Kubeflow 를 사용한 쿠버네티스 기반 머신러닝 모델 개발

임호근, 김영한*
*숭실대학교

limhk@dcn.ssu.ac.kr, *younghak@ssu.ac.kr

Kubernetes based Machine Learning Model Development Using Kubeflow

Ho Keun Lim, Younghan Kim*
*Soongsil Univ.

요 약

머신러닝 알고리즘의 발전에 따라 머신러닝 개발을 위한 대용량 컴퓨팅 성능 및 대용량 데이터를 다루는 것이 중요해졌다. 이를 수행하기 위해서는 GPU 등 특수한 하드웨어가 필요하고 이 자원들을 효율적으로 관리하기 위해서 클라우드 위에서 머신러닝 개발을 하는 것이 필요해졌다. 클라우드에서 효율적인 머신러닝 개발을 위해 MLOps 라는 개념이 나타났고 이를 지원하기 위한 툴들이 생겨났다. 본 논문에서는 쿠버네티스 환경에서 효율적인 머신러닝 개발을 위한 플랫폼인 Kubeflow 를 이용하여 Fashion MNIST 분류 모델 개발 과정을 보임으로써 MLOps 및 이를 위한 플랫폼의 중요성을 보인다.

Ⅰ. 서 론

머신러닝 알고리즘의 발전에 따라 머신러닝 모델의 성능을 올리기 모델을 개발하기 위해선 대용량 컴퓨팅 성능 및 GPU 등 특수한 하드웨어 리소스가 필요하다. 하지만 이러한 하드웨어들은 비용이 높고 효율적으로 관리하기 힘들다. 이를 위해 머신러닝 개발을 위한 서버를 각각 관리하는 것이 아닌 전체적으로 관리할 수 있게 프라이빗 클라우드를 구축했다.[1][2] 따라서 머신러닝 개발을 클라우드 위에서 진행하게 되었다. 하지만 머신러닝 개발자들은 이러한 클라우드 인프라에서 워크로드를 수행하기 위해서 인프라에 대한 지식이 필요해졌으며 추가적으로 머신러닝 개발을 효율적으로 하기 위한 MLOps 라는 개념이 생겨났다. 이에 따라 머신러닝 개발자들은 인프라의 이해 없이 머신러닝 모델 개발 및 수행할 수 있게 되었다. 본 논문에서는 쿠버네티스 환경에서 효율적인 머신러닝 개발을 위한 오픈소스 플랫폼인 Kubeflow 에 대한 소개와 Fashion MNIST 분류 모델 개발과정을 보이면서 MLOps 의 필요성 및 이를 위한 플랫폼의 중요성을 보인다.[3]

Ⅱ. Kubeflow

Kubeflow 는 쿠버네티스 환경에서 머신러닝 워크플로우를 만드는데 있어 필요한 오픈소스 툴들을 한 곳에 모아둔 프로젝트이다.[4] 본 논문에서는 머신러닝 모델 개발에 있어 Kubeflow 에서 제공하는 툴 중 일부에 대해 설명한다.

1. Fairing

Faring 은 ML 모델을 쿠버네티스 환경에서 배포 할 수 있도록 하는 파이썬 패키지이다. 패키지를 설치 함으로써 ML 모델 생성, 학습, 배포 등의 작업을 쿠버네티스 클러스터로 요청을 하게 된다. 본 툴을 이용하여 주피터 노트북을 사용하여 코드를 개발하고 이를 컨테이너화 한 후 쿠버네티스 클러스터에 구동하는 과정이 축소되었다.

2. Katib

Katib 는 머신러닝 모델 개발에 있어 하이퍼파라미터 최적화를 위한 툴이다. 이 과정은 전체 ML 워크플로우에서 최적값이 나올 때까지 과정이 반복되기 때문에 많은 시간을 차지한다. 따라서 이 과정을 자동화하여 설정한 최적값이 나올 때까지 계속 반복함으로써 수동으로 이 값을 찾아냈던 과정들을 자동화시킨다.

3. Pipeline

파이프라인은 컨테이너 기반의 end-to-end ML 워크플로우를 만들고 배포할 수 있는 쿠버네티스 기반 툴이다. 파이프라인은 ML 워크플로우의 컴포넌트들과 그것을 그래프 형태로 결합한 것이며 파이프라인을 실행하게 되면 각 단계에 대한 쿠버네티스 파드를 생성하게 되며 순서에 따라 각 컨테이너들이 실행된다.

Ⅲ. Kubeflow 를 활용한 Fashion MNIST 분류 모델 생성

1. 실험 환경

본 논문에서 Fashion MNIST 분류 모델을 생성하기 위한 환경은 표 1 과 같다. 표 1 과 같은 서버 환경에서 오픈스택을 All-in-one 형태로 구축하고 이 위에 가상머신을 생성하고 minkube 를 설치하여 쿠버네티스 클러스터를 구축하였다. 쿠버네티스 클러스터에 Kubeflow 를 설치 및 Kubeflow 에서 제공되는 툴들을 사용하여 Fashion MNIST 머신러닝 모델을 개발했다.

Processor	Intel(R) Xeon(R) CPU D-1587 @ 1.70GHz
Memory	32GB DDR4 2400MHz
OS	Ubuntu 18.04
Openstack Version	victoria
Kuberentes Version	v1.20.2
Kubeflow Version	V1.0.2

표 1. 실험 환경

2. Fashion MNIST 분류 모델 생성

본 논문에서는 Fashion MNIST 모델 생성을 위한 코드를 작성한 후에 이를 Fairing 을 활용하여 쿠버네티스 환경에서 수행할 수 있도록 코드를 그림 1 과 같이 작성했다. config.set_deployer 코드를 사용하여 Job 을 만듬으로써 Kubeflow 의 Fairing 이 컨테이너화 및 자동적으로 ML 워크로드를 생성하여 컨테이너를 생성하기 위한 매니페스트 파일을 작성하지 않도록 했다.

```
if __name__ == '__main__':
    if os.getenv('FAIRING_RUNTIME', None) is None:
        from kubeflow import fairing
        from kubeflow.fairing.kubernetes import utils as k8s_utils

        DOCKER_REGISTRY = 'kubeflow-registry.default.svc.cluster.local:30000'
        fairing.config.set_builder(
            'append',
            image_name='fairing-job',
            base_image='brightfly/kubeflow-jupyter-lab:tf2.0-gpu',
            registry=DOCKER_REGISTRY,
            push=True)
        # cpu 2, memory 5GiB
        fairing.config.set_deployer('job',
                                    namespace='hk',
                                    pod_spec_mutators=[
                                        k8s_utils.get_resource_mutator(cpu=2,
                                                                       memory=5)]

                                    )
        fairing.config.run()
    else:
        remote_train = MyFashionMnist()
        remote_train.train()
```

그림 1. Fairing 코드

그 후 하이퍼파라미터 최적화를 위해서 Katib 를 사용하여 이를 수행했다. 최적의 값을 찾기 위한 파라미터를 지정하고 어떠한 최적화 함수를 쓸 것인지 설정하여 모델의 성능을 올렸다. 본 논문에서는 learning late 및 drop out 의 값을 최적화 하기 위하여 최적화 함수는 sgd 또는 adam 중 하나를 사용하도록 했고 랜덤 탐색 알고리즘을 사용했다. 하이퍼파라미터 최적화를 위한 코드는 그림 2 와 같다.

```
def train(self):
    parser = argparse.ArgumentParser()
    parser.add_argument('--learning_rate', required=False, type=float, default=0.001)
    parser.add_argument('--dropout_rate', required=False, type=float, default=0.2)
    parser.add_argument('--opt', required=False, type=int, default=1)

    args = parser.parse_args()

    (x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
    x_train, x_test = x_train / 255.0, x_test / 255.0

    model=tf.keras.models.Sequentail([
        tf.keras.layers.Flatten(input_shape=(28,28)),
        tf.keras.layers.Dense(128,activation='relu'),
        tf.keras.layers.Dropout(args.dropout_rate),
        tf.keras.layers.Dense(10,activation='softmax')
    ])

    model.summary()

    sgd=tf.keras.optimizers.SGD(lr=args.learning_rate)
    adam=tf.keras.optimizers.Adam(lr=args.learning_rate)

    optimizers=[sgd,adam]
    model.compile(optimizer=optimizers[args.opt],
                  loss='sparse_categorical_crossentropy',
                  metrics=['acc'])
```

```
parameters:
  - name: --learning_rate
    parameterType: double
    feasibleSpace:
      min: "0.005"
      max: "0.015"
  - name: --dropout_rate
    parameterType: double
    feasibleSpace:
      min: "0.1"
      max: "0.9"
  - name: --opt
    parameterType: int
    feasibleSpace:
      min: "0"
      max: "1"
```

그림 2. 하이퍼파라미터 최적화 코드

마지막으로 파이프라인을 작성함으로써 이에 대한 워크플로우를 구성하였고 마지막에 추론 서버를 파이프라인에 추가함으로써 개발된 모델을 사용하여 추론할 수 있도록 구성했다. 파이프라인의 각 단계는 각각 실행되어 다른 단계에 영향을 미치지 않는다. 따라서 중간 단계가 고장 나더라도 처음부터 돌리는 것이 아닌 그 부분만 수정하게 되면 다시 재가동된다. 파이프라인 구성 모습은 그림 3 과 같다.

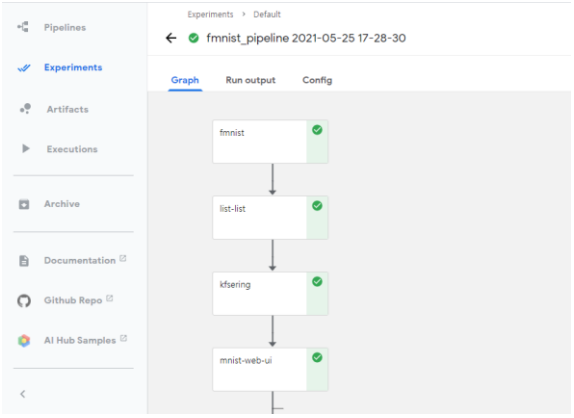

그림 3. 파이프라인 구성도

Ⅳ. 결 론

본 논문에서는 클라우드 환경에서 머신러닝 모델 개발을 위한 머신러닝 플랫폼인 Kubeflow 를 이용하여 머신러닝 모델의 개발과정을 보였다. 오픈소스를 바탕으로 프라이빗 클라우드에서 머신러닝 개발을 위한 환경을 구축하고 머신러닝 플랫폼을 통해서 MLOps 를 구현함으로써 머신러닝 개발과정을 세분화하고 반복할 수 있게 되었고 클라우드를 사용함으로써 리소스 또한 효율적으로 관리되게 했다.

ACKNOWLEDGMENT

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2020-0-00946, 하이브리드 클라우드 환경에서의 고속, 자동 서비스 복구 및 이전 소프트웨어 개발)

참 고 문 헌

[1] M. Xue et al., "Scalable GPU Virtualization with Dynamic Sharing of Graphics Memory Space," in IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 8, pp. 1823-1836

[2] Hong Y., Kim J., “Workflow Improvement for KubeFlow DL Performance over Cloud-Native SmartX AI Cluster,” in EIDWT 2020. Lecture Notes on Data Engineering and Communications Technologies, vol 47

[3] Y. Zhou, Y. Yu and B. Ding, "Towards MLOps: A Case Study of ML Pipeline Platform," 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE), 2020, pp. 494-500

[4] Kubeflow, "Introduction to Kubeflow," 2021, (https://www.kubeflow.org/docs/about/kubeflow/).