Kafka-ML: 데이터 스트림을 ML/AI 프레임워크와 연결

Cristian Mart' n, Peter Langendoerfer, Pouya Soltani Zarrin, Manuel D' az 및 Bartolome Rubio'

개요 - 기계 학습(ML)과 인공 지능(AI)은 알고리즘을 통해 훈련하고, 개선하고, 예측하기 위해 데이터 소스에 의존합니다. 디지털 혁명과 사물 인터넷과 같은 최신 패러다임으로 인해 이러한 정보는 정적 데이터에서 지속적인 데이터 스트림으로 바뀌고 있습니다. 그러나 현재 사용되는 대부분의 ML/AI 프레임워크는 이러한 혁명에 완전히 준비되어 있지 않습니다. 본 논문에서는 데이터 스트림(Apache Kafka)을 통해 TensorFlow ML/AI 파이프라인 관리를 가능하게 하는 오픈소스 프레임워크인 Kafka-ML을 제안했습니다. Kafka-ML은 사용자가 쉽게 ML 모델을 정의하고 추론을 위해 학습, 평가 및 배포할 수 있는 액세스 가능하고 사용자 친화적인 웹 사용자 인터페이스를 제공합니다. Kafka-ML 자체와 배포된 구성 요소는 컨테이너화 기술을 통해 완벽하게 관리되므로 이식성과 손쉬운 배포, 내결함성 및 고가용성과 같은 기타 기능이 보장됩니다. 마지막으로, 데이터 스트림을 관리하고 재사용하기 위한 새로운 접근 방식이 도입되었으며, 이로 인해 데이터 스토리지 및 파일 시스템이 활용되지 않을 수도 있습니다.

색인 용어 - Kafka-ML, Apache Kafka, 기계 학습, 인공 지능, 데이터 스트림, TensorFlow, Docker, Kubernetes

I. 소개

이 디지털 시대에 정보는 다양한 출처에서 다양한 목적과 부문을 통해 지속적으로 수집되고 처리됩니다. 이러한 의미에서 기계 학습(ML)과 인공 지능(AI)[1]은 원시 정보를 유용한 예측 및 권장 사항으로 변환하여 무엇보다도 비즈니스 운영과 시민의 삶을 개선하는 데 결정적인 역할을 하고 있습니다. 예를 들어 Facebook과 같은 회사에서는 부적절한 콘텐츠를 감지하기 위해 매일 수백만 장의 사진을 처리합니다. 이는 ML/AI 알고리즘 및 시스템에 대한 지속적인 정보 데이터 스트림을 생성합니다.

최근에는 사물 인터넷(IoT)[2]이 확산되면서 인터넷 시대에 새로운 데이터 소스가 활성화되었으며, 2030년까지 연결된 장치가 5,000억 개에 달할 것으로 예상됩니다[3]. 인더스트리 4.0, 커넥티드카, 스마트시티 등의 패러다임이 가능해지고 있으며, 무엇보다도 물리적 세계의 서비스와 현상의 디지털화에 기여하고 있습니다. 결과적으로 데이터 스트림은 지속적으로 증가했으며 앞으로 몇 년 동안 엄청난 확장이 예상됩니다.

전통적으로 ML/AI 알고리즘의 설계 및 개발을 뒷받침하는 대부분의 ML/AI 프레임워크는 데이터 스트림에서 작동하도록 설계되지 않았지만 지속성 데이터 세트 및 정적 데이터에서 작동하도록 설계되었습니다. 요즘에도 PyTorch, Theano, TensorFlow와 같은 인기 있는 Python 프레임워크는 가장 인기 있는 데이터 스트림 시스템인 Apache Kafka[4]와 같은 데이터 스트림 시스템을 제공하지 않거나 부분적으로만 지원합니다.

여기에는 ML 모델 교육뿐만 아니라 추론, 테스트, 평가 등 ML/AI 파이프라인의 일부일 수 있는 나머지 단계도 포함됩니다. 이러한 과제에 대처하기 위해 데이터 스트림을 통해 ML/AI 파이프라인을 관리하는 오픈 소스 프레임워크인 Kafka-ML1이, 제시됩니다. Kafka-ML은 Apache Kafka를 활용하며 현재 데이터 스트림과 ML/AI를 통합하기 위한 ML 프레임워크로 TensorFlow를 지원합니다. 그러나 목표는 가까운 시일 내에 ML/AI 프레임워크에 대한 지원을 확장하는 것입니다. Kafka-ML은 ML/AI 전문가와 비전문가 모두를 위해 ML/AI 파이프라인을 관리하기 위해 액세스 가능하고 사용자 친화적인 웹 사용자 인터페이스(AutoML 이니셔티브와 유사한 접근 방식)를 제공합니다. 사용자는 알고리즘을 훈련, 비교, 평가하고 추론하기 위해 ML 모델 코드 몇 줄만 작성하면 됩니다. 또한 이 프레임워크는 Apache Kafka에서 데이터 스트림을 관리하기 위한 새로운 접근 방식을 사용합니다. 이는 구성된 횟수만큼 재사용될 수 있으므로 Kafka-ML의 데이터 세트에 대한 데이터 저장소나 파일 시스템이 필요하지 않습니다. 마지막으로 Kafka-ML은 컨테이너화 및 컨테이너 오케스트레이션 플랫폼을 활용하여 시스템 로드를 분산하고 구성 요소 배포를 촉진하는 동시에 내결함성과 고가용성을 제공합니다.

따라서 이 백서의 주요 기여는 다음과 같습니다. 1) 데이터 스트림을 통해 ML/AI 파이프라인을 관리하기 위한 액세스 가능하고 사용자 친화적인 오픈 소스 프레임워크인 Kafka-ML을 제시합니다.

2) 데이터 저장이나 파일 시스템이 필요 없이 ML/AI 파이프라인의 데이터 스트림을 관리하는 새로운 접근 방식 문서의 나머지 부분은 다음과 같이 구성됩니다. 섹션 II에서는 Kafka-ML의 배경을 제시합니다. 섹션 III에서는 이 작업의 동기를 제시합니다. 섹션 IV에서는 Kafka-ML의 ML/AI 파이프라인이 소개됩니다. 그런 다음 섹션 V에서는 Kafka-ML 아키텍처와 해당 구성 요소가 제시됩니다. Apache Kafka의 데이터 스트림 관리 접근 방식은 섹션 VI에 나와 있습니다. Kafka-ML의 검증은 섹션 VII에서 분석되고 관련 작업은 섹션 VIII에서 논의됩니다. 마지막으로 우리의 결론과 향후 연구는 섹션 IX에 제시되어 있습니다.

II. 배경

Apache Kafka는 대량의 메시지를 전달하고 소비할 수 있는 분산 메시징 시스템(게시/구독)입니다.

Cristian Mart' n, Manuel D' az 및 Bartolome Rubio'는 스페인 말라가 소재 말라가 대학의 ITIS 소프트웨어 및 언어 및 컴퓨터 과학부에서 근무하고 있습니다. Peter Langendoerfer와 Pouya Soltani Zarrin은 독일 프랑크푸르트(오데르)의 IHP - Leibniz-Institut fr Innovative Mikroelektronik에 있습니다. Peter Langendoerfer는 독일 Cottbus의 Brandenburg University of Technology Cottbus-Senftenberg에도 근무하고 있습니다. 이메일: Cristian Mart' n (cmf@lcc.uma.es).

1https://github.com/ertis-research/kafka-ml

낮은 대기 시간의 데이터. 기존 메시지 대기열은 주제당 여러 소비자를 추가하여 높은 비율의 메시지 소비를 지원할 수 있지만 한 번에 한 명의 소비자만 각 메시지를 수신합니다. 메시지 대기열과 마찬가지로 게시/구독 시스템은 생산자에서 소비자로 정보를 교환합니다.

그럼에도 불구하고 메시지 대기열과 달리 게시-구독 시스템에서는 여러 소비자가 주제의 각 메시지를 수신할 수 있습니다. 요즘 빅데이터 시대에는 스트림 데이터가 일괄 처리, 스트림 처리 등 여러 시스템으로 이동하지만 낮은 지연 시간도 요구됩니다. 따라서 두 기능이 모두 필요하며 Apache Kafka가 이를 제공하는 방법은 다음과 같습니다.

- 다중 고객 배포. 게시/구독 시스템인 Apache Kafka를 사용하면 여러 클라이언트와 고객을 메시지에 연결할 수 있습니다. 또한 Apache Hadoop, Apache Storm, TensorFlow 등과 같은 광범위한 솔루션에 대한 통합 및 지원 덕분에 이 기능은 확실히 가능한 것 이상입니다. • 메시지 발송 비율이 높습니다. 이는 다음과 같은 기능을 결합하여 달성됩니다. 1) 메시지 세트 추상화: 메시지는 한 번에 단일 메시지를 보내는 대신 네트워크 왕복 오버헤드를 상각하여 함께 그룹화됩니다. 2) 이
진 메시지 형식: 데이터 청크를 수정 없이 전송할 수 있습니다. 3) 제로 복사 최적화: 페이지 캐시의 많은 복사본을 방지합니다. 그러나 가장 주목할만한 기능 중 하나는 Kafka 소비자 그룹으로, 이를 통해 메시지 대기열처럼 Apache Kafka에서 관리하는 고객 클러스터에 메시지를 배포할 수 있습니다.

주제는 Kafka의 메시지 스트림으로, 생산자는 메시지를 게시할 수 있고 소비자는 이를 수신하기 위해 구독할 수 있습니다. 많은 분산 대기열 프레임워크와 달리 생산자가 메시지를 Kafka로 보내면 Kafka는 구성 가능한 보존 정책을 사용하여 메시지를 디스크에 저장하므로 나중에 구성 요소에서 데이터를 검색할 수 있습니다. 이는 분산 로그로 널리 알려져 있으며 소비자가 필요한 대로 로그를 살펴볼 수 있도록 합니다. Kafka-ML의 ML 훈련과 같은 일부 경우에는 모든 데이터가 한 번에 처리되고, 이 프로세스 중에 오류가 발생하더라도 고객이 데이터 손실 및 파일 시스템에 저장하지 않고도 다시 시작할 수 있기 때문에 이 기능이 적합합니다.

로드 밸런싱 및 내결함성은 주제의 파티션에서도 수행됩니다. 여기서 각 주제는 여러 파티션으로 나눌 수 있고 각 파티션은 여러 복제본을 가질 수 있습니다. 파티션을 사용하면 로그를 더 작은 단위로 나누고 로드 밸런싱을 제공할 수 있으며 토픽 복제본을 사용하면 내결함성이 가능합니다. Apache Kafka 클러스터는 파티션과 복제본을 공유하는 브로커의 P2P 네트워크로 구성됩니다. 소비자 그룹이 있는 경우 파티션을 고객과 연결하여 높은 파견 비율을 달성할 수 있습니다.

Apache Kafka는 또한 '최대 1회', '최소 1회', '정확히 1회'와 같은 다양한 정책을 통합하여 메시지 발송을 위한 맞춤형 QoS 정책을 가능하게 합니다.

Apache Kafka의 인기, 다수의 구현 및 많은 클라우드 컴퓨팅 시스템과의 통합, 커뮤니티에서의 큰 수용으로 인해 Apache Kafka는 시스템 상호 연결, 데이터 수집 및 정보 전달을 위한 사실상의 솔루션으로 전환되었습니다.

III. 동기 부여

ML/AI 애플리케이션이 알고리즘을 훈련하고 평가하기 위해 비디오 데이터 소스가 있는 경우 많은 양의 데이터가 필요하다는 것은 의심의 여지가 없습니다. 전통적으로 ML/AI 데이터는 데이터 저장소와 대용량 파일에서 발견되므로 사용자와 애플리케이션 간의 공유가 어렵습니다. 결과적으로 대부분의 경우 사용자와 개발자는 데이터 세트의 로컬 복사본을 보유하거나(중견기업에 대한 영향을 가정) 공유 인프라를 사용하여 데이터 세트를 처리하고 액세스합니다. 반면, 다양한 알고리즘과 접근 방식을 평가하려면 동일한 데이터 세트가 필요할 수 있으며, 이는 데이터 전송용으로 설계되지 않은 시스템에서는 제한될 수 있습니다.

IoT가 확산되면서 데이터는 정적 소스에서 데이터 스트림으로 바뀌고 있습니다. 인터넷은 정보와 데이터 스트림의 또 다른 거대한 소스입니다. 그러나 현재 ML 프레임워크의 대부분은 데이터 스트림과 함께 작동하도록 설계되지 않았습니다. 즉, 이를 직접 지원하지 않거나 커넥터만 제공하지만 ML/AI 파이프라인과 제대로 통합되지 않았습니다. 여기에는 데이터 스트림을 사용하여 수신 데이터를 예측할 수 있는 추론 작업뿐만 아니라 ML/AI 파이프라인의 교육 및 평가 단계를 위한 데이터 스트림 수집도 포함됩니다. 네트워크 링크를 따라 데이터 스트림을 사용하여 작업하면 데이터 손실이 발생할 수 있는데, 이는 로컬 파일 시스템에는 없을 수 있으므로 고려해야 합니다.

또한, 우리는 모든 사람이 개방적이고 접근 가능한 ML 및 AI를 제공하는 것을 목표로 합니다. 이러한 의미에서 Google Cloud AutoML과 같은 AutoML 이니셔티브는 경험이 부족한 개발자에게 다양한 비즈니스 요구에 맞게 조정된 고품질 모델과 솔루션을 제공하는 데 기여해 왔습니다. 그러나 이러한 이니셔티브는 아직 데이터 스트림 파이프라인과 제대로 통합되지 않았습니다. 사용자가 ML 및 AI에 접근할 수 있는 길을 열어주는 또 다른 방법은 다양한 모델과 구성을 평가하는 데 사용할 수 있는 훈련된 모델과 지표(예: 손실 및 정확도)를 공유할 수 있는 누락된 생태계를 제공하는 것입니다.

ML/AI 솔루션에는 일반적으로 개인용 컴퓨터에서 달성하기 어려운 요구 사항이 필요하며 개발자는 애플리케이션을 배포하기 위해 공유 인프라를 채택하는 경향이 있습니다.

또한 ML/AI 미션 크리티컬 애플리케이션에는 고가용성, 로드 밸런싱 및 내결함성이 필요할 수 있으며 사용자에게 투명한 방식으로 제공되어야 합니다. 이를 통해 사용자와 개발자는 ML 모델과 비즈니스 로직에 집중할 수 있어 프로덕션 시스템에서 ML/AI 애플리케이션의 수명 주기가 촉진됩니다.

요약하자면, 우리는 데이터 스트림과 ML 프레임워크 간의 격차를 줄이고, AutoML 기능을 채택하고, 고성능 및 고가용성 인프라에서 ML 모델, 측정항목 및 결과를 공유할 수 있는 시스템을 구상합니다.

IV. KAFKA-ML 의 ML 모델 파이프라인

이 섹션에서는 수명 주기를 나타내는 Kafka-ML의 ML 모델 파이프라인을 소개합니다. 그림 1은 수행할 파이프라인과 단계를 보여줍니다. A) ML 모델 설계 및 정의; B) ML 모델의 훈련 구성을 생성하는 것, 즉 훈련할 ML 모델 세트를 선택하는 것;

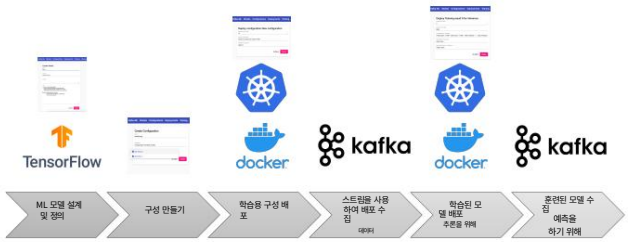

그림 1. Kafka-ML의 ML/AI 파이프라인

C) 훈련을 위한 구성을 배포합니다. D) 섭취
교육 및 선택적으로 평가를 통해 배포된 구성
Apache Kafka를 통해 데이터를 스트리밍합니다. E) 훈련된 인력 배치
추론 모델; F) 그리고 마지막으로 배치된 먹이를 공급합니다.
데이터를 사용해 예측하기 위한 추론용 훈련 모델
스트림. 이전 단계의 대부분은 RESTful API를 사용하므로,
파이프라인은 자동화될 수 있으며, 이와 관련된 모든 단계는
ML 모델 공급(훈련 및 추론)이 수행됩니다.
데이터 스트림으로. 데이터스토어가 더 이상 필요하지 않을 수 있음
(섹션 VI). 다음에서는 이러한 각 단계에 대해 자세히 설명합니다.

A. ML 모델 설계 및 정의

처음부터 우리는 이 단계를 다음과 같이 만들고 싶었습니다.
ML 개발자가 ML 모델에 집중할 수 있도록 최대한 간단하게 만듭니다.
새로운 라이브러리를 배우거나 복잡한 파이프라인을 사용하는 대신
ML을 쉽게 테스트하고 검증할 수 있는 도구
ML 개발자를 위한 모델은
일하고 그들이 전문가인 분야에 집중할 수 있도록 해줄 것입니다.
에. 이러한 이유로 필요한 유일한 소스 코드는 ML입니다.
그림과 같이 널리 사용되는 ML 프레임워크의 모델 정의 자체
목록 1에서.

목록 1. Kafka-ML용 ML 코드 예
모델 = tfkeras . 순차([
tf 밀도(32 , 케라스 . 레이어활성 . , 입력 어둡게 = 100
, 화 = tf . nn . 렐루) ,
tf . 케라스 . 레이어활성 . 조밀한 (1 ,
화 = tf . nn . 시 gm oi d)])
모델 . c om pil e (최적화 = '아담' 손실 = , ,
sparsecategoricalcros sentropy metrics = [ ])
정확성

Listing 1 소스 코드는 친숙해 보일 수 있습니다. 사실, 그것은
숨겨진 레이어가 있는 간단한 Python TensorFlow/Keras 모델,
단일 출력 및 훈련용 컴파일. 카프카-ML
현재 Python TensorFlow [5]를 지원합니다.
TensorFlow/IO2를 통한 Apache Kafka . 현재 아파치는
Kafka 통합은 개발 중이며 추가 ML을 수신하여 Kafka-ML 도메인이 확장되고 있습니다.
프레임워크.

ML 편집기(예: Jupyter)를 사용하여 모델이 정의되면
모델의 TensorFlow/Keras 소스 코드를 삽입할 수 있습니다.
모델 생성을 위해 Kafka-ML 웹 UI(사용자 인터페이스)에
그림 2와 같이 모델을 정의할 수도 있습니다.
Kafka-ML에서 직접 검증하는 것이 좋습니다.

2https://www.tensorflow.org/io

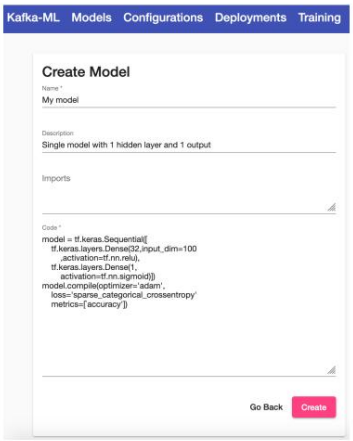

그림 2. Kafka-ML의 ML 모델 정의

더 강력한 다른 ML 통합을 사용하여 미리
개발 환경(IDE) 또는 편집자. 기타 필수
모델에 대한 기능(있는 경우)을 가져오기에 삽입할 수 있습니다.
필드. 모델이 제출되면 소스 코드가
유효한 TensorFlow 모델로 확인되어
카프카-ML. 모델이 성공적으로 정의된 경우
파이프라인은 다음 단계로 계속될 수 있습니다.

B. 구성 생성

구성은 Kafka-ML 모델의 논리적 집합입니다.
훈련을 위해 그룹화됩니다. 필요할 때 유용할 수 있습니다.
측정항목(예: 손실 및 정확도)을 평가하고 비교합니다.
Kafka-ML 모델 세트 또는 단순히 모델 그룹 정의
동일하고 고유한 데이터 스트림으로 훈련할 수 있는
평행한. 따라서 n개의 ML 모델이 있는 경우
모두 훈련을 위한 데이터 스트림이 필요하며 단 하나만 필요합니다.
구성이 필요한 경우 데이터 스트림을 Apache Kafka로 보내야 합니다.
n개의 모델로 정의되었습니다. 구성에 유의하세요.
모델만으로 정의할 수도 있습니다. 구성은 다음과 같습니다.
그림 3과 같이 Kafka-ML 웹 UI에서 생성됩니다.

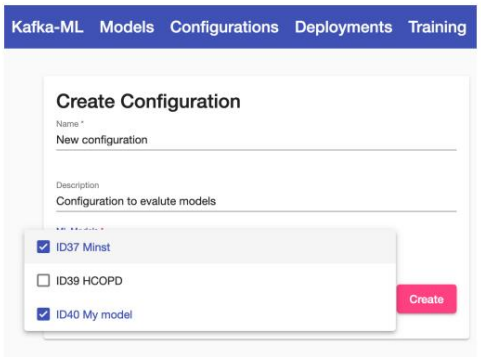

그림 3. Kafka-ML에서 구성 생성

C. 학습을 위한 구성 배포

배치 크기와 같은 일부 학습 매개변수를 설정한 후
세대 및 반복 횟수, 선택적으로 Kafka-ML 웹 UI(그림 4)에서 평가를 위한 일부 매개변수,

학습을 위해 구성을 배포할 준비가 되었습니다. 그렇다면 Kafka-ML 모델별로 작업이 배포됩니다. 그런 다음 배포된 각 작업이 수행하는 첫 번째 단계 중 하나는 Kafka-ML 아키텍처에서 해당 ML 모델을 가져와서 로드하여 훈련을 시작하는 것입니다. 마지막으로 훈련 및 선택적으로 평가 데이터가 포함된 데이터 스트림이 Apache Kafka를 통해 수신될 때까지 작업을 재개할 수 있습니다. 이를 통해 데이터 스트림이 전송될 때 학습 가능한 ML 모델을 보유하고 데이터 스트림이 이미 Kafka에 있는 경우 직접 학습을 수행할 수 있습니다.

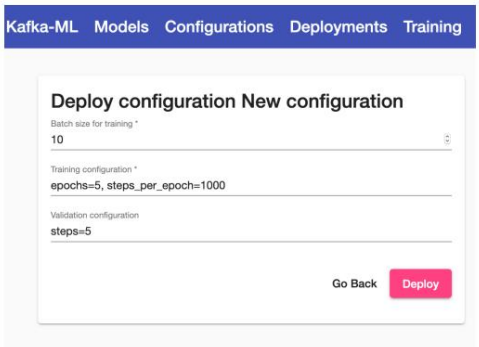

그림 4. Kafka-ML에서 훈련을 위한 구성 배포

D. 스트림 데이터를 사용하여 배포 수집 모델이 배포되면 파

이프라인을 계속하기 위해 훈련을 위해 데이터 스트림을 보내야 합니다. 모델 배포 전에 제출할 수도 있습니다. Kafka 스트림 커넥터는 시작 시 스트림 데이터를 가질 것으로 예상하므로 Kafka에서 데이터 스트림을 사용할 수 있을 때까지 훈련을 시작할 수 없습니다. 우리는 이를 극복하기 위해 최소한 두 가지 Kafka 주제를 사용했습니다. 1) 훈련 및 평가에 필요한 훈련 및 평가 데이터 스트림만 포함하는 데이터 주제; 2) 및 데이터 스트림을 교육 및 평가에 사용할 수 있는 시기와 위치에 대한 제어 메시지를 통해 배포된 ML 모델에 알리는 제어 주제입니다. 섹션 VI에서는 이에 대해 자세히 논의할 것입니다. 제어 메시지에는 최소한 다음 정보가 포함되어야 합니다. • 배포 ID: 배포된 구성의 ID입니다.

데이터 스트림이 이동합니다.

- 주제: 학습 및 평가 데이터가 있는 Kafka 주제
  스트림이 있습니다.
- 입력 형식: 데이터 스트림 형식(예: RAW,
  AVRO).
- 입력 구성: 선택한 데이터 형식(예: Avro의 구성표)에 필요한 구성입니다. • 검증 비율: 평가에 사용될 스트림 데이터의 비
  율입니다. 검증률이 0이면 훈련만 수행됩니다.
- 총 메시지 수: 데이터 스트림에서 전달된 메시지 수입니다. 전송된 메시지 수는 Kafka-ML 라이브러리에 의해 계산됩니다.

Kafka-ML은 현재 RAW 형식(이미지와 같이 모양 변경을 요청할 수 있는 단일 입력 데이터 스트림에 적합)과 Apache Avro(구성표가 데이터 스트림 디코딩 방법을 지정하는 복잡한 다중 입력 데이터 세트에 적합)를 지원합니다. 새로운 데이터 지원을 위해 열려 있습니다.

형식. 각 경우에 디코딩을 위한 정보는 제어 메시지(입력 구성)에 포함됩니다(예: Avro 형식에 대한 훈련 및 레이블 데이터 체계). 우리는 이 두 가지 데이터 형식에 대한 라이브러리를 개발했습니다. 이는 데이터 스트림이 전송될 때 제어 메시지를 보내는 것과 같은 Kafka-ML 측면을 처리하므로 데이터 스트림 전달을 더 쉽게 만듭니다.

IoT 장치 또는 게이트웨이에서 제공되는 라이브러리, 데이터 세트 또는 정보 소스를 해당 배포 ID로 전달한 후 구성에 그룹화된 모든 ML 모델이 훈련 및 평가를 시작합니다(검증률 > 0).

E. 추론을 위해 훈련된 모델 배포

훈련 및 평가 직후, 훈련된 모델 자체와 정의된 측정항목(예: 손실 및 정확도)이 각 훈련 작업에 의해 Kafka-ML 아키텍처에 제출됩니다.

결과는 그림 5와 같이 Kafka-ML 웹 UI에서 시각화할 수 있습니다. 각 결과에 대해 사용자는 결과를 편집하거나 훈련된 모델을 다운로드하거나 추론을 위해 배포할 수 있습니다.

추론 배포(그림 6)에서 사용자는 배포할 추론 복제본 수를 선택할 수 있습니다. 이는 Apache Kafka의 소비자 그룹 기능을 활용하여 추론을 위한 로드 밸런싱 및 내결함성을 활성화합니다. 또한 모든 상호 작용은 Apache Kafka를 통해 수행되며 사용자는 입력 항목(예측할 값)과 출력 항목(예측)을 구성해야 합니다.

F. 추론을 위한 스트림 데이터와 함께 배포된 학습 모델 수집

마지막으로, 학습된 모델이 준비되고 배포되어 데이터 스트림을 통해 예측 및 추천을 수행하면 ML/AI 파이프라인이 종료됩니다. 이 경우 입력 및 출력 주제 이후에는 제어 메시지를 보낼 필요가 없으며 입력 형식 및 구성은 Web-UI에서 미리 정의되어 있습니다(그림 6). 사용자와 시스템은 입력 주제에 정의된 데이터 형식으로 인코딩된 데이터 스트림을 전송하기만 하면 되며, 모델 예측이 구성된 출력 주제에 대한 추론 결과가 즉시 전송됩니다.

V. KAFKA-ML 아키텍처

Kafka-ML 아키텍처는 마이크로서비스 아키텍처를 구성하는 단일 책임 원칙을 기반으로 하는 구성 요소 집합으로 구성됩니다. 이러한 모든 구성 요소는 Docker 컨테이너로 실행될 수 있도록 컨테이너화되었습니다. 이는 아키텍처의 간편한 이식성, 인스턴스 간 격리, 다양한 플랫폼에 대한 빠른 설정 지원뿐만 아니라 Kubernetes와 같은 컨테이너 오케스트레이션 플랫폼을 통한 관리 및 모니터링도 가능하게 합니다[6].

Kubernetes는 컨테이너와 해당 복제본을 지속적으로 모니터링하여 정의된 상태와 지속적으로 일치하는지 확인하는 동시에 고가용성 및 로드 밸런싱과 같은 프로덕션 환경에 대한 기타 기능을 허용합니다. Kubernetes는 Kafka-ML 및 해당 구성 요소의 수명 주기를 관리합니다. Kafka-ML은 오픈 소스 프로젝트이며 구현, 구성, Kubernetes 배포 파일입니다.

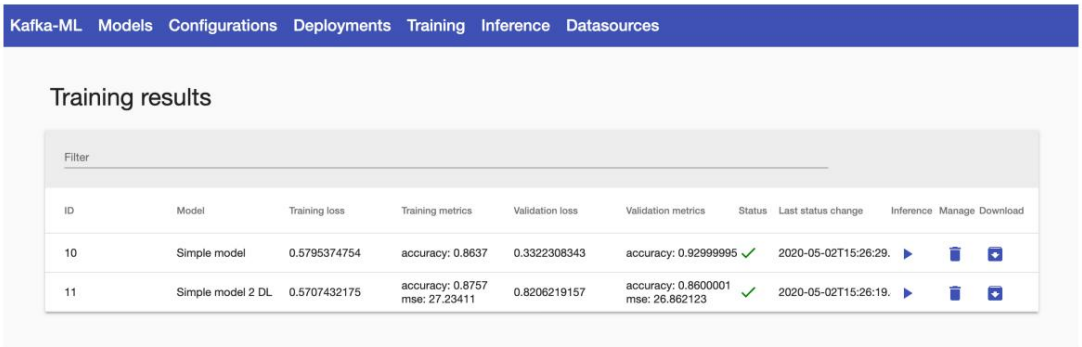

그림 5. Kafka-ML의 훈련 관리 및 시각화

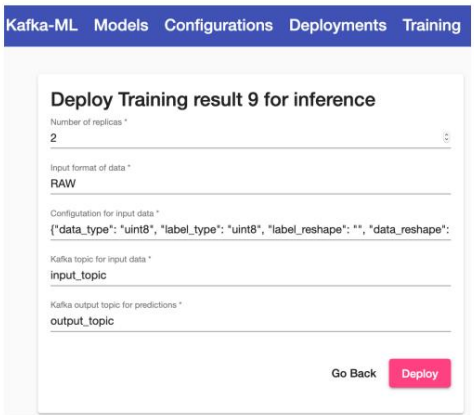

그림 6. Kafka-ML에서 추론을 위해 훈련된 ML 모델 배포

일부 예는 GitHub 저장소3에서 찾을 수 있습니다 .

Kafka-ML 아키텍처의 개요는 그림 7에 나와 있으며 아래에는 각 구성 요소가 자세히 설명되어 있습니다.

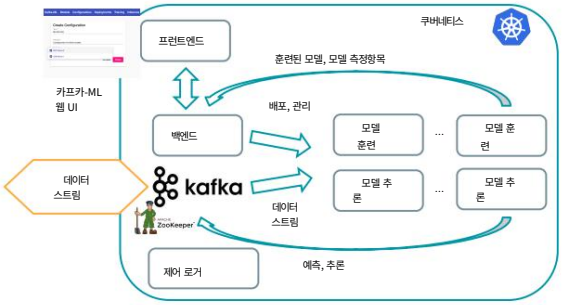

그림 7. Kafka-ML 아키텍처 개요

A. 프론트엔드

프런트 엔드는 사용자가 ML 모델 및 구성 생성, 교육 및 추론을 위한 배포 등 Kafka-ML에서 사용할 수 있는 작업을 사용자 친화적이고 접근 가능한 방식으로 수행할 수 있는 관리 웹 UI를 제공합니다. 프런트엔드는 백엔드에서 제공하는 RESTful API를 사용하며 웹 개발용으로 널리 사용되는 TypeScript 프레임워크인 Angular를 사용하여 구현되었습니다.

프런트엔드와 백엔드가 명확하게 구별되었기 때문에 이 아키텍처는 타사 애플리케이션을 통합하고 새로운 프런트엔드(예: 스마트폰 애플리케이션)를 생성할 수 있는 문을 열어줍니다.

B. 백엔드

백엔드 구성 요소는 ML 모델, 구성, 배포 등 Kafka-ML에 포함된 모든 정보를 관리하기 위해 RESTful API를 제공합니다. 이 구성 요소는 해당 Kubernetes API와 연결되어 사용자가 주문할 때 구성 및 ML 모델의 훈련 및 추론을 배포하고 관리합니다. 또한 백엔드는 구성을 학습한 후 학습된 ML 모델 및 측정항목도 수신합니다. 이러한 훈련된 모델은 나중에 추론하기 위해 다운로드하거나 배포할 수 있습니다. 이 구성 요소는 Kubernetes 구성 요소의 배포 및 관리를 위해 Kubernetes4용 공식 Python 클라이언트 라이브러리와 함께 Python 웹 프레임워크 Django를 통해 구현되었습니다 .

C. 모델 훈련

백엔드가 구성을 배포하면 Kubernetes의 배포 가능한 단위인 작업이 Docker 컨테이너를 훈련하고 컨테이너화하기 위해 Kafka-ML 모델별로 실행됩니다.

알고리즘 1은 학습 Job의 절차를 설명합니다. 단순화를 위해 예외 관리 및 데이터 스트림 디코딩과 같은 일부 단계는 포함되지 않았습니다. 먼저 Job은 백엔드에서 ML 모델을 다운로드합니다. 다음으로 예상되는 제어 스트림을 수신할 때까지, 즉 수신된 배포 ID와 일치할 때까지 제어 스트림 수신을 시작합니다.

교육 및 선택적으로 평가는 제어 스트림 메시지에서 수신된 데이터 스트림을 통해 수행됩니다. 마지막으로 작업은 훈련된 모델과 훈련 및 평가 지표를 백엔드에 제출합니다.

D. 모델 추론

ML 모델을 훈련하고 백엔드를 통해 추론을 위해 배포한 후, 복제본이 설정된 복제 컨트롤러(지정된 수의 복제본이 항상 실행되도록 보장하는 Kubernetes 구성 요소)가 해당 모델과 함께 Kubernetes에서 실행됩니다. 도커

3https://github.com/ertis-research/kafka-ml

4https://github.com/kubernetes-client/python

데이터: 모델 URL, 학습 kwargs, 평가 kwargs, 배포 ID, 스트림 데이터
결과: 훈련된 ML 모델과 훈련 및 평가 지표 모델 ←

downloadModelFromBackend(model url); 훈련됨 ← 거짓; 훈련받지 않은 동안

msg ← readControlStreams(); 배포 id == msg.deployment id 이면 훈련 스트림 ← readStream(msg.topic); msg.validation rate > 0 이면
훈련 스트림 ← take(데이터 스트림, msg.validation rate); 평가 스트림 ← 분할(데이터 스트림, msg.검증 비율)
훈련
종료 res ← trainModel(모델, 훈련 kwargs, 훈련 스트림); msg.validation rate > 0 이면 평가 res ← 평가모델(모델, 평가 kwargs, 평가 스트림) ;
끝
uploadTrainedModelAndMetrics(모델 URL, 모델, 교육 해상도, 평가 해상도); 훈련됨 ← 사실입니다. 끝 끝

알고리즘 1: Kafka-ML의 학습 알고리즘

컨테이너. 알고리즘 2는 Kafka-ML의 추론 절차를 설명합니다. 백엔드에서 훈련된 모델을 다운로드하면 이 구성 요소는 스트림 데이터 수신을 시작한 다음 수신된 스트림으로 예측을 수행하고 구성된 Kafka 출력 항목을 통해 전송합니다. Kafka 브로커와 파티션이 여러 개 있는 경우 복제 컨트롤러는 복제본과 파티션을 일치시켜 로드 밸런싱과 더 높은 데이터 수집을 제공함으로써 Apache Kafka의 소비자 그룹 기능을 활용합니다.

데이터: 모델 URL, 입력 주제, 출력 주제, 입력 구성, 스트림 데이터
결과: Apache Kafka 모델에 대한 예측 ←

downloadTrainedModelFromBackend(모델 URL); deserializer ← getDeserializer(입력 구성); while True do stream ←
readStreams(입력 주제); 데이터 ← 디코드(디시리얼라이저, 스트림); 예측 ← 예측(모델, 데이터); sendToKafka(예측, 출력 주제); 끝

알고리즘 2: Kafka-ML의 추론 알고리즘

E. 제어 로거

제어 로거 구성 요소는 Kafka-ML에서 수신된 제어 메시지를 소비하고 전송하는 역할만 담당합니다. 백엔드로 이동하세요. 이러한 제어 메시지는 백엔드에서 두 가지 목적으로 사용됩니다. 1) 전체 교육 및 평가 데이터 스트림을 보낼 필요 없이 배포된 다른 구성으로 메시지를 다시 보낼 수 있습니다. 이는 Apache Kafka가 데이터 스트림을 분산 로그; 2) 제어 메시지에서 수신된 정보를 사용하여 추론 입력 형식 및 구성을 자동 구성합니다. 입력 형식 및 구성은 Kafka-ML 웹 UI에서 직접 구성되지 않지만 제어 메시지에 정의되므로 추론을 위해 ML 모델을 배포할 때 사용자가 데이터 매개 변수를 정의하는 작업을 용이하게 합니다. 훈련 중에 Jobs는 제어 데이터 스트림과 이에 따른 정보를 받습니다.

F. Apache Kafka 및 ZooKeeper

Apache Kafka의 배포 및 관리를 용이하게 하고 Kubernetes가 제공하는 가능성을 활용하기 위해 Kubernetes의 Docker 컨테이너를 사용하여 Apache Kafka5 및 Apache ZooKeeper6 (동기화를 위해 Apache Kafka에 필요)를 작업으로 배포했습니다. 우리는 Kubernetes 서비스를 통해 내부적으로는 나머지 구성 요소에 노출되고 외부적으로는 다른 시스템이 데이터 스트림을 보낼 수 있도록 활성화했습니다.

6. Apache를 통한 데이터 스트림 관리 KAFKA 분산 로그

섹션 II에서 설명한 것처럼 Apache Kafka에서 제공되는 분산 로그를 통해 소비자는 원하는 대로 로그를 따라 이동하고 데이터 스트림을 읽을 수 있습니다. 이는 모든 데이터를 한 번에 처리해야 하는 구성요소/시스템(예: 훈련 작업)이 실패하여 모든 데이터 스트림을 복구해야 하는 경우에 유용합니다. 각 메시지가 소비된 후 삭제될 수 있는 기존 메시지 대기열 시스템에서는 이러한 상황에서 데이터 손실이 없도록 보장하기 위해 데이터 저장소가 필요할 수 있습니다.

반면, 데이터 스트림은 로그에 보관되도록 구성할 수 있으므로 전체 데이터 스트림을 다시 보낼 필요 없이 배포된 다른 구성 및 ML 모델에 대한 교육에 이러한 스트림을 재사용할 수 있습니다. 유일한 요구 사항은 보존 정책이 설정된 Apache Kafka에서 데이터 스트림을 사용할 수 있는 한 해당 제어 메시지(수십 바이트)를 원하는 배포 구성으로 보내는 것입니다. 이 기능의 예가 그림 8에 나와 있습니다. 먼저 첫 번째 데이터 스트림(녹색 데이터)이 해당 제어 메시지(C1)와 함께 배포된 구성 D1로 전송되었습니다. 구성 D2가 동일한 데이터 스트림을 사용할 수 있도록 제어 메시지 C1이 다시 전송되었습니다. 현재 분산 로그 상태에서 이 데이터 스트림은 만료되며 배포된 다른 구성에 더 이상 재사용할 수 없습니다.

제어 메시지 C2와 연관된 데이터 스트림은 배포된 구성 D3 및 D5로 전송되었으며, 이 스트림 데이터를 다시 사용하려는 새 구성에 계속 재사용할 수 있습니다. 마지막으로 훈련 및 평가 스트림의 왼쪽에 있는 데이터 스트림이 이제 분산 로그에 입력되고 있으며 데이터 스트림이 완료되지 않았기 때문에 제어 메시지가 아직 전송되지 않았습니다.

5https://github.com/wurstmeister/kafka-docker
6https://hub.docker.com/_/zookeeper

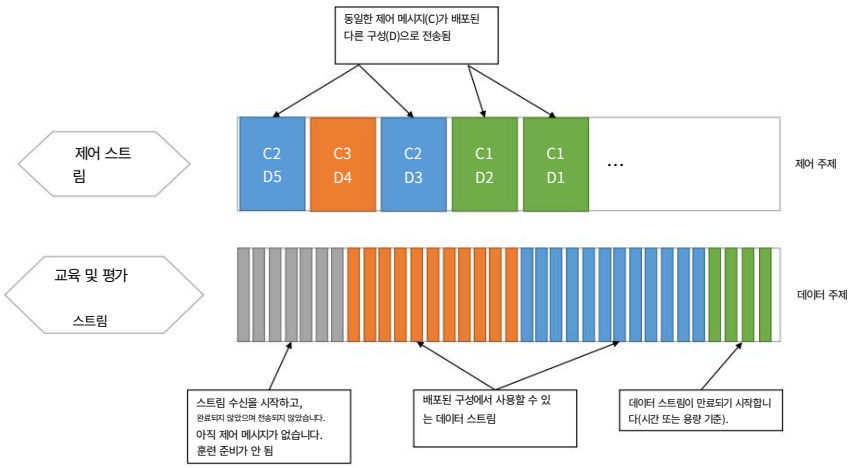

그림 8. Kafka-ML의 데이터 스트림 관리

훈련 및 평가 작업이 스트림 데이터를 따라 자유롭게 이동할 수 있도록 제어 메시지는 데이터 스트림이 있는 주제뿐만 아니라 배포 로그에서의 위치도 지정합니다. 이는 Ten-sorFlow/IO의 KafkaDataset 커넥터에서 제공하는 다음 목록 형식을 따릅니다: [topic:partition:offset:length]. 예를 들어, [kafka-ml:0:0:70000] 예제는 kafka-ml 주제와 해당 파티션 0에서 데이터 스트림이 오프셋 위치 0에서 70000까지 사용 가능함을 지정합니다. 이 제어 메시지를 통해 Kafka-ML 데이터 스트림이 정확히 어디에 있는지 배포된 구성을 알려줍니다. Kafka-ML 웹 UI에서는 사용자가 Kafka-ML로 전송된 데이터 스트림 목록을 볼 수 있는 양식을 사용할 수 있으며, 이는 다른 구성에 재사용할 수 있습니다.

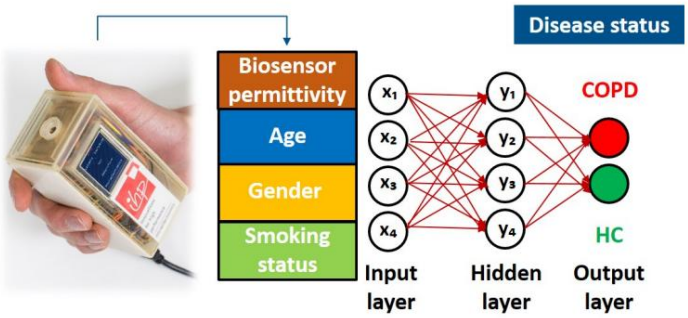

그림 9. IHP의 유전율 바이오센서를 사용하여 Exasens 데이터세트에서 사용할 수 있는 COPD 및 HC 샘플 분류에 사용되는 신경망 모델의 아키텍처

논의한 대로 이 동작은 Apache Kafka에 설정된 보존 정책에 따라 달라집니다. Apache Kafka 삭제 보존 정책 내의 현재 보존 전략은 다음과 같습니다.

1) 보존 바이트: Kafka가 공간을 확보하기 위해 오래된 로그 세그먼트를 삭제하기 전에 파티션이 커질 수 있는 최대 크기를 제어합니다. 기본값은 적용되지 않습니다.
2) 보존 시간(ms): 공간을 확보하기 위해 오래된 로그 세그먼트를 삭제하기 전에 로그가 보존되는 최대 시간을 제어합니다. 기본값은 7일입니다.

Apache Kafka는 압축 정책이라는 또 다른 보존 정책을 제공합니다. 이는 Kafka가 단일 주제 파티션의 각 메시지 키에 대해 최소한 마지막으로 알려진 값을 유지하도록 보장합니다. 그럼에도 불구하고 데이터 손실이나 압축이 필요하지 않기 때문에 Kafka-ML에는 삭제 보존 정책이 선호됩니다.

VII. 확인

UCI 기계 학습 저장소7에서 사용할 수 있는 오픈 액세스 Exasens 데이터세트는 만성 폐쇄성 폐질환(COPD) 환자와 건강 대조군(HC)의 타액 샘플 분류를 통해 Kafka-ML을 검증하기 위해 이 연구에서 사용되었습니다. 이 새로운 데이터 세트에는 그룹에서 수집한 타액 샘플에 대한 정보가 포함되어 있습니다. COPD 및 HC를 포함한 호흡기 환자의 [7]. 피험자 분류에 사용된 데이터세트의 속성에는 그림 9와 같이 환자의 인구통계학적 정보(나이, 성별, 흡연 상태)와 각 클래스에 대해 특징화된 타액 샘플의 유전 특성이 포함됩니다.]. 계산 목적을 위해 진단, 성별, 흡연 상태의 비정량적 속성은 진단(COPD (1)HC (0)), 성별(남성(1)여성(0)), 흡연 상태 라벨을 사용하여 숫자 값으로 변환되었습니다. (흡연자(3)전흡연자(2)비흡연자(1)). 이 작업에서 분석에 사용된 데이터의 측정항목, 모델 및 섹션은 GitHub8에서 확인할 수 있습니다 .

ML 분류기의 성능을 향상시키기 위해 데이터 세트는 평균 및 단위 분산이 0인 가우스 표준 정규 분포로 표준화되었습니다. 또한, 본 연구에서는 모델 평가를 위해 5겹 교차 검증 방법을 구현하여 과적합을 방지하고 신뢰할 수 있는 결과를 제공했습니다. 따라서 모든 교차 검증 접기에 대해 데이터 세트는 각각 2080% 비율로 테스트-학습 하위 집합으로 분할되었습니다. 데이터 준비 및 ML 구현은 Keras 2.2.5 및 Scikit-learn 0.22 Python 라이브러리를 사용하여 JupyterLab 환경에서 수행되었습니다.

7https://archive.ics.uci.edu/ml/datasets/Exasens

8https://github.com/Pouya-SZ/HCOPD

Avro 인코딩 형식은 두 가지 교육 모두를 위해 설계되었습니다.
입력 데이터에는 여러 변수가 있으므로 레이블 데이터입니다. 그만큼
이 검증의 소스 코드는 다음 예제에 포함되어 있습니다.
카프카-ML9 . Keras 모델 소스 코드는 다음과 같이 삽입되었습니다.
HCOPD GitHub 저장소에 정의되어 있습니다(목록 2).

목록 2. Kafka-ML에 사용되는 HCOPD Keras ML 코드

```
모델 = tf . 순차( [ . 케라스
tf . 케라스 · 레이어 D ropo ut ( 0 . 2 ,
inputshape = ( 4 , ) ) ,
tf · keraslayers 밀도(4 활성화 = 'si gm oi ,
d') ,
tf . 케라스 · 레이어활성 ; 조밀한( 2이 ,
화 = 므로 ftma x ' )
] )
모델 · c om pil e ( tf ( lr · 케라스 · 최적화 프로그램 . 아담
= . 0 0 0 1,) ,
손실 = sparsecategoricalcros sentropymetrics = [ ] ) ,
정확성
```

이 예를 통해 우리는 측정하고 평가했습니다.
데이터 스트림에 대한 Kafka-ML의 응답 시간 및
컨테이너화. 연구를 위해 지연 응답을 측정했습니다.
다음과 같은 경우 훈련 및 추론의 영향:
1) Kafka 통합 없음(데이터 스트림 없음) 2) 와
데이터 스트림 통합; 3) 그리고 데이터 스트림 모두
통합 및 컨테이너화. 훈련 응답에는 데이터 스트림 수집과 추론이 포함됩니다.

응답에는 데이터 전송 사이의 대기 시간 응답이 포함됩니다.
예측이 수신될 때까지. 훈련이 실시되었습니다
배치 크기가 10이고 이 구성이
Kafka-ML 웹 UI: “epochs=1000, epoch당 단계=22, shuffle=True, verbose=0”.

검증은 단일 Kubernetes 클러스터에서 수행되었습니다.
16GB MacBook Pro 노트북에서 실행됩니다. 훈련 및 추론에 관한 Exasens 데이터세트의 지연 응답
각각 표 I과 II에 나와 있습니다. 데이터의 경우
스트림의 경우 대기 시간 응답이 허용될 수 있습니다.
ML/AI 파이프라인에서 볼 수 있는 이점을 고려하세요. 에서
컨테이너화의 경우 지연 시간이 데이터보다 약간 높음
특히 훈련을 위한 스트림입니다. 추론의 경우 이후로 더 낮습니다.
Kafka는 Kubernetes에 배포되므로 네트워크 지연이 더 작습니다. 컨테이너화를 개선하는 방법을 연구하겠습니다.
배포 및 GPU 지원을 통한 교육.

표 I
훈련 지연 응답 (S)

정상	데이터 스트림	데이터 스트림 및 컨테이너화
27,37	29,61	31,44

Exasens 데이터 세트 및 클래식 ML을 사용하여 수행됨
모델에서는 Kafka-ML을 사용하여 ML/AI 파이프라인을 관리할 수 있습니다.
평가 및 데이터 스트림을 촉진하는 다른 ML 작업
공개적으로 사용 가능한 GitHub 소스 코드를 사용하여 관리합니다.

9https://github.com/ertis-research/kafka-ml/tree/master/examples/HCOPD Avro 형식

표 2
추론 지연 응답 (S)

정상	데이터 스트림	데이터 스트림 및 컨테이너화
0,079	0,374	0,335

VIII. 관련된 일

우리가 아는 한, Kafka-ML은 최초의 것입니다.
통합할 ML/AI 파이프라인 솔루션을 제공하는 프레임워크
머신러닝 및 데이터 스트림, TensorFlow 및 Apache
카프카. 그럼에도 불구하고 다른 접근법은 비슷한 목표를 가지고 있거나
아래 설명된 대로 Kafka-ML이 제공하는 일부 기능을 제공했습니다.

NVIDIA 딥 러닝 GPU 트레이닝 시스템(DIGITS)
[9] 훈련 및 추론을 위한 대화형 웹 UI 제공
다중 GPU 시스템의 심층 신경망(DNN).
Kafka-ML과 달리 DIGITS는 프레임워크 자체는 아니지만
NVCaffe, Torch 및 TensorFlow에 대한 래퍼입니다.
대신 해당 프레임워크에 웹 인터페이스를 제공합니다.
명령줄에서 직접 처리합니다. 주요
DIGITS의 장점은 GPU에 대한 기본 지원과
세 가지 ML 프레임워크, 사전 학습된 모델 출시 및
정확도와 손실을 실시간으로 확인할 수 있는 기능입니다.
그럼에도 불구하고 DIGITS는 학습 및 추론을 지원하지 않습니다.
데이터 스트림을 통해(대신 데이터세트를 가져와야 함)
확장을 위해 컨테이너를 통해 이러한 작업을 배포합니다.
GPU에 대한 종속성이 있으며 소스 작성이 필요할 수 있습니다.
이러한 프레임워크 위에 코드를 작성하세요.

Kubeflow[10]는 Kubernetes를 위한 강력한 ML 툴킷입니다.
Kubeflow에서 사용자는 ML/AI의 여러 단계를 구성할 수 있습니다.
하이퍼파라미터, 전처리, 트레이닝 등의 파이프라인
추론. 그러나 Kubeflow 파이프라인을 실행할 때는 다음과 같습니다.
Google Cloud Platform의 공식 사례입니다(그림 1).
10), 요구되지 않는 일부 단계가 있을 수 있습니다.
Kafka-ML 파이프라인(그림 1), 특히 다음이 필요한 파이프라인
훈련 및 추론을 위한 컨테이너를 구축합니다. 카프카-ML에서는
사용자는 교육을 위해 웹 UI와 상호작용하기만 하면 됩니다.
그리고 추론. 또한 데이터 스트림 지원은
Kubeflow ML 개발자가 수동으로 개발하고
사용자. Kafka-ML에서는 데이터 스트림 관리를 통해
Apache Kafka는 모든 파이프라인에서 지원됩니다. 어쨌든,
Kubeflow는 Kubernetes 및 ML에 대한 탁월한 지원을 제공합니다.
다중 프레임워크이며 대규모 생태계에서 지원됩니다.
범위와 기능을 훨씬 넘어서는 커뮤니티
Kafka-ML에서 제공하므로 해당 방식을 연구해 볼 가치가 있습니다.
가까운 시일 내에 두 시스템을 통합할 예정입니다.

MOA[11]는 온라인 학습 및 데이터를 위한 프레임워크입니다.
스트림 마이닝. MOA는 그래픽 인터페이스를 제공합니다.
사용자는 분류를 위한 ML 알고리즘 구현 모음을 포함하여 ML 작업을 실행하고 시각화할 수 있습니다.
회귀 및 클러스터링. Kafka-ML이지만
데이터 스트림을 지원하지만 Kafka-ML 및 TensorFlow는 좋지 않습니다.
(아직) 온라인 학습을 지원합니다. 반면 카프카ML은
TensorFlow/Keras 모델을 지원하며 대규모
커뮤니티를 통해 새로운 프레임워크를 만드는 대신
채택을 제한할 수 있는 소스 코드. Scikit-멀티플로우 [12]
온라인 학습을 위한 또 다른 프레임워크입니다.

그림 10. GCP용 Kubeflow 파이프라인 예시 출처 https://www.kubeflow. 조직/문서/gke/gcp-e2e/

널리 사용되는 프레임워크인 scikit-learn은 웹 인터페이스나 ML/AI 파이프라인에 대한 전체 제어 기능을 제공하지 않습니다.

Kafka-ML은 Apache SAMOA[13], Apache Flink[14] 및 Lambda 아키텍처[15], [16]와 같은 다른 분산 데이터 스트림 프레임워크와는 다른 접근 방식을 따릅니다. Apache SAMOA는 현재 Apache에서 인큐베이션을 진행 중이며 기본 처리 엔진(예: Apache Storm 및 Apache Samza)의 복잡성을 직접 처리하지 않고도 데이터 스트림을 통해 ML 알고리즘을 개발할 수 있도록 하는 것을 목표로 합니다. Apache Flink는 규모와 상관없이 메모리 내 속도로 데이터 스트림에 대한 계산을 수행할 수 있는 프레임워크를 제공합니다. 그리고 Lambda 아키텍처를 사용하면 실시간 및 일괄 처리 계층을 통해 대량의 데이터를 실시간으로 처리할 수 있습니다. 일반적으로 이러한 프레임워크는 데이터 스트림을 사용하여 모든 종류의 계산을 배포하기 위한 분산 엔진을 제공하지만 지원이 제한적이거나 ML/AI 파이프라인 및 TensorFlow와 같은 인기 있는 ML/AI 프레임워크를 촉진하는 데 특별히 중점을 두지 않습니다. Kafka-ML처럼 광범위한 ML/AI 솔루션 및 커뮤니티가 있습니다. 또한 Kafka-ML을 사용하면 고가용성 및 내결함성 ML/AI 파이프라인을 배포할 수도 있습니다.

마지막으로 Kafka-ML은 OpenML[17], Google Cloud AutoML[18]과 같은 AutoML 프로젝트와 어느 정도 관련이 있습니다.
OpenML은 사용자가 결과, 과학 작업, 데이터 분석 흐름 및 데이터 세트를 공개적으로 공유, 업로드 및 탐색할 수 있는 웹 플랫폼입니다. ML 모델의 결과와 메트릭은 Kafka-ML에서 구성을 사용하여 공유하고 비교할 수도 있습니다.

또한 섹션 VI에서 볼 수 있듯이 데이터 스트림을 관리하고 공유할 수도 있습니다. Google Cloud AutoML은 적은 노력과 주제에 대한 고급 지식 없이도 고품질 ML 모델을 제공합니다. 이러한 모델의 품질에 도달하는 것은 Kafka-ML의 범위를 벗어납니다. 그러나 Kafka-ML은 ML/AI 파이프라인을 시작하는 데 몇 줄의 ML 모델 소스 코드만 있으면 되는 액세스 가능하고 사용자 친화적인 플랫폼을 제공합니다. 또한 Kafka-ML은 ML/AI 전문가와 비전문가 모두가 사용할 수 있는 오픈 소스 프로젝트입니다.

IX. 결론 및 향후 연구

본 논문에서는 데이터 스트림을 통해 ML/AI 애플리케이션의 파이프라인을 관리하는 오픈 소스 프레임워크인 Kafka-ML을 제시했습니다. Kafka-ML은 널리 사용되는 데이터 스트림을 활용합니다. Apache Kafka 시스템과 Python ML 프레임워크 Tensor-Flow를 사용하여 ML/AI와 데이터 스트림을 모두 통합합니다. Kafka-ML은 단 몇 줄의 소스 코드만으로 사용자가 웹 UI에서 ML 모델을 생성하여 ML/AI 파이프라인을 제어하고 다양한 ML 모델을 평가하기 위한 구성을 생성하고 훈련, 검증할 수 있으므로 접근성과 사용이 쉬운 것이 특징입니다. 추론을 위해 훈련된 모델을 배포합니다. 또한 Kafka-ML에서 수신된 데이터 스트림을 완전히 제어하기 위해 Apache Kafka의 분산 로그를 기반으로 하는 새로운 접근 방식을 채택하여 ML/AI 애플리케이션이 이러한 데이터 스트림을 재사용하고 데이터 저장소에 대한 종속성을 제거할 수 있습니다. 파일 시스템. Kafka-ML은 완전히 컨테이너화되어 있으며 해당 구성 요소(훈련 및 추론)를 배포합니다. Docker와 Kubernetes는 각각 내결함성과 고가용성을 위해 Kafka-ML 아키텍처의 컨테이너화와 조정을 담당합니다. Kafka-ML은 데이터 스트림을 채택하는 ML/AI에 대한 전문가와 비전문가 모두가 사용하고 개선할 수 있도록 GitHub에서 공개적으로 제공됩니다.

향후 작업으로 다음과 같은 과제를 지적했습니다.
Kafka-ML의 기능 및 개선 사항:

- 분산 추론. 심층 신경망 계층은 여러 개의 독립적인 ML 모델로 분할될 수 있으며 중간 출구[19]를 통해 포그, 엣지 및 클라우드 컴퓨팅 패러다임에서 실행을 최적화할 수 있습니다. 목표는 Kafka-ML에서 ML 모델을 훈련하고 분할하여 나중에 IoT-클라우드 연속체에 배포하는 것입니다. 레이어 간 전체 데이터 흐름을 지원하는 새로운 아키텍처도 필요합니다.

- 분산 교육. 현재 훈련은 단일 컨테이너에서 수행되므로 대규모 신경망에는 충분하지 않을 수 있습니다. 이와 관련하여 Kubeflow 및 GPU 지원과 같은 Kubernetes의 분산 교육을 위한 다른 접근 방식을 살펴봐야 합니다. • 다른 ML 프레임워크를 지원합니다. 이는 TensorFlow가 TensorFlow/IO에서 했던 것처럼 Apache Kafka를 활성화하는 다른 ML 프레임워크의 개발에 따라
달라집니다.

어떤 경우든 다른 ML 프레임워크에 대한 새로운 데이터 스트림 커넥터를 탐색할 수 있습니다.

- IoT 및 ML/AI. IoT는 온디바이스 추론을 위한 uTen-sor10 및 TensorFlow lite11 과 같은 이니셔티브에서 입증된 것처럼 ML/AI 파이프라인에서 발생하고 있습니다 . IoT 장치용 ML 모델 생성과 Kafka-ML에서의 설치까지 ML/AI 파이프라인을 IoT까지 확장할 수 있습니다. • 다른 처리 작업을 통합합니다. 마지막으로 구조적 상태 모니터링과 같은 많은 애플리케이션에서는 ML/
AI를 사용할 수 있지만 동일한 데이터 스트림이 필요할 수 있는 기타 통계 및 처리 작업도 사용할 수 있습니다. 따라서 Kafka-ML은 이러한 비ML/AI 작업을 관리하여 사용되는 데이터 스트림과 통합할 수도 있습니다.

감사의 말

이 작업은 스페인 프로젝트 RT2018-의 자금 지원을 받습니다.
099777-B-100(“rFOG: 대기 시간 및 안정성 개선

10https://github.com/uTensor/uTensor
11https://www.tensorflow.org/lite

중요 서비스를 위해 계산을 FOG로 오프로드") 및 UMA18FEDERJA-215("Fog의 딥 러닝 서비스 기반 고급 모니터링 시스템"). Cristian Mart' n은 스페인 프로젝트 TIC-1572("MIsTIca: 무선 기술 기반 중요 인프라 모니터링")에서 박사후 연구원 보조금을 받고 있으며 IHP에서의 연구 기간은 University of Malaga 및 IHP 자금. Kafka 및 TensorFlow와의 결합에 대한 수많은 기사와 GitHub 저장소를 통해 영감과 아이디어를 주신 Kai Whner에게 익명으로 감사의 말씀을 전하고 싶습니다.

참고자료

[1] Y. Lu, "인공 지능: 진화, 모델, 응용 프로그램 및 미래 동향에 대한 조사", Journal of Management Analytics, vol. 6, 아니. 1, 2019년 1~29페이지.

[2] M. D' az, C. Mart' n 및 B. Rubio, "사물 인터넷과 클라우드 컴퓨팅의 통합에 대한 최첨단, 과제 및 공개 문제"
네트워크 및 컴퓨터 응용 저널, vol. 67, pp. 99–117, 2016.

[3] “한 눈에 보는 사물 인터넷”, 온라인 이용 가능: https://emarsonindia. com/wp-content/uploads/2020/02/Internet-of-Things.pdf, (2020년 5월 15일 액세스)

[4] “Apache kafka,” 온라인 이용 가능: http://kafka.apache.org/(2020 년 5월 13일 액세스).

[5] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard 외, “Tensorflow: A 대규모 기계 학습을 위한 시스템”, 제12회 {USENIX} 운영 체제 설계 및 구현에 관한 심포지엄({OSDI} 16), 2016년, 265-283페이지.

[6] “Kubernetes”, 온라인 이용 가능: https://kubernetes.io/, (15일 접속) 2020년 5월).

[7] PSZNR . C. Wenger, "Exasens: copd 환자의 타액 샘플 분류를 위한 새로운 데이터 세트", 2020. [온라인].
이용 가능: http://dx.doi.org/10.21227/7t0z-pd65 [8]
P. Soltani Zarrin, F. Ibne Jamal, N. Roeckendorf 및 C. Wenger, "신속한 감지를 위한 휴대용 유전체 바이오센서 개발" Copd 환자의 타액 샘플과 건강한 대조군을 사용한 점도 변화 및 시험관 내 평가,” Healthcare, vol. 7, 아니. 1, p. 2019년 11월 11일

[9] L. Yeager, J. Bernauer, A. Gray 및 M. Houston, "Digits: 딥 러닝 GPU 교육 시스템", ICML 2015 AutoML 워크샵, 2015년 7월 11일, 프랑스 릴.

[10] 'Kubeflow' 온라인 이용 가능: https://www.kubeflow.org/(2020 년 5월 13일 액세스).

[11] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, "MOA: 대규모 온라인 분석", J. Mach. 배우다. Res., vol. 11, pp. 1601–1604, 2010.
[온라인]. 사용 가능: http://portal.acm.org/citation.cfm?id=1859903 [12]
J. Montiel, J. Read, A. Bifet 및 T. Abdessalem, “Scikit-multiflow: 다중 출력 스트리밍 프레임워크 ,” 기계 학습 연구 저널, vol. 19, 아니. 72, pp. 1–5, 2018. [온라인]. 이용 가능: http://jmlr.org/papers/v19/18-251.html

[13] "Apache samoa" 온라인 이용 가능: https://samoa.incubator.apache.org/(2020 년 5월 13일 접속)

[14] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi 및 K. Tzoumas, "Apache flink: 단일 엔진의 스트림 및 일괄 처리", IEEE 컴퓨터 협회 기술 위원회 게시판 데이터 엔지니어링, vol. 36, 아니. 2015년 4월 4일.

[15] M. D' az, C. Mart' n 및 B. Rubio, "λ-coap: 람다 아키텍처 및 coap에 기반한 사물 인터넷 및 클라우드 컴퓨팅 통합", 공동 컴퓨팅: 네트워킹, 애플리케이션, 작업 공유: 제11회 국제 컨퍼런스, CollaborateCom 2015, 우한, 2015년 11월 10~11일, 중국. 절차, S. Guo, X. Liao, F. Liu 및 Y. Zhu, Eds. Cham: Springer International Publishing, 2016, pp. 195–206.

[16] S. Ge, H. Isah, F. Zulkernine 및 S. Khan, 2019년 IEEE 43차 연례 컴퓨터 소프트웨어 및 애플리케이션 컨퍼런스(COMPSAC)에서 "딥 러닝을 사용한 다단계 스트리밍 데이터 분석을 위한 확장 가능한 프레임워크", 7월 15일 -19, 미국 위스콘신주 밀워키, vol. 2. IEEE, 2019, 189-194페이지.

[17] "Openml: 함께하는 기계 학습", 온라인 이용 가능: https://www. openml.org/(2020 년 5월 13일 액세스).

[18] 'Google 클라우드 automl', 온라인 이용 가능: https://cloud.google.com/ automl (2020년 5월 13일에 액세스함).

[19] S. Teerapittayanon, B. McDanel 및 H.-T. Kung, "클라우드, 에지 및 최종 장치를 통한 분산 심층 신경망", 2017 IEEE 37차 분산 컴퓨팅 시스템(ICDCS) 국제 회의, 6월 58일, 미국 조지아주 애틀랜타. IEEE, 2017, pp. 328-339.