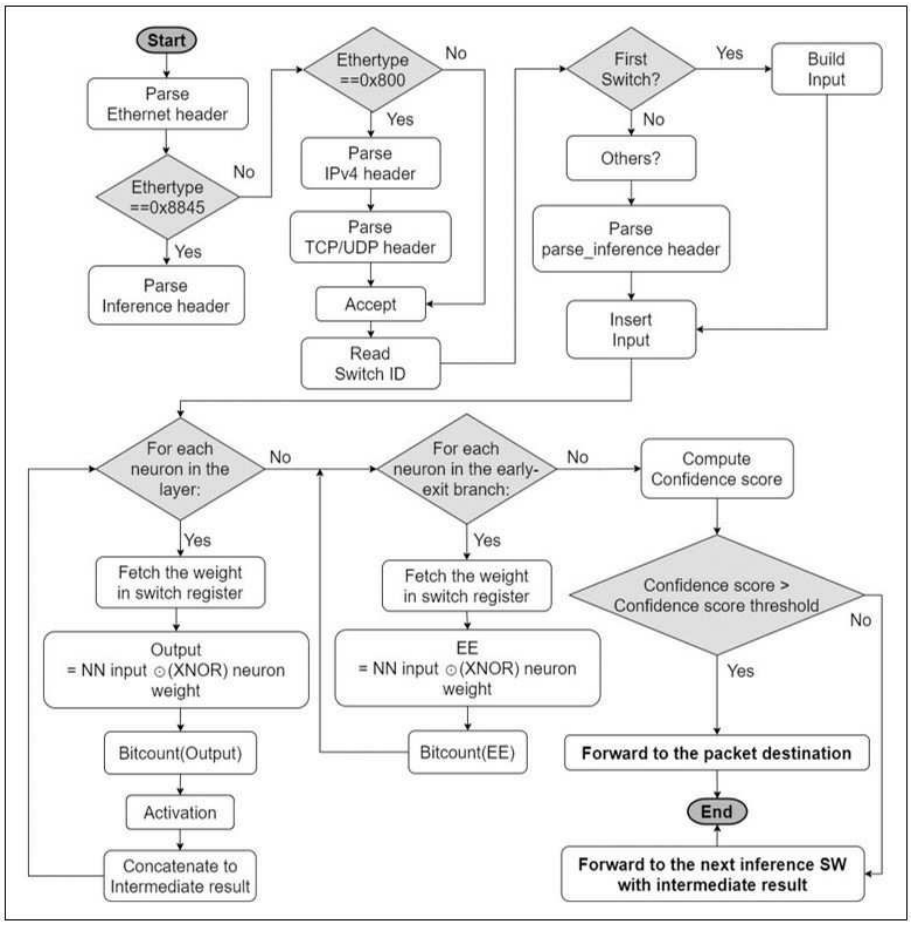
도면2b


Input Building
BNN Inference
(1) First switch
parse_ethernet{}
parse_ipv4{}
parse_tcp{}
...
0101
110
...
010
build_input( )
(2) The others
parse_ethernet{}
parse_inference{}
parse_ipv4{}
...
0101110 ... 010
Switch
input
neuron0
output0
bitcount( )
80
activation( )
XNOR
Early-Exit
exit0
111 ... 011
exit1
010 ... 101
XNOR & bitcount
56
85
table confidence{ }
56: (63~74)=0.6
56: (74~85)=0.7
56: (85~97)=0.8
Cont.
Exit

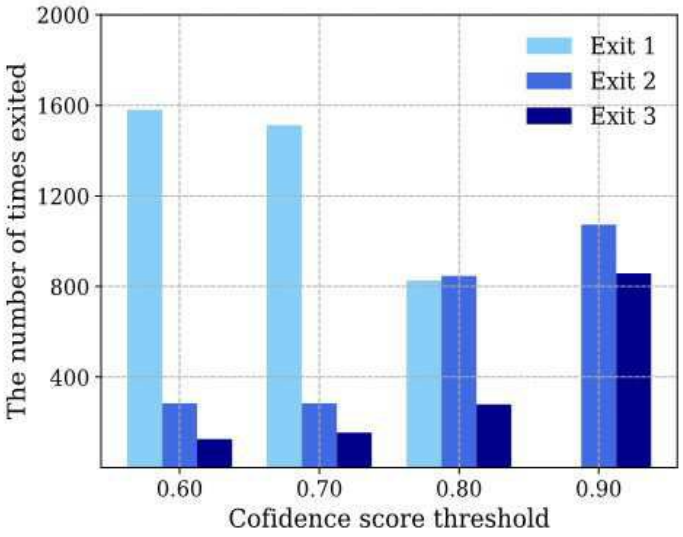
도면3


Programmable Switch
XNOR &
Bitcount
56
85
L1
E1
Confidence score: 0.7
Match-Action Table (Confidence score table)
56: (63~74) = 0.6
56: (74~85) = 0.7
56: (85~97) = 0.8

도면4


Start
Parse Ethernet header
Ethertype ==0x8845
No
Yes
Parse Inference header
Ethertype ==0x800
No
Yes
Parse IPv4 header
Parse TCP/UDP header
Accept
Read Switch ID
First Switch?
Yes
Build Input
No
Others?
Parse parse_inference header
Insert Input
For each neuron in the layer:
No
Yes
Fetch the weight in switch register
Output = NN input ⊙(XNOR) neuron weight
Bitcount(Output)
Activation
Concatenate to Intermediate result
For each neuron in the early-exit branch:
No
Yes
Fetch the weight in switch register
EE = NN input ⊙(XNOR) neuron weight
Bitcount(EE)
Compute Confidence score
Confidence score > Confidence score threshold
No
Yes
Forward to the packet destination
End
Forward to the next inference SW with intermediate result

도면5


2000
1600
1200
800
400
The number of times exited
Exit 1
Exit 2
Exit 3
0.60
0.70
0.80
0.90
Cofidence score threshold