(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2025년06월27일
(11) 등록번호 10-2826736
(24) 등록일자 2025년06월25일

(51) 국제특허분류(Int. Cl.)
G06N 99/00 (2019.01) *G06N 3/08* (2023.01)
G06V 10/40 (2022.01)
(52) CPC특허분류
G06N 20/00 (2021.08)
G06F 18/2413 (2023.01)
(21) 출원번호 10-2018-7005492
(22) 출원일자(국제) 2016년08월11일
심사청구일자 2021년07월27일
(85) 번역문제출일자 2018년02월23일
(65) 공개번호 10-2018-0044295
(43) 공개일자 2018년05월02일
(86) 국제출원번호 PCT/US2016/046576
(87) 국제공개번호 WO 2017/034820
국제공개일자 2017년03월02일
(30) 우선권주장
62/209,859 2015년08월25일 미국(US)
14/863,410 2015년09월23일 미국(US)
(56) 선행기술조사문헌
WO2005022343 A2*
US20140358831 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
탈라티 사친 수바쉬
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
바르탁 아니켓
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
(74) 대리인
특허법인코리아나

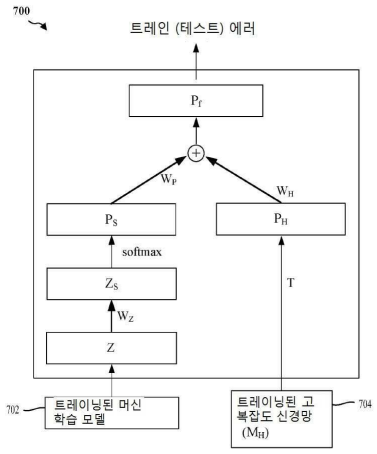
전체 청구항 수 : 총 28 항 심사관 : 박요한

(54) 발명의 명칭 **트레이닝된 머신 학습 모델의 성능을 개선시키는 방법**

(57) 요 약

트레이닝된 머신 학습 모델의 성능을 개선시키는 방법은 제 1 목적 함수를 갖는 제 1 분류자에 제 2 목적 함수를 갖는 제 2 분류자를 추가하는 단계를 포함한다. 제 1 분류자에 대한 에러들의 함수를 최소화하기 보다는, 제 2 목적 함수는 제 1 분류자의 에러들의 수를 직접 감소시키는데 이용된다.

대 표 도 - 도7



(52) CPC특허분류
G06N 3/082 (2023.01)
G06V 10/454 (2023.08)

명 세 서

청구범위

청구항 1

트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법으로서,

제 2 목적 함수를 갖는 제 2 분류자에서, 제 1 목적 함수를 갖는 제 1 분류자로부터 제 1 분류 출력을 수신하는 단계로서, 상기 제 1 분류 출력은 상기 제 1 분류자에서 수신된 입력에서 식별된 객체를 분류함으로써 생성된, 상기 제 1 분류 출력을 수신하는 단계; 및

상기 제 2 분류자에서, 상기 제 1 분류 출력을 상기 제 2 분류자의 가중치들과 컨볼루션함으로써 생성된 피처들에 따른 상기 객체를 분류하는 단계로서, 상기 제 2 분류자는 상기 제 1 분류 출력의 분류 에러들의 수를 직접 감소시키는, 상기 객체를 분류하는 단계를 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 2

제 1 항에 있어서,

상기 제 1 목적 함수는 미분가능한, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 3

제 1 항에 있어서,

상기 제 2 목적 함수는 미분가능하지 않은, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 4

제 1 항에 있어서,

상기 제 2 목적 함수는 상기 제 1 분류자와 상기 제 2 분류자의 에러들 간의 차이의 함수인, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 5

제 1 항에 있어서,

상위 복잡도 모델로부터의 확률들의 혼합에 적어도 부분적으로 기초하여 상기 제 2 목적 함수를 결정하는 단계를 더 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 6

제 1 항에 있어서,

상기 제 1 분류자를 재트레이닝함이 없이 상기 제 2 분류자를 추가하는 단계를 더 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 7

제 1 항에 있어서,

상기 제 1 분류자에 외부적으로 상기 제 2 분류자를 추가하는 단계를 더 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 8

제 1 항에 있어서,

상기 제 1 분류자에 의해 트레이닝된 모델에 의해 생성된 피처들에 대한 가중치들을 아이덴티티 값으로 배정하는 단계를 더 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 9

제 8 항에 있어서,

고 복잡도 모델의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 0 으로 배정하는 단계를 더 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 10

제 1 항에 있어서,

상기 제 2 분류자의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 배정하는 단계를 더 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 11

제 1 항에 있어서,

고 복잡도 모델의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 0 으로 배정하는 단계를 더 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 12

제 11 항에 있어서,

상기 제 2 분류자의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 배정하는 단계를 더 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 13

제 1 항에 있어서,

고정된 온도 T 에 의해 상위 복잡도 모델에 의해 생성된 확률 벡터들을 스케일링하는 단계를 더 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 컴퓨터-구현 방법.

청구항 14

트레이닝된 머신 학습 모델의 성능을 개선시키는 장치로서,

메모리; 및

상기 메모리에 커플링된 적어도 하나의 프로세서를 포함하고,

상기 적어도 하나의 프로세서는,

제 2 목적 함수를 갖는 제 2 분류자에서, 제 1 목적 함수를 갖는 제 1 분류자로부터 제 1 분류 출력을 수신하게 하는 것으로서, 상기 제 1 분류 출력은 상기 제 1 분류자에서 수신된 입력에서 식별된 객체를 분류함으로써 생성된, 상기 제 1 분류 출력을 수신하게 하고, 및

상기 제 2 분류자에서, 상기 제 1 분류 출력을 상기 제 2 분류자의 가중치들과 컨볼루션함으로써 생성된 피처들에 따른 상기 객체를 분류하게 하는 것으로서, 상기 제 2 분류자는 상기 제 1 분류 출력의 분류 에러들의 수를 직접 감소시키는, 상기 객체를 분류하게 하도록 구성되는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 15

제 14 항에 있어서,

상기 제 1 목적 함수는 미분가능한, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 16

제 14 항에 있어서,

상기 제 2 목적 함수는 미분가능하지 않은, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 17

제 14 항에 있어서,

상기 제 2 목적 함수는 상기 제 1 분류자와 상기 제 2 분류자의 에러들 간의 차이의 함수인, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 18

제 14 항에 있어서,

상기 적어도 하나의 프로세서는 또한,

상위 복잡도 모델로부터의 확률들의 혼합에 적어도 부분적으로 기초하여 상기 제 2 목적 함수를 결정하도록 구성되는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 19

제 14 항에 있어서,

상기 적어도 하나의 프로세서는 또한,

상기 제 1 분류자를 재트레이닝함이 없이 상기 제 2 분류자를 추가하도록 구성되는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 20

제 14 항에 있어서,

상기 적어도 하나의 프로세서는 또한,

상기 제 1 분류자에 외부적으로 상기 제 2 분류자를 추가하도록 구성되는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 21

제 14 항에 있어서,

상기 적어도 하나의 프로세서는 또한,

상기 제 1 분류자에 의해 트레이닝된 모델에 의해 생성된 피처들에 대한 가중치들을 아이덴티티 값으로 배정하도록 구성되는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 22

제 21 항에 있어서,

상기 적어도 하나의 프로세서는 또한,

고 복잡도 모델의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 0 으로 배정하도록 구성되는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 23

제 14 항에 있어서,

상기 적어도 하나의 프로세서는 또한,

상기 제 2 분류자의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 배정하도록 구성되는, 트레이닝된 머신

학습 모델의 성능을 개선시키는 장치.

청구항 24

제 14 항에 있어서,

상기 적어도 하나의 프로세서는 또한,

고 복잡도 모델의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 0 으로 배정하도록 구성되는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 25

제 24 항에 있어서,

상기 적어도 하나의 프로세서는 또한,

상기 제 2 분류자의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 배정하도록 구성되는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 26

제 14 항에 있어서,

상기 적어도 하나의 프로세서는 또한,

고정된 온도 T 에 의해 상위 복잡도 모델에 의해 생성된 확률 벡터들을 스케일링하도록 구성되는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 27

트레이닝된 머신 학습 모델의 성능을 개선시키는 장치로서,

제 2 목적 함수를 갖는 제 2 분류자에서, 제 1 목적 함수를 갖는 제 1 분류자로부터 제 1 분류 출력을 수신하기 위한 수단으로서, 상기 제 1 분류 출력은 상기 제 1 분류자에서 수신된 입력에서 식별된 객체를 분류함으로써 생성된, 상기 제 1 분류 출력을 수신하기 위한 수단; 및

상기 제 2 분류자에서, 상기 제 1 분류 출력을 상기 제 2 분류자의 가중치들과 컨볼루션함으로써 생성된 피처들에 따른 상기 객체를 분류하기 위한 수단으로서, 상기 제 2 분류자는 상기 제 1 분류 출력의 분류 에러들의 수를 직접 감소시키는, 상기 객체를 분류하기 위한 수단을 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 장치.

청구항 28

트레이닝된 머신 학습 모델의 성능을 개선시키기 위한 프로그램 코드를 인코딩한 비일시적 컴퓨터 판독가능 저장 매체로서,

상기 프로그램 코드는 프로세서에 의해 실행되며, 제 2 목적 함수를 갖는 제 2 분류자에서, 제 1 목적 함수를 갖는 제 1 분류자로부터 제 1 분류 출력을 수신하기 위한 프로그램 코드로서, 상기 제 1 분류 출력은 상기 제 1 분류자에서 수신된 입력에서 식별된 객체를 분류함으로써 생성된, 상기 제 1 분류 출력을 수신하기 위한 프로그램 코드; 및

상기 제 2 분류자에서, 상기 제 1 분류 출력을 상기 제 2 분류자의 가중치들과 컨볼루션함으로써 생성된 피처들에 따른 상기 객체를 분류하기 위한 프로그램 코드로서, 상기 제 2 분류자는 상기 제 1 분류 출력의 분류 에러들의 수를 직접 감소시키는, 상기 객체를 분류하기 위한 프로그램 코드를 포함하는, 트레이닝된 머신 학습 모델의 성능을 개선시키기 위한 프로그램 코드를 인코딩한 비일시적 컴퓨터 판독가능 저장 매체.

발명의 설명

기 술 분 야

[0001] 관련 출원의 상호 참조

[0002] 본 출원은 2015년 8월 25일 출원되고 발명의 명칭이 "METHOD FOR IMPROVING PERFORMANCE OF A TRAINED MACHINE LEARNING MODEL" 인 미국 가특허 출원 제62/209,859호의 이익을 주장하며, 그 전체 내용을 표현상 여기서는 참조로서 포함한다.

[0003] 기술분야

[0004] 본 개시의 특정 양태들은 일반적으로 머신 학습에 관한 것이고, 보다 구체적으로는, 트레이닝된 머신 학습 모델의 성능을 개선시키는 시스템들 및 방법들에 관한 것이다.

배 경 기 술

[0005] 머신 학습 모델, 이를 테면, 인공 신경들 (예를 들어, 뉴런 모델들) 의 상호접속된 그룹을 포함할 수도 있는 인공 신경 네트워크는 계산 디바이스이거나 또는 계산 디바이스에 의해 수행되는 방법을 표현한다.

[0006] 컨볼루션 신경 네트워크들은 피드-포워드 인공 신경 네트워크의 유형이다. 컨볼루션 신경 네트워크들은 각각이 개개의 필드를 갖고, 입력 공간을 총괄적으로 타일화하는 뉴런들의 집합들을 포함할 수도 있다. 컨볼루션 신경 네트워크들 (Convolutional neural networks; CNNs) 은 다수의 애플리케이션들을 갖는다. 특히, CNN들은 패턴 인식 및 분류의 영역에 널리 이용되어 왔다.

[0007] 딥 러닝 아키텍처, 이를 테면, 딥 빌리프 네트워크들 및 딥 컨볼루션 네트워크들은 계층화된 신경 네트워크들 아키텍처들이며, 여기에서 뉴런들의 제 1 계층의 출력은 뉴런들의 제 2 계층에 대한 입력으로 되며, 뉴런들의 제 2 계층의 출력은 뉴런들의 제 3 계층에 대한 입력으로 되며, 이하 동일하게 이루어진다. 딥 신경 네트워크들은 피처들의 계층구조를 인식하도록 트레이닝될 수도 있기 때문에, 이들은 객체 인식 애플리케이션들에 점점더 이용되고 있다. 컨볼루션 신경 네트워크들과 유사하게, 이들 딥 러닝 아키텍처에서의 계산은 프로세싱 노드들의 모집단에 걸쳐 분산될 수도 있고, 이 노드들은 하나 이상의 계산 체인들로 구성될 수도 있다. 이들 다층화된 아키텍처들은 한번에 한 층씩 트레이닝되며, 역전파를 이용하여 미세조정될 수도 있다.

[0008] 다른 모델들이 또한 객체 인식에 이용가능하다. 예를 들어, 지원 벡터 머신들 (support vector machines; SVMs) 은 분류에 적용될 수 있는 러닝 툴들이다. 지원 벡터 머신들은 데이터를 카테고리화하는 분리 하이퍼플레인 (예를 들어, 결정 바운더리) 를 포함한다. 하이퍼플레인은 지도 학습에 의해 정의된다. 원하는 하이퍼플레인은 트레이닝 데이터의 마진을 증가시킨다. 즉, 하이퍼플레인은 트레이닝 예들까지의 가장 최소의 거리를 가져야 한다.

[0009] 이들 솔루션들은 다수의 분류 벤치마크들에 대해 우수한 결과들을 가져오지만, 이들 계산 복잡도는 확률적으로 매우 높을 수 있다. 추가적으로, 모델들의 트레이닝이 도전과제가 될 수도 있다.

발명의 내용

과제의 해결 수단

[0010] 본 개시의 일 양태에서, 트레이닝된 머신 학습 모델의 성능을 개선시키는 방법이 제공된다. 본 방법은 제 1 목적 함수를 갖는 제 1 분류자에 제 2 목적 함수를 갖는 제 2 분류자를 추가하는 단계를 포함한다. 제 2 목적 함수는 제 1 분류자의 에러들을 직접 감소시키는데 이용된다.

[0011] 다른 양태에서, 트레이닝된 머신 학습 모델의 성능을 개선시키기 위한 장치가 제공된다. 장치는 메모리 및 메모리에 커플링된 적어도 하나의 프로세서를 포함한다. 프로세서(들)은 제 1 목적 함수를 갖는 제 1 분류자에 제 2 목적 함수를 갖는 제 2 분류자를 추가하도록 구성된다. 제 2 목적 함수는 제 1 분류자의 에러들을 직접 감소시키는데 이용된다.

[0012] 또 다른 양태에서, 트레이닝된 머신 학습 모델의 성능을 개선시키기 위한 장치가 제공된다. 본 장치는 제 1 목적 함수를 갖는 제 1 분류자에 제 2 목적 함수를 갖는 제 2 분류자를 추가하기 위한 수단을 포함한다. 제 2 목적 함수는 제 1 분류자의 에러들을 직접 감소시키는데 이용된다. 장치는 트레이닝된 머신 학습 모델을 통하여 수신된 입력에 기초하여 제 2 분류자로부터 피처 벡터를 출력하기 위한 수단을 더 포함한다.

[0013] 또 다른 양태에서, 비일시적 컴퓨터 판독가능 매체가 제공된다. 비일시적 컴퓨터 판독가능 매체는 그 매체상에서 트레이닝된 머신 학습 모델의 성능을 개선시키기 위한 프로그램 코드를 인코딩한다. 프로그램 코드는 프로세서에 의해 실행되며, 제 1 목적 함수를 갖는 제 1 분류자에 제 2 목적 함수를 갖는 제 2 분류자를 추가하기 위한 프로그램 코드를 포함한다. 제 2 목적 함수는 제 1 분류자의 에러들을 직접 감소시키는데 이용

된다.

[0014] 본 개시의 추가적인 특징들과 이점들은 하기에 설명될 것이다. 당업자라면, 본 개시가 본 개시의 동일한 목적을 수행하기 위한 다른 구조들을 수정하거나 설계하는 기초로서 쉽게 활용될 수도 있음이 이해되어야만 한다. 당업자라면, 이러한 등가의 구성들이 하기의 특허청구범위에서 설명되는 본 개시의 교시들을 벗어나지 않는다는 것을 알 수 있을 것이다. 동작의 구성 및 방법들 양자에 관한 본 개시의 특징으로 여겨지는 신규의 특징들은, 다른 목적들 및 이점들과 함께, 첨부된 도면과 연계한 하기의 설명으로부터 더욱 명확해질 것이다. 그러나, 각 도면은 도해 및 설명의 목적으로만 제공된 것이며 본 개시의 제한들의 정의로서 의도된 것은 아님이 명확히 이해되어져야만 한다.

도면의 간단한 설명

[0015] 본 개시의 특징들, 속성, 및 이점들은 유사한 도면 부호들이 전체에 걸쳐 대응적으로 식별하는 도면들과 연계하여 취해질 때 하기에 제시된 상세한 설명으로부터 보다 명백해질 것이다.

도 1 은 본 개시의 특정 양태들에 따라 범용 프로세서를 포함하는 시스템-온-칩 (system-on-a-chip; SOC) 을 이용한 신경 네트워크를 설계하는 예시적 구현을 예시한다.

도 2 는 본 개시의 양태들에 따른 시스템의 예시적 구현을 예시한다.

도 3a 는 본 개시의 양태들에 따른 신경 네트워크를 예시하는 다이어그램이다.

도 3b 는 본 개시의 양태들에 따른 예시적인 딥 컨볼루션 네트워크 (deep convolutional network; DCN) 를 예시하는 블록도이다.

도 4 는 본 개시의 양태들에 따라 인공 지능 (artificial intelligence; AI) 기능들을 모듈화할 수도 있는 예시적인 소프트웨어 아키텍처를 예시하는 블록도이다.

도 5 는 본 개시의 양태들에 따라 스마트폰 상에서의 AI 애플리케이션의 런타임 동작을 예시하는 블록도이다.

도 6a 및 도 6b 는 본 개시의 양태들에 따라 머신 학습 모델의 성능을 개선시키기 위해 제 1 분류자에 제 2 분류자를 추가하는 변형예들을 예시하는 블록도들이다.

도 7 은 본 개시의 양태들에 따라 트레이닝된 머신 학습 모델의 성능을 개선시키기 위한 예시적 분류자의 개략도이다.

도 8 은 본 개시의 양태들에 따라 트레이닝된 머신 학습 모델의 성능을 개선시키기 위한 방법을 예시한다.

도 9 는 본 개시의 양태들에 따라 트레이닝된 머신 학습 모델의 성능을 개선시키기 위한 방법을 예시하는 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0016] 첨부된 도면들과 연계하여 하기에 설명되는 상세한 설명은, 여러 구성들의 설명으로서 의도된 것이며 본원에서 설명되는 개념들이 실시될 수도 있는 구성들만을 나타내도록 의도된 것은 아니다. 상세한 설명은 여러 개념들의 철저한 이해를 제공하기 위한 목적으로 특정 상세들을 포함한다. 그러나, 이들 개념들이 이들 특정 상세들 없이 실시될 수도 있음이 당업자에게는 명백할 것이다. 일부 사례들에서, 이러한 개념들을 모호하게 하는 것을 방지하기 위해 공지의 구조들 및 컴포넌트들이 블록도의 형태로 도시된다.

[0017] 독립적으로 또는 본 개시물의 임의의 다른 양태들과 결합하여 구현될지 또는 독립적으로 구현될지에 따라, 본 교시들에 기초하여, 당업자들은 본 개시의 범위가 본 개시의 임의의 양태를 포괄하고자 함을 이해해야 할 것이다. 예를 들어, 전술된 임의의 수의 양태들을 이용하여 장치가 구현될 수도 있거나 방법이 실시될 수도 있다. 또한, 본 개시의 범위는 본원에 제시된 개시의 다양한 양태들에 더해 또는 양태들 이외의 다른 구조, 기능성, 또는 구조와 기능성을 이용하여 실시되는 그러한 장치 또는 방법을 포괄하고자 한다. 본원에 개시된 개시의 임의의 양태는 청구항의 하나 이상의 요소들에 의해 구체화될 수도 있다.

[0018] 단어 "예시적인" 은 "예, 사례 또는 예시의 역할을 하는 것" 을 의미하는 것으로 본원에서 이용된다. "예시적인" 것으로 본원에서 설명된 임의의 양태는 반드시 다른 양태들에 비해 바람직하거나 유리한 것으로 반드시 간주되는 것은 아니다.

[0019] 비록 특정 양태들이 본원에서 설명되지만, 이들 양태들의 많은 변형예들 및 치환예들이 본 개시의 범위 내에 속한다. 비록 바람직한 양태들의 일부 이득들 및 이점들이 언급되었지만, 본 개시의 범위는 특정 이득들, 이용들, 또는 목적들로 제한되기 위한 것이 아니다. 다만, 본 개시들의 양태들은 상이한 기술들, 시스템 구성들, 네트워크들, 및 프로토콜들에 널리 적용되고자 하며, 본 개시의 양태들 중 일부는 도면들에서 그리고 다음의 바람직한 양태들의 설명에서 예로서 예시된다. 상세한 설명 및 도면들은 제한하는 것이기 보다는 단지 본 개시의 예시일 뿐이며, 본 개시의 범위는 첨부된 청구항들 및 그의 균등물들에 의해 정의된다.

[0020] 본 개시의 양태들은 트레이닝된 하위 복잡도의 머신 학습 모델의 성능을 개선시키는 것을 개시한다. 본 개시의 양태들에 따르면, 모델 성능은 하위 복잡도 분류자의 분류자 에러들의 수를 직접 최소화하거나 또는 감소시키도록 구성된 제 2 분류자를 추가하는 것에 의해 개선될 수도 있다. 즉, 표준 기술 (예를 들어, 기울기 강하) 을 이용하여 통상의 코스트 함수 (예를 들어, 제곱 합 (sum of squares; SSE) 또는 음의 로그 우도 (negative log likelihood)) 에 의해 주어지는 바와 같이 에러들의 함수를 최소화하기 보다는, 추가된 분류자에 대한 새로운 목적 함수가 에러들의 수를 직접 최소화하거나 또는 감소시키도록 정의된다. 예를 들어, 7 개의 정확한 분류들 및 3 개의 부정확한 분류들이 있는 분류 동작들이 수행되면, 목적 함수는 3 개의 에러들을 0 으로 감소시키도록 설계될 수도 있다.

[0021] 추가적으로, 본 개시의 양태들에 따르면, 트레이닝된 하위 복잡도 머신 학습 모델의 성능은 상위 복잡도 모델의 소프트 확률을 이용하여 추가로 개선될 수도 있다.

[0022] 소프트 확률들

[0023] 소프트 확률들은 비최대 확률 값들 또는 확률 벡터의 다크 값들 (dark values) 이다. 많은 통상의 분류 시스템들에서, 확률 벡터는 클래스 라벨을 예측하는데 이용된다. 이러한 통상의 시스템들에서, 클래스 라벨은 확률 벡터에서 최고 또는 최대 확률 값을 이용하여 예측된다. 비최대 확률 값들 또는 소프트 확률들은 무시된다.

[0024] 예를 들어, 분류의 지도 머신 학습 문제를 고려하여 보며, 여기에서 머신 학습 모델 ($M_\lambda(W)$) 은 N 개의 샘플들의 입력 데이터 ($X^{tr}= [x_0,x_1,x_2, \ldots x_{N-1}]$) (여기에서 $x_i \in \mathbb{R}^D$ 임) 및 대응하는 N 개의 트레이닝된 샘플들의 C-라벨링된 출력 데이터 ($y^{tr}= [y_0,y_1,y_2, \ldots y_{N-1}]$) (여기에서 $y_i \in [0,C-1]$) 로 구성된 트레이닝 데이터를 이용하여 트레이닝된다. 통상적으로, 머신 학습 모델 (예를 들어, 신경 네트워크) 의 아키텍처를 정의하는 파라미터들 (λ) 및 모델을 트레이닝하는 학습 프로세스의 파라미터들은 미리 정해진다. 그 후, 트레이닝 데이터 $\{X^{tr}, y^{tr}\}$ 는 모델 (M_λ) 의 가중치들 (W) 을 학습하는데 이용된다. 트레이닝은 $p_j \in \mathbb{Z}_2^C$ (여기에서, $y_j = k$ 이고 $\sum_{k=0}^{C-1} p_{jk} = 1$ 이면 $p_{jk} = 1$ 이다) 이도록 $P = [p_0,p_1, \ldots p_{N-1},]$ 로의 1-K 인코딩을 이용하여 라벨링된 데이터 ($y = [y_0,y_1,y_2, \ldots y_{N-1}]$) 를 인코딩하는 것을 포함할 수도 있다.

[0025] 입력 (x) 이 주어지면, 머신 학습 모델 (M_λ) 은 하기와 같이 표현될 수도 있는 출력 확률에 대한 추정치를 생성하여:

[0026] $$\hat{p} = M_\lambda(x, W) \quad (1)$$

[0027] 하기에 의해 주어지는 멀티-클래스 엔트로피 함수를 최소화하게 한다:

[0028] $$C = \sum_{i=0}^{N-1}\sum_{j=0}^{C-1} p_{ij} log(\hat{p}_{ij}), \quad (2)$$

[0029] 출력 클래스 라벨은 하기와 같이 획득된다:

[0030] $$\hat{y} = \underset{j}{argmax}[\hat{p}] \quad (3)$$

[0031] 이로써, 하드 확률 (hard-probability) 로 지칭되는 벡터 ($\hat{p}$) 의 최대 값의 인덱스만이 추론에 이용되고, 비최

대값들은 무시된다.

[0032] 본 개시의 양태들은 분류 성능을 향상시키기 위해 소프트 확률들을 이용한다. 일부 양태들에서, 소프트 확률들은 온도 스케일링을 사용하여 추출될 수도 있다. 예를 들어, 신경 네트워크 모델에 의해 생성된 확률들 ($\hat{p}$) 은 하기와 같이 softmax 함수를 통한 추정치이다:

$$\hat{p}_k = \frac{exp\left(a_{out,k}\right)}{\sum_{j=0}^{C-1} exp\left(a_{out,j}\right)}$$

[0033] (4)

[0034] 여기에서 $a_{out} = [a_{out,0}, a_{out,1}, \ldots a_{out,C-1}]$ 는 신경 네트워크의 출력 노드로부터의 활성화 값들이다.

[0035] 트레이닝된 머신 학습 모델 (예를 들어, 신경 네트워크) 에 의해 생성된 출력 확률들은 하기와 같이 소프트 확률에 숨겨진 정보를 추출하기 위해 온도 T 로 스케일링될 수도 있다:

$$\hat{p}_k^{Te} = \frac{exp\left(\frac{a_{out,k}}{T}\right)}{\sum_{j=0}^{C-1} exp\left(\frac{a_{out,j}}{T}\right)}$$

[0036] (5)

[0037] 하나의 목적은 트레이닝된 모델에 의해 생성된 확률 벡터 ($\hat{p}$) 의 분포를 소프트화하는 것이다. 온도 T 를 통한 스케일링은 확률들의 분포를 평탄화하여 소프트 확률에서의 정보가 활용되게 할 수 있다.

[0038] 일단 추출되면, 소프트 확률들이 분류 성능을 개선시키는데 이용될 수도 있다. 예를 들어, 일 예들에서, W_m 및 b_m 이, 소프트 확률에서의 정보를 함께 풀링하는데 이용되는 가중치들의 세트 및 바이어스를 나타내는 경우에, 혼합 확률은 하기에 의해 주어질 수도 있다:

$$\tilde{p}^{Te} = \frac{1}{1 + exp\left(-(W_m \hat{p}^{Te} + b_m)\right)}$$

[0039] (6)

[0040] 혼합 확률들은 하기와 같이 트레이닝된 머신 학습 모델에 의해 출력 클래스 라벨을 예측하는데 이용될 수도 있다:

$$\tilde{y} = \underset{j}{argmax}\,[\tilde{p}^{Te}]$$

[0041] (7)

[0042] 트레이닝 데이터 ($\{X^{tr}, y^{tr}\}$) 는 소프트 확률들의 혼합을 생성하는데 이용되는 가중치들 및 바이어스들에 대한 값들을 추정하는데 이용될 수도 있다. 출력 레이블들이 하드 확률들 (e_d)(식 3) 만을 이용하여 예측될 때의 트레이닝된 머신 학습 모델에 의해 생성된 부분적 트레이닝 에러, 및 출력 레이블들이 소프트 확률들 (e)(식 7) 을 이용하여 예측될 때의 부분적 트레이닝 에러는 하기에 의해 주어진다:

$$e_d = \frac{1}{N} \sum_{j=0}^{N-1} \mathbb{I}_{\hat{y}_j \neq \hat{y}_j}$$

[0043] (8)

$$e = \frac{1}{N} \sum_{j=0}^{N-1} \mathbb{I}_{\hat{y}_j \neq \hat{y}_j}$$

[0044] (9)

[0045] 코스트 함수 (C) 는 분류 에러들을 감소시키는데 이용될 수도 있다. 즉, 코스트 함수 (C) 는 소프트 확률들의 혼합에 의해 생성된 출력 라벨들에 대한 예측 값들을 이용할 때의 트레이닝 데이터에 대한 에러들이 코스트 함수가 양의 0이 아닌 값을 취하는 확률들을 이용하는 것에 의해 획득된 에러보다 더 작게 되도록 설계될 수도 있다. 코스트 함수는 다음과 같이 표현될 수도 있다:

[0046] $$C = max\,(0, (e_d - e) / e_d) \quad (10)$$

[0047] 소프트 확률들의 혼합에 대한 개선된 또는 최적의 가중치 및 바이어스들은 다음의 최적화 문제를 푸는 것에 의해 획득될 수도 있다:

[0048] $$\{W_m^*, b_m^*\} = \underset{\{w_M, b_m\}}{argmin}\,[1 - C] \quad (11)$$

[0049] 식 11 의 최적화 문제는 초기 조건들 $\{W_m(0), b_m(0)\} = \{I, 0\}$ 을 가진 기울기 값들을 이용하지 않는 표준의 비제약된 최적화 프로세스들의 어느 것을 이용하여 풀 수도 있다. 일부 양태들에서, 최적화 기술들은 또한 소프트 확률들을 생성하기 위한 개선된 또는 최적의 온도를 결정하도록 채택될 수도 있다. 예를 들어, 식 11 의 최적화 문제는 다음과 같이 수정될 수도 있다.

[0050] $$\{T^*, W_m^*\, b_m^*\} = \underset{\{T, W_m, b_m\}}{argmin}\,[1 - C] \quad (12)$$

[0051] 표준의 비제약된 최소화 프로세스를 이용하는 것은 온도의 초기 선택에 대한 C 에 대한 국부적 최소값들인 솔루션을 가져온다. 수렴 전략은 온도의 초기 선택 주변의 국부적 최소값들을 없애는데 이용될 수도 있다. 예를 들어, 일부 양태들에서, 전략은 파라미터들의 초기 세트: $\{T(0), W_m(0), b_m(0)\}$ 로 시작하여, 식 11 을 이용하여 가중치들 및 바이어스들 ($\{W_m^{*,T(0)}, b_m^{*,T(0)}\}$) 에 대한 최적의 값들을 풀 수 있다. 초기 조건 (T'(0)) 으로부터 시작하여, 코스트 함수: C=max (0,(e-e')/e) 를 최적화하며 여기에서, e 는 $\{T'(0), W_m^{*,T(0)}, b_m^{*,T(0)}\}$ 와 함께 식 11 을 이용하여 연산되고, e' 는 $\{T'(0), W_m^{*,T(0)}, b_m^{*,T(0)}\}$ 와 함께 식 11 을 이용하여 연산된다. 시퀀스는 수렴때까지 반복될 수도 있다.

[0052] 일부 양태들에서, 앙상블 평균화는 머신 학습 모델들을 따라 및/또는 단일의 머신 학습 모델에서의 로지스틱 회귀 계층들을 따라 구현될 수도 있다. 일 예에서, 다수의 머신 학습 모델들 (M > 1) 은 M 개의 트레이닝된 모델들에 의해 생성된 출력 확률들 ($\{\hat{p}_0, \hat{p}_1 \cdots \hat{p}_{M-1}\}$) 과 함께 트레이닝된 데이터를 이용하여 트레이닝된다. 이들 모델들 각각에 대해, 소프트 확률들의 최적의 혼합은 절차 최적화 기술들 및/또는 위의 수렴 전략을 이용하여 생성될 수도 있다. 결과적인 혼합 확률들 $\{\tilde{p}_0^{Te_0}, \tilde{p}_1^{Te_1} \cdots \tilde{p}_{M-1}^{Te_{M-1}}\}$ 은 다음과 같이 출력 라벨을 예측하는데 이용될 수도 있다:

[0053] $$y^{pred} = \underset{j}{\text{argmax}} \left[\sum_k w_k \tilde{p}_k^{Te_k}\right] \quad (13)$$

[0054] $\{w_k\}$ 에 대한 하나의 선택은 (k=(1,2, ...M-1) 에 대해) w_k=1/M 이다. 대안으로서, 최적화 기술 및 위의 수렴 전략 또는 다른 유사한 기술들은 멀티-모델 확률 혼합 가중치들 ($\{w_k\}$) 의 최적화 세트를 추정하는데 이용될 수도 있다.

[0055] 다른 예에서, 다수의 로지스틱 회귀 출력 계층들을 갖추었지만 단일한 머신 학습 모델에서, 최적화 기술, 수렴 전략 등은 모델의 상이한 로지스틱 회귀 계층들로부터 유발된 소프트 확률들을 개선 또는 최적화하기 위해 이용될 수도 있다.

[0056] 일부 양태들에서, 추론들은 클래스들의 수가 클 때 (예를 들어, C ≫1) 소프트 확률을 이용하여 개선될 수도 있다. C^2 으로서 소프트 확률 스케일의 최적 혼합을 생성하기 위한 파라미터들의 수는 추론을 위한 소프트 확률들의 혼합을 추정할 때의 문제일 수 있다. 이 경우에, 유용한 정보를 포함하는 것으로 믿어지는 각각의 클래스에 대한 최고 소프트 확률들의 서브세트 (P << C) 는 분류 성능을 개선시키기 위해 활용될 수도 있다. 이어서, 식 11 은 추정될 파라미터들의 총수가 P(P+1) 이도록 가중치들 및 바이어스들을 획득하기 위해 풀 수도 있다. 추론 시간에 또는 추론 시간 주변에서, 상단 P 개의 소프트 확률들의 인덱스는 최적의 가중치들 및 바이어스들을 이용하여 추정된 혼합 확률들을 통하여 추적되고 첨부될 수도 있다.

[0057] 도 1 은 본 개시의 특정 양태들에 따른 범용 프로세서 (CPU) 또는 멀티-코어 범용 프로세서들 (CPUs) (102) 를 포함할 수도 있는, 시스템-온-칩 (SOC)(100) 을 이용하여 트레이닝된 머신 학습 모델의 성능을 개선시키는 상술한 방법의 예시적 구현을 예시한다. 변수들 (예를 들어, 모델 가중치들), 계산 디바이스와 연관된 시스템 파라미터들 (예를 들어, 가중치들을 갖는 머신 학습 모델), 지연들, 주파수 빈 정보, 및 태스크 정보는 신경 프로세싱 유닛 (NPU)(108) 과 연관된 메모리 블록에, CPU (102) 와 연관된 메모리 블록에, 그래픽 프로세싱 유닛 (graphics processing unit; GPU)(104) 과 연관된 메모리 블록에, 디지털 신호 프로세서 (digital signal processor; DSP)(106) 와 연관된 메모리 블록에, 전용 메모리 블록 (118) 에 저장될 수도 있거나 또는 다수의 블록들에 걸쳐 분산될 수도 있다. 범용 프로세서 (102) 에서 실행되는 명령들은 CPU (102) 와 연관된 프로그램 메모리로부터 로딩될 수도 있거나 전용 메모리 블록 (118) 으로부터 로딩될 수도 있다.

[0058] SOC (100) 는 또한 GPU (104), DSP (106), 제 4 세대 롱 텀 에볼루션 (4G LTE) 접속성, 비허가 Wi-Fi 접속성, USB 접속성, 블루투스 접속성 등을 포함할 수도 있는 접속성 블록 (110), 및 예를 들어, 제스처들을 검출 및 인식할 수도 있는 멀티미디어 프로세서 (112) 와 같은 특정의 기능들로 맞추어진 추가적인 프로세싱 블록들을 포함할 수도 있다. 일 구현에서, NPU 는 CPU, DSP, 및/또는 GPU 에서 구현된다. SOC (100) 는 또한 센서 프로세서 (114), 이미지 신호 프로세서들 (ISP들), 및/또는 글로벌 위치확인 시스템을 포함할 수도 있는 네비게이션 (120) 을 포함할 수도 있다.

[0059] SOC (100) 는 ARM 명령 세트에 기초할 수도 있다. 본 개시의 양태에서, 범용 프로세서 (102) 로 로딩된 명령들은 제 1 목적 함수 (예를 들어, 코스트) 를 갖는 제 1 분류자에 제 2 목적 함수 (예를 들어, 코스트) 를 갖는 제 2 분류자를 추가하기 위한 코드를 포함할 수도 있다. 제 2 목적 함수는 제 1 분류자의 에러들을 직접 감소시키는데 이용된다.

[0060] 도 2 는 본 개시의 특정 양태들에 따른 시스템 (200) 의 예시적 구현을 예시한다. 도 2 에 예시된 바와 같이, 시스템 (200) 은 본원에 설명된 방법들의 여러 동작들을 수행할 수도 있는 다수의 로컬 프로세싱 유닛들 (202) 을 가질 수도 있다. 각각의 로컬 프로세싱 유닛 (202) 은 신경 네트워크의 파라미터들을 저장할 수도 있는 로컬 상태 메모리 (204) 및 로컬 파라미터 메모리 (206) 를 포함할 수도 있다. 또한, 로컬 프로세싱 유닛 (202) 은 로컬 모델 프로그램을 저장하기 위한 로컬 (뉴런) 모델 프로그램 (local model program; LMP) 메모리 (208), 로컬 학습 프로그램을 저장하기 위한 로컬 학습 프로그램 (local learning program; LLP) 메모리 (210), 및 로컬 접속 메모리 (212) 를 가질 수도 있다. 또한, 도 2 에 예시된 바와 같이, 각각의 로컬 프로세싱 유닛 (202) 은 로컬 프로세싱 유닛의 로컬 메모리들을 위한 구성들을 제공하기 위한 구성 프로세서 유닛 (214) 과, 그리고 로컬 프로세싱 유닛들 (202) 사이에 라우팅을 제공하는 라우팅 접속 프로세싱 유닛 (216) 과 인터페이스할 수도 있다.

[0061] 딥 러닝 아키텍처들은 각각의 계층에서 추상화의 연속적으로 더 높은 레벨들에서 입력들을 표현하기 위해 학습에 의한 객체 인식 태스크를 수행하고, 이것에 의해 입력 데이터의 유용한 특징 표현을 구축할 수도 있다. 이러한 방식으로, 딥 러닝은 전통적인 머신 학습의 주요한 병목효과를 해결한다. 딥 러닝의 도래 이전에, 객체 인식 문제에 대한 머신 학습 접근법은 아마도 쉘로우 (shallow) 분류자와 결합하여, 인간 조작된 (human engineered) 피처들에 크게 의존할 수도 있다. 쉘로우 분류자는 예를 들어 2-클래스 선형 분류자일 수도 있으며, 여기서 피처 벡터 성분들의 가중된 합은 입력이 어느 클래스에 속하는지를 예측하기 위해 임계값과 비교될 수도 있다. 인간 조작된 피처들은 도메인 전문 기술을 갖는 엔지니어들에 의해 특정의 문제 도메인으로 맞추어진 템플릿들 또는 커널들일 수도 있다. 딥 러닝 아키텍처들은 대조적으로, 트레이닝을 통해서만 인간 엔지니어가 설계할지도 모르는 것과 유사한 피처들을 표현하는 것을 학습할 수도 있다. 또한, 딥 네트워크는 인간이 고려하지 못했을 수도 있는 새로운 타입들의 피처들을 표현 및 인식하는 것을 학습할 수도 있다.

[0062] 딥 러닝 아키텍처는 피처들의 계층 구조를 학습할 수도 있다. 예를 들어, 시각적 데이터가 제시되면, 제 1 계층은 입력 스트림에서 에지들과 같은 비교적 간단한 피처들을 인식하는 것을 학습할 수도 있다. 다른 예에서, 청각적 데이터가 제시되는 경우, 제 1 계층은 특정의 주파수들에서의 스펙트럼 파워를 인식하는 것을 학습할 수도 있다. 제 2 계층은, 제 1 계층의 출력을 입력으로서 취하여, 시각적 데이터에 대한 단순 형상들 또는 청각적 데이터에 대한 사운드들의 조합들과 같은 피처들의 조합들을 인식하는 것을 학습할 수도 있다. 예를 들어, 상위 계층들은 시각적 데이터에서 복잡한 형상들 또는 청각적 데이터에서 단어들을 표현하는 것을 학습할 수도 있다. 더 상위의 계층들은 공통 시각적 객체들 또는 발화된 어구들을 인식하는 것을 학습할 수도 있다.

[0063] 딥 러닝 아키텍처들은 자연의 계층적 구조를 갖는 문제들에 적용될 때 특히 잘 수행될 수도 있다. 예를 들어, 자동차들의 분류는 처음에 바퀴들, 전면 유리들, 및 다른 피처들을 인식하는 것을 학습하는 것으로부터 이익을 얻을 수도 있다. 이들 피처들은 차량들, 트럭들, 및 비행기들을 인식하기 위해 상이한 방식들로 상위 계층들에서 결합될 수도 있다.

[0064] 신경 네트워크들과 같은 머신 학습 모델들은 다양한 접속 패턴들로 설계될 수도 있다. 피드-포워드 네트워크들에서, 정보는 하위 계층으로부터 상위 계층으로 전달되며, 주어진 계층에서의 각각의 뉴런은 상위 계층들에서의 뉴런들로 통신한다. 계층 구조 표현은 상술된 바와 같이 피드-포워드 네트워크의 연속적인 계층들에서 구축될 수도 있다. 신경 네트워크들은 또한 회귀 또는 피드백 (또한 탑-다운 (top-down) 으로도 불림) 접속들을 가질 수도 있다. 회귀 접속에서, 주어진 계층에서의 뉴런으로부터의 출력은 동일한 계층에서의 다른 뉴런으로 통신될 수도 있다. 회귀 아키텍처는 순차적으로 신경 네트워크로 전달되는 입력 데이터 청크들 중 하나 보다 많은 청크에 걸쳐 있는 패턴들을 인식하는데 있어서 유용할 수도 있다. 주어진 계층에서의 뉴런으로부터 하위 계층에서의 뉴런으로의 접속은 피드백 (또는 탑-다운) 접속으로 불린다. 다수의 피드백 접속들을 갖는 네트워크는 하이 레벨 개념의 인식이 입력의 특정의 로우 레벨 피처들을 판별하는데 도움을 줄 수도 있는 경우에 유용할 수도 있다.

[0065] 도 3a 를 참조하여 보면, 신경 네트워크의 계층들 사이의 접속들은 완전히 접속되거나 (302) 또는 국부적으로 접속될 수도 있다 (304). 완전히 접속된 네트워크 (302) 에서, 제 1 계층에서의 뉴런은 제 2 계층에서의 모든 뉴런에 그 출력을 통신할 수도 있어, 제 2 계층에서의 각각의 뉴런이 제 1 계층에서의 모든 뉴런으로부터의 입력을 수신하게 한다. 대안적으로, 국부적으로 접속된 네트워크 (304) 에서는, 제 1 계층에서의 뉴런은 제 2 계층에서의 제한된 수의 뉴런들에 접속될 수도 있다. 컨볼루션 네트워크 (306) 는 국부적으로 접속될 수도 있고, 또한, 제 2 계층에서의 각각의 뉴런에 대한 입력들과 연관된 접속 강도들이 공유되도록 구성된다. 보다 일반적으로, 네트워크의 국부적으로 접속된 계층은 계층에서의 각각의 뉴런이 동일하거나 유사한 접속 패턴을 갖지만, 상이한 값들 (예를 들어, 310, 312, 314, 및 316) 을 가질 수도 있는 접속 강도들을 갖도록 구성될 수도 있다. 주어진 영역에서의 상위 계층 뉴런들이 네트워크에 대한 총 입력의 제한된 부분의 특성들로 트레이닝을 통해 튜닝되는 입력들을 수신할 수도 있기 때문에, 국부적으로 접속된 접속 패턴은 상위 계층에서 공간적으로 별개의 수용성 필드들을 발생시킬 수도 있다.

[0066] 국부적으로 접속된 신경 네트워크들은 입력들의 공간적 로케이션들이 의미있는 문제들에 매우 적합될 수도 있다. 예를 들어, 차량-탑재 카메라로부터 시각적 피처들을 인식하도록 설계된 네트워크 (300) 는 상이한 특성들을 갖는 하이 계층 뉴런들을, 이미지의 하위 대 상위 부분과의 그들의 연관에 따라 전개할 수도 있다. 예를 들어, 이미지의 하위 부분과 연관된 뉴런들은 차선 표시들을 인식하는 것을 학습할 수도 있는 반면, 이미지의 상위 부분과 연관된 뉴런들은 교통 신호등, 교통 표지판 등을 인식하는 것을 학습할 수도 있다.

[0067] DCN 은 지도 학습으로 트레이닝될 수도 있다. 트레이닝 동안, DCN 에는 속도 제한 표지판의 크롭핑된 이미지와 같은 이미지가 제시될 수도 있고, "포워드 패스 (forward pass)" 는 그 후 출력 (322) 를 생성하기 위해 컴퓨팅될 수도 있다. 출력 (322) 은 "표지판", "60", 및 "100" 과 같은 피처들에 대응하는 값들의 벡터일 수도 있다. 네트워크 설계자는 DCN 이 출력 피처 벡터, 예를 들어 트레이닝되었던 네트워크 (300) 에 대한 출력 (322) 에서 도시된 바와 같은 "표지판" 및 "60" 에 대응하는 것들에서 뉴런들의 일부에 대해 높은 스코어를 출력하기를 원할 수도 있다. 트레이닝 전에, DCN 에 의해 생성된 출력은 정확하지 않기 쉽고, 따라서 에러가 실제의 출력과 목표 출력 사이에 계산될 수도 있다. DCN 의 가중치들은 그 후 DCN 의 출력 스코어들이 목표와 더 가깝게 정렬되도록 조정될 수도 있다.

[0068] 가중치들을 조정하기 위해, 학습 알고리즘은 가중치들에 대한 기울기 벡터를 연산할 수도 있다. 기울기는 가중치가 약간 조정되었다면, 에러가 증가하거나 감소할 양을 나타낼 수도 있다. 상단 계층에서, 기울기는 페널티메이트 (penultimate) 계층에서의 활성화된 뉴런 및 출력 계층에서의 뉴런을 접속하는 가중치의 값에 직접 대응할 수도 있다. 하위 계층들에서, 기울기는 가중치들의 값에 및 상위 계층들의 연산된 에러 기울기들에 의존할 수도 있다. 가중치들은 그 후 에러를 감소시키기 위해 조정될 수도 있다. 가중치들을 조정하는 이러한 방식은 이것이 신경 네트워크를 통한 "백워드 패스" 를 수반하기 때문에 "역 전파 (back propagation)" 로 지칭될 수도 있다.

[0069] 실제로, 가중치들의 에러 기울기는 계산된 기울기가 실제의 에러 기울기에 근사화하도록 적은 수의 예들에 대해 계산될 수도 있다. 이 근사화 방법은 확률론적 기울기 하강으로 지칭될 수도 있다. 확률론적 기울기 하강은 전체 시스템의 달성가능한 에러 레이트가 감소하기를 중단할 때까지 또는 에러 레이트가 목표 레벨에 도달

할 때까지 반복될 수도 있다.

[0070] 학습 후, DCN 에는 새로운 이미지들 (326) 이 제시될 수도 있고, 네트워크를 통한 포워드 패스는 DCN 의 추론 또는 예측에 고려될 수도 있는 출력 (322) 을 발생시킬 수도 있다.

[0071] 딥 빌리프 네트워크들 (DBNs) 은 은닉 노드들의 다수의 계층들을 포함하는 확률 모델이다. DBN들은 트레이닝 데이터 세트들의 계층구조적 표현을 추출하는 데 이용될 수도 있다. DBN 은 제한된 볼츠만 머신들 (RBMs) 의 계층들을 적층하는 것에 의해 획득될 수도 있다. RBM 은 입력들의 세트에 대한 확률 분포를 학습할 수 있는 일종의 인공 신경 네트워크이다. RBM들은 각각의 입력이 카테고리화되어야 하는 클래스에 대한 정보의 부재시 확률 분포를 학습할 수 있기 때문에 RBM들은 비지도 학습에 종종 이용된다. 하이브리드 비지도 및 지도 패러다임을 사용하여 DBN 의 하부 RBM들은 비지도 방식으로 트레이닝될 수 있으며, 피처 추출기로 역할을 할 수도 있고 상단 RBM 은 지도 방식으로 (이전 계층 및 목표 클래스들로부터의 입력들의 연결 분포 상에서) 트레이닝될 수도 있고 분류자로서 역할을 할 수도 있다.

[0072] 딥 컨볼루션 네트워크들 (DCNs) 은 컨볼루션 네트워크의 네트워크들이고, 추가적인 풀링 및 정규화 계층들로 구성된다. DCN들은 많은 작업들에서 최첨단 성능을 실현시켰다. DCN들은 입력 및 출력 타겟 양쪽이 많은 예시자들로 알려져 있고 기울기 강하 방법들의 사용에 의해 네트워크의 가중치들을 수정하는데 사용되는 지도 학습을 사용하여 트레이닝될 수도 있다.

[0073] DCN들은 피드-포워드 네트워크들일 수도 있다. 또한, 전술한 바와 같이, DCN 의 제 1 층의 뉴런으로부터 다음 상위 계층의 뉴런들의 그룹으로의 연결들은 제 1 계층의 뉴런들에 걸쳐 공유된다. DCN들의 피드 포워드 및 공유 접속들은 고속 프로세싱을 위해 활용될 수 있다. DCN 의 계산 부담은 예를 들어, 회귀 또는 피드백 접속을 포함하는 유사한 사이즈의 신경 네트워크의 계산 부담보다 훨씬 적을 수 있다.

[0074] 컨벌루션 네트워크의 각각의 계층의 프로세싱은 공간적 불변 템플릿 또는 베이시스 프로젝션으로 고려될 수 있다. 입력이 컬러 이미지의 적색, 녹색 및 청색 채널과 같은 다수의 채널들로 먼저 분해되면, 그 입력에 대해 트레이닝된 컨볼루션 네트워크는 3차원인 것으로서 고려될 수도 있고, 2 개의 공간 차원은 이미지의 축을 따르고, 세번째 차원은 컬러 정보를 캡처한다. 컨볼루션 접속들의 출력들은, 후속 계층 (318 및 320) 에서의 피처 맵을 형성하는 것으로 고려될 수도 있고, 피처 맵 (예를 들어, 320) 의 각각의 엘리먼트는 이전 계층 (예를 들어, 318) 에서의 뉴런들의 범위로부터 그리고 다수의 채널들 각각으로부터 입력을 수신한다. 피처 맵에서의 값들은 정류, max(0,x) 와 같은 비선형성으로 추가 프로세싱될 수 있다. 인접 뉴런들의 값들은 다운 샘플링에 대응하는, 추가 풀링될 수 있으며 추가 로컬 불변성 및 차원 감소를 제공할 수도 있다. 화이트닝에 대응하는 정규화는 피처 맵에서 뉴런들 사이의 측면 억제를 통해 적용될 수도 있다.

[0075] 딥 러닝 아키텍처들의 성능은 더 많은 라벨링된 데이터 포인트들이 이용가능함에 따라 계산적 전력이 증가함에 따라 증가할 수 있다. 현대의 딥 신경 네트워크들은 단지 15 년전에 전형적인 연구자가 이용할 수 있었던 것보다 수천 배나 많은 계산 자원들로 루틴하게 트레이닝된다. 새로운 아키텍처들 및 트레이닝 패러다임은 딥 러닝의 성능을 한층 더 부스팅할 수도 있다. 정류된 선형 유닛들은 소멸하는 기울기로서 알려진 트레이닝 문제를 감소시킬 수도 있다. 새로운 트레이닝 기법들은 오버피팅을 감소시킬 수도 있고, 이에 따라 양호한 일반화를 실현하도록 대형 모델들을 실현할 수도 있다. 캡슐화 기술들은 주어진 수신 필드에서 데이터를 추상화할 수도 있고 전반적인 성능을 더욱 부스팅할 수 있다.

[0076] 도 3b 는 예시적인 딥 컨벌루션 네트워크 (350) 를 예시하는 블록도이다. 딥 콘볼루션 네트워크 (350) 는 접속성 및 가중치 공유에 기초하여 다수의 상이한 유형의 층들을 포함할 수 있다. 도 3b 에 도시된 바와 같이, 예시적인 딥 컨벌루션 네트워크 (350) 는 다수의 컨벌루션 블록들 (예를 들어, C1 및 C2) 을 포함한다. 각각의 컨볼루션 블록들은 컨볼루션 계층, 정규화 계층 (LNorm) 및 풀링 계층으로 구성될 수 있다. 컨볼루션 계층들은 하나 이상의 콘볼루션 필터들을 포함할 수 있으며, 이는 피처 맵을 생성하기 위해 입력 데이터에 적용될 수 있다. 비록 2 개의 컨볼루션 블록들만이 도시되어 있지만, 본 발명은 이에 제한되지 않고 임의의 수의 컨볼루션 블록들이 설계 선호도에 따라 딥 컨벌루션 네트워크 (350) 에 포함될 수 있다. 정규화 계층은 컨볼루션 필터들의 출력을 정규화하는데 이용될 수도 있다. 예를 들어, 정규화 계층은 화이트닝 또는 측면 억제를 제공할 수 있다. 풀링 계층은 로컬 불변성 및 차원 감소를 위해 공간에 대해 다운 샘플링 애그리게이션을 제공할 수 있다.

[0077] 예를 들어, 딥 컨볼루션 네트워크의 병렬 필터 뱅크들은 선택적으로 ARM 명령 세트에 기초하여 SOC (100) 의 CPU (102) 또는 GPU (104) 상에 로딩되어 고성능 및 저전력 소비를 실현할 수도 있다. 다른 실시형태에서,

병렬 필터 뱅크들은 SOC (100) 의 DSP (106) 또는 ISP (116) 상에 로딩될 수 있다. 또한, DCN 은 센서들 (114) 및 네비게이션 (120) 에 전용된 프로세싱 블록과 같이 SOC 상에 존재할 수 있는 다른 프로세싱 블록들에 액세스할 수 있다.

[0078] 또한 딥 컨볼루션 네트워크 (350) 는 하나 이상의 완전히 접속된 계층들 (예를 들어, FC1 및 FC2) 을 포함할 수 있다. 딥 컨볼루션 네트워크 (350) 는 로지스틱 회귀 (LR) 계층을 더 포함할 수 있다. 딥 컨볼루션 네트워크 (350) 의 각 계층 사이에는 업데이트될 가중치 (도시 생략) 가 있다. 각 계층의 출력은 제 1 컨볼루션 블록 (C1) 에 제공된 입력 데이터 (즉, 이미지, 오디오, 비디오, 센서 데이터 및/또는 다른 입력 데이터) 로부터 계층구조적 피처 표현들을 학습하기 위해 딥 컨볼루션 네트워크 (350) 에서 후속 계층의 입력으로서의 역할을 할 수 있다.

[0079] 도 4 는 인공 지능 (AI) 기능들을 모듈화할 수 있는 예시적인 소프트웨어 아키텍처 (400) 를 예시하는 블록도이다. 아키텍처를 사용하여, 애플리케이션 (402) 은 SOC (420)(예를 들어, CPU (422), DSP (424), GPU (426) 및/또는 NPU (428)) 의 다양한 프로세싱 블록들로 하여금 애플리케이션 (402) 의 런타임 동작 동안 계산을 지원하는 것을 수행하게 할 수도 있다.

[0080] AI 애플리케이션 (402) 은, 예를 들어, 디바이스가 현재 동작하는 로케이션을 나타내는 장면의 검출 및 인식을 제공할 수 있는 사용자 공간 (404) 에 정의된 기능을 호출하도록 구성될 수 있다. AI 애플리케이션 (402) 은 예를 들어, 인식된 장면이 사무실, 강당, 레스토랑, 또는 호수와 같은 실외 설정인지 여부에 따라 다르게 마이크 및 카메라를 구성할 수 있다. AI 애플리케이션 (402) 은 현재 장면의 추정값을 제공하기 위해 장면 검출(SceneDetect) 애플리케이션 프로그래밍 인터페이스 (API) (406) 에 정의된 라이브러리와 연관된 컴파일 된 프로그램 코드에 요청할 수 있다. 이 요청은 궁극적으로 예를 들어 비디오 및 위치 데이터에 기초하여 장면 추정을 제공하도록 구성된 딥 신경 네트워크의 출력에 의존할 수 있다.

[0081] 런타임 프레임워크 (Runtime Framework) 의 컴파일된 코드일 수 있는 런타임 엔진 (408) 은 AI 애플리케이션 (402) 에 추가로 액세스가능할 수 있다. AI 애플리케이션 (402) 은 예를 들어, 런타임 엔진으로 하여금 특정 시간 간격에서 또는 애플리케이션의 사용자 인터페이스에 의해 검출된 이벤트에 의해 트리거링된 장면 추정을 요청하게 할 수도 있다. 장면을 추정하게 할 때, 런타임 엔진은 이어서 SOC (420) 상에서 실행되는 리눅스 커널 (412) 과 같은 오퍼레이팅 시스템 (410) 에 신호를 전송할 수 있다. 이어서, 오퍼레이팅 시스템 (410) 은 CPU (422), DSP (424), GPU (426), NPU (428), 또는 이들의 조합에 대해 연산이 수행되게 할 수도 있다. CPU (422) 는 오퍼레이팅 시스템에 의해 직접 액세스될 수 있고, 다른 프로세싱 블록들은 DSP (424), GPU (426) 또는 NPU (428) 에 대한 드라이버 (414-418) 와 같은 드라이버를 통해 액세스될 수 있다. 예시적인 예에서, 딥 신경 네트워크는 CPU (422) 및 GPU (426) 와 같은 프로세싱 블록들의 조합 상에서 실행하도록 구성될 수 있거나, 존재한다면 NPU (428) 상에서 실행될 수 있다.

[0082] 도 5 는 스마트 폰 (502) 상의 AI 애플리케이션의 런타임 동작 (500) 을 나타내는 블록도이다. AI 애플리케이션은 이미지 (506) 의 포맷을 변환하고 그 다음 이미지 (508) 를 크롭 및/또는 리사이징하기 위해 (예를 들어, 자바 프로그래밍 언어를 사용하여) 구성될 수 있는 프리-프로세스 모듈 (504) 을 포함할 수 있다. 프리-프로세싱된 이미지는 시각적 입력에 기초하여 장면을 검출하고 분류하기 위해 (예를 들어, C 프로그래밍 언어를 사용하여) 구성될 수 있는 장면 검출 백엔드 엔진 (512) 을 포함하는 분류 애플리케이션 (510) 에 전달될 수 있다. 장면 검출 백엔드 엔진 (512) 은 스케일링 (516) 및 크롭핑 (518) 에 의해 이미지를 추가로 프리-프로세싱 (514) 하도록 구성될 수 있다. 예를 들어, 결과 이미지가 224 픽셀 x 224 픽셀이도록 이미지가 스케일링 및 크롭핑될 수도 있다. 이들 차원들은 신경 네트워크의 입력 차원에 맵핑될 수 있다. 신경 네트워크는 SOC (100) 의 다양한 프로세싱 블록들이 딥 신경 네트워크로 이미지 픽셀들을 추가로 프로세싱하게 하도록 딥 신경 네트워크 블록 (520) 에 의해 구성될 수도 있다. 그 후, 딥 신경 네트워크의 결과들은 임계화 (522) 되고 분류 애플리케이션 (510) 의 지수적 평활화 블록 (524) 을 통과할 수도 있다. 그 다음, 평활화된 결과들은 스마트 폰 (502) 의 설정 및/또는 디스플레이의 변경을 야기할 수 있다.

[0083] 일 구성에서, 머신 학습 모델은 제 1 목적 (예를 들어, 코스트) 함수를 갖는 제 1 분류자에 제 2 목적 (예를 들어, 코스트) 함수를 갖는 제 2 분류자를 추가하기 위하여 구성될 수도 있고, 제 2 목적 함수는 제 1 분류자의 에러들을 직접 감소시키는데 이용된다. 머신 학습 모델은 트레이닝된 머신 학습 모델을 통하여 수신된 입력에 기초하여 제 2 분류자로부터 피처 벡터를 출력하기 위하여 또한 구성된다. 머신 학습 모델은 추가 수단 및/또는 출력 수단을 포함한다. 일 양태에서, 추가 수단 및/또는 출력 수단은 범용 프로세서 (102), 범용 프로세서 (102) 와 연관된 프로그램 메모리, 메모리 블록 (118), 로컬 프로세싱 유닛 (202) 및/또는 연결된 기

능들을 수행하도록 구성된 라우팅 접속 프로세싱 유닛 (216) 일 수도 있다. 다른 구성에서, 전술한 수단은 전술한 수단에 의해 열거된 기능을 수행하도록 구성된 임의의 모듈 또는 임의의 장치일 수 있다.

[0084] 본 개시의 특정 양태들에 따르면, 각각의 로컬 프로세싱 유닛 (202) 은 네트워크의 소망하는 하나 이상의 기능성 피처들에 기초하여 네트워크의 파라미터들을 결정하고, 그리고 결정된 파라미터들이 또한 적응, 튜닝 및 업데이트될 때 소망하는 기능성 피처들 측을 향해 하나 이상의 기능성 피처들을 전개하도록 구성될 수도 있다.

[0085] 도 6a 및 도 6b 는 신경 네트워크 모델과 같은 머신 학습 모델의 성능을 개선시키기 위해 제 1 분류자에 제 2 분류자를 추가하는 변형예들을 예시하는 블록도들이다. 도 6a 및 도 6b 를 참조하면, 트레이닝된 머신 학습 모델 (606) 의 제 1 분류자 (604)에 제 2 분류자 (602) 가 추가될 수 있다. 일부 양태들에서, 머신 학습 모델 (606) 은 국부적으로 접속된 (L-C) 계층 또는 다른 머신 학습 모델을 포함하는 딥 컨볼루션 네트워크 (DCN) 를 포함할 수 있다. 머신 러니 모델은 복잡도가 낮을 수 있다. 일부 예시적인 양태들에서, 10억 미만의 승산-누적 연산들 (MACs) 을 갖는 머신 학습 모델은 저 복잡도 모델로 간주될 수 있다. 한편 10억이 넘는 승산-누적 연산을 갖는 머신 학습 모델은 고 복잡도 모델로 간주될 수 있다. 물론, 모델들의 상대적 복잡도 (예를 들어, 파라미터들의 수, 스테이지들 (계층들) 의 수, 및/또는 스테이지들의 유형) 를 결정하는데 다른 메트릭이 이용될 수도 있다.

[0086] 트레이닝된 머신 학습 모델 (606) 은 입력 (예를 들어, 이미지) (도시되지 않음) 을 수신하도록 구성될 수 있다. 머신 학습 모델 (606) 은 입력으로부터 피처 세트를 추출하기 위해 이미지를 프로세싱할 수 있다. 입력에 대응하는 피처 벡터는 제 1 분류자 (604) 에 공급될 수 있다. 제 1 분류자 (604) 는 분류 정확도를 향상시키기 위해 사용될 수 있는 미분가능 (예를 들어, 기울기가 결정가능한) 목적 함수로 구성될 수 있다. 이어서, 제 1 분류자 (604) 는 출력 클래스 라벨을 결정하는데 사용될 수 있는 확률 벡터 (Pc) 를 생성할 수도 있다.

[0087] 제 1 분류자 (604) 의 성능 및 정확도를 향상시키기 위해, 제 2 분류자 (602) 가 추가될 수 있다. 제 2 분류자 (602) 는 미분가능하지 않은 (예를 들어, 기울기가 없음) 목적 함수로 구성될 수 있다. 목적 함수는 제 1 분류자 (604) 에 의해 생성된 에러들의 수를 직접 감소시키도록 구성될 수도 있다. 즉, 제 1 분류자 (604) 에 대한 에러들의 함수 또는 코스트 함수를 최소화하려고 시도하기 보다는, 제 2 분류자 (602) 는 에러의 총 수를 감소시킨다. 예를 들어, 일부 양태들에서, 제 2 분류자 (602) 에 대한 목적 함수는 하기와 같이 표현될 수도 있다:

[0088] $$Objective\ function: argmax\ [max(0,(e_d - e)))] \quad (14)$$

[0089] 목적 함수는 전술한 바와 같이 제약되지 않은 최소화 기술을 사용하여 제 2 분류자 (602) 에 대한 가중치 및 바이어스 항을 결정하는 데 사용될 수 있다. 따라서, 제 2 분류자 (602) 로부터의 출력 클래스 라벨은 제 2 분류자 (602) 만을 통해 생성된 것보다 적은 에러를 포함할 수 있다.

[0090] 이 구성은 이전에 트레이닝된 머신 학습 모델을 재트레이닝하지 않고 분류 성능의 향상을 달성할 수 있기 때문에 특히 유용할 수 있다. 대신, 제 2 분류자 (602) 를 오직 재트레이닝하는 것에 의해 개선될 수도 있다.

[0091] 일부 양태들에서, 도 6b 에 도시된 바와 같이, 대안적으로 제 2 분류자 (602) 는 (예를 들어, 트레이닝된 머신 학습 모델로부터의 모델의 계층으로서) 트레이닝된 머신 학습 모델 (606) 내에 제공될 수 있다. 또한, 일부 양태들에서, 머신 학습 모델 (606)(도 6a 및 도 6b 에 도시됨) 의 성능은 고 복잡도 모델 (608) 을 통해 공급되는 소프트 확률을 사용하여 더욱 개선될 수 있다.

[0092] 도 7 은 본 개시의 양태들에 따라 트레이닝된 머신 학습 모델 (예를 들어, 신경 네트워크) 의 성능을 개선시키기 위한 예시적 분류자 (700) 의 개략도를 제시한다. 도 7 을 참조하면, 신경 네트워크의 분류자 (회귀) 계층의 출력에서 미분가능하지 않은 목적 함수 (O) 가 추가된다. 목적 함수는 주어진 트레이닝 (또는 테스트) 데이터세트에 대한 목적 함수에 대한 최대 0 이 아닌 값이, 트레이닝 (테스트) 에러들의 수가 원래 트레이닝된 신경 네트워크에 대해 얻어진 것 미만일 때에만 발생하도록 규정될 수 있다.

[0093] 입력 X $\in \mathbb{R}^D$ 이 주어지면, 머신 학습 모델 (702) 은 C 클래스들 중 하나로 입력을 분류하도록 구성될 수도 있다. 1-핫 인코딩과 같은 인코딩 방식을 이용하여, 클래스 라벨들은 확률 벡터들 $P \in \mathbb{Z}_2^C$ 로 표기될 수도 있어, 주어진 클래스 라벨 1< C 에 대해, $P = [p_1 p_2 \cdots p_C]^T$ (여기에서, i = 1 그리고 $\sum_{i=2}^{C} p_i = 1$ 이면

p_i=1) 이도록 한다. 트레이닝된 머신 학습 모델 (예를 들어, 신경 네트워크) $M: X \in \mathbb{R}^D \rightarrow Z \in \mathbb{R}^C$ 가 주어지면, 추정 확률 벡터 ($\hat{P}$) 는 Z 로부터 $\hat{P} = \sigma(Z)$ 와 같이 획득될 수도 있고 여기에서, σ 는 soft-max 비선형성이다.

[0094] 위에 논의된 바와 같이, 통상의 접근 방식은 $\hat{P}$ 를 사용하여 클래스 라벨을 $\hat{l} = \text{argmax}[\hat{P}]$ 로서 예측한다.

U 트레이닝 샘플들을 갖는 주어진 데이터세트에 대해, 트레이닝 에러는 이후 $e_d^{tr} = \frac{1}{U}\sum_{i=1}^{U} \mathbb{I}_{\hat{l} \neq l}$ 로서 획득되고, 그리고 V 테스트 샘플들에 대한 테스팅 에러는 유사하게 $e_d^{ts} = \frac{1}{V}\sum_{i=1}^{V} \mathbb{I}_{\hat{l} \neq l}$ 로서 획득된다. e_d^{tr} 및 e_d^{ts} 에 대한 값들은 모델 M 의 양호도 또는 정확도를 결정한다. 트레이닝된 모델 M 에 대한 하나의 양호도 또는 정확도 메트릭은, $e_d^{tr} = 0$ 그리고 $e_d^{ts} << 1$ 이다. 본 개시의 양태들은 트레이닝된 모델 M 의 성능을 개선시키기 위한 것이며, 이 모델에 대해 $e_d^{tr} \neq 0$ 이다.

[0095] 본 개시의 양태들에 따르면, 트레이닝된 모델 (702) 을 통하여 생성된 피처 표현은 분류자 (700) 에 제공될 수도 있다. 분류자 (700) 는 피처 벡터 Z 를 수신하며, 이 벡터는 모델 가중치 Wz 와 혼합되어 새로운 피처 벡터 $Z_s = W_z^T Z$ 를 생성할 수도 있다. 피처 벡터 Z_s 는 이후 확률 벡터 $P_s = \sigma(Z_s)$ 를 추정하는데 이용될 수도 있다. 확률 피처 벡터 $P_f = W_p^T P_s$ 는 이후 하기와 같이 트레이닝 세트에서의 추정된 예측 에러를 연산하는데 이용될 수도 있고: $e^{tr} = \frac{1}{U} = \sum_{i=l}^{U} \mathbb{I}_{\hat{l}_f \neq l}$, 여기에서 $\hat{l}_f = argmax[P_f]$ 이다. 파라미터들 (λ= [W_z,W_p]) 은 다음의 목적 함수를 통하여 최적화하는 것에 의해 추정된다:

[0096] $$O = MAX\left(0, (e_d^{tr} - e^{tr})\right) \quad (14)$$

[0097] 일부 양태들에서, 고 복잡도 모델 (704) 은 소프트 확률 벡터 P_H 를 머신 학습 모델 (702) 에 제공할 수도 있다. 소프트 확률 벡터는 모델 가중치 (W_h) 와 혼합될 수도 있다. 이어서, 확률 피처 벡터 $P_f = W_p^T P_s + W_h^T P_H$ 는 하기와 같이 트레이닝 세트 상의 추정된 예측 에러를 연산하는데 이용될 수도 있다: $e^{tr} = \frac{1}{U} = \sum_{i=l}^{U} \mathbb{I}_{\hat{l}_f \neq l}$, 여기에서, $\hat{l}_f = argmax[P_f]$ 이다. 파라미터들, λ= [W_z,W_p,W_h,T] 은 식 14 의 목적 함수를 통하여 최적화하는 것에 의해 추정될 수도 있다.

[0098] 0 가 미분가능하지 않은 함수라고 하면, 비제약적 최소화 프로세스가 최적 λ^* 를 하기와 같이 푸는데 이용될 수도 있다: λ^* = argmax [0]. 0 에 대한 0 이 아닌 컨버전스 값은 $e_d^{tr} < e^{tr}$ 이고 이에 따라 파라미터들의 부가 세트를 추정하는 코스트에서 오리지널 모델보다 더 양호한 성능을 갖는 결과적인 모델을 생성할 수 있음을 함축한다.

[0099] 일부 양태들에서, λ (예를 들어, W_z,W_p,W_h, 또는 T) 에서의 파라미터들의 일부는 선험적으로 설정될 수도 있다. 이로써, 수개의 새로운 파라미터들의 부가를 오버피팅하는 문제들이 완화 또는 감소될 수도 있다.

[0100] 일부 양태들에서, 여러 단순화들은 설계 선호도에 따라 성능을 개선시키면서 채택될 수도 있다. 예를 들어, 트레이닝된 학습 모델에 의해 생성된 피처들에 대응하는 가중치들은 아이덴티티 값 (identity value) 으로 설정될 수도 있다. 이로써 트레이닝된 머신 학습 모델에 의해 생성된 피처 벡터들의 혼합은 고려되지 않을 것이다. 한편, 제 2 예에서, 트레이닝된 머신 학습 모델을 통하여 생성된 피처 벡터들의 혼합만이 고려될 수도 있다.

[0101] 제 3 예에서, 트레이닝된 학습 모델에 의해 생성된 피처들에 대응하는 가중치들은 아이덴티티 값으로 설정될 수

도 있고, 고 복잡도 모델 (704) 로부터 이용가능한 소프트 확률 정보는 무시될 수도 있다.

[0102] 제 4 예에서, 고 복잡도 모델 (704) 로부터의 소프트 확률 (P_H) 은 주어진 온도 값에 의해 재스케일링될 수도 있다 (예를 들어, $T = \alpha, \alpha > 1$).

[0103] 도 8 은 트레이닝된 머신 학습 모델의 성능을 개선시키기 위한 방법 (800) 을 예시한다. 블록 802 에서, 프로세스는 제 1 목적 함수 (예를 들어, 코스트) 를 갖는 제 1 분류자에 제 2 목적 함수 (예를 들어, 코스트) 를 갖는 제 2 분류자를 추가한다. 제 2 목적 함수는 제 1 분류자의 에러들을 직접 감소시키는데 이용된다.

[0104] 제 1 목적 함수는 미분가능하고, 제 2 목적 함수는 미분가능하지 않다. 일부 양태들에서, 제 2 목적 함수는 제 1 분류자와 제 2 분류자의 에러들 간의 차이의 함수일 수도 있다. 다른 양태들에서, 제 2 목적 함수는 상위 복잡도 모델로부터의 확률들의 혼합에 기초하여 결정될 수도 있다.

[0105] 일부 양태들에서, 제 2 분류자는 제 1 분류자에 외부적으로 추가될 수도 있다. 대안으로서, 제 2 분류자는 제 1 분류자 (예를 들어, 제 1 분류자의 계층) 내에 포함될 수도 있다. 또한, 제 2 분류자는 제 1 분류자를 재트레이닝함이 없이 추가될 수도 있다.

[0106] 블록 804 에서, 프로세스는 트레이닝된 머신 학습 모델을 통하여 수신된 입력에 기초하여 제 2 분류자로부터 피처 벡터를 출력한다.

[0107] 일부 양태들에서, 프로세스는 오버피팅 문제들을 감소 또는 완화하기 위해 여러 단순화 기법을 구현할 수도 있다. 예를 들어, 프로세스는 아이덴티티 값에, 제 1 분류자에 의해 트레이닝되는 모델에 의해 생성된 피처들에 대한 가중치들을 배정할 수도 있다. 프로세스는 또한 고 복잡도 모델의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 0 으로 배정할 수도 있다. 프로세스는 또한 제 2 분류자의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 배정할 수도 있다. 프로세스는 또한 고 복잡도 모델의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 0 으로 배정할 수도 있다. 프로세스는 또한 제 2 분류자의 확률 벡터에 의해 생성된 피처들에 대한 가중치들을 배정할 수도 있다. 프로세스는 고정된 온도 T 에 의해 상위 복잡도 모델에 의해 생성된 확률 벡터들을 스케일링할 수도 있다.

[0108] 도 9 는 본 개시의 양태들에 따라 트레이닝된 머신 학습 모델의 성능을 개선시키기 위한 방법 (900) 을 예시하는 블록도이다. 블록 902 에서, 프로세스는 트레이닝된 머신 학습 모델을 통하여 머신 학습 벡터들을 머신 학습 모델 (예를 들어, 분류자) 에서 수신한다. 확률 벡터들은 트레이닝된 머신 학습 모델에서 수신된 입력들에 대응한다. 블록 904 에서, 머신 학습 모델의 파라미터들, 이를 테면, 모델 가중치들 및 바이어스들은 트레이닝된 머신 학습 모델의 에러들을 직접 감소시키는 목적 함수에 기초하여 연산될 수도 있다. 즉, 목적 함수는 트레이닝된 머신 학습 모델에 대한 에러들의 함수보다는 에러들의 수를 직접 감소시키도록 설계된다. 이로써, 머신 학습 모델의 목적 함수는 미분가능하지 않다.

[0109] 일부 양태들에서, 트레이닝된 머신 학습 모델 및/또는 고 복잡도 모델로부터의 소프트 확률들이 파라미터들을 연산하는데 이용될 수도 있다.

[0110] 블록 906 에서, 프로세스는 머신 학습 모델의 파라미터들을 업데이트할 수도 있다. 그 후, 블록 908 에서, 머신 학습 모델은 수신된 확률 벡터들에 대응하는 입력들에 대한 클래스 라벨들을 생성할 수도 있다. 이로써, 업데이팅에 후속하는 분류 에러들은 동일 입력들에 대한 트레이닝된 머신 학습 모델에 의해 생성된 에러들 미만일 수도 있다. 따라서, 트레이닝된 머신 학습 모델의 성능이 개선될 수도 있다.

[0111] 위에서 설명된 방법들의 다양한 동작들은 대응하는 기능들을 수행할 수 있는 임의의 적절한 수단으로 수행될 수도 있다. 수단은 주문형 집적 회로 (ASIC), 또는 프로세서를 포함하여 다양한 하드웨어 및/또는 소프트웨어 컴포넌트(들) 및/또는 모듈(들)을 포함하나, 이로 제한되지는 않는다. 일반적으로, 도면들에 대응하는 동작들이 있는 경우, 이러한 동작들은 대응하는 상대 수단 + 동일한 번호를 갖는 기능 컴포넌트들을 가질 수도 있다.

[0112] 본원에서 이용되는 바와 같이, 용어 "결정하는" 은 매우 다양한 액션들을 포괄한다. 예를 들어, "결정하는" 은 계산하는, 컴퓨팅, 프로세싱, 도출하는, 조사하는, 검색하는(예를 들어, 테이블, 데이터베이스, 또는 다른 데이터 구조에서 검색하는), 확인하는 등을 포함할 수도 있다. 또한, "결정하는"은 수신하는 (예를 들면, 정보를 수신하는), 액세스하는 (메모리의 데이터에 액세스하는) 등을 포함할 수 있다. 또한, "결정하는" 은 해결하는, 선택하는, 고르는, 확립하는 등을 포함할 수 있다.

[0113] 본원에서 이용되는 바와 같이, 아이템들의 리스트 중 "그 중 적어도 하나" 를 지칭하는 구절은 단일 멤버들을 포함한, 이들 아이템들의 임의의 조합을 지칭한다. 일 예로서, "a, b, 또는 c: 중의 적어도 하나" 는 a, b, c, a-b, a-c, b-c, 및 a-b-c 를 포함하고자 한다.

[0114] 본원 개시와 연계하여 설명된 여러 예시적인 논리 블록들, 모듈들, 및 회로들은 본원에서 개시된 기능들을 수행하도록 디자인된 범용 프로세서, 디지털 신호 프로세서 (DSP), 주문형 반도체 (ASIC), 필드 프로그래밍가능 게이트 어레이 (FPGA) 또는 다른 프로그래밍가능 로직 디바이스, 이산 게이트 또는 트랜지스터 로직, 이산 하드웨어 컴포넌트들, 또는 이들의 임의의 조합에 의해 구현되거나 수행될 수도 있다. 범용 프로세서는 마이크로프로세서일 수도 있으나, 대안으로서, 프로세서는 임의의 상업적으로 이용가능한 프로세서, 제어기, 마이크로컨트롤러, 또는 상태 머신일 수도 있다. 프로세서는 또한 컴퓨팅 디바이스들의 조합, 예를 들면, DSP와 마이크로프로세서의 조합, 복수의 마이크로프로세서들, DSP 코어와 연계한 하나 이상의 마이크로프로세서들, 또는 임의의 다른 그러한 구성으로 구현될 수도 있다.

[0115] 본 개시와 연계하여 설명된 방법의 단계들 또는 알고리즘은 하드웨어에서, 프로세서에 의해 실행되는 소프트웨어 모듈에서, 또는 이들 양쪽의 조합에서 직접 구현될 수도 있다. 소프트웨어 모듈은 공지된 임의의 형태의 저장 매체 내에 있을 수도 있다. 이용될 수도 저장 매체들의 일부 예들은, 랜덤 액세스 메모리 (RAM), 판독 전용 메모리 (ROM), 플래시 메모리, EPROM (erasable programmable read-only memory) 메모리, EEPROM (electrically erasable programmable read-only memory) 메모리, 레지스터들, 하드 디스크, 탈착가능 디스크, CD-ROM 등을 포함한다. 소프트웨어 모듈을 단일 명령 또는 다수의 명령들을 포함할 수도 있고, 수개의 상이한 코드 세그먼트들 상에서, 상이한 프로그램들 간에 그리고 다수의 저장 매체에 걸쳐 분포될 수도 있다. 저장 매체는 프로세서에 커플링되어, 프로세서가 저장 매체로부터 정보를 판독하거나 저장 매체에 정보를 기록할 수 있게 한다. 대안에서, 저장 매체는 프로세서에 통합될 수도 있다.

[0116] 본원에서 개시된 방법들은 상술된 방법을 실현하기 위한 하나 이상의 단계들 또는 액션들을 포함한다. 방법 단계들 및/또는 액션들은 청구항들의 범위를 벗어나지 않으면서 서로 상호 교환될 수도 있다. 즉, 단계들 또는 액션들에 대한 특정 순서가 달리 명시되지 않는 한, 특정 단계들 및/또는 액션들의 순서 및/또는 이용은 청구항들의 범위로부터 벗어남이 없이 수정될 수도 있다.

[0117] 설명된 기능들은 하드웨어, 소프트웨어, 펌웨어, 또는 이들의 임의의 조합으로 구현될 수도 있다. 하드웨어로 구현되면, 예시적인 하드웨어 구성은 디바이스에서의 프로세싱 시스템을 포함할 수도 있다. 프로세싱 시스템은 버스 아키텍처로 구현될 수도 있다. 버스는 프로세싱 시스템의 특정 애플리케이션 및 전체 설계 제약들에 의존하여 임의의 수의 상호접속 버스들 및 브리지들을 포함할 수도 있다. 버스는 프로세서, 머신 판독가능 매체 및 버스 인터페이스를 포함한 여러 회로들을 함께 링크할 수도 있다. 버스 인터페이스는 다른 무엇보다도, 네트워크 어댑터를 버스를 통하여 프로세싱 시스템에 접속하는데 이용될 수도 있다. 네트워크 어댑터는 신호 프로세싱 기능들을 구현하는데 이용될 수도 있다. 특정 양태들에서, 사용자 인터페이스 (예를 들어, 키패드, 디스플레이, 마우스, 조이스틱 등) 가 또한 버스에 접속될 수도 있다. 버스는 또한 각종 다른 회로들, 예컨대, 타이밍 소스들, 주변기기들, 전압 레귤레이터들, 및 전력 관리 회로들 등을 링크할 수도 있으며, 이는 공지되어 있으므로, 더 이상 설명되지 않을 것이다.

[0118] 프로세서는 머신 판독가능 매체 상에 저장된 소프트웨어의 실행을 포함하는 범용 프로세싱 및 버스를 관리하는 것을 담당할 수도 있다. 프로세서는 하나 이상의 범용 및/또는 특수 목적 프로세서들로 구현될 수도 있다. 예들은 마이크로프로세서들, 마이크로컨트롤러들, DSP 프로세서들, 및 소프트웨어를 실행할 수 있는 다른 회로부를 포함한다. 소프트웨어는, 소프트웨어, 펌웨어, 미들웨어, 마이크로코드, 하드웨어 기술 언어 또는 그 외의 것으로 지칭되던지 간에 명령들, 데이터 또는 이들의 임의의 조합을 의미하도록 널리 간주되어야 한다.

머신 판독가능 매체는 예를 들어, RAM (Random Access Memory), 플래시 메모리, ROM (Read Only Memory), PROM (Programmable Read-Only Memory), EPROM (Erasable Programmable Read-Only Memory), EEPROM (Electrically Erasable Programmable Read-Only Memory), 레지스터들, 자기 디스크들, 광 디스크들, 하드 드라이브들, 또는 임의의 다른 적절한 저장 매체, 또는 이들의 임의의 조합을 포함할 수도 있다. 머신 판독가능 매체는 컴퓨터 프로그램 제품에서 구현될 수도 있다. 컴퓨터 프로그램 제품은 패키징 재료들을 포함할 수도 있다.

[0119] 하드웨어 구현에서, 머신 판독가능 매체는 프로세서와 분리된 프로세싱 시스템의 부분일 수도 있다. 그러나, 당해 기술 분야의 당업자가 쉽게 이해할 바와 같이, 머신 판독가능 매체, 또는 이들의 임의의 부분은 프로세싱 시스템 외부에 있을 수도 있다. 예를 들어, 머신 판독가능 매체는 송신 라인, 데이터에 의해 변조

된 반송파, 및/또는 디바이스로부터 분리된 컴퓨터 제품을 포함할 수도 있고, 이들 모두는 버스 인터페이스를 통하여 프로세서에 의해 액세스될 수도 있다. 대안으로서, 또는 추가적으로, 머신 판독가능 매체 또는 이들의 임의의 부분은 프로세서 내에 통합될 수도 있고, 이러한 경우는 캐시 및/또는 일반 레지스터 파일들과 함께 있을 수도 있다. 설명된 여러 컴포넌트들이 특정 로케이션, 이를 테면, 로컬 컴포넌트를 갖는 것으로서 설명될 수도 있지만, 이들은 또한 여러 방식들로 이를 테면, 특정 컴포넌트들이 분산된 컴퓨터 시스템의 부분으로서 구성되는 것으로서 구성될 수도 있다.

[0120] 프로세싱 시스템은 프로세서 기능을 제공하는 하나 이상의 마이크로프로세서들, 및 머신 판독가능 매체의 적어도 일부분을 제공하는 외부 메모리를 갖춘 범용 프로세싱 시스템으로서 구성될 수도 있고, 모두는 외부 버스 아키텍처를 통하여 다른 지원 회로부와 함께 링크된다. 대안으로서, 프로세싱 시스템은 본원에 설명된 뉴런 모델들 및 신경 시스템의 모델들을 구현하기 위한 하나 이상의 뉴로모픽 프로세서들을 포함할 수도 있다. 다른 대안으로서, 프로세싱 시스템은 프로세서, 버스 인터페이스, 사용자 인터페이스, 지원 회로부 및 단일 칩 내에 통합된 머신 판독가능 매체의 적어도 일부분를 구비한 ASIC (Application Specific Integrated Circuit) 에 의해 또는 하나 이상의 FPGA들 (Field Programmable Gate Arrays), PLD들 (Programmable Logic Devices), 제어기들, 상태 머신들, 게이트 로직, 이산 하드웨어 컴포넌트들 또는 임의의 다른 적절한 회로부 또는 본 개시물 전반에 걸쳐 설명된 여러 기능을 수행할 수 있는 회로들의 임의의 조합에 의해 구현될 수도 있다. 당해 기술 분야의 당업자는 전체 시스템 상에 부여되는 전체 설계 제약들 및 특정 애플리케이션에 따라 그 설명된 기능성을 최상으로 구현하는 방법을 알고 있을 것이다.

[0121] 머신 판독가능 매체는 복수의 소프트웨어 모듈들을 포함할 수도 있다. 소프트웨어 모듈들은 프로세서에 의해 실행될 때 프로세싱 시스템으로 하여금 각종 기능들을 수행하게 하는 명령들을 포함한다. 소프트웨어 모듈들은 송신 모듈 및 수신 모듈을 포함할 수도 있다. 각각의 소프트웨어 모듈은 단일 저장 디바이스에 있을 수도 있거나 또는 다수의 저장 디바이스들에 걸쳐 분산될 수도 있다. 예를 들어, 소프트웨어 모듈은 트리거링 이벤트를 발생할 때 하드 드라이브로부터 RAM 으로 로딩될 수도 있다. 소프트웨어 모듈의 실행 동안, 프로세서는 액세스 속도를 증가시키기 위해 캐시 내에 명령들의 일부를 로딩할 수도 있다. 그 후, 하나 이상의 캐시 라인들은 프로세서에 의한 실행을 위해 일반 레지스터 파일 내에 로딩될 수도 있다. 아래의 소프트웨어 모듈의 기능을 언급할 때, 이러한 기능은 소프트웨어 모듈로부터의 명령들을 실행할 때 프로세서에 의해 구현되는 것임을 알 것이다. 또한, 본 개시의 양태들은 프로세서, 컴퓨터, 머신 또는 이러한 양태들을 구현하는 다른 시스템의 기능에 대한 개선들을 가져온다는 것을 알아야 한다.

[0122] 소프트웨어로 구현되면, 상기 기능들은 하나 이상의 명령들 또는 코드로서 컴퓨터 판독가능한 매체 상에 저장되거나 또는 전송될 수도 있다. 컴퓨터 판독가능 매체들은 한 장소에서 다른 장소로 컴퓨터 프로그램의 전송을 가능하게 하는 임의의 매체를 포함한 통신 매체들 및 컴퓨터 저장 매체들 양쪽을 포함한다. 저장 매체는 컴퓨터에 의해 액세스될 수 있는 임의의 이용 가능한 매체일 수도 있다. 비제한적인 예로서, 이러한 컴퓨터 판독 가능한 매체는 RAM, ROM, EEPROM, CD-ROM 또는 다른 광학 디스크 스토리지, 자기 디스크 스토리지 또는 다른 자기 스토리지 디바이스들, 또는 요구되는 프로그램 코드를 명령들 또는 데이터 구조들의 형태로 이송 또는 저장하기 위해 사용될 수 있으며 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체를 포함할 수 있다. 또한, 임의의 접속은 컴퓨터 판독 가능한 매체라고 적절히 지칭된다. 예를 들어, 소프트웨어가 동축 케이블, 광섬유 케이블, 연선, 디지털 가입자 회선 (DSL), 또는 적외선 (IR), 무선, 및 마이크로파와 같은 무선 기술들을 사용하여 웹사이트, 서버, 또는 다른 원격 소스로부터 전송되면, 동축 케이블, 광섬유 케이블, 연선, DSL, 또는 적외선, 무선, 및 마이크로파와 같은 무선 기술들은 매체의 정의 내에 포함된다. 본원에서 이용된 바와 같이, 디스크 (disk) 와 디스크 (disc) 는, 컴팩트 디스크 (CD), 레이저 디스크, 광학 디스크, 디지털 다기능 디스크 (DVD), 플로피디스크 및 Blu-ray® 디스크를 포함하며, 여기서 디스크 (disk) 는 통상 자기적으로 데이터를 재생하고, 디스크 (disc) 는 레이저를 이용하여 광학적으로 데이터를 재생한다. 따라서, 일부 양태들에서 컴퓨터 판독가능 매체들은 비일시적 컴퓨터 판독가능 매체들 (예를 들어, 유형의 매체들) 을 포함할 수도 있다. 또한, 다른 양태들에서 컴퓨터 판독가능 매체들은 일시적 컴퓨터 판독가능 매체들 (예를 들어, 신호) 을 포함할 수도 있다. 위의 조합들도 컴퓨터 판독가능 매체들의 범위 내에 포함되어야 한다.

[0123] 따라서, 특정 양태들은 본원에 제시된 동작들을 수행하는 컴퓨터 프로그램 제품을 포함할 수도 있다. 예를 들어, 이러한 컴퓨터 프로그램 제품은 저장된 (및/또는 인코딩된) 명령들을 갖는 컴퓨터 판독가능 매체를 포함할 수도 있으며, 명령들은 본원에 설명된 동작들을 수행하기 위해 하나 이상의 프로세서들에 의해 실행가능할 수도 있다. 특정 양태들에서, 컴퓨터 프로그램 제품은 패키징 재료를 포함할 수도 있다.

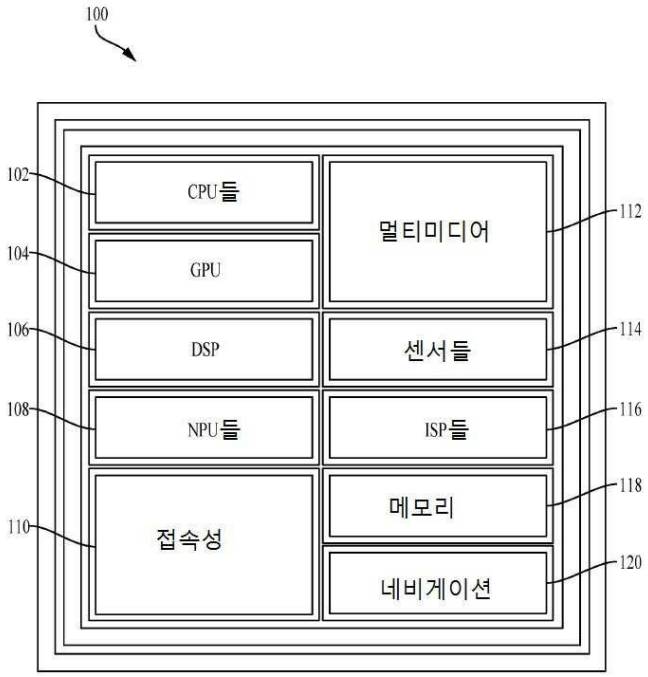
[0124] 또한, 본원에 설명된 방법들 및 기법들을 수행하는 모듈들 및/또는 다른 적절한 수단은 다운로드될 수도 있고/

있거나, 그렇지 않으면 가능한 적용가능한 사용자 단말 및/또는 기지국에 의해 획득될 수도 있다. 예를 들어, 본원에서 설명된 방법들을 수행하기 위한 수단의 전송을 용이하게 하기 위한 서버에 디바이스가 커플링될 수도 있다. 대안으로, 본원에 설명된 다양한 방법들이 저장 수단 (예를 들어, RAM, ROM, 물리적 컴팩트 디스크 (CD) 나 플로피 디스크와 같은 물리적 저장 매체 등) 을 통해 제공될 수도 있어, 사용자 단말 및/또는 기지국은 디바이스에 커플링할 시에 또는 디바이스에 저장 수단을 제공할 시에 다양한 방법들을 획득할 수 있다. 또한, 본원에서 설명된 상기 방법들 및 기술들을 디바이스에 제공하기 위한 임의의 다른 적절한 기술들이 활용될 수 있다.

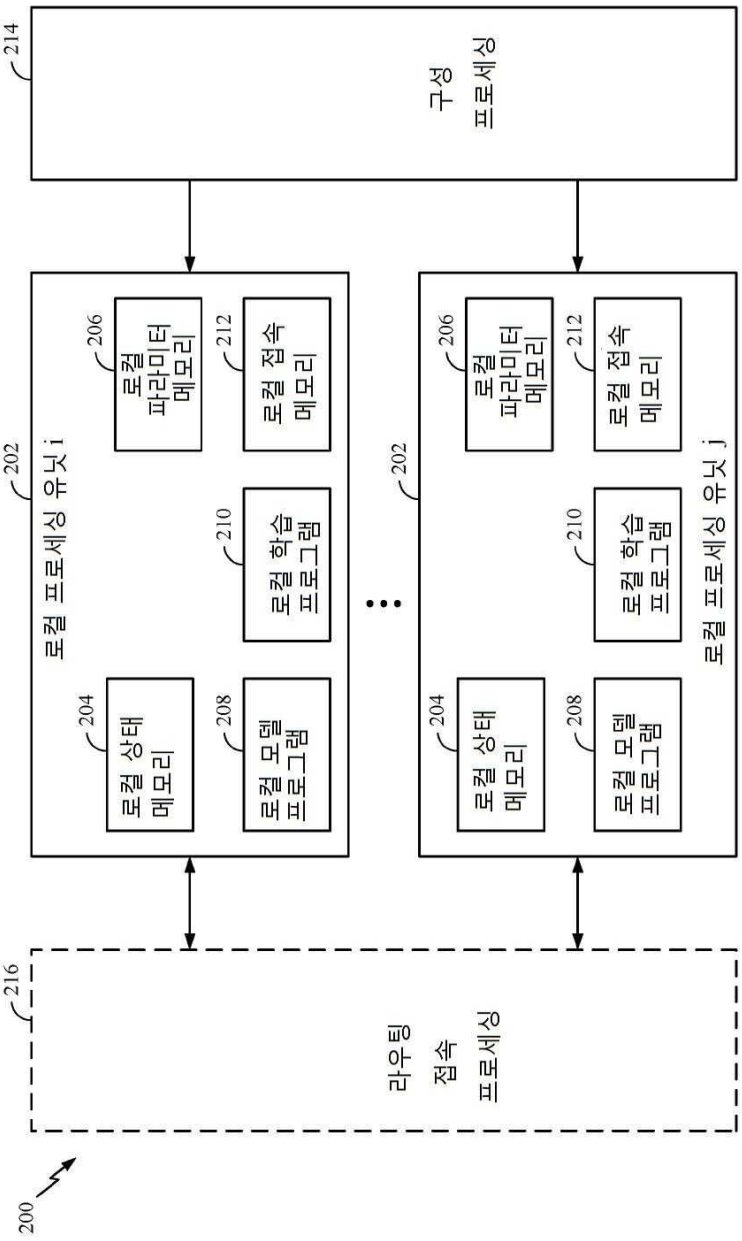
[0125] 하기의 특허청구범위는 상기 설명된 정확한 구성 및 컴포넌트들로 제한되는 것이 아님을 이해해야 한다. 특허청구범위의 범위를 벗어나지 않으면서, 본원에서 설명된 시스템들, 방법들 및 장치들의 배치, 동작 및 상세들에서 다양한 수정예들, 변경예들 및 변형예들이 행해질 수도 있다.

도면

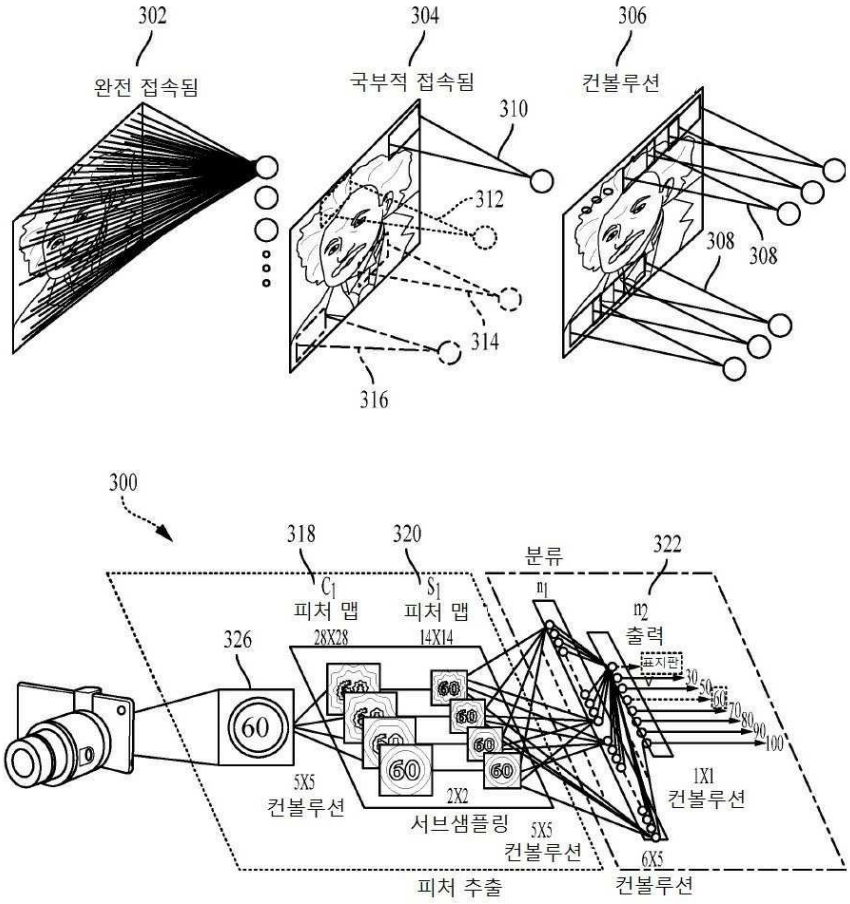
도면1



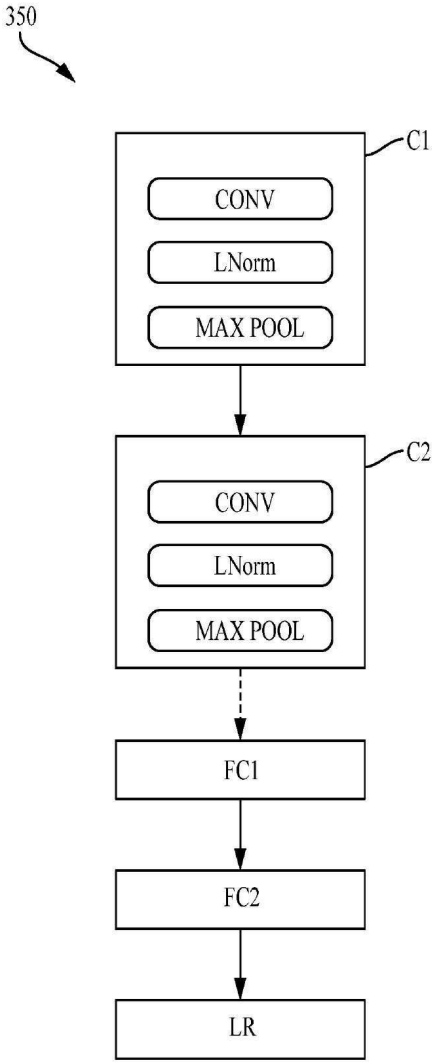
도면2


214
구성
프로세싱
202
로컬 프로세싱 유닛 i
206
로컬
파라미터
메모리
212
로컬 접속
메모리
210
로컬 학습
프로그램
204
로컬 상태
메모리
208
로컬 모델
프로그램
202
로컬 프로세싱 유닛 j
206
로컬
파라미터
메모리
212
로컬 접속
메모리
210
로컬 학습
프로그램
204
로컬 상태
메모리
208
로컬 모델
프로그램
216
라우팅
접속
프로세싱
200

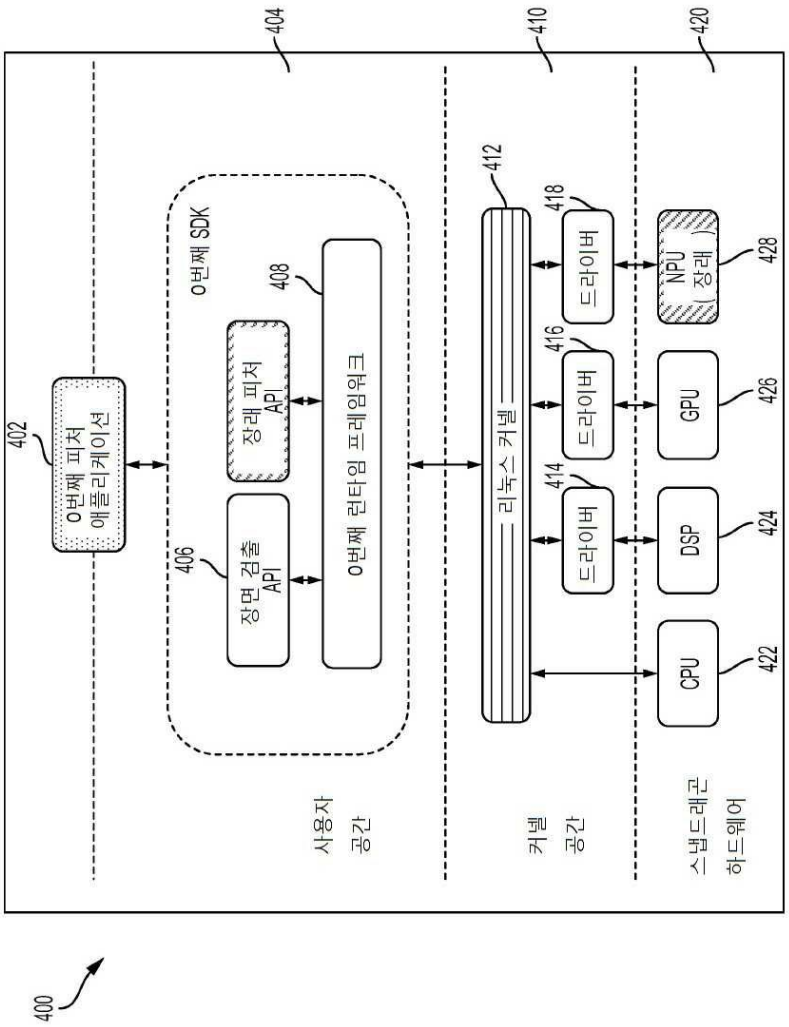
도면3a


302
완전 접속됨
304
국부적 접속됨
306
컨볼루션
310
312
314
316
308
308
300
318
320
분류
322
C_1
피처 맵
S_1
피처 맵
n_1
n_2
출력
28X28
14X14
326
60
표지판
30
50
70
80
90
100
5X5
컨볼루션
2X2
서브샘플링
5X5
컨볼루션
1X1
컨볼루션
6X5
컨볼루션
피처 추출

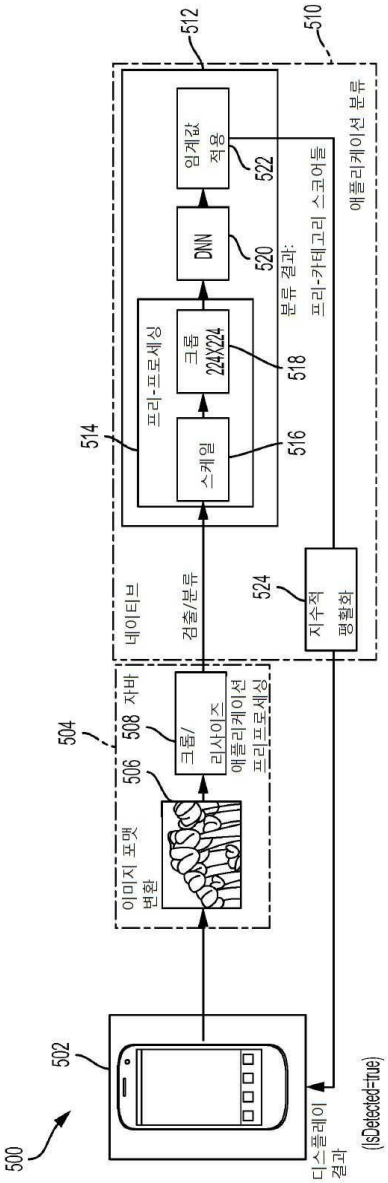
도면3b



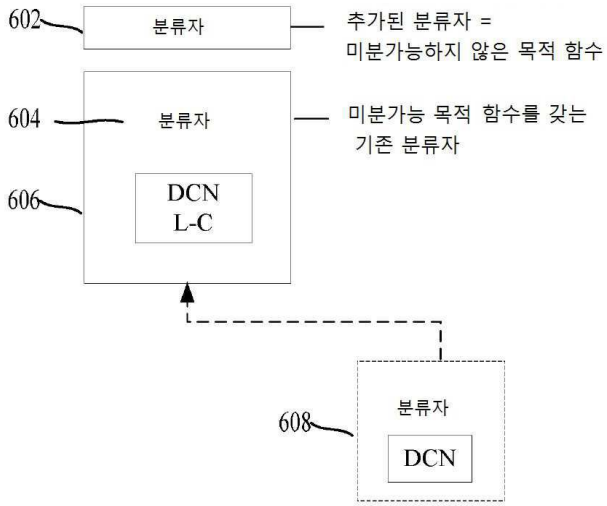
도면4


400
404
410
420
402
O번째 피처 애플리케이션
O번째 SDK
408
장래 피처 API
406
장면 검출 API
O번째 런타임 프레임워크
412
리눅스 커널
418
드라이버
416
드라이버
414
드라이버
NPU (장래)
428
GPU
426
DSP
424
CPU
422
사용자 공간
커널 공간
스냅드래곤 하드웨어

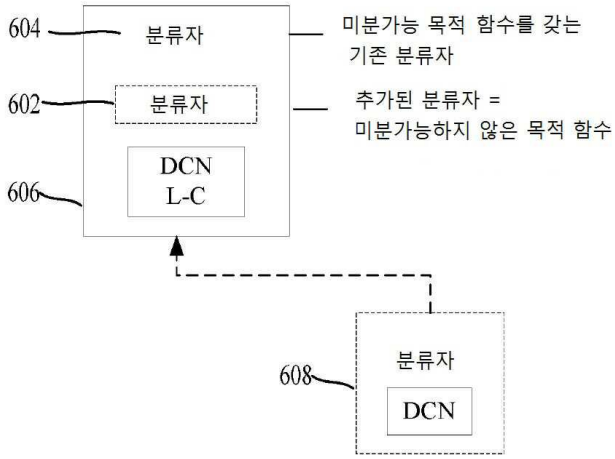
도면5


500
502
504
506
508
이미지 포맷 변환
자바 애플리케이션 프리프로세싱
크롭/ 리사이즈
네이티브
검출/분류
514
프리-프로세싱
스케일
516
크롭 224X224
518
DNN
520
분류 결과: 프리-카테고리 스코어들
애플리케이션 분류
임계값 적용
522
510
512
524
지수적 평활화
디스플레이 결과
(isDetected=true)

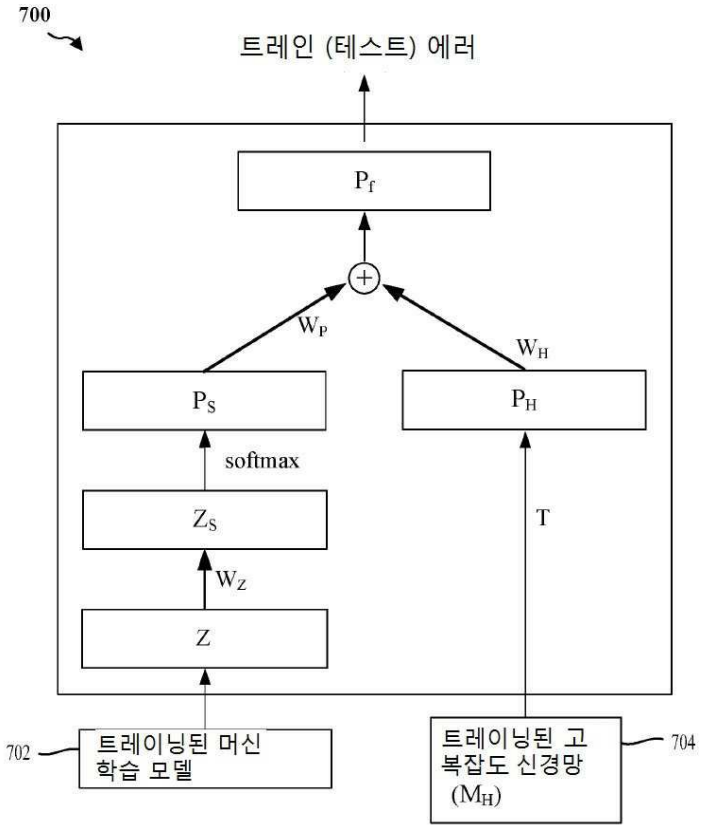
도면6a


602
분류자
추가된 분류자 =
미분가능하지 않은 목적 함수
604
분류자
미분가능 목적 함수를 갖는
기존 분류자
606
DCN
L-C
608
분류자
DCN

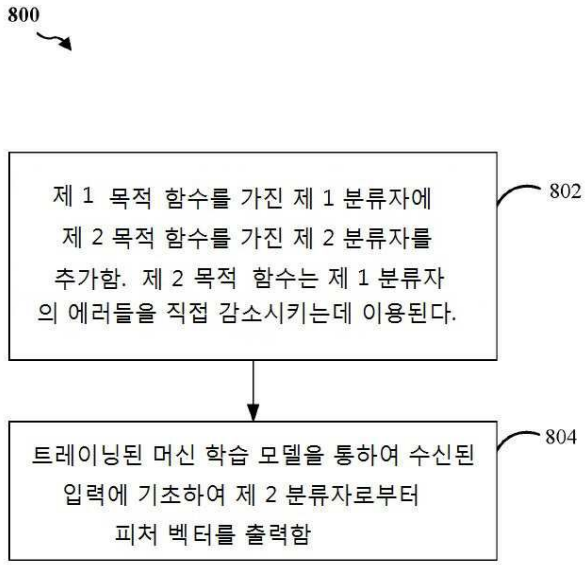
도면6b


604
분류자
미분가능 목적 함수를 갖는
기존 분류자
602
분류자
추가된 분류자 =
미분가능하지 않은 목적 함수
606
DCN
L-C
608
분류자
DCN

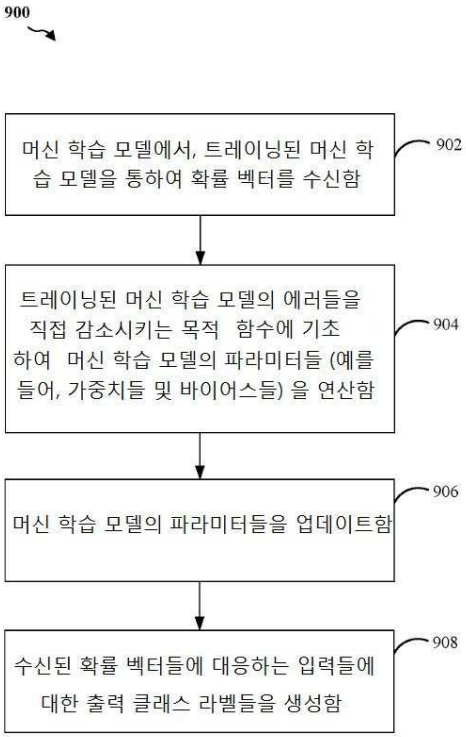
도면7


700
트레인 (테스트) 에러
P_f
W_P
W_H
P_S
P_H
softmax
Z_S
T
W_Z
Z
702
트레이닝된 머신 학습 모델
704
트레이닝된 고 복잡도 신경망 (M_H)

도면8


800
제 1 목적 함수를 가진 제 1 분류자에 제 2 목적 함수를 가진 제 2 분류자를 추가함. 제 2 목적 함수는 제 1 분류자의 에러들을 직접 감소시키는데 이용된다.
802
트레이닝된 머신 학습 모델을 통하여 수신된 입력에 기초하여 제 2 분류자로부터 피처 벡터를 출력함
804

도면9


900
머신 학습 모델에서, 트레이닝된 머신 학습 모델을 통하여 확률 벡터를 수신함
902
트레이닝된 머신 학습 모델의 에러들을 직접 감소시키는 목적 함수에 기초하여 머신 학습 모델의 파라미터들 (예를 들어, 가중치들 및 바이어스들) 을 연산함
904
머신 학습 모델의 파라미터들을 업데이트함
906
수신된 확률 벡터들에 대응하는 입력들에 대한 출력 클래스 라벨들을 생성함
908