출간계획서

책 제목과 부제

제목 딥러닝 첫 걸음

부제 Microsoft CNTK

저자 소개

김태형

현재 kt융합기술원의 AI Tech Center의 책임연구원. 국제신경망협회 (International Neural Network Society)의 회장을 역임한 Donald C. Wunsch II의 지도 하에 인공신경망/딥러닝 관련 연구를 배웠다. 대화, 추천, IoT, 컴퓨터 바둑, 무선 네트워크 관련 프로젝트를 수행했다.

미주리 과학기술대 전기공학과 박사, 연세대학교 전기전자전파공학과 석사, 연세대학교 전파공학과 학사. Siemens Student Program 4기.

한국에서 태어나, 일본, 독일, 미국을 배회하다 지금은 한국에 거주 중이다. 영어 및 일본어에 능통하며, 몸과 마음을 관리하기 위해 서핑과 신디사이저 연주를 한다.

타깃독자

- 딥러닝을 배우려는 이공계 학사 고학년 및 대학원생

- 딥러닝을 업무에 활용하려는 개발자

- AI에 대한 보다 깊은 이해를 원하는 기술적 소양을 가진 일반인

출간목적:

딥러닝의 이후로 딥러닝 프레임워크가 인기이다. 구글에서 2015년 텐서플로를 출시했을 때, 한국에선 인기이다.

NVIDIA Deep Learning Institute에서 CNTK를 접했을 때, 텐서플로에 비해 다양한 장점을 가진 CNTK에 매료되었다. 첫째, 스크립트가 보다 직관적이고 간단하다. 둘째, 필요한 프로그래밍 언어가 C++만으로, . 셋째, C++ 프로그램인 CNTK은 NVIDIA의 다양한 library와 호환성이 좋은 수 밖에 없는 태생적인 구조를 가지고 있다. 넷째, 진정한 딥러닝의 파워를 발휘하기 위한 Multi-Node Multi-GPU환경에 보다 적합한 태생적 목적을 가지고 있다.

집필원칙: 이 책을 집필하면서 다음과 같은 원칙을 지키고자 한다.

CNTK의 소개서에 그치지 않도록 딥러닝의 주요개념을 놓치지 않고 설명한다.

특정 application은 해당 분야의 전문가에게 반드시 감수 (Cross-validation)를 받는다. 예를 들면, MNIST, CIFAR-10예제는 이미지처리 분야, 는 음성처리

육하원칙에 근거해서 설명한다. A는 무엇이고 (What), 왜 배워야하는지 (Why), 어떻게 하면 되고 (How), 누가 발명했으며 (Who), 언제/어디서 발명되었는지 (When/Where).

1. 이론과 실습부분으로 나눈다.

2. 이론:

1. 직관적인 설명으로 개념을 먼저 잡고 이론부분을 시작한다.
2. 개념을 뒷받침할 수 있는 이론을 간략히 소개한다.
3. 이론에 필요한 핵심 공식은 반드시 포함하되 최소화한다.

그 이상 필요 시 부록에 포함한다.

3. 실습:

1. 이론을 뒷받침하는 실습을 위한 예제를 제공한다.
2. 예제의 CNTK의 스크립트를 제공한다.
3. CNTK 스크립트의 설명을 한다.
4. Software Container Platform인 Docker로 해당 문제를 제공한다.

2. 반드시 매일 30분이상 쓴다.

컨셉: 이 책은 무엇이 다른가? (차별화 포인트)

- 딥러닝을 모르는 엔지니어가

- C++프로그래머가 쉽게 넘어올 수 있도록 한다.

분량: 약 220쪽, A4, 기본여백, 글자크기 11포인트 기준

일정

- 출간 희망일: 10월 중순~11월 중순

- 목차, 서문, 샘플 원고 (3꼭지): 별도로 첨부함

목차

1.

2. 신경망 개론

3. 데이터셋

3.1. MNIST

3.2. CIFAR-10

3.3. Pascal VOC

4. 예: MNIST 예제를 활용한

서문

샘플원고

상세일정

MNIST

<https://github.com/Microsoft/CNTK/tree/v2.0.beta15.0/Examples/Image/GettingStarted>

1주일

<https://github.com/Microsoft/CNTK/tree/master/Examples/Image/Classification/ResNet#cntk-examples-imageclassificationresnet>

ResNet

ResNet20

ResNet110

ResNet50

ResNet101

ResNet152

Docker는 개발환경을 설정하는 반복적인 작업을 자동화해줘서 개발자들이 개발을 하는데 집중할 수 있게 해준다. 이에 실습을 진행하는데 있어 실습자 모두가 환경설정을 하는 것이 아니라, 저자가 한번 설정하고 실습자는 이 환경을 (설정하지 않고) 이용만 할 수 있게 한다.

이렇게 Docker를 이용한 실습환경에는 두가지 이점이 있습니다.

1. 환경설정에 걸리는 번거로움이 없습니다.
2. 설정 시 문제가 발생했을 때, 문제해결을 할 필요가 없습니다.

즉, 설정한 환경이 들어있는 Docker 이미지를 다운받아 실행시키면 된다.

문제가 발생했을 때, 리눅스 환경이 익숙한 분이라면 문제해결을 할 수 있지만 시간이 지체된다는 단점이 있습니다. 익숙하지 않으시다면 문제해결이 안 되어 실습을 할 수 없다는 문제에 봉착하게 되는데요. 더 큰 문제는 이런 문제를 해결하는데 시간과 노력을 들이다보면 학습의욕이 떨어질 수도 있다는 것에 진정한 문제가 있다고 봅니다.

부수적인 장점으로 Docker라는 훌륭한 툴을 배울 수 있게 되는 장점도 있습니다. Docker는 OS 전체를 설치하지 않고, 라이브러리와 설정 등만을 쓸 수 있습니다. 즉, 리눅스커널 등이 빠집니다. 기존에 동일한 환경을 제공하기 위해서는 가상화머신을 다운로드 받아서 설치했어야 하는데요. N개의 실습을 위해 N개의 가상화머신을 설치해야했습니다. 이렇게 되면 리눅스 커널이 추가로 N번 설치되어 비효율적입니다.

a full operating system - only libraries and settings r