Keras를 이용한 Python과 C#의 딥러닝 성능 비교 분석

이성진* · 문상호

부산외국어대학교

Comparative analysis of deep learning performance for Python and C# using Keras

Sung-jin Lee · Sang-Ho Moon

Dept. of Computer Engineering, Busan University of Foreign Studies

E-mail : sjlee@bufs.ac.kr

요 약

최근에 Kaggle ML & DS Survey에 따르면 기계 학습 및 데이터 과학을 위한 프레임워크에서 TensorFlow와 Keras의 비율이 각각 41.82%, 34.09%로 비중을 차지하고 있으며, 개발 프로그래밍의 경우 약 82%로 Python을 사용하는 것으로 나타났다. 상당수의 기계 학습 및 딥러닝의 구조가 Keras 프레임워크와 Python을 활용하고 있으나, Python의 경우에는 스크립트 언어인 관계로 인해 배포 및 실행을 Python 스크립트 환경에 제한되어 동작하므로 다양한 환경에서 동작하기 어려운 개연성이 있을 수 있다.

본 논문에서는 Visual Studio 2019에서 동작하는 C#과 Keras를 활용한 기계 학습 및 딥러닝 시스템을 구현하였으며, 세부적으로 Mnist 데이터셋을 활용하여 파이썬 3.8.2와 C# .NET 5.0 환경에서 20번의 테스트를 진행하였다. 테스트 수행 결과, Python은 최소 시간 1.86초, 최대 시간 2.38초, 평균 시간 1.98초 총 시간 39.53초가 소요되었으며, C#은 최소 시간 1.78초, 최대 시간 2.11초 평균 시간 1.85초 총 시간 37.02초가 소요되었다. 결론적으로 C#의 성능이 Python보다 6% 정도 향상되었음을 확인하였으며, 이를 통해 실행파일 추출이 가능하여 활용도가 높을 것으로 기대한다.

ABSTRACT

According to the 2018 Kaggle ML & DS Survey, among the proportions of frameworks for machine learning and data science, TensorFlow and Keras each account for 41.82%. It was found to be 34.09%, and in the case of development programming, it is confirmed that about 82% use Python. A significant number of machine learning and deep learning structures utilize the Keras framework and Python, but in the case of Python, distribution and execution are limited to the Python script environment due to the script language, so it is judged that it is difficult to operate in various environments.

This paper implemented a machine learning and deep learning system using C# and Keras running in Visual Studio 2019. Using the Mnist dataset, 100 tests were performed in Python 3.8,2 and C# .NET 5.0 environments, and the minimum time for Python was 1.86 seconds, the maximum time was 2.38 seconds, and the average time was 1.98 seconds. Time 1.78 seconds, maximum time 2.11 seconds, average time 1.85 seconds, total time 37.02 seconds. As a result of the experiment, the performance of C# improved by about 6% compared to Python, and it is expected that the utilization will be high because executable files can be extracted.

키워드

Keras, C#, Performance Evaluation, Deep Learning

Ⅰ. 서 론

최근 Kaggle ML & DS Survey에 따르면 기계 학습 및 데이터 과학을 위한 프레임워크에서 TensorFlow와 Keras의 비율이 각각 41.82%, 34.09%로 비중을 차지하고 있으며, 개발 프로그래밍은 약 82%로 Python의 활용이 높은 편이다[1]. 일반적으로 Keras는 딥러닝과 기계 학습에서 활용

* speaker

되고 있으며, Python에서 주로 동작한다. C#은 마이크로소프트에서 만든 개체 지향 프로그래밍 언어로 2000년 7월에 발표했으며, 전 세계 개발자가 오랫동안 사용하고 있는 프로그래밍 언어 중 하나이다. 세부적으로 C#은 소프트웨어 개발에 특화된 언어로 활용되고 있으며, TensorFlow 2.0에서 Java, Go, Rust, C# 등의 바인딩을 지원하면서, C#을 위한 Keras 라이브러리가 배포되었다[2].

본 논문에서는 Keras를 이용한 Python과 C#의 딥러닝 성능 비교 분석을 위하여 먼저 Visual Studio 2019에서 동작하는 C#과 Keras를 활용한 기계 학습 및 딥러닝 시스템을 구현하였다. 그리고 Mnist 데이터셋을 기반으로 파이썬 3.8.2와 C# .NET 5.0 환경에서 딥러닝 성능을 비교 분석하였다.

II. 관련 연구

2.1 Keras

Keras는 딥러닝 모델을 만들기 위하여 고수준의 구성 요소를 제공하는 모델 수준의 라이브러리이다. 세부적으로 텐서 조작이나 미분 같은 저수준의 연산을 다루지 않고, Keras의 백엔드 엔진(Backend engine)에서 제공하는 텐서 라이브러리를 사용한다[2]. 그림 1은 Tensorflow 구조와 Keras 구조를 비교한 것으로, (a)는 Tensorflow 구조를 (b)는 Keras 구조를 각각 나타내고 있다[3].

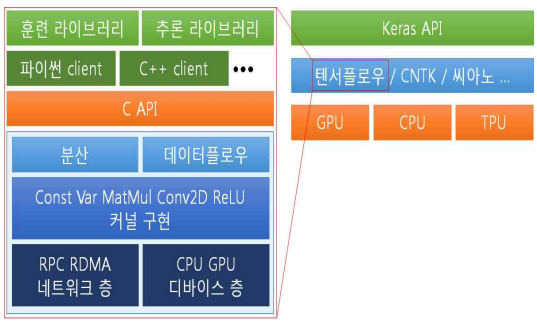

Fig. 1. Tensorflow & Keras structure

2.2 Python

Python은 1991년 네덜란드계 프로그래머인 귀도 반 로섬이 발표한 고급 프로그래밍 언어로, 플랫폼에 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어다. 다양한 플랫폼에서 사용할 수 있고, 라이브러리나 모듈이 풍부하여, 대학을 비롯한 여러 교육 기관, 연구 기관 및 산업계에서 활용이 증가하고 있는 추세이다. 또한 Python은 순수한 프로그램 언어로서의 기능 외에도 다른 언어로 쓰인 모듈들을 연결하는 용도로도 자주 이용되고 있으며, 기계 학습 및 딥러닝에서 Keras와 같이 활용되고 있다. Python은 스크립트 기반으로 동작하므로, 실행 시간이 고급 언어에 비해 수행 속도가 느린 단점이 있으나 Pyrex, Psyco, NumPy 등을 이용하여 특정 부분에서 연산을 향상시킬 수 있어, 과학, 공학 분야에서도 많이 이용되고 있다[4].

2.3 C#

C#은 2000년 7월에 개최되었던 Professional Developers Conference(PDC)에서 .NET 프로젝트와 함께 발표된 객체 지향 프로그래밍 언어로, 2002년 1월 .NET Framework 1.0을 기점으로 발전하였다. 윈도우 전용 개발 언어로 진행하다가 2015년 7월 .NET Core 1.0을 시작으로 리눅스를 지원하면서, 2021년 4월에 .NET Core과 .NET Framework 기능을 합친 .NET 5.0이 출시되었다. 수많은 라이브러리와 특화된 IDE(Visual Studio)를 통해 생산성이 높은 언어이지만, 가상 머신을 거쳐서 처리되므로, C/C++ 등의 언어에 비해 속도가 조금 느린편이다. 윈도우 기반 시각화 및 응용 프로그램 개발에서 많이 이용되고 있다[5].

III. 성능비교 실험

3.1 MNIST 데이터셋을 이용한 성능 비교 설계

본 논문에서 성능 비교를 위하여 MNIST 데이터셋을 활용하며, MNIST는 인공지능 연구자인 LeCun교수가 만든 데이터 셋으로, 60,000개의 트레이닝 셋과 10,000개의 테스트 셋으로 이루어져 있다. 세부적으로 트레이닝 셋을 학습데이터로 사용하고 테스트 셋을 신경망을 검증하는 데에 사용한다. 실험에 사용한 MNIST 데이터 셋은 총 1만개의 배치 사이즈와 5번의 에폭 학습을 거치고, 총 20번을 반복시킨 후, 최소 시간과 최대 시간, 평균 시간과 총 시간을 비교하여 성능을 분석한다[6]. 성능 분석을 위한 실행 코드로 그림 2에서 Python, 그림 3에서 C#로 각각 구현하였다.

```python
from tensorflow import keras
from tensorflow.keras import layers, models
import time

minTime = 0
maxTime = 0
totalTime = 0
count = 0
for x in range(20):
    fashion_mnist = keras.datasets.fashion_mnist
    (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

    train_images = train_images / 255.0
    test_images = test_images / 255.0

    model = models.Sequential()
    model.add(layers.Flatten())
    model.add(layers.Dense(128, activation='relu'))
    model.add(layers.Dense(10, activation='softmax'))

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    start = time.time()

    model.fit(train_images, train_labels, batch_size=10000, epochs=5)

    end = time.time()

    checkTime = end - start

    if minTime == 0 or minTime > checkTime:
        minTime = checkTime
    if maxTime == 0 or maxTime < checkTime:
        maxTime = checkTime

    totalTime += checkTime
    count = count + 1

print("최소 시간 : %.10f" % minTime)
print("최대 시간 : %.10f" % maxTime)
averageTime = totalTime / 20
print("평균 시간 : %.10f" % averageTime)
print("총 시간 : %.10f" % totalTime)
```

Fig. 2. Python Source Code

```csharp
using System;
using Keras.Datasets;
using Keras.Models;
using Keras.Utils;
using Numpy;
using Keras.Layers;

namespace Keras_Test
{
    참조 0개
    class Program
    {
        참조 0개
        static void Main(string[] args)
        {
            double minTime = 0;
            double maxTime = 0;
            double totalTime = 0;
            for (int i = 0; i < 20; i++)
            {
                var ((train_images, train_labels), (test_images, test_labels)) = FashionMNIST.LoadData();

                train_images = train_images / 255.0;
                test_images = test_images / 255.0;

                var model = new Sequential();

                model.Add(new Flatten());

                model.Add(new Dense(128, null, "relu"));
                model.Add(new Dense(10, null, "softmax"));

                model.Compile(optimizer: "adam", loss: "sparse_categorical_crossentropy", metrics: new string[] { "accuracy" });

                long start = DateTime.Now.Ticks;

                model.Fit(train_images, train_labels, 10000, 5);

                long end = DateTime.Now.Ticks;

                double checkTime = (double)(end - start) / 10000000.0F;

                if (minTime == 0 || minTime > checkTime)
                    minTime = checkTime;
                if (maxTime == 0 || maxTime < checkTime)
                    maxTime = checkTime;
                totalTime += checkTime;
            }
            Console.WriteLine("최소 시간 : {0}", minTime);
            Console.WriteLine("최대 시간 : {0}", maxTime);
            Console.WriteLine("평균 시간 : {0}", totalTime/ 20);
            Console.WriteLine("총 시간 : {0}", totalTime);
        }
    }
}
```

Fig. 3. C# Source Code

3.2 실험 결과 및 결과 고찰

본 논문에서 성능비교를 위하여 수행한 실험 결과는 그림 4와 그림 5에서 확인할 수 있다. 세부적으로 Python은 최소 시간 1.86초, 최대 시간 2.38초, 평균 시간 1.98초 총 시간 39.53초가 소요되었으며, C#은 최소 시간 1.78초, 최대 시간 2.11초 평균 시간 1.85초 총 시간 37.02초가 소요되었다. 이를 통하여 Python에 비하여 C#이 최소 시간 4.3%, 최대 시간 11.3%, 평균 시간 6.5%, 총 시간 6.3% 더 향상되었음을 확인할 수 있다.

```
loss: 0.8529 - accuracy: 0.7206
Epoch 4/5
6/6 [==============================] - 0s 42ms/step -
loss: 0.7360 - accuracy: 0.7539
Epoch 5/5
6/6 [==============================] - 0s 38ms/step -
loss: 0.6672 - accuracy: 0.7777
Epoch 1/5
6/6 [==============================] - 0s 41ms/step -
loss: 1.9427 - accuracy: 0.3949
Epoch 2/5
6/6 [==============================] - 0s 40ms/step -
loss: 1.1760 - accuracy: 0.6407
Epoch 3/5
6/6 [==============================] - 0s 42ms/step -
loss: 0.8902 - accuracy: 0.6927
Epoch 4/5
6/6 [==============================] - 0s 41ms/step -
loss: 0.7636 - accuracy: 0.7286
Epoch 5/5
6/6 [==============================] - 0s 37ms/step -
loss: 0.6890 - accuracy: 0.7592
최소 시간 : 1.8861079216
최대 시간 : 2.3831362724
평균 시간 : 1.9769130826
총 시간 : 39.5382616520
```

Fig. 4. Python Execution Result

```
1/6 [====>.....................] - ETA: 0s - loss: 0.9173 - accuracy: 0.6864
3/6 [==============>.............] - ETA: 0s - loss: 0.8872 - accuracy: 0.6940
5/6 [========================>.....] - ETA: 0s - loss: 0.8601 - accuracy: 0.7000
6/6 [==============================] - ETA: 0s - loss: 0.8475 - accuracy: 0.7038
6/6 [==============================] - 0s 48ms/step - loss: 0.8475 - accuracy: 0
.7038
Epoch 4/5
1/6 [====>.....................] - ETA: 0s - loss: 0.7612 - accuracy: 0.7391
2/6 [=========>....................] - ETA: 0s - loss: 0.7559 - accuracy: 0.7375
4/6 [==================>...........] - ETA: 0s - loss: 0.7421 - accuracy: 0.7421
5/6 [========================>.....] - ETA: 0s - loss: 0.7357 - accuracy: 0.7441
6/6 [==============================] - 0s 49ms/step - loss: 0.7317 - accuracy: 0
.7449
Epoch 5/5
1/6 [====>.....................] - ETA: 0s - loss: 0.6711 - accuracy: 0.7712
3/6 [==============>.............] - ETA: 0s - loss: 0.6733 - accuracy: 0.7678
5/6 [========================>.....] - ETA: 0s - loss: 0.6678 - accuracy: 0.7689
6/6 [==============================] - 0s 44ms/step - loss: 0.6622 - accuracy: 0
.7716
최소 시간 : 1.7851021
최대 시간 : 2.1111208
평균 시간 : 1.8511558799999999
총 시간 : 37.0231176
```

Fig. 5. C# Execution Result

V. 결 론

본 논문에서는 Keras를 이용한 Python과 C#의 딥러닝 성능 비교 분석을 위하여 MNIST 데이터셋을 기반으로 실험을 수행하였다. 실험 결과, C#의 성능이 Python보다 약 6% 정도 향상되었음을 확인하였다. 또한, C#의 경우에는 실행 파일 추출 및 시각화 프로그램 작성 등이 가능하여 활용도가 높은 장점이 있다. 현재 본 논문을 통한 성능 비교는 단순한 형태였지만, 차후 C#의 딥러닝 라이브러리가 다양해지면, 실행 파일 추출 및 지원등으로 인해 멀티플랫폼 타입의 딥러닝 개발 언어로 성장할 수 있음을 기대해본다.

References

[1] Ana Trisovic, Matthew K. Lau, Thomas Pasquire and Merce Crosas, “A large-scale study on research code quality and execution” *Scientific Data,* 60, Feb, 2022.
[2] Keras IO, “Developer guides”, https://keras.i2o/guides/
[3] TensorFlow.org “End-to-End Open Source ML Platform”, https://www.tensorflow.org/
[4] Python org, “Welcome to Python World”, https://www.python.org/
[5] 위키백과, “C 샤프”, https://ko.wikipedia.org/wiki/C_샤프
[6] MNIST dataset, “THE MNIST DATABASE of handwritten digits”, http://yann.lecun.com/exdb/mnist/