JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 11, pp. 19-27, November 2022
https://doi.org/10.9708/jksci.2022.27.11.019

MLOps workflow language and platform for time series data anomaly detection

Jung-Mo Sohn*, Su-Min Kim*

*Researcher, Epozen's research institute, Seoul, Korea
*Researcher, Epozen's research institute, Seoul, Korea

[Abstract]

In this study, we propose a language and platform to describe and manage the MLOps(Machine Learning Operations) workflow for time series data anomaly detection. Time series data is collected in many fields, such as IoT sensors, system performance indicators, and user access. In addition, it is used in many applications such as system monitoring and anomaly detection. In order to perform prediction and anomaly detection of time series data, the MLOps platform that can quickly and flexibly apply the analyzed model to the production environment is required. Thus, we developed Python-based AI/ML Modeling Language (AMML) to easily configure and execute MLOps workflows. Python is widely used in data analysis. The proposed MLOps platform can extract and preprocess time series data from various data sources (R-DB, NoSql DB, Log File, etc.) using AMML and predict it through a deep learning model. To verify the applicability of AMML, the workflow for generating a transformer oil temperature prediction deep learning model was configured with AMML and it was confirmed that the training was performed normally.

▸ **Key words**: AI, MLOps, Workflow, Time series data, Anomaly detection

[요 약]

본 연구에서는 시계열 데이터 이상 탐지 수행을 위한 MLOps(Machine Learning Operations) 워크플로를 기술하고 관리할 수 있는 언어와 플랫폼을 제안한다. 시계열 데이터는 IoT 센서, 시스템 성능 지표, 사용자 접속량 등 많은 분야에서 수집되고 있다. 또한, 시스템 모니터링 및 이상 탐지 등 많은 응용 분야에 활용 중이다. 시계열 데이터의 예측 및 이상 탐지를 수행하기 위해서는 분석된 모델을 빠르고 유연하게 운영 환경에 적용할 수 있는 MLOps 플랫폼이 필요하다. 이에, 최근 데이터 분석에 많이 활용되고 있는 Python 기반의 AMML(AI/ML Modeling Language)을 개발하여 손쉽게 MLOps 워크플로를 구성하고 실행할 수 있도록 제안한다. 제안하는 AI MLOps 플랫폼은 AMML을 이용하여 다양한 데이터 소스(R-DB, NoSql DB, Log File 등)에서 시계열 데이터를 추출, 전처리 및 예측을 수행할 수 있다. AMML의 적용 가능성을 검증하기 위해, 변압기 오일 온도 예측 딥러닝 모델을 생성하는 워크플로를 AMML로 구성하고 학습이 정상적으로 수행됨을 확인하였다.

▸ **주제어**: AI, MLOps, 워크플로, 시계열 데이터, 이상탐지

• First Author: Jung-Mo Sohn, Corresponding Author: Jung-Mo Sohn
*Jung-Mo Sohn (jmsohn@epozen.com), Epozen's research institute
*Su-Min Kim (ksum_3@epozen.com), Epozen's research institute
• Received: 2022. 09. 22, Revised: 2022. 11. 16, Accepted: 2022. 11. 16.

I. Introduction

기계 학습은 인공지능의 한 분야로서, 컴퓨터가 학습할 수 있도록 알고리즘과 기술을 개발하는 분야이다[1]. 즉, 학습된 모델 등을 통해 가치 있는 패턴이나 예측 등의 결과물을 얻는 것이다. 기계 학습은 학습 방법에 따라 지도학습(supervised learning)과 비지도 학습(unsupervised learning)으로 구분된다.

지도학습은 입력 데이터와 정답 라벨이 있는 데이터에 대해 모델을 학습하는 것이다. 대표적 알고리즘으로 SVM(support vector machine), 의사 결정 트리, 딥러닝 알고리즘 등이 있다. 반면 비지도 학습은 입력 데이터만 있는 경우에 학습하는 것이다. 대표적으로는 군집화 계열의 알고리즘이 존재하며 k-평균군집(k-means), DBSCAN(density-based spatial clustering of applications with noise)등이 있다[2].

기계 학습 모델의 적절한 운영을 위해서는 학습을 위한 자동화된 데이터의 확보 방안과 학습된 모델을 운영 환경에 지속해서 적용/관리하는 것이 중요하다. 최초 수집된 데이터로 모델을 만들 수는 있지만, 운영 환경 적용 후 점진적으로 데이터가 변화하는 Data drift 상황이 발생할 수 있기 때문이다[3]. 이는 모델의 정확도를 낮아지게 만들며, 데이터 변화에 유연하게 적응할 수 없도록 한다. 따라서 변화하는 데이터를 운영 환경에서 지속적으로 수집 관리하고 모델을 최신화해야 할 필요성이 있다. 특히, 시스템 모니터링 시, 이상 상황의 사전 징후 변화 확인 및 추후 모델 학습을 위해 데이터를 저장 및 관리해야 한다. 일반적으로 시스템 이상 발생 시, 담당자는 이상 상황 해결에 주안점을 두기 때문에, 데이터 확보 및 관리에 어려움이 있다. 이를 해결하기 위해서는 이상 데이터만 따로 저장 및 관리할 수 있어야 하며, 이상 발생 시 분석에 필요한 데이터를 자동으로 저장할 수 있어야 한다. 또한, 이렇게 확보된 데이터를 활용하여 학습된 모델을 운영 환경에 빠르게 적용할 수 있어야 한다.

Makinen et al. (2021)의 설문 조사에 따르면 기계 학습(machine learning, ML) 도메인에서 종사하는 전문가 중 40%의 응답자가 ML 모델과 운영을 위한 인프라를 모두 사용한다고 응답하였다. 또한, 분석에 사용되는 데이터는 주로 관계형 데이터와 시계열 데이터가 가장 많은 것으로 조사되었다[4]. 이는 관계형 및 시계열 운영 환경 데이터의 저장 관리 및 지속적인 모델 학습/적용이 필요함을 시사한다.

시계열 데이터는 데이터가 시간적인 움직임을 포함하고 있을 때, 즉 어떤 기간에 걸쳐서 취해진 데이터라고 할 수 있다. 이러한 시계열 데이터를 예측하는 것은 전례 없는 변동과 불완전한 정보로 인해 어려움이 있다[5]. 예를 들어, 주가 정보와 가격 예측 등은 외부의 정치 및 경제적 요소 등 예측하기 어려운 외란에 의해 영향을 받기 때문에 예측이 어려운 측면이 있다. 시계열 분석 기법으로는 통계적 기법과 딥러닝을 활용한 기법들이 있다. 통계적 기법으로는 회귀 분석, ARIMA(autoregressive integrated moving average) 기법 등이 있으며 딥러닝 기법으로는 LSTM(long short-term memory) 등 다양한 기법이 사용되고 있다.

본 논문에서는 시계열 데이터 분석을 위한 MLOps 플랫폼을 제안한다. 세부적으로는 자동화된 운영 데이터 저장과 신속한 모델을 전개를 위한 Python 기반의 워크플로 기술 언어 AMML(AI/ML modeling language)과 실행 엔진, 모델 관리 방안에 대해 제안한다. 또한, AMML을 사용하여 시계열 데이터 셋인 변압기 오일 온도 데이터 셋을 LSTM 모델로 학습 및 생성하고 예측하는 과정을 통해 검증을 수행하였다. 2장에서는 관련 연구를 소개한다. 3장에서는 제안하는 MLOps(machine learning operations) 플랫폼에 대한 세부 사항과 LSTM 모델을 사용한 사례를 기술한다. 4장 MLOps 플랫폼에 대한 결론과 추후 연구 방향을 제시한다.

II. Preliminaries

1. Related works

1.1 MLOps

MLOps는 운영 환경에서 기계 학습 모델을 배포하기 위한 기술 및 도구들의 모음으로 DevOps와 기계 학습의 조합이라 할 수 있다. DevOps는 소프트웨어 개발과 운영의 차이를 줄이는 것을 주요 목적으로 하고 있으며 두 가지 주 원칙으로 CI(continuous integration)/CD (continuous deploy)가 있다[6]. CI는 개발조직이 빈번하게 통합하여 개발하는 방식이다. 이를 통해 지속적으로 테스트를 수행하고 오류를 조금씩 개선해 나갈 수 있으며 결과적으로 소프트웨어의 개발 기간을 단축할 수 있다. CD는 운영 환경 적용 및 테스트를 위한 새로운 버전이 지속적으로 생성되는 방식이다. 이러한 방식은 새로운 기능과 지속적인 통합(CI)으로 고객에게 더욱 빠른 서비스 도달을 제공한다.

MLOps는 이러한 DevOps의 CI/CD 기능에 지속적인 모델 재학습이라는 CT(continuous training) 개념을 추가한 것이다[6]. CT는 새로운 데이터를 사용하여 주기적으

로 모델을 재학습하는 것으로 새로운 모델의 품질 평가를 포함하는 개념이다[7].

MLOps의 자동화 수준에 따라 성숙도 수준을 나타낼 수 있다. MLOps 성숙도 수준에 대한 표준은 현재 없지만, 참조할만한 주요 성숙도 모델로서, 구글과 마이크로소프트에서 정의한 MLOps 성숙도 수준이 있다[6]. 구글의 성숙도 모델은 Fig. 1과 같이 총 3단계로 수준 0: 수동 프로세스(Manual process), 수준 1: ML 파이프라인 자동화(ML pipeline automation), 수준 2: CI/CD 파이프라인 자동화(CI/CD pipeline automation) 단계로 구분된다[8]. 마이크로소프트의 성숙도 단계는 Fig. 2와 같이 총 5단계로 구성된다. 각 수준은 수준 0: MLOps 없음(No MLOps), 수준 1: DevOps는 있지만 MLOps 없음(DevOps but no MLOps), 수준 2: 자동화된 학습(Automated Training), 수준 3: 자동화된 모델 배포(Automated Model Deployment), 수준 4: 전체 MLOps 자동화된 작업(Full MLOps Automated Operations)으로 구성된다[9].

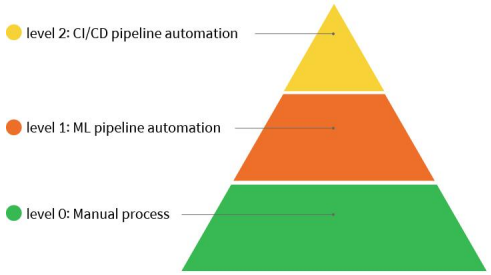

Fig. 1. Google's MLOps Maturity Level

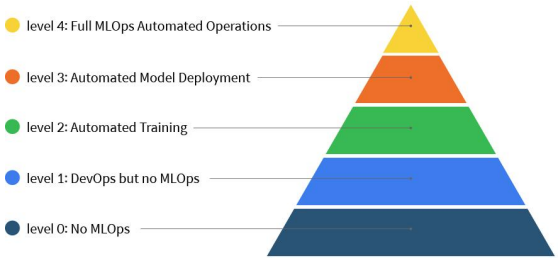

Fig. 2. Microsoft's MLOps Maturity Level

MLOps는 다양한 오픈소스 및 기업에서 솔루션이 개발되고 있다. 오픈소스 솔루션으로는 MLflow와 Kubeflow 등이 활발히 개발되고 있다. MLflow는 python으로 개발되었다. 주요 구성요소로는 Trackings, Projects, Models, Model Registry로 구성되어 있다. Tracking은 기계 학습 실험의 매개변수, 코드 및 결과를 기록하는 API와 UI이다. Projects는 재사용 가능한 코드를 패키징하기 위한 표준 형식이다. Models는 기계 학습 모델을 여러 가지 방식으로 패키징하기 위한 규칙과 도구를 제공한다. Model Registry는 MLflow 모델의 전체 수명 주기를 공동으로 관리하기 위한 중앙 집중식 모델 저장소, API 및 UI를 제공한다[10]. Kubeflow는 Kubernetes에서 기계 학습 워크플로를 이식 가능하며 확장할 수 있게 배포한다. 주요 구성요소로는 Notebooks, TensorFlow model training, Model serving, Katib 가 있다. Notebooks는 주피터 노트북을 생성하고 관리할 수 있다, TensorFlow model training은 딥러닝 모델을 학습할 수 있다. Model serving은 KFServing과 Seldon Core를 사용하여 모델을 서빙한다. Pipelines는 컨테이너를 기반으로 확장 가능한 end-to-end 기계 학습 워크플로를 구축한다. Katib는 하이퍼 파라미터를 튜닝할 수 있게 하고, 신경망 아키텍처 탐색 기능을 제공한다[11]. 상용솔루션으로는 마이크로소프트의 Azure Machine Learning, AWS의 SageMaker 및 구글의 Vertex AI 등이 있다. 이들 서비스는 클라우드 상에서 모델 개발 및 운영을 할 수 있게 한다. 주피터 노트북을 이용하여 데이터 분석 환경을 지원하며, UI를 통한 워크플로 설정과 모델 관리 기능을 제공하고 있다. 하지만, UI로 만들어진 워크플로를 이해관계자와 공유하고 재활용하기에 어려운 점이 있다.

1.2 ML-Schema

ML-Schema는 W3C(world wide web consortium) Machine learning schema community group에 의해 제안된 것으로, 기계 학습 개발자 간의 상호운용성을 제공하는 것이 목적이다. 이를 위해 기계 학습 알고리즘, 데이터 셋 및 테스트에 대한 정보를 표현하고 교환하는 데 사용할 수 있는 클래스, 속성 및 제한 집합을 제공하는 공유 스키마를 제시하고 있다[12].

Mun et al. (2021)에서는 ML-Schema와 PROV-ML기반으로 딥러닝 모델을 정의하고 학습할 수 있는 언어인 DL2IA(deep learning description language for image analysis)을 제안하였다[13]. DL2IA는 XML로 만들어진 언어이며 번호판 인식을 위한 딥러닝 모델을 표현 및 사용하여 학습과 검증을 수행할 수 있음을 보였다. 하지만 XML은 인간이 쉽게 이해하고 사용하는 어려운 점이 있으며 표현이 길어지는 단점이 있다. 또한, XML에 워크플로와 모델 정보가 모두 포함되어 있어 이해가 어렵다. 이에 본 연구에서는 간략화된 AMML을 제안한다. AMML은 간략한 워크플로와 모델 표현 방식을 제공하여, 이해당사자 간의 공유를 편리하게 한다.

1.3 LSTM

LSTM은 딥러닝 분야에서 시계열 데이터 분석에서 주로 사용된다. LSTM은 RNN(recurrent neural network)을 개선한 모델이다. RNN은 기존 신경망의 세포가 상태를 저장하지 않는 것과는 다르게 세포가 상태를 저장하는 것이 특징이다. 또한, 이전 시간의 상태를 입력으로 사용한다. 모델은 BPTT(backpropagation through time) 알고리즘을 통해 수행한다. BPTT 알고리즘은 기존의 backpropagation 알고리즘을 RNN에서 사용할 수 있도록 수정한 학습 알고리즘이다. 하지만, RNN은 기존 신경망 알고리즘의 문제점이었던 시간이 지날수록 영향도가 폭발하거나 줄어드는 문제가 발생한다[14]. LSTM은 이 문제를 해결하기 위해 Forget gate, Input gate, Output gate를 사용하여 모델의 성능을 향상시킨다. Forget gate는 이전 시간의 입력에 대한 처리를 수행하고, Input gate는 현재 입력을 처리하며, Output gate는 모델의 출력에 대한 처리를 수행한다[14][15].

1.4 YAML(YAML ain't markup language)

XML은 컴퓨터를 통해 파싱이 쉽고 구조화되어 있는 장점이 있지만, 사람보다는 컴퓨터가 이해하기 쉬운 형태로 되어 있고, 작성 시 각 시작 태그와 종료 태그를 작성하여야 해 사용이 어려운 측면이 있다. YAML은 사람이 이해하기 쉬운 형태의 언어를 지향하고 있다[16]. 키-값 형태 및 목록 형태의 다양한 데이터를 YAML로 손쉽게 표현할 수 있다. Fig. 3은 구매 목록을 YAML(왼쪽)과 XML(오른쪽)로 데이터를 표현한 것으로, XML에 비해 YAML이 간결하고 이해하기 쉽게 표현되는 것을 확인할 수 있다.

```
# Products purchased
- item : Super Hoop
  quantity: 1
- item : Basketball
  quantity: 4
- item : Big Shoes
  quantity: 1
```

```
<!-- Products purchased -->
<items>
  <item>
    <name>Super Hoop</name>
    <quantity>1</quantity>
  </item>
  <item>
    <name>Basketball</name>
    <quantity>4</quantity>
  </item>
  <item>
    <name>Big Shoes</name>
    <quantity>1</quantity>
  </item>
</items>
```

Fig. 3. Products purchased list using YAML and XML

III. The Proposed Scheme

1. AMML for workflow

AMML은 MLOps의 워크플로를 유연하게 구성하기 위해 워크플로 구성언어와 딥러닝 모델을 정의하는 언어로 나뉜다. 워크플로 구성언어는 리눅스 커맨드 라인과 유사하게 구성할 수 있도록 하였으며, 모델 정의 언어는 YAML을 사용하였다.

AMML 워크플로 구성언어는 컴포넌트 간의 연결 관계를 파이프('|')를 이용하여 표현하였다. 즉, 파이프 이전 컴포넌트의 출력이 이후 컴포넌트의 입력이 되는 방식이다. Fig. 4는 워크플로 구성언어의 예이다. 3개의 normalsource로부터 데이터를 입력받아 평균과 표준편차를 출력하는 워크플로 구성의 예이다. normalsource는 설정된 평균과 표준편차로 정규분포를 가지는 데이터를 생성하는 컴포넌트이다. 워크플로를 다양한 형태로 구성할 수 있도록 여러 플로를 합칠 수 있는 'join' 컴포넌트와 하나의 플로를 여러 플로로 나눌 수 있는 'split' 컴포넌트를 지원하도록 설계하였다.

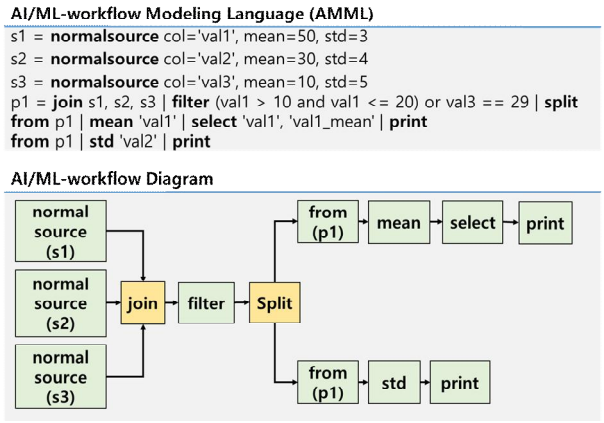

Fig. 4. Example of AMML and Workflow Diagram

컴포넌트는 컴포넌트 명과 콜백 스크립트로 구성되어 있다. 컴포넌트 명은 데이터를 처리할 컴포넌트에 대한 명칭이다. 콜백 스크립트는 컴포넌트가 데이터를 처리할 때 호출되는 스크립트이다. 예를 들어, 'filter cond = col1 > 1'으로 정의되어 있으면, filter 컴포넌트는 실행시 데이터를 하나씩 수신받아 col1의 값이 1 보다 큰 경우에만 다음 파이프라인으로 데이터를 전송한다. 이때, filter가 컴포넌트 명이며 'cond = col1 > 1'은 콜백 스크립트이다. 이러한 워크플로 구성언어는 워크플로 구성을 위한 바이트 코드와 콜백 스크립트 바이트 코드로 컴파일된다. 워크플로 컴포넌트에는 콜백 스크립트 바이트 코드가 설정되어 실

행 시 수행한다. Fig. 5는 워크플로 컴파일 수행 예이다. ①은 워크플로의 filter 컴포넌트의 생성 명령어이며, ②는 워크플로의 filter 컴포넌트의 조건인 'cond = col1 > 1' 스크립트의 컴파일된 바이트 코드이다. 콜백 스크립트는 사칙 연산 및 비교, boolean 연산 등 기본적인 연산을 지원하며, built-in으로 제공하는 메소드 호출을 지원한다.

```
normalsource col='val1', mean=0, std=1 |
filter cond=val1 > 1 | print
```

```
WF_MAKE_COMPONENT normalsource
WF_DEF_CALLBACK
CONST_DEF_START
0          4          sval1
CONST_DEF_END
INSTRUCTION_START
LOAD_CONST 0
ASSIGN col
INSTRUCTION_END
WF_ADD_CALLBACK
...
WF_MAKE_COMPONENT filter
WF_DEF_CALLBACK
CONST_DEF_START
0          1          i1
CONST_DEF_END
INSTRUCTION_START
LOAD_VALUE val1
LOAD_CONST 0
CMP_GT
ASSIGN cond
INSTRUCTION_END
...
WF_MAKE_COMPONENT print
WF_ADD_COMPONENT
WF_ASSIGN
WF_END
```

Fig. 5. Workflow and Callback Script bytecode

Fig. 6은 컴포넌트의 상속 구조와 각 컴포넌트를 표현한 것이다. 컴포넌트는 AbstractComponent를 최상위 추상 클래스로 하여 AbstractIngress, AbstractProgress, AbstractEgress로 분류하였다. AbstractIngress는 데이터 소스로부터 데이터를 가져와서 다음 컴포넌트로 전달하는 역할을 한다. 새로운 형태의 데이터 소스를 만들기 위해서는 AbstractIngress를 상속받아 구현할 수 있다. AbstractProgress는 이전 컴포넌트로부터 데이터를 받아 처리를 수행하고 처리된 데이터를 다음 컴포넌트로 전달하는 역할을 한다. 예를 들어, filter, minmax scaler, zscore scaler, LSTM 등 데이터 필터링 및 변환 작업들과 예측 및 학습을 수행하는 컴포넌트이다. AbstractEgresss는 이전 컴포넌트에서 입력받은 데이터를 처리하고 종료하는 컴포넌트이다. print, alert 등의 컴포넌트가 여기에 속한다. 즉, 다음 컴포넌트로 전달할 데이터가 없는 경우 사용되며 보통 워크플로의 마지막 컴포넌트로 사용된다. 각 컴포넌트는 multiprocess 형태로 독립적으로 구동된다. 데이터의 수발신은 프로세스 사이의 queue를 통해 이루어진다.

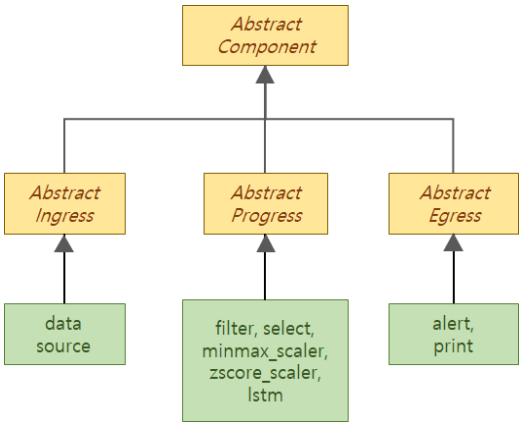

Fig. 6. Component Class Diagram

각 컴포넌트는 데이터 소스로부터 사전(dictionary) 형태의 데이터를 수신받아 한 건씩 처리하도록 하였다. 사전 형태의 데이터는 컬럼명과 데이터가 key-value 형식으로 되어 있는 데이터 형태로 python에서 기본적으로 지원하는 데이터 타입이다. 데이터 소스는 사전 형태로 표현할 수 있는 것이면 RDB 뿐만 아니라 NoSql 및 로그 파일 등 다양한 데이터 소스를 추가로 구현하여 사용할 수 있다.

워크플로 구성언어는 모델의 학습 및 적용을 위한 워크플로를 구성하며, 데이터 필터링, 컬럼 추출 등 다양한 전처리 작업(minmax, z-score scaler등) 및 이상 발생 시 알람 발생, 스냅샷 생성 등을 제공하여 프로세스를 다양하게 구성할 수 있도록 한다.

이 중 스냅샷 생성 컴포넌트는 내부의 버퍼에 입력된 데이터를 저장하고 있다. 다른 컴포넌트에서 스냅샷 저장 이벤트가 발생하면 데이터를 저장하는 방식이다. 주로 운영 환경에서 이상 현상 발생시 버퍼의 데이터를 데이터베이스에 즉시 저장하여 추후 분석이나 모델 재학습에 활용할 수 있도록 한다. Fig. 7은 Alert 발생시 데이터를 저장하는 방식을 표시한 것이다. ①은 감시 컴포넌트(surveillance component)로부터 Alert이 발생한 것이다. ②는 alert 정보를 저장하는 과정이다. ③에서는 Snapshot 컴포넌트에 alert 발생 event를 전송한다. ④는 Snapshot 컴포넌트의 버퍼 내의 데이터를 저장하는 과정이다.

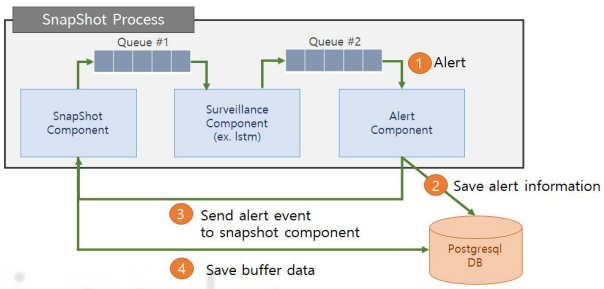

Fig. 7. Process of snapshot event

2. AMML for model

딥러닝 모델은 YAML을 사용하여 정의한다. 모델의 정의 구조는 모델과 학습으로 구분된다. dataset 및 전처리 과정 등은 워크플로를 통해 구성되기 때문에 따로 정의하지 않는다. Fig. 8은 일반적인 DNN(deep neural network)을 YAML로 표시한 것이다. 총 6개의 입력 필드에서 데이터를 받아 1개의 출력을 발생하는 구조이다.

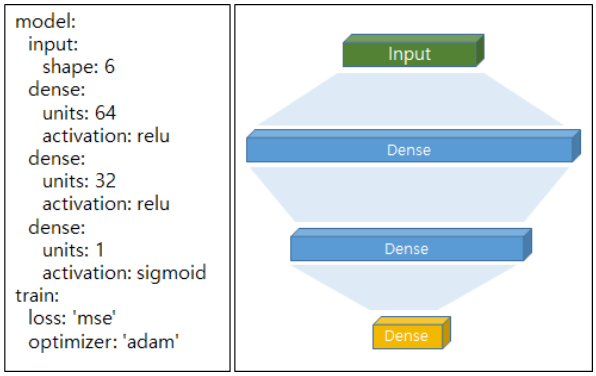

Fig. 8. Example of AMML for model

모델은 딥러닝 모델의 layer를 설정하여 모델의 구조를 정의한다. 각 layer는 활성화 함수 및 unit 개수 등 파라미터를 설정하고 조절할 수 있다. 본 연구에서는 시계열 데이터를 분석하기 위한 Input, LSTM 및 Dense 컴포넌트를 사용하여 모델을 구성하였다. 학습은 모델 훈련시 필요한 loss 함수 및 optimizer 등 파라미터를 설정하여 구성할 수 있다. 모델 학습 컴포넌트는 모델과 학습 정보를 이용하여 Keras 라이브러리를 사용하여 모델을 구성하고 학습시킨다. 사용된 Keras는 2.9.0 버전을 사용하였다.

정의된 딥러닝 모델의 YAML은 모델명과 함께 데이터베이스에 저장된다. LSTM 학습 컴포넌트에서는 설정된 모델명을 참조하여 모델을 데이터베이스에 YAML을 적재한다. 로드된 YAML을 사용하여 모델의 구조를 생성한 후 데이터 소스로부터 입력된 데이터를 수신받아 학습을 수행한다.

3. Dataset

시계열 모델 테스트 및 검증을 위해 ETDataset (electricity transformer dataset)을 활용하였다[17][18]. ETDataset은 변압기 오일의 온도를 예측하여 변압기 예방보존에 활용하기 위한 데이터 셋으로 2년간 실제 플랫폼에서 데이터를 수집한 것이다. 본 실험에서는 ETT(electricity transformer temperature)-Small dataset 중 시간 단위로 측정된 ETT-h1 dataset을 사용하여 학습하였다. 데이터 셋의 구성은 날짜와 오일 온도 포함하여 총 8개 컬럼으로 구성된다. Fig. 9는 ETT-h1의 컬럼 구조와 샘플 데이터를 표시한 것이다.

date	HUFL	HULL	MUFL	MULL	LUFL	LULL	OT
2016-07-01 00:00:00	5.827	2.009	1.599	0.462	4.203	1.340	30.531000
2016-07-01 01:00:00	5.693	2.076	1.492	0.426	4.142	1.371	27.787001
2016-07-01 02:00:00	5.157	1.741	1.279	0.355	3.777	1.218	27.787001
2016-07-01 03:00:00	5.090	1.942	1.279	0.391	3.807	1.279	25.044001
2016-07-01 04:00:00	5.358	1.942	1.492	0.462	3.868	1.279	21.948000
2016-07-01 05:00:00	5.626	2.143	1.528	0.533	4.051	1.371	21.174000
2016-07-01 06:00:00	7.167	2.947	2.132	0.782	5.026	1.858	22.792000

Field	date	HUFL	HULL	MUFL	MULL	LUFL	LULL	OT
Description	The Recorded date	High UseFul Load	High UseLess Load	Middle UseFul Load	Middle UseLess Load	Low UseFul Load	Low UseLess Load	Oil Temperature (target)
Type	datetime	float						

Fig. 9. Structure and sample of ETT-h1

4. AMML model for ETDataset analysis

AMML을 이용한 ETDataset 분석을 위한 워크플로는 학습 워크플로와 예측 워크플로로 구성하였다. Fig. 10은 학습 워크플로의 구조를 표현하고 있다. 학습 워크플로는 학습용 데이터 소스에서 ETT-h1의 데이터를 로드하여, 전처리 작업인 min-max 스케일 작업을 수행한다. min-max scaler는 컬럼의 min, max 값을 사용하여 데이터를 0과 1사이의 값으로 변환한다. 전처리 작업이 완료된 데이터를 LSTM 모델 학습 컴포넌트에 입력하여 모델을 학습 및 생성한다. 모델은 시계열 데이터를 분석에서 자주 사용되는 LSTM을 사용하여 구성하였다.

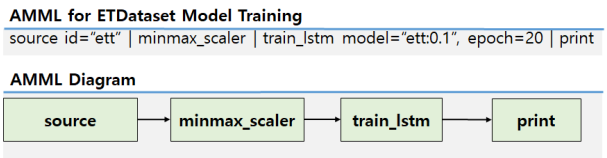

Fig. 10. Workflow for model training

Fig. 11은 모델의 구조와 학습 관련 YAML 설정을 보여준다. LSTM의 activation 함수는 'tanh'로 설정하였다. LSTM 모델 학습 시, loss 함수로는 'MAE(mean absolute error)'를 사용하였으며, optimizer는 'adam'을 사용하였다. 학습은 총 20 epoch로 수행하였다. LSTM 모델 학습 컴포넌트는 epoch별 오차(loss)를 출력으로 하며, 다음 컴포넌트로 전달한다.

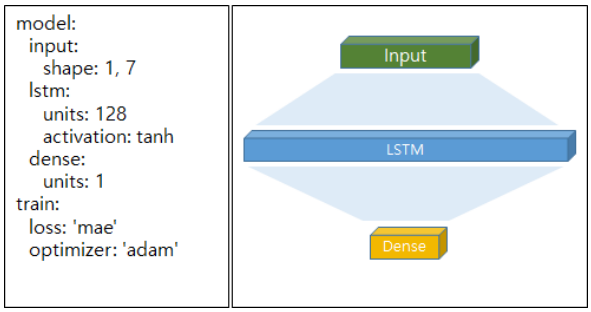

Fig. 11. Model for ETT-h1 analysis

학습된 모델 파일은 하둡 파일 시스템(HDFS, hadoop file system)에 저장하고, 모델 메타 정보(모델명, 모델의 하둡 저장위치)는 데이터베이스(PostgreSQL)에 저장하여 추후 변압기 온도 예측 시 사용할 수 있도록 한다. Fig. 12는 학습된 모델의 저장 과정을 보여준다. ① LSTM training 컴포넌트에서 모델 학습이 종료되면 모델과 메타데이터를 등록하기 위해 Model Manager에게 요청한다. ② Model Manager는 메타데이터를 PostgreSQL 데이터베이스에 저장한다. 이는 학습된 모델을 검색하여 사용할 수 있도록 해준다. ③ 메타데이터 저장이 완료되면 모델을 HDFS 저장하여 완료한다.

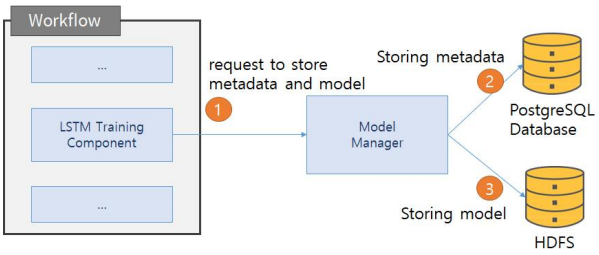

Fig. 12. Process of model storing

예측 워크플로는 Fig. 13과 같이 구성하였다. 학습 워크플로와 유사하게 데이터 소스로부터 ETT-h1 데이터를 로드하여 min-max 스케일 작업을 수행한다. 이후 LSTM 컴포넌트를 이용하여 예측을 수행한다. LSTM 컴포넌트는 모델이 저장된 Model Manager에게 모델을 요청하여 로딩한다. 예측된 값에 대해 minmax_inverse 컴포넌트를 통해 원래 스케일로 변경 후 예측값을 출력한다.

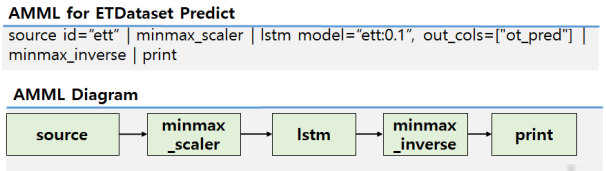

Fig. 13. Workflow for ETT-h1 predict

Fig. 14은 모델의 로딩 과정을 보여준다. ① Model Manager에 모델을 요청한다. ② Model Manager는 메타데이터(HDFS의 위치 정보)를 가져온다. ③ HDFS에서 모델을 가져온다. ④ LSTM 컴포넌트로 모델을 보낸다. 최종적으로 LSTM 컴포넌트는 로딩된 모델을 사용하여 입력 데이터에 대해 예측을 수행하고 결과를 출력한다.

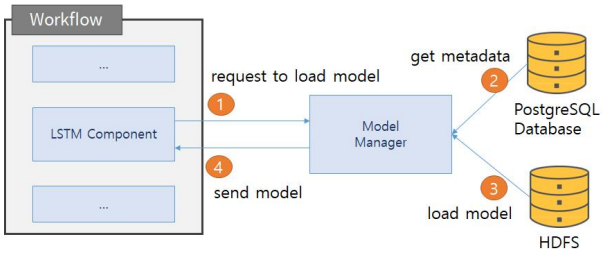

Fig. 14. Process of model loading

5. LSTM model training and results

AMML로 만들어진 학습 워크플로를 수행하였다. Fig. 15는 모델 학습을 위한 워크플로를 등록하고 수행하는 화면이다. 워크플로는 Fig. 10의 워크플로와 동일한 "source id="ett" | minmax_scaler | train_lstm model="ett:0.1", epoch=20 | print"을 사용하였다.

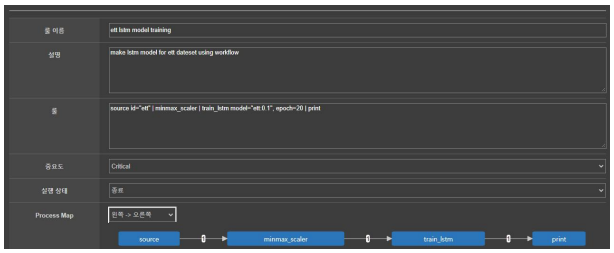

Fig. 15. Workflow for training LSTM model

Fig. 16은 Fig. 15의 워크플로로 학습 수행 시 loss의 변화를 보여주는 그래프이다. 학습 최종 loss는 0.0130이었다. Fig. 17은 모델이 예측한 값과 실제 값과 비교하는 그래프로 상당히 유사하게 예측된 것을 확인할 수 있다. 예측 오차는 0.0075로 측정되었다.

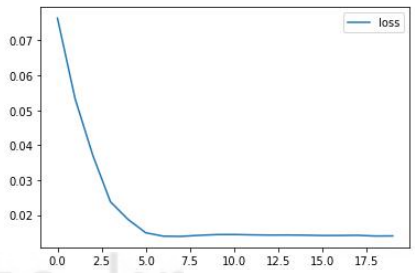

Fig. 16. loss of model training

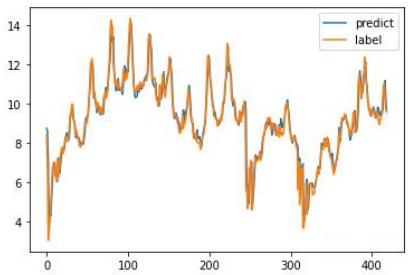

Fig. 17. ett-h1 predict and label data

AMML의 설계 원칙은 최대한 간결하게 워크플로 파이프라인을 구성하고 설정을 최소화하여 사용성을 높이는 데 중점을 두었다. Google의 Vertex AI 플랫폼은 Python SDK를 제공하여, Pipeline을 구축할 수 있도록 지원하고 있다[19]. 하지만, Python과 같은 일반적인 언어는 워크플로의 표현이 AMML에 비해 상대적으로 복잡하다. 또한, MicroSoft는 Azure ML에서 워크플로 구성을 위한 GUI를 제공한다[20]. 제안하는 AMML은 단순 복사하여 공유함으로써 다른 이해관계자와 협업이 편리하다. 전처리 및 분석 컴포넌트를 다양하게 변경하여 유연하게 적용해 볼 수 있다. 또한, AMML로 이루어진 워크플로를 운영 환경에 손쉽게 배포할 수 있다.

제안하는 시스템은 데이터 전처리 및 수집, AI 모델 학습, AMML을 이용한 운영 환경 배포 등을 지원한다. 하지만, 자동화된 룰 배포는 구현되지 않은 상태이다. 이는 구글의 MLOps 수준 1: ML 파이프라인 자동화(ML pipeline automation), 마이크로소프트의 수준 2: 자동화된 학습(Automated Training) 단계를 지원한다고 할 수 있다.

IV. Conclusions

많은 경우 기계 학습 전문가들은 모델 개발과 더불어 모델을 운영 환경에 적용한다. 이를 위해 학습용 데이터의 수집 및 가공 작업을 수행하며, 지속적으로 모델을 학습 및 개선한다. 학습된 모델은 주기적으로 운영 환경에 적용되어야 한다. MLOps는 이러한 과정들을 자동화하여 업무 효율성을 향상하고 개발자와 전문가들의 상호운용성을 강화할 수 있다. 본 연구에서는 MLOps를 구현하는 방안으로 AMML을 제시하였으며, 적용 가능성을 판단하기 위해 변압기 오일 온도 예측 LSTM 모델을 학습하고 예측을 수행하였다. 이를 통해 AMML을 사용한 학습 및 예측 워크플로를 운영 환경에 유연하게 적용할 수 있음을 확인하였다.

하지만, 구현된 시스템은 시계열 분석을 위한 구현만 수행하였다. 이미지 분석을 위한 CNN(convolutional neural network)등 다양한 모델을 지원하지 못한 한계가 있다. 추가적인 전처리를 위해 시계열뿐만 아니라 이미지 처리를 위한 컴포넌트 구현이 필요하다. CNN의 경우 기존 시계열 데이터와는 달리 컴포넌트 간 효율적인 이미지 전달 및 저장 방법이 중요하다. 이에 NoSql DB를 활용하여 이미지를 저장하고, NoSql DB의 이미지 저장 위치 정보를 컴포넌트에 전달하는 방안에 관한 연구를 추가로 수행할 예정이다. 또한, MLOps를 온전히 구현하기 위해서는 운영 환경에 워크플로를 자동배포 기능으로 배포하는 기능이 추가로 필요하다. 워크플로 AMML 스크립트 저장, 버저닝(versioning) 관리 및 통제 방안 마련 등의 연구를 수행하려 한다.

REFERENCES

[1] Lee Yo-Seob and Moon Phil-Joo, “A Comparison and Analysis of Deep Learning Framework,” The Journal of the Korea institute of electronic communication sciences, Vol. 12, No. 1, pp. 115–122, 2017, DOI: https://doi.org/10.13067/JKIECS.2017.12.1.115

[2] Bae Seong-Wan and Yu Jung-Suk, “Predicting the real estate price index using machine learning methods and time series analysis model,” Housing Studies Review, Vol. 26, No. 1, pp. 107-133, 2018, doi: http://dx.doi.org/10.24957/hsr.2018.26.1.107

[3] Samuel Ackerman, Orna Raz, Marcel Zalmanovici and Aviad Zlotnick, “Automatically detecting data drift in machine learning classifiers,” arXiv preprint arXiv:2111.05672, 2021, DOI: https://doi.org/10.48550/arXiv.2111.05672

[4] Sasu Makinen, Henrik Skogstrom, Eero Laaksonen and Tommi Mikkonen, “Who needs MLOps: What data scientists seek to accomplish and how can MLOps help?,” In: 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN), IEEE, pp. 109-112, 2021, DOI: 10.1109/WAIN52551.2021.00024

[5] Sima Siami-Namini, Neda Tavakoli and Akbar Siami Namin, “A Comparison of ARIMA and LSTM in Forecasting Time Series,” 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1394-1401, 2018. DOI: 10.1109/ICMLA.2018.00227

[6] Georgios Symeonidis, Evangelos Nerantzis, Apostolos Kazakis and George A. Papakostas, “MLOps - Definitions, Tools and Challenges,” 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0453-0460, 2022. DOI: 10.1109/CCWC54503.2022.9720902.

[7] Dominik Kreuzberger, Niklas Kuhl and Sebastian Hirschl, “Machine Learning Operations (MLOps): Overview, Definition, and Architecture,” arXiv preprint arXiv:2205.02302, 2022. DOI: https://doi.org/10.48550/arXiv.2205.02302

[8] Google, “MLOps: Continuous delivery and automation pipelines in machine learning,” https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning?hl=en

[9] Microsoft, “Machine Learning operations maturity model,” https://docs.microsoft.com/en-us/azure/architecture/example-scenario/mlops/mlops-maturity-model

[10] MLflow Project, “MLflow - A platform for the machine learning lifecycle,” https://mlflow.org

[11] The Kubeflow Authors, "Kubeflow," https://www.kubeflow.org

[12] Gustavo Correa Publio, Diego Esteves, Agnieszka Ławrynowicz, Panče Panov, Larisa Soldatova, Tommaso Soru, Joaquin Vanschoren And Hamid Zafar, “ML-schema: exposing the semantics of machine learning with schemas and ontologies,” arXiv preprint arXiv:1807.05351, 2018. DOI: https://doi.org/10.48550/arXiv.1807.05351

[13] Mun Jong-Hyeok, Kim Do-Hyung, Choi Jong-Sun and Choi Jae-Young, “Deep Learning Description Language for Referring to Analysis Model Based on Trusted Deep Learning,” KIPS Transactions on Software and Data Engineering, vol. 10, no. 4, pp. 133–142, Apr. 2021. DOI: https://doi.org/10.3745/KTSDE.2021.10.4.133

[14] Sepp Hochreiter and Jurgen Schmidhuber, “Long Short-Term Memory,” in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735

[15] Sima Siami-Namini, Neda Tavakoli and Akbar Siami Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series," 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1394-1401, 2018. DOI: 10.1109/ICMLA.2018.00227

[16] YAML Language Development Team, "YAML Ain’t Markup Language (YAML™) version 1.2,", https://yaml.org/spec/1.2.2/

[17] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong and Wancai Zhang, “Electricity Transformer Dataset (ETDataset),”, https://github.com/zhouhaoyi/ETDataset

[18] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong and Wancai Zhang, “Informer: Beyond Efficient Transformer for Long SequenceTime-Series Forecasting,” In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 12, pp. 11106-11115, May 2021. DOI: https://doi.org/10.48550/arXiv.2012.07436

[19] Google, “Tabular Workflows on Vertex AI,” https://cloud.google.com/vertex-ai/docs/tabular-data/tabular-workflows/overview?hl=en

[20] Microsoft, “Create and run machine learning pipelines using components with the Azure Machine Learning studio(Preview),” https://docs.microsoft.com/en-us/azure/machine-learning/how-to-create-component-pipelines-ui

Authors

Jung-Mo Sohn received the B.S. and M.S. degrees in Industrial Automation from Inha University, Korea, in 1998 and 2000, respectively. Sohn has been working at Epozen's research institute since 2018.

He is interested in artificial intelligence, cloud computing, and information security.

Su-Min Kim received the B.S degrees in Computer Systems & Engineering from Inha Technical College in 2021. Kim has been working at Epozen's research institute since 2020.

She is interested in artificial intelligence, cloud computing.