(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2023-0167943
(43) 공개일자 2023년12월12일

(51) 국제특허분류(Int. Cl.)
G06F 3/06 (2006.01)
(52) CPC특허분류
G06F 3/0608 (2013.01)
G06F 3/0604 (2013.01)
(21) 출원번호 10-2022-0068223
(22) 출원일자 2022년06월03일
심사청구일자 2022년06월03일

(71) 출원인
창원대학교 산학협력단
경상남도 창원시 의창구 창원대학로 20(퇴촌동)
(72) 발명자
강동현
경상남도 창원시 성산구 원이대로 495, 211동 1303호(반림동, 트리비앙아파트)
권철현
울산광역시 울주군 청량읍 덕정1길 8, 102동 2403호(유림아파트)
신재하
경상남도 창원시 의창구 용지로285번길 3, 2층 201호(사림동)
(74) 대리인
양성보

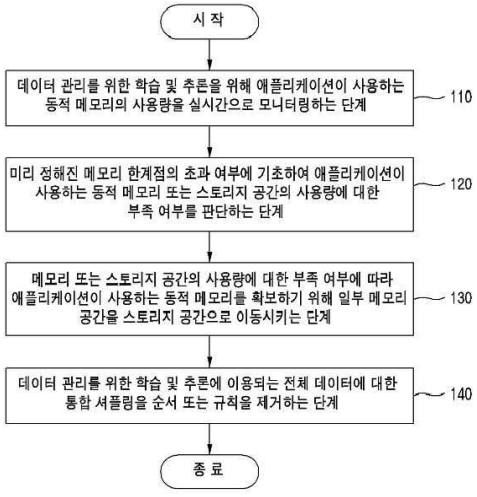
전체 청구항 수 : 총 14 항

(54) 발명의 명칭 **텐서플로우 라이트의 메모리 관리 방법 및 시스템**

(57) 요 약

텐서플로우 라이트의 메모리 관리 방법 및 시스템이 제시된다. 본 발명에서 제안하는 텐서플로우 라이트의 메모리 관리 방법은 모니터링부가 데이터 관리를 위한 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 모니터링하는 단계, 판단부가 미리 정해진 메모리 한계점(Threshold)의 초과 여부에 기초하여 상기 애플리케이션이 사용하는 동적 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부를 판단하는 단계, 메모리 관리부가 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 단계 및 통합 셔플링부가 데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 통해 순서 또는 규칙을 제거하는 단계를 포함한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 3/0626 (2013.01)
G06F 3/0653 (2013.01)
G06F 3/0658 (2013.01)
G06F 2212/1016 (2013.01)
G06F 2212/1044 (2013.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1345349237
과제번호 2021R1I1A3047006
부처명 교육부
과제관리(전문)기관명 한국연구재단
연구사업명 지역대학우수과학자지원사업
연구과제명 그린 AI 환경을 위한 딥 러닝 기반 시스템 소프트웨어 연구
기 여 율 1/1
과제수행기관명 창원대학교
연구기간 2022.03.01 ~ 2023.02.28

명 세 서

청구범위

청구항 1

모니터링부가 데이터 관리를 위한 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 모니터링하는 단계;

판단부가 미리 정해진 메모리 한계점(Threshold)의 초과 여부에 기초하여 상기 애플리케이션이 사용하는 동적 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부를 판단하는 단계;

메모리 관리부가 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 단계; 및

통합 셔플링부가 데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 통해 순서 또는 규칙을 제거하는 단계

를 포함하는 메모리 관리 방법.

청구항 2

제1항에 있어서,

상기 메모리 관리부가 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 단계는,

상기 애플리케이션이 미리 정해진 메모리 한계점(Threshold)을 초과하여 사용된다면, 메모리 공간의 용량을 차지하는 원인인 학습 과정에 사용되는 이미지 데이터를 .tflite 파일기반으로 추출된 특성과 학습 및 추론을 위한 해당 데이터의 라벨을 생성하고, 생성된 라벨을 메모리에서 파일 데이터로 작성하여 상기 추출된 특성과 함께 스토리지에 저장하는

메모리 관리 방법.

청구항 3

제2항에 있어서,

상기 메모리에서 작성되는 파일 데이터는 고정된 크기를 가짐으로써, 크기와 무관하게 항상 고정된 크기 값으로 파일 I/O을 수행하여, 상기 미리 정해진 메모리 한계점 이상의 메모리를 사용하지 않으면서 실행이 지속되는

메모리 관리 방법.

청구항 4

제2항에 있어서,

상기 스토리지에 데이터를 저장하고 읽어 들이는 오버헤드(Overhead)를 줄이기 위해 상기 생성된 라벨 및 상기 추출된 특성을 플로트(float) 배열 형식으로 저장하는

메모리 관리 방법.

청구항 5

제1항에 있어서,

상기 메모리 관리부가 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 단계는,

상기 애플리케이션이 미리 정해진 메모리 한계점(Threshold)을 초과하지 않는다면, 메모리에서 텐서플로우 라이트의 학습 및 추론을 위한 모든 과정이 수행 및 완료되는

메모리 관리 방법.

청구항 6

제1항에 있어서,

상기 통합 셔플링부가 상기 데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 통해 순서 또는 규칙을 제거하는 단계는,

상기 통합 셔플링은 학습 및 추론 과정에서 필요한 전처리(Pre-processing) 과정을 수행할 때 전체 데이터의 인덱스(Index)를 기준으로 임의로 데이터를 추출하여 셔플링을 진행하는

메모리 관리 방법.

청구항 7

제6항에 있어서,

상기 통합 셔플링을 사용하여 과적합(Overfitting) 현상을 방지하기 위해 상기 데이터의 저장 위치에 상관없이 전체 데이터의 인덱스를 기준으로 데이터에 대한 인덱스의 위치가 메모리인 경우, 메모리 I/O를 수행하고 인덱스의 위치가 스토리지인 경우, 파일 I/O를 수행하는

메모리 관리 방법.

청구항 8

데이터 관리를 위한 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 모니터링하는 모니터링부;

미리 정해진 메모리 한계점(Threshold)의 초과 여부에 기초하여 상기 애플리케이션이 사용하는 동적 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부를 판단하는 판단부;

상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 메모리 관리부; 및

데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 통해 순서 또는 규칙을 제거하는 통합 셔플링부

를 포함하는 메모리 관리 시스템.

청구항 9

제8항에 있어서,

상기 메모리 관리부는,

상기 애플리케이션이 미리 정해진 메모리 한계점(Threshold)을 초과하여 사용된다면, 메모리 공간의 용량을 차지하는 원인인 학습 과정에 사용되는 이미지 데이터를 .tflite 파일기반으로 추출된 특성과 학습 및 추론을 위한 해당 데이터의 라벨을 생성하고, 생성된 라벨을 메모리에서 파일 데이터로 작성하여 상기 추출된 특성과 함께 스토리지에 저장하는

메모리 관리 시스템.

청구항 10

제9항에 있어서,

상기 메모리 관리부는,

상기 메모리에서 작성되는 파일 데이터는 고정된 크기를 가짐으로써, 크기와 무관하게 항상 고정된 크기 값으로 파일 I/O을 수행하여, 상기 미리 정해진 메모리 한계점 이상의 메모리를 사용하지 않으면서 실행이 지속되는

메모리 관리 시스템.

청구항 11

제9항에 있어서,

상기 메모리 관리부는,

상기 스토리지에 데이터를 저장하고 읽어 들이는 오버헤드(Overhead)를 줄이기 위해 상기 생성된 라벨 및 상기 추출된 특성을 플로트(float) 배열 형식으로 저장하는

메모리 관리 시스템.

청구항 12

제8항에 있어서,

상기 메모리 관리부는,

상기 애플리케이션이 미리 정해진 메모리 한계점(Threshold)을 초과하지 않는다면, 메모리에서 텐서플로우 라이트의 학습 및 추론을 위한 모든 과정이 수행 및 완료되는

메모리 관리 시스템.

청구항 13

제8항에 있어서,

상기 통합 셔플링부는,

상기 통합 셔플링은 학습 및 추론 과정에서 필요한 전처리(Pre-processing) 과정을 수행할 때 전체 데이터의 인덱스(Index)를 기준으로 임의로 데이터를 추출하여 셔플링을 진행하는

메모리 관리 시스템.

청구항 14

제13항에 있어서,

상기 통합 셔플링부는,

상기 통합 셔플링을 사용하여 과적합(Overfitting) 현상을 방지하기 위해 상기 데이터의 저장 위치에 상관없이 전체 데이터의 인덱스를 기준으로 데이터에 대한 인덱스의 위치가 메모리인 경우, 메모리 I/O를 수행하고 인덱스의 위치가 스토리지인 경우, 파일 I/O를 수행하는

메모리 관리 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 텐서플로우 라이트의 메모리 관리 방법 및 시스템에 관한 것이다.

배 경 기 술

[0002] 최근 딥 러닝(Deep Learning) 기술을 지원하기 위한 프레임워크(Framework)가 개발 및 공개됨에 따라서 다양한 분야에서 해당 프레임워크를 적용 및 활용하기 위한 연구가 진행되고 있다[1]. 텐서플로우 라이트(TensorFlow Lite) 역시 모바일(Mobile) 환경에서 딥 러닝 기술을 쉽게 구현할 수 있도록 도와주는 프레임워크 중 하나이다. 일반적으로, 모바일 디바이스는 서버 환경보다 제한된 하드웨어(예를 들어, CPU/메모리/배터리)로 구성되기 때문에 텐서플로우 라이트는 텐서플로우 프레임워크보다 제한된 기능으로 동작한다. 예를 들어, 텐서플로우 라이트의 경우 기본적인 학습(Learning) 과정은 주로 서버 환경에서 수행한다. 그리고, 학습된 모델(Model) 결과를 기반으로 추가적인 학습 및 추론(Inference)을 모바일 디바이스에서 수행하는 전이 학습(Transfer Learning) 방법으로 딥 러닝 기술을 지원한다[2].

[0003] 그러나, 일반적인 텐서플로우 라이트 기반 모바일 애플리케이션(Application)은 추가적인 학습 및 추론 데이터

의 양이 증가하는 경우, 메모리 사용량이 증가하여 비정상적으로 종료되는 문제점을 가지고 있다. 이는, 모바일 디바이스의 운영체제인 안드로이드의 메모리 관리 정책 때문이다[3]. 즉, 안드로이드는 실행중인 애플리케이션이 제한된 메모리보다 많은 양의 메모리를 사용하는 경우, 해당 애플리케이션을 종료시킨다. 그러나, 애플리케이션의 학습 및 추론 과정에서 필요한 메모리 사용량을 예측하거나 사전에 제한하는 것은 현실적으로 어려운 일이다. 또한, 추론 데이터의 양이 많은 애플리케이션의 경우, 추론 과정을 완료하기 전 애플리케이션이 종료되는 결과를 초래한다.

[0004] 텐서플로우 라이트는 오픈소스 라이브러리인 텐서플로우(TensorFlow)의 경량화 버전이며, 제한된 하드웨어 기반의 환경(예를 들어, 모바일, IoT 등)에서 주로 사용된다[5]. 일반적으로, 딥 러닝 모델에 대한 학습 및 추론을 정상적으로 수행하기 위해서는 많은 양의 학습 및 추론 데이터셋(Dataset)과 고사양의 하드웨어 지원이 필요하다[6]. 이러한 제한된 하드웨어 기반의 모바일 환경에서는 추론 및 학습을 수행하기 위해 전이 학습(Transfer Learning) 방법을 주로 사용한다. 전이 학습은 대용량의 데이터셋을 사용하여 모델에 대한 사전 학습을 진행하고 그 결과(예를 들어, 가중치 등)에 기초하여 추가적인 학습 및 추론을 진행함으로써, 저사양의 하드웨어 환경에서도 빠르고 정확한 학습 및 추론을 지원한다. 일반적으로, 모바일 환경에서 학습 및 추론을 수행하기 위해서는 서버 기반의 텐서플로우(TensorFlow) 프레임워크에서 사전 학습을 진행하고 사전 학습 모델에 함수를 추가하여 .tflite 파일로 변환하는 과정이 필요하다[5]. 마지막으로, 모바일 애플리케이션에서 .tflite 파일에 저장된 함수를 호출함으로써, 추가적인 학습 및 추론을 수행한다.

[0005] 안드로이드 운영체제는 멀티태스킹 환경을 효율적으로 유지하기 위해 각 애플리케이션에 대한 동적 메모리(즉, 힙(heap)) 영역의 크기를 엄격하게 제한한다[3]. 만약, 애플리케이션이 제한된 힙 공간(예를 들어, 256MB)을 초과하여 메모리를 요청 및 사용하게 되면 안드로이드는 해당 애플리케이션을 강제로 종료시킴으로써, 전체 시스템 메모리에 대한 가용 공간(Free Space)을 유지한다. 이에, 제한된 힙 공간 이상의 메모리 공간이 필요한 텐서플로우 라이트 기반 애플리케이션의 경우, 정상적으로 전이 학습을 완료할 수 없는 문제가 발생한다. 물론, 이러한 애플리케이션들은 안드로이드의 largeHeap 옵션을 설정하여, 힙 공간을 확장(예를 들어, 512MB)할 수 있다. 그러나 확장된 힙 공간보다 큰 데이터셋을 대상으로 학습 및 추론을 수행하는 경우, 동일한 문제가 여전히 발생하기 때문에 영구적인 해결책이라고 보기 어렵다.

발명의 내용

해결하려는 과제

[0006] 본 발명이 이루고자 하는 기술적 과제는 동적 메모리 사용량이 많은 텐서플로우 라이트 기반의 애플리케이션이 학습 및 추론 과정을 정상적으로 수행 및 완료할 수 있도록 애플리케이션의 메모리가 부족한 경우, 메모리의 내용을 스토리지로 일부 이동시킴으로써 사용 가능한 여유 공간을 확보하는 텐서플로우 라이트의 메모리 관리 방법 및 시스템을 제공하는데 있다. 또한, 모델의 높은 정확도(Accuracy)를 보장하기 위해 학습 및 추론 과정에서 필요한 셔플링(Shuffling) 데이터를 메모리와 스토리지의 데이터로 임의 선택하는 방안을 제안한다.

과제의 해결 수단

[0007] 일 측면에 있어서, 본 발명에서 제안하는 텐서플로우 라이트의 메모리 관리 방법은 모니터링부가 데이터 관리를 위한 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 모니터링하는 단계, 판단부가 미리 정해진 메모리 한계점(Threshold)의 초과 여부에 기초하여 상기 애플리케이션이 사용하는 동적 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부를 판단하는 단계, 메모리 관리부가 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 단계 및 통합 셔플링부가 데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 통해 순서 또는 규칙을 제거하는 단계를 포함한다.

[0008] 상기 메모리 관리부가 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 단계는 상기 애플리케이션이 미리 정해진 메모리 한계점(Threshold)을 초과하여 사용된다면, 메모리 공간의 용량을 차지하는 원인인 학습 과정에 사용되는 이미지 데이터를 .tflite 파일기반으로 추출된 특성과 학습 및 추론을 위한 해당 데이터의 라벨을 생성하고, 생성된 라벨을 메모리에서 파일 데이터로 작성하여 상기 추출된 특성과 함께 스토리지에 저장한다.

[0009] 상기 메모리에서 작성되는 파일 데이터는 고정된 크기를 가짐으로써, 크기와 무관하게 항상 고정된 크기 값으로

파일 I/O을 수행하여, 상기 미리 정해진 메모리 한계점 이상의 메모리를 사용하지 않으면서 실행이 지속된다.

[0010] 상기 스토리지에 데이터를 저장하고 읽어 들이는 오버헤드(Overhead)를 줄이기 위해 상기 생성된 라벨 및 상기 추출된 특성을 플로트(float) 배열 형식으로 저장한다.

[0011] 상기 메모리 관리부가 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 단계는 상기 애플리케이션이 미리 정해진 메모리 한계점(Threshold)을 초과하지 않는다면, 메모리에서 텐서플로우 라이트의 학습 및 추론을 위한 모든 과정이 수행 및 완료된다.

[0012] 상기 통합 셔플링부가 상기 데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 통해 순서 또는 규칙을 제거하는 단계는 상기 통합 셔플링은 학습 및 추론 과정에서 필요한 전처리(Pre-processing) 과정을 수행할 때 전체 데이터의 인덱스(Index)를 기준으로 임의로 데이터를 추출하여 셔플링을 진행한다.

[0013] 상기 통합 셔플링을 사용하여 과적합(Overfitting) 현상을 방지하기 위해 상기 데이터의 저장 위치에 상관없이 전체 데이터의 인덱스를 기준으로 데이터에 대한 인덱스의 위치가 메모리인 경우, 메모리 I/O를 수행하고 인덱스의 위치가 스토리지인 경우, 파일 I/O를 수행한다.

[0014] 또 다른 일 측면에 있어서, 본 발명에서 제안하는 텐서플로우 라이트의 메모리 관리 시스템은 데이터 관리를 위한 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 모니터링하는 모니터링부, 미리 정해진 메모리 한계점(Threshold)의 초과 여부에 기초하여 상기 애플리케이션이 사용하는 동적 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부를 판단하는 판단부, 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 메모리 관리부 및 데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 통해 순서 또는 규칙을 제거하는 통합 셔플링부를 포함한다.

발명의 효과

[0015] 본 발명의 실시예들에 따른 텐서플로우 라이트의 메모리 관리 방법 및 시스템을 통해 동적 메모리 사용량이 많은 텐서플로우 라이트 기반의 애플리케이션이 학습 및 추론 과정을 정상적으로 수행 및 완료할 수 있도록 애플리케이션의 메모리가 부족한 경우, 메모리의 내용을 스토리지로 일부 이동시킴으로써 사용 가능한 여유 공간을 확보할 수 있다. 또한, 학습 및 추론 과정에서 필요한 셔플링(Shuffling) 데이터를 메모리와 스토리지의 데이터로 임의 선택함으로써 모델의 높은 정확도(Accuracy)를 보장할 수 있다.

도면의 간단한 설명

[0016] 도 1은 본 발명의 일 실시예에 따른 텐서플로우 라이트의 메모리 관리 방법을 설명하기 위한 흐름도이다.

도 2는 본 발명의 일 실시예에 따른 텐서플로우 라이트의 메모리 관리 시스템의 구성을 나타내는 도면이다.

도 3은 본 발명의 일 실시예에 따른 통합 셔플링 과정을 설명하기 위한 도면이다.

도 4는 본 발명의 일 실시예에 따른 전체 애플리케이션의 수행 시간을 비교하기 위한 그래프이다.

도 5는 본 발명의 일 실시예에 따른 학습 손실을 비교하기 위한 그래프이다.

발명을 실시하기 위한 구체적인 내용

[0017] 본 발명에서는 동적 메모리 사용량이 많은 텐서플로우 라이트 기반의 애플리케이션이 학습 및 추론 과정을 정상적으로 수행 및 완료할 수 있도록 새로운 기법을 제안한다. 제안하는 기법은 애플리케이션의 메모리가 부족한 경우, 메모리의 내용을 스토리지로 일부 이동시킴으로써, 사용 가능한 여유 공간을 확보한다. 또한, 학습 및 추론 과정에서 필요한 셔플링(Shuffling) 데이터를 메모리와 스토리지의 데이터로 임의 선택함으로써, 모델의 높은 정확도(Accuracy)를 보장한다. 이하, 본 발명의 실시 예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0019] 도 1은 본 발명의 일 실시예에 따른 텐서플로우 라이트의 메모리 관리 방법을 설명하기 위한 흐름도이다.

[0020] 대부분의 모바일 애플리케이션들은 학습 및 추론을 수행하기 위해 텐서플로우 라이트(TensorFlow Lite)를 사용

하며, 전이 학습을 통해 제한된 하드웨어(CPU, 메모리, 등)를 기반으로 추가 학습 및 추론을 진행한다. 그러나, 애플리케이션의 학습 및 추론 데이터가 증가하는 경우, 안드로이드 운영체제의 메모리 관리 정책에 의해 해당 애플리케이션이 종료되는 문제점이 존재한다. 이는 안드로이드가 애플리케이션별 동적 메모리 할당에 대한 최대 사용량을 제한하고 있기 때문이다.

[0021] 따라서, 본 발명에서는 대용량의 학습 및 추론이 필요한 모바일 애플리케이션이 종료되지 않고 정상적으로 수행될 수 있도록 새로운 기법을 제안한다. 제안 기법은 메모리와 스토리지 공간을 효율적으로 사용하는 데이터 관리 정책과 학습 및 추론을 위한 통합 셔플링 기능을 포함하고 있다.

[0022] 제안하는 텐서플로우 라이트의 메모리 관리 방법은 모니터링부가 데이터 관리를 위한 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 모니터링하는 단계(110), 판단부가 미리 정해진 메모리 한계점(Threshold)의 초과 여부에 기초하여 상기 애플리케이션이 사용하는 동적 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부를 판단하는 단계(120), 메모리 관리부가 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 단계(130) 및 통합 셔플링부가 데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 통해 순서 또는 규칙을 제거하는 단계(140)를 포함한다.

[0023] 단계(110)에서, 모니터링부가 데이터 관리를 위한 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 모니터링한다.

[0024] 제안하는 기법은 애플리케이션의 부족한 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시킴으로써, 안드로이드에 의해 애플리케이션이 종료되지 않도록 보호한다. 이를 위해, 제안 기법은 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 감시(Monitoring)한다.

[0025] 단계(120)에서, 판단부가 미리 정해진 메모리 한계점(Threshold)의 초과 여부에 기초하여 상기 애플리케이션이 사용하는 동적 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부를 판단한다.

[0026] 제안 기법은 안드로이드와 별개의 메모리 한계점(Threshold)을 지정하고 있으며, 한계점의 초과 여부를 기반으로 메모리 또는 스토리지 공간의 사용 여부를 판단한다. 제안 기법이 별도의 메모리 한계점을 사용하는 이유는 안드로이드 가상 머신의 가비지 컬렉션(Garbage Collection)을 제안 기법이 실행되는 중에 실행시키지 않기 위함이다.

[0027] 단계(130)에서, 메모리 관리부가 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시킨다.

[0028] 만약, 애플리케이션의 메모리 사용량이 제안 기법의 한계점을 초과하지 않는다면, 기존 텐서플로우 라이트의 학습 및 추론 과정과 동일하게 메모리에서 모든 과정을 수행 및 완료된다.

[0029] 반면, 애플리케이션이 제안 기법의 한계점을 초과하여 사용한다면, .tflite 파일 기반으로 학습 및 추론을 위한 라벨(Label)을 생성하고 그 결과를 메모리에서 파일 데이터로 작성한다.

[0030] 이때, 작성되는 데이터는 고정된 크기를 갖도록 설계하였다. 즉, 실제 작성되는 데이터의 크기와 무관하게 항상 고정된 크기인 MAX(예를 들어, 250,896Byte)값으로 파일 I/O을 수행한다. 그 결과, 텐서플로우 라이트 기반 애플리케이션은 제안 기법의 메모리 한계 값 이상의 메모리를 사용하지 않으며 실행이 지속될 수 있다.

[0031] 다시 말해, 제안 기법에서는 딥러닝 애플리케이션에서 힙 공간이 한계점에 도달하면 힙 공간의 용량을 차지하는 원인인 학습 과정에 사용되는 이미지 데이터를 .tflite 파일을 기반으로 추출한 특성과 해당 데이터의 라벨과 함께 스토리지에 저장하며 이때, 스토리지에 데이터를 저장하고 읽어 들이는 오버헤드(Overhead)를 줄이기 위해 플로트(float) 배열 형식으로 저장한다. 따라서 기존에 발행하는 메모리 부족(Out of Memory)의 발생 및 애플리케이션 강제 종료 없이 안정적으로 딥 러닝을 진행할 수 있다.

[0032] 단계(140)에서, 통합 셔플링부가 데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 통해 순서 또는 규칙을 제거한다.

[0033] 딥 러닝에서 데이터에 대한 특정 순서 또는 규칙이 존재할 경우, 정상적인 추론 결과를 기대하기 어렵다[7]. 이에, 딥 러닝에서 대부분의 학습 및 추론에 사용되는 전체 데이터를 셔플 과정을 통해 순서 또는 규칙을 제거하는 과정을 거친다. 그러나, 제안 기법의 경우 일부의 데이터는 메모리에 저장되고 일부의 데이터는 스토리지에 저장되기 때문에 셔플링 대상이 특정 순서 또는 규칙을 가질 수 있다. 이러한 문제를 해결하기 위해 제안 기법

에서는 통합 셔플링 기능을 수행한다.

[0034] 제안하는 통합 셔플링 과정은 학습 및 추론 과정에서 필요한 전처리(Pre-processing) 과정을 수행할 때 전체 데이터의 인덱스(Index)를 기준으로 임의로 데이터를 추출하여 셔플링을 진행한다.

[0035] 데이터의 저장 위치에 상관없이 전체 데이터의 인덱스를 기준으로 데이터에 대한 인덱스의 위치가 메모리인 경우, 메모리 I/O를 수행하고 인덱스의 위치가 스토리지인 경우, 파일 I/O를 수행하는 통합 셔플링을 사용하여 과적합(Overfitting) 현상을 방지한다.

[0036] 본 발명의 실시예에 따르면, 제안 기법은 데이터가 한계점을 초과할 경우 메모리의 일부를 스토리지에 저장하여 메모리 부족의 발생을 방지하고, 학습 시 통합 셔플링을 적용하여 모델의 과적합을 방지한다. 그 결과 딥 러닝 애플리케이션에서 사용 가능한 학습 데이터 수를 대폭 향상시킬 수 있고, 기존의 애플리케이션과 유사한 학습 성능을 보장할 수 있다.

[0038] 도 2는 본 발명의 일 실시예에 따른 텐서플로우 라이트의 메모리 관리 시스템의 구성을 나타내는 도면이다.

[0039] 제안하는 텐서플로우 라이트의 메모리 관리 시스템(200)은 모니터링부(210), 판단부(220), 메모리 관리부(230) 및 통합 셔플링부(240)를 포함한다.

[0040] 본 발명의 실시예에 따른 모니터링부(210)는 데이터 관리를 위한 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 모니터링한다.

[0041] 제안하는 기법은 애플리케이션의 부족한 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시킴으로써, 안드로이드에 의해 애플리케이션이 종료되지 않도록 보호한다. 이를 위해, 제안 기법은 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 감시(Monitoring)한다.

[0042] 본 발명의 실시예에 따른 판단부(220)는 미리 정해진 메모리 한계점(Threshold)의 초과 여부에 기초하여 상기 애플리케이션이 사용하는 동적 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부를 판단한다.

[0043] 제안 기법은 안드로이드와 별개의 메모리 한계점(Threshold)을 지정하고 있으며, 한계점의 초과 여부를 기반으로 메모리 또는 스토리지 공간의 사용 여부를 판단한다. 제안 기법이 별도의 메모리 한계점을 사용하는 이유는 안드로이드 가상 머신의 가비지 컬렉션(Garbage Collection)을 제안 기법이 실행되는 중에 실행시키지 않기 위함이다.

[0044] 본 발명의 실시예에 따른 메모리 관리부(230)는 상기 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 상기 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시킨다.

[0045] 본 발명의 실시예에 따른 메모리 관리부(230)는 애플리케이션의 메모리 사용량이 제안 기법의 한계점을 초과하지 않는다면, 기존 텐서플로우 라이트의 학습 및 추론 과정과 동일하게 메모리에서 모든 과정을 수행 및 완료된다.

[0046] 반면, 본 발명의 실시예에 따른 메모리 관리부(230)는 애플리케이션이 제안 기법의 한계점을 초과하여 사용한다면, .tflite 파일 기반으로 학습 및 추론을 위한 라벨(Label)을 생성하고 그 결과를 메모리에서 파일 데이터로 작성한다.

[0047] 이때, 작성되는 데이터는 고정된 크기를 갖도록 설계하였다. 즉, 실제 작성되는 데이터의 크기와 무관하게 항상 고정된 크기인 MAX(예를 들어, 250,896Byte)값으로 파일 I/O을 수행한다. 그 결과, 텐서플로우 라이트 기반 애플리케이션은 제안 기법의 메모리 한계 값 이상의 메모리를 사용하지 않으며 실행이 지속될 수 있다.

[0048] 다시 말해, 제안 기법에서는 딥러닝 애플리케이션에서 힙 공간이 한계점에 도달하면 힙 공간의 용량을 차지하는 원인인 학습 과정에 사용되는 이미지 데이터를 .tflite 파일을 기반으로 추출한 특성과 해당 데이터의 라벨과 함께 스토리지에 저장하며 이때, 스토리지에 데이터를 저장하고 읽어 들이는 오버헤드(Overhead)를 줄이기 위해 플로트(float) 배열 형식으로 저장한다. 따라서 기존에 발행하는 메모리 부족(Out of Memory)의 발생 및 애플리케이션 강제 종료 없이 안정적으로 딥 러닝을 진행할 수 있다.

[0049] 본 발명의 실시예에 따른 통합 셔플링부(240)는 데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 통해 순서 또는 규칙을 제거한다.

[0050] 본 발명의 실시예에 따른 통합 셔플링부(240)는 학습 및 추론 과정에서 필요한 전처리(Pre-processing) 과정을 수행할 때 전체 데이터의 인덱스(Index)를 기준으로 임의로 데이터를 추출하여 셔플링을 진행한다.

[0051] 본 발명의 실시예에 따른 통합 셔플링부(240)는 데이터의 저장 위치에 상관없이 전체 데이터의 인덱스를 기준으로 데이터에 대한 인덱스의 위치가 메모리인 경우, 메모리 I/O를 수행하고 인덱스의 위치가 스토리지인 경우, 파일 I/O를 수행하는 통합 셔플링을 사용하여 과적합(Overfitting) 현상을 방지한다.

[0052] 앞서 설명된 바와 같이, 안드로이드의 메모리 공간을 초과할 경우 애플리케이션이 강제로 종료된다. 이는 안드로이드에서 멀티태스킹을 효율적으로 관리하기 위한 방법이다. 따라서 제안하는 기법은 애플리케이션의 부족한 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시킴으로써, 안드로이드에 의해 애플리케이션이 종료되지 않도록 보호한다.

[0053] 만약, 애플리케이션의 메모리 사용량이 제안 기법의 한계점을 초과하지 않는다면, 기존 텐서플로우 라이트의 학습 및 추론 과정과 동일하게 메모리에서 모든 과정이 수행 및 완료된다.

[0054] 반면에, 애플리케이션이 제안 기법의 한계점을 초과하여 사용된다면, .tflite 파일기반으로 학습 및 추론을 위한 라벨을 생성하고 그 결과를 메모리에서 파일 데이터로 작성한다. 이때, 작성되는 데이터는 고정된 크기를 갖도록 설계하였다. 다시 말해 실제 작성되는 데이터의 크기와 무관하게 항상 고정된 크기인 MAX(예를 들어, 250,896Byte) 값으로 파일I/O을 수행한다. 그 결과, 텐서플로우 라이트 기반 애플리케이션은 제안 기법의 메모리 한계 값 이상의 메모리를 사용하지 않으며 실행이 지속될 수 있다.

[0056] 도 3은 본 발명의 일 실시예에 따른 통합 셔플링 과정을 설명하기 위한 도면이다.

[0057] 제안하는 통합 셔플링 과정은 도 3과 같이 학습 및 추론 과정에서 필요한 전처리(Pre-processing) 과정을 수행할 때 전체 데이터의 인덱스(Index)를 기준으로 임의로 데이터를 추출하여 셔플링을 진행한다. 다시 말해, 데이터에 대한 인덱스의 위치가 메모리(310)인 경우, 메모리 I/O를 수행하고 인덱스의 위치가 스토리지(320)인 경우, I/O를 수행한다. 이와 같이 저장 위치에 상관없이 임의로 데이터를 추출(331)하여 셔플링을 진행함으로써, 최초 배치(332)에 모든 클래스의 데이터가 포함되도록 한 후 학습을 수행한다(333).

[0058] 또한, 스토리지에 저장된 파일 내에서 인덱스에 대한 데이터의 위치(File Offset)는 "인덱스*MAX(고정된 파일 I/O 크기)"로 쉽게 계산하도록 설계하였다.

[0059] 본 발명에서 제안하는 기법을 평가하기 위해 텐서플로우 라이트의 예제 애플리케이션을 최신 모바일 디바이스(구글 픽셀 6 프로)에 구현하였으며, largeHeap 옵션을 설정하여 기본 애플리케이션도 최대 512MB의 힙 메모리를 사용할 수 있도록 설정하였다[3, 4, 8]. 또한, 기본 애플리케이션에 제안 기법을 추가로 구현하여 평가를 진행하였으며, 제안 기법의 메모리 한계점은 425MB으로 설정하였다. 평가를 진행하기 위해 예제 애플리케이션의 사전 학습 모델을 사용하였으며, 전이 학습의 학습 및 추론에 사용되는 사진은 직접 촬영하여 최대 3000장까지 확보하였다. 마지막으로, 시험에 대한 공정한 평가를 진행하기 위해, 시험 전 모바일 디바이스를 재부팅하였으며, 자동화된 실험 방법을 사용하였다.

[0060] 평가 결과, 제안 기법은 학습 및 추론에 필요한 데이터의 양이 증가함에도 정상적으로 동작하는 것을 확인할 수 있었다. 또한, 전체 애플리케이션의 수행 시간을 비교함으로써, 제안 기법의 오버헤드가 크지 않다는 것을 확인하였다.

[0062] 도 4는 본 발명의 일 실시예에 따른 전체 애플리케이션의 수행 시간을 비교하기 위한 그래프이다.

[0063] 도 4를 참조하면, 기본 애플리케이션과 제안 기법이 적용된 애플리케이션에 대한 실험 결과를 나타내었다. 실험은 학습 및 추론에 사용되는 사진의 수를 1500장부터 3000장까지 증가시키며 진행되었다. 도 4에서 보여주듯이, 1500장의 사진에서는 제안 기법(proposed scheme)과 기본(base) 애플리케이션의 수행 시간의 차이가 존재하지 않는다. 이는, 제안 기법이 적용된 애플리케이션이 메모리 한계점을 초과하지 않았기 때문이다. 다시 말해, 모든 학습 및 추론이 메모리에서 수행되었다. 반면에, 2000장의 학습 및 추론 데이터에서는 제안 기법이 적용된 애플리케이션의 수행 시간이 기본 애플리케이션보다 0.38초 증가되었다. 수행 시간 차이를 확인하기 위해 분석한 결과, 제안 기법이 적용된 애플리케이션의 경우, 1700장 이후 메모리 한계점을 초과하여 스토리지에 데이터를 저장되는 것을 확인할 수 있었다. 한편, 전체 학습 및 추론에 사용되는 사진이 2500장을 넘어가는 경우에는

기본 애플리케이션은 실행 중 안드로이드에 의해 종료되어 수행 시간 측정이 불가능하였다. 이는 약 2200장의 사진 데이터가 사용될 때 애플리케이션이 최대 사용 가능한 512MB의 힙 메모리 공간을 모두 사용하였기 때문이다. 반면에, 제안 기법이 적용된 애플리케이션은 종료 없이 정상적으로 모든 데이터에 대한 학습 및 추론을 완료하였다.

[0065] 도 5는 본 발명의 일 실시예에 따른 학습 손실을 비교하기 위한 그래프이다.

[0066] 제안 기법은 메모리와 스토리지 공간을 동시에 사용하고 통합 셔플링을 통해 학습 및 추론에 사용되는 데이터에 대한 순서 또는 규칙을 제거한다. 이에, 통합 셔플링에 따른 모델의 정확도가 감소하는지 평가하기 위한 기 때문에 통합 셔플링에 대한 모델 손실(Loss) 정도를 확인하기 위해 추가 실험을 진행하였다. 도 5는 사진 2000장에 대한 실험을 진행할 때 측정된 손실을 에포크(Epoch) 단위로 보여준다. 도 5에서 확인할 수 있듯이, 제안 기법(proposed scheme)이 적용된 애플리케이션의 모델 손실이 기본(base) 애플리케이션과 비슷한 결과를 보이는 것을 확인할 수 있다. 이는, 메모리와 스토리지 공간을 모두 사용하는 경우, 통합 셔플링 기능에 대한 필요성을 보여준다.

[0067] 앞서 설명된 바와 같이, 본 발명에서는 안드로이드 애플리케이션의 부족한 메모리 공간을 추가적으로 확보하기 위한 정책과 기능을 제안한다. 제안 기법의 학습 및 추론 성능을 평가하기 위해 텐서플로우 라이트의 전이 학습 예제 애플리케이션을 구현 및 수정하여 기본 애플리케이션과 비교하는 실험을 진행한 결과, 애플리케이션의 메모리 공간이 부족한 경우, 제안 기법에서는 스토리지 공간을 효율적으로 확보함으로써, 정상적인 학습 및 추론을 진행할 수 있음을 확인할 수 있었다.

[0069] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate 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(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 기술분야에서 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.

[0070] 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, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치에 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.

[0071] 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), 및 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.

[0072] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

[0073] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

[0075] <참 고 문 헌>

[0076] [1] Giorgos Demosthenous et al., "Continual Learning on the Edge with TensorFlow Lite," arXiv preprint arXiv:2105.01946, 2021.

[0077] [2] On-Device Training with TensorFlow Lite [Online]. Avaliable: https://www.tensorflow.org/lite/examples/on_device_training/overview?hl=ko (downloaded 2022, Apr. 28)

[0078] [3] Overview of memory management [Online]. Available: https://developer.android.com/topic/performance/memory-overview (downloaded 2022, Apr. 28)

[0079] [4] Google Pixel 6 Pro - Full phone specifications

[0080] https://www.gsmarena.com/google_pixel_6_pro-10918.php

[0081] [5] TensorFlow Lite Guide [Online]. Available: https://www.tensorflow.org/lite/guide?hl=ko (downloaded 2022, Apr. 28)

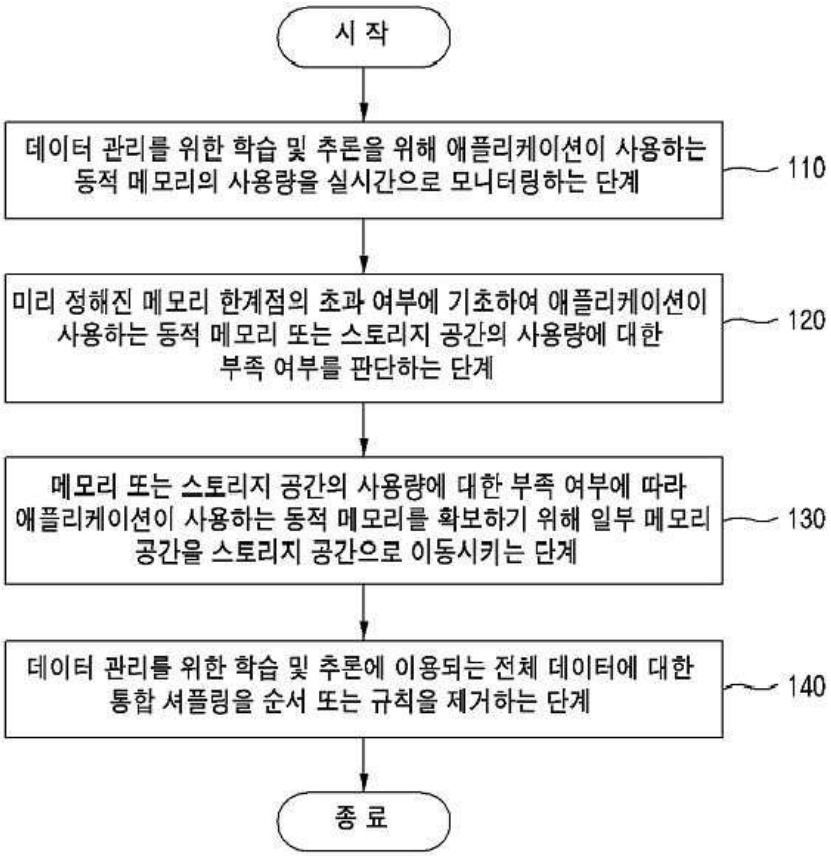
[0082] [6] Iacovos G. Kolokasis et al., "Say Goodbye to Off-heap Caches! On-heap Caches Using Memory-Mapped I/O,"12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 20), 2020.

[0083] [7] Tom Dietterich, "Overfitting and undercomputing in machine learning," ACM Computmg Survevs1, Vol. 27, No. 3, pp. 326-327, 1995.

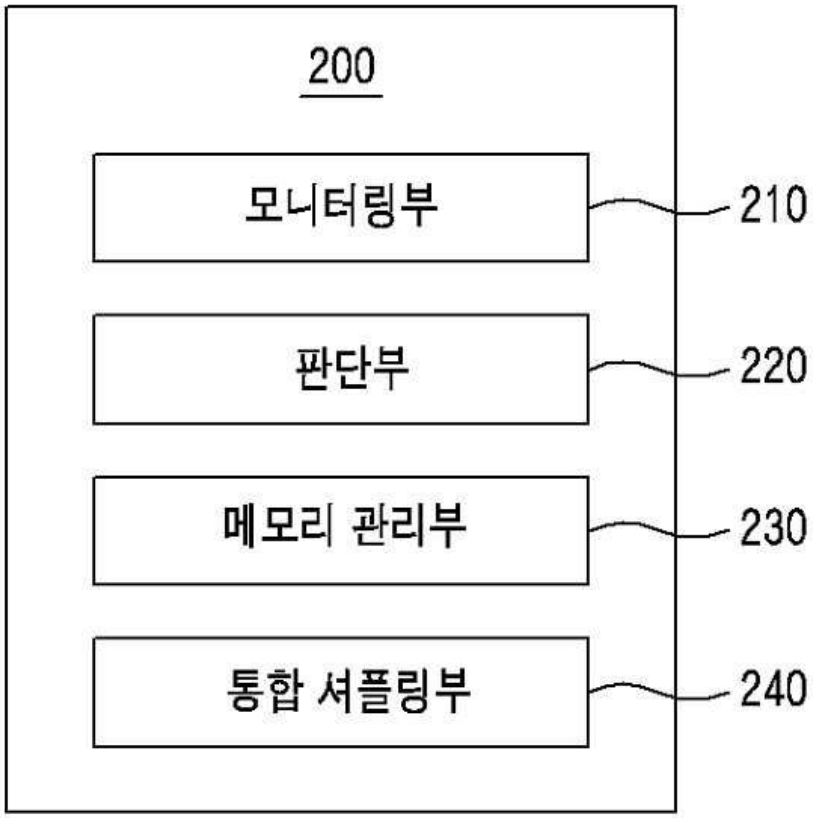
[0084] [8] TensorFlow Lite Example On-device Model Personalization [Online]. Available: https://github.com/tensorflow/examples/tree/master/lite/examples/model_personalization (downloaded 2022, Apr. 28)

도면

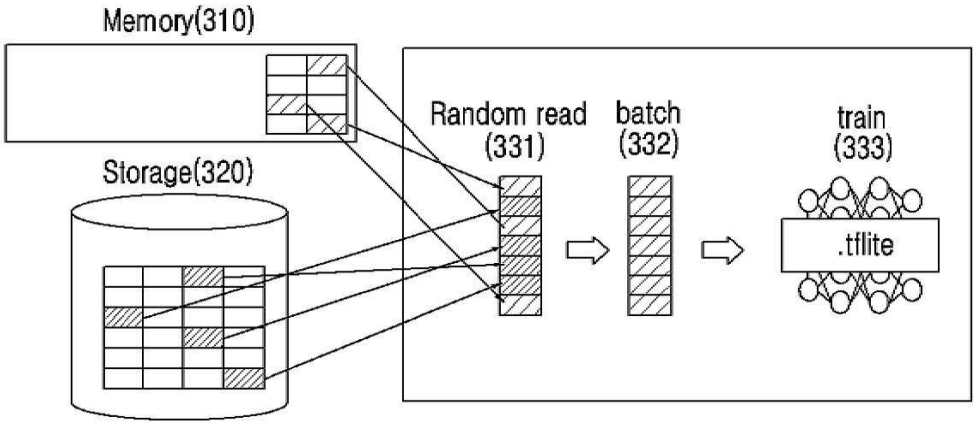
도면1


시 작
데이터 관리를 위한 학습 및 추론을 위해 애플리케이션이 사용하는 동적 메모리의 사용량을 실시간으로 모니터링하는 단계
110
미리 정해진 메모리 한계점의 초과 여부에 기초하여 애플리케이션이 사용하는 동적 메모리 또는 스토리지 공간의 사용량에 대한 부족 여부를 판단하는 단계
120
메모리 또는 스토리지 공간의 사용량에 대한 부족 여부에 따라 애플리케이션이 사용하는 동적 메모리를 확보하기 위해 일부 메모리 공간을 스토리지 공간으로 이동시키는 단계
130
데이터 관리를 위한 학습 및 추론에 이용되는 전체 데이터에 대한 통합 셔플링을 순서 또는 규칙을 제거하는 단계
140
종 료

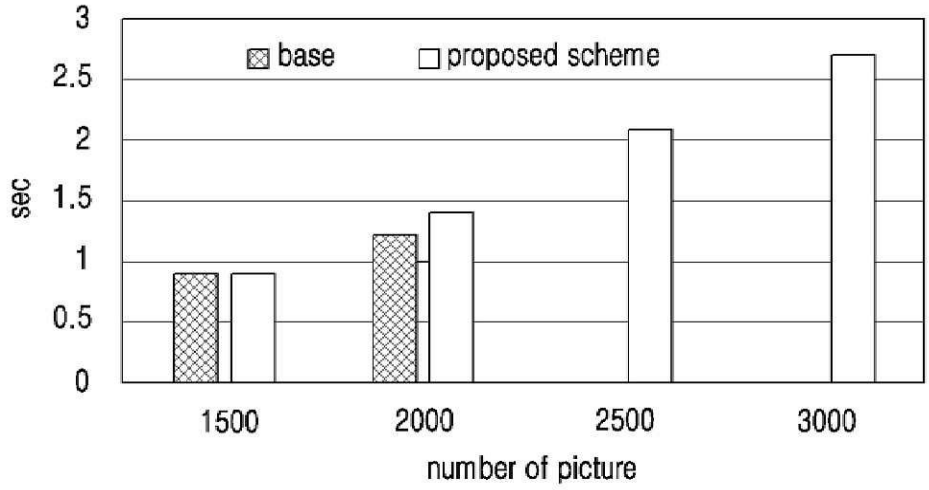
도면2


200
모니터링부
210
판단부
220
메모리 관리부
230
통합 셔플링부
240

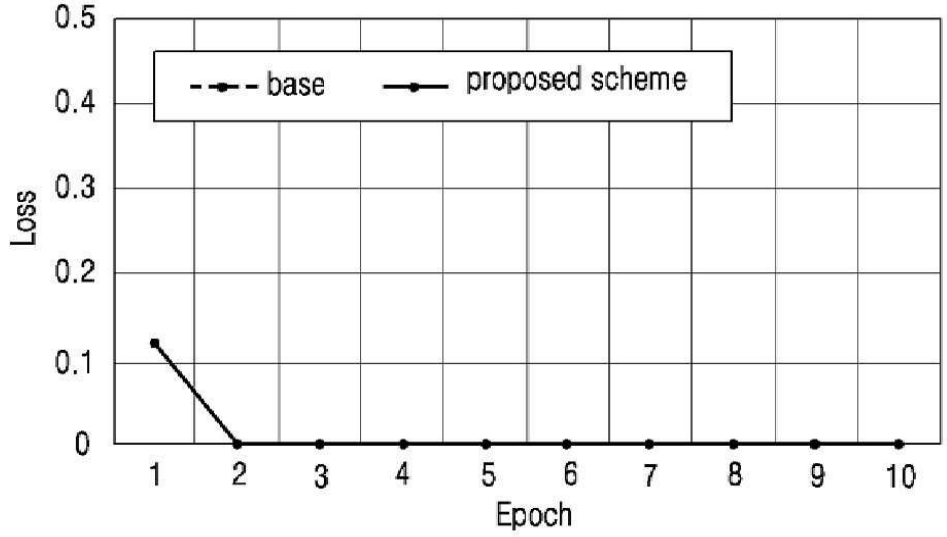
도면3


Memory(310)
Storage(320)
Random read
(331)
batch
(332)
train
(333)
.tflite

도면4


3
2.5
2
1.5
1
0.5
0
sec
base
proposed scheme
1500
2000
2500
3000
number of picture

도면5


0.5
0.4
0.3
0.2
0.1
0
Loss
base
proposed scheme
1
2
3
4
5
6
7
8
9
10
Epoch