

02

텐서플로 2.0 설치

이 책의 모든 코드는 구글 코랩과 깃허브를 통해 제공됩니다. 구글 코랩의 코드는 별도의 설치 없이 인터넷만 연결돼 있으면 바로 실행 가능하지만 로컬 환경에서 작업해야 할 경우를 대비해서 윈도우와 macOS 환경에서 텐서플로 2.0의 설치법을 간단히 설명하겠습니다.

2.1 윈도우

윈도우 환경에서 텐서플로를 설치할 때 가장 유의해야 할 점은 GPU 가속을 사용할 수 있도록 텐서플로를 설치하는 것입니다. 이를 위해서는 그래픽 드라이버, CUDA¹, cuDNN²을 올바르게 설치할 필요가 있습니다.

여기서는 기존에 사용하던 엔비디아(NVIDIA) 그래픽 카드를 사용하는 윈도우 10 컴퓨터에 텐서플로를 설치한다고 가정하겠습니다. 이 설치 가이드는 윈도우 10 환경, 다수의 그래픽 카드와 시스템을 대상으로 테스트한 후 작성됐습니다.

2.1.1 기존 엔비디아 드라이버 제거

엔비디아 그래픽 카드를 사용하고 있었다면 높은 확률로 엔비디아 그래픽 드라이버를 사용하고 있었을 것이고, 엔비디아 드라이버가 이미 설치돼 있는 경우 높은 확률로 CUDA의 설치가 정상적으로 되지 않

¹ 엔비디아 그래픽 카드의 GPU에 프로그램 명령을 실행할 수 있게 해주는 표준 소프트웨어입니다.

² 딥러닝 신경망을 위한 NVIDIA의 GPU 가속 라이브러리로서 CUDA를 기반으로 합니다. 텐서플로는 cuDNN을 사용하기 때문에 GPU를 사용하려면 반드시 설치해야 합니다.

는 경우가 발생합니다. 이미 CUDA 설치를 시도했지만 잘 되지 않았던 분들만 이번 절의 내용을 시도해주시기 바랍니다. CUDA 설치가 잘 된다면 굳이 기존 엔비디아 드라이버를 제거할 필요는 없습니다.

먼저 윈도우 키를 눌러서 시작 메뉴를 연 뒤 “프로그램 추가/제거” 메뉴를 선택합니다.

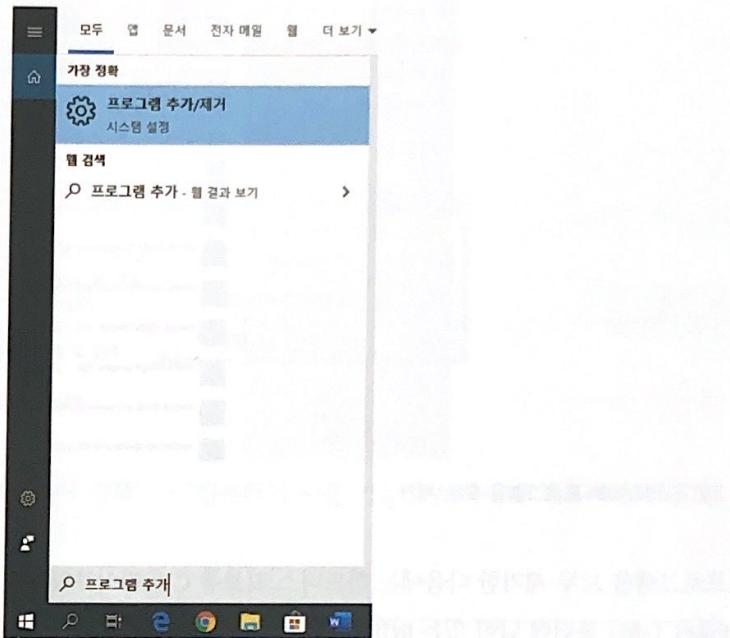


그림 2.1 프로그램 추가/제거 메뉴 선택

앱 및 기능 메뉴가 나오면 검색을 통해 이름이 “NVIDIA”로 시작하는 프로그램을 찾아서 모두 제거합니다. 이 과정에서 몇 번의 재부팅이 필요할 수 있습니다.

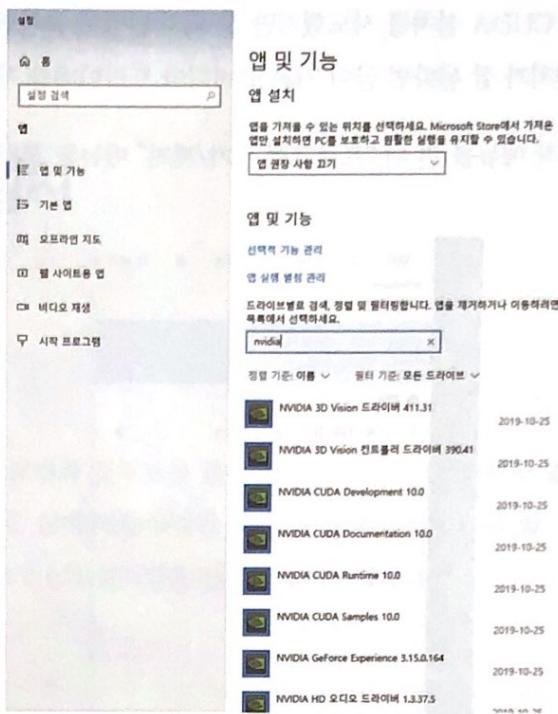


그림 2.2 NVIDIA 프로그램을 모두 제거

프로그램을 모두 제거한 다음에는 하드디스크(보통 C 드라이브와 D 드라이브)의 Program Files, Program Files (x86) 폴더에 남아 있는 NVIDIA로 시작하는 폴더를 모두 지웁니다.

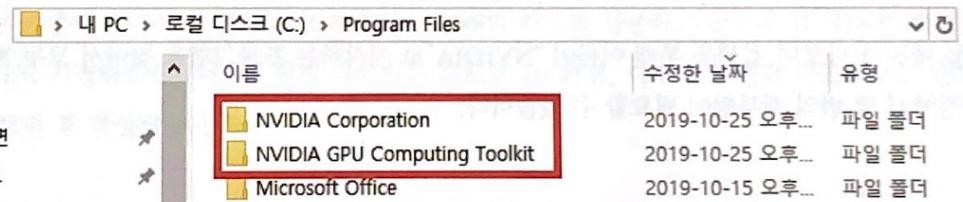


그림 2.3 프로그램 파일 폴더에 남아있는 NVIDIA 폴더를 제거

그다음에는 시작 메뉴에서 “장치 관리자”를 선택해서 엽니다.

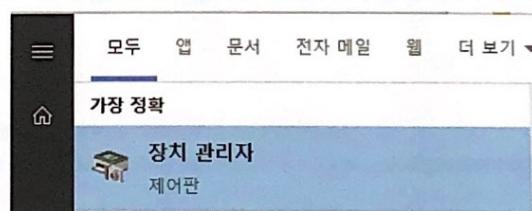


그림 2.4 장치 관리자 열기

장치 관리자에서 디스플레이 어댑터 하단에 있는 그래픽 카드를 선택한 후, 마우스 오른쪽 버튼을 클릭하고 “드라이버 업데이트”를 선택합니다.

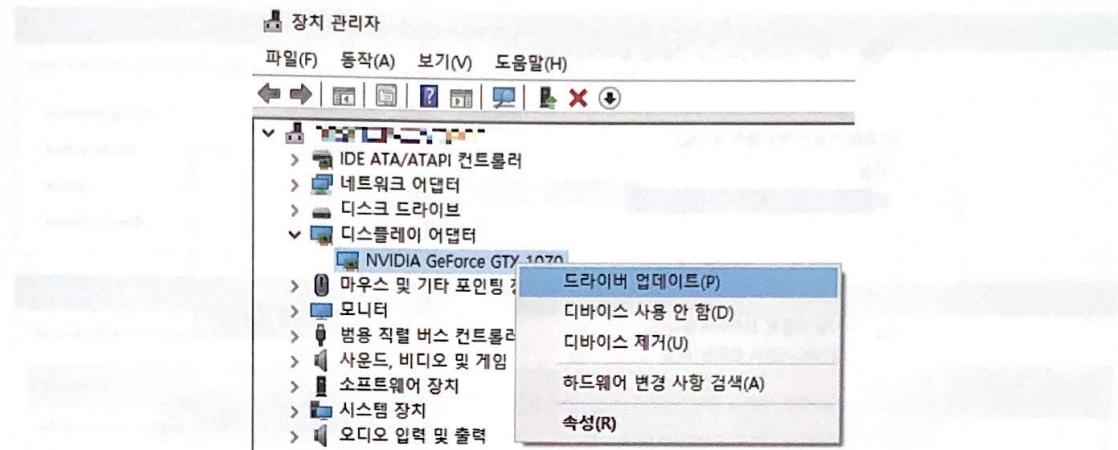


그림 2.5 그래픽 카드에서 “드라이버 업데이트” 선택

그다음 “컴퓨터에서 드라이버 소프트웨어 검색” → “컴퓨터의 사용 가능한 드라이버 목록에서 직접 선택”을 클릭합니다.

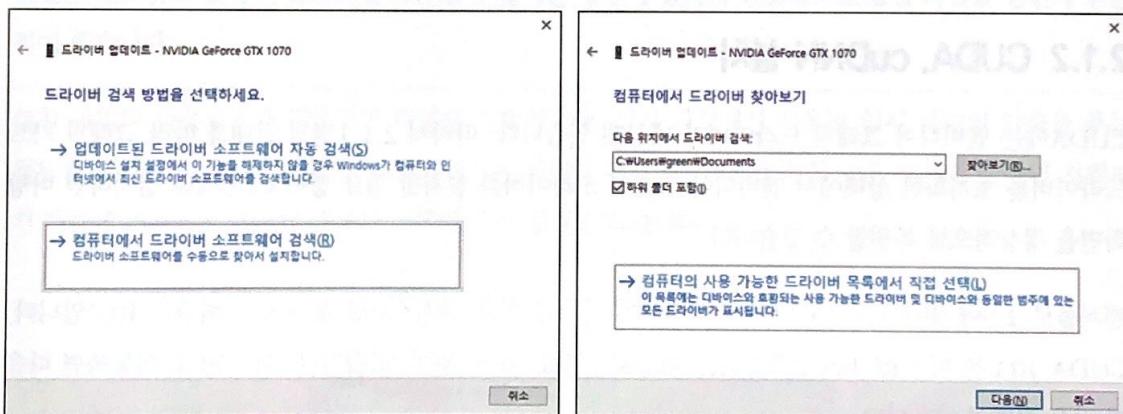


그림 2.6 그래픽 카드의 드라이버 업데이트 메뉴

“Microsoft 기본 디스플레이 어댑터”를 선택합니다. 위의 과정을 거치고 나면 NVIDIA 드라이버는 이 화면에 표시되지 않아야 합니다.

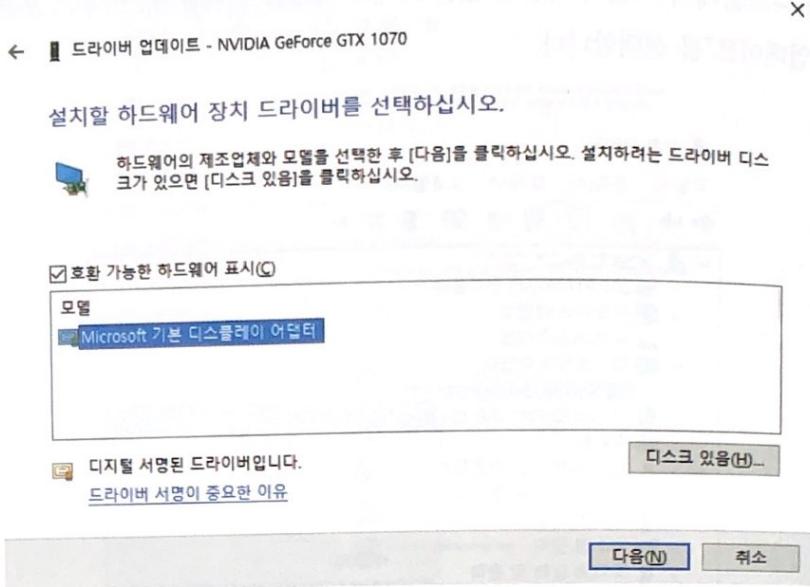


그림 2.7 “Microsoft 기본 디스플레이 어댑터” 선택

시스템을 재부팅하면 바탕화면의 구성요소 크기가 자연스럽게 커졌을 것입니다. 엔비디아 그래픽 카드 드라이버가 완전히 제거된 것으로, 이제 CUDA와 cuDNN을 정상적으로 설치할 수 있습니다.

2.1.2 CUDA, cuDNN 설치

CUDA에는 엔비디아 그래픽 드라이버가 내장돼 있습니다. 따라서 2.1.1절의 안내를 따라 그래픽 카드 드라이버를 초기화한 상태에서 엔비디아 그래픽 드라이버를 설치할 필요 없이 CUDA만 설치하면 바탕화면을 정상적으로 복원할 수 있습니다.

텐서플로 2.0에 맞는 CUDA 버전을 선택해야 하는데 현재 안정적으로 동작하는 버전은 10.0입니다. CUDA 10.0은 이 URL(<http://bit.ly/33dsXxL>)에서 내려받을 수 있습니다. 이 URL로 이동하면 다음과 같은 화면이 나옵니다.

CUDA Toolkit 10.0 Archive

The screenshot shows the 'Select Target Platform' section with four rows of filters:

- Operating System: Windows (selected), Linux, Mac OSX
- Architecture: x86_64 (selected)
- Version: 10 (selected), 8.1, 7, Server 2016, Server 2012 R2
- Installer Type: exe [network] (selected), exe [local] (disabled)

Below this is the 'Download Installer for Windows 10 x86_64' section, which contains the following text and a download button:

The base installer is available for download below.

Base Installer

Download [2.1 GB]

Installation Instructions:

1. Double click cuda_10.0.130_411.31_win10.exe
2. Follow on-screen prompts

그림 2.8 CUDA 10.0 설치 파일 다운로드

여기서 자신의 시스템에 맞는 옵션을 1~4번 순서대로 선택한 후 5번의 설치 파일을 내려받습니다. 설치 파일은 네트워크 설치 타입으로 받으면 종종 끊기는 경우가 있어서 안정적으로 로컬 파일을 한번에 받는 것이 좋습니다.

설치 파일의 다운로드가 완료되면 파일을 실행합니다. 설치 과정에서 사용될 임시 파일의 압축을 푸는 경로를 지정하는 메뉴가 나오면 적절한 위치를 선택한 뒤 “OK” 버튼을 누릅니다. 설치가 끝나면 지워지기 때문에 기본으로 표시된 경로를 선택해도 큰 문제는 없습니다.

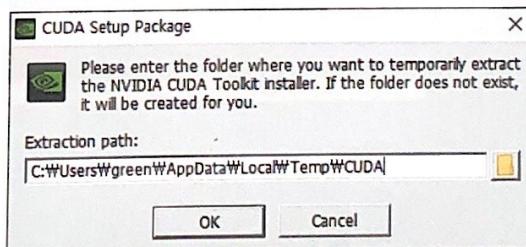


그림 2.9 설치용 임시 파일의 압축을 풀 경로를 선택

설치가 진행될 때 옵션에서 “사용자 정의 설치”를 선택합니다.

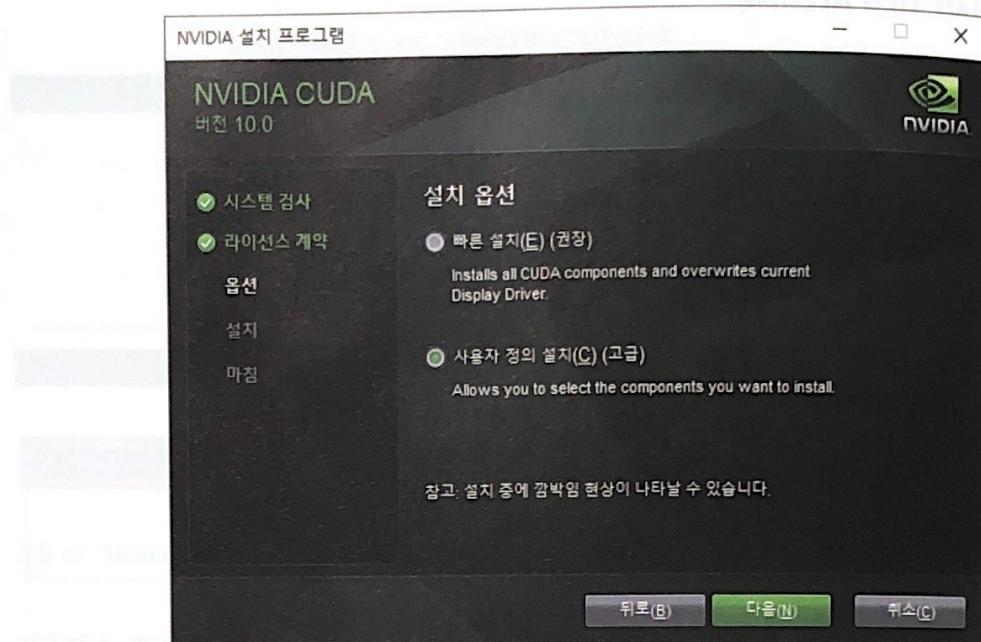


그림 2.10 사용자 정의 설치 선택

다음 단계에서 “CUDA” → “Visual Studio Integration”을 선택 해제합니다. 이 부분이 해제되지 않았을 때 CUDA 설치가 정상적으로 되지 않는 경우가 있기 때문입니다.

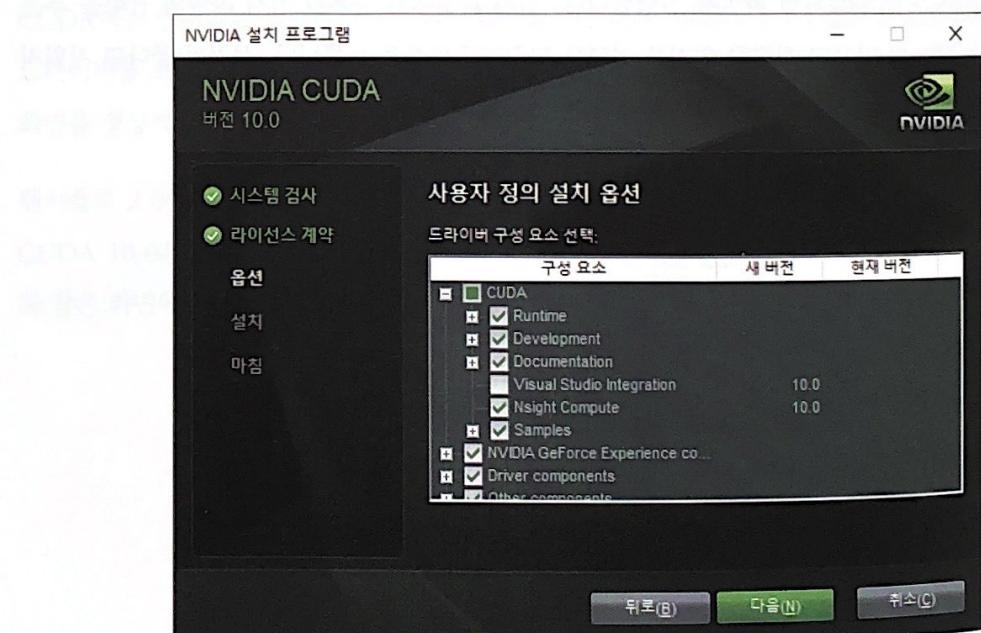


그림 2.11 “CUDA” → “Visual Studio Integration” 선택 해제

설치가 정상적으로 완료되면 시스템을 재부팅해보고, 그래픽 드라이버가 함께 설치되어 바탕화면이 복원된 것을 확인합니다.

이제 cuDNN을 설치할 차례입니다. 2019년 11월 현재 텐서플로 2.0, CUDA 10.0과 맞는 cuDNN 버전은 7.6.0입니다. 이 버전 정보는 텐서플로 버전업에 따라 변경될 수 있습니다. cuDNN은 이 URL(<http://bit.ly/34or6X2>)에서 내려받을 수 있습니다. 참고로 cuDNN을 다운로드하기 위해서는 회원 가입이 필요합니다. 한 번만 가입해놓으면 제한 없이 계속 내려받을 수 있기 때문에 귀찮더라도 나중을 위해 가입하는 것이 좋습니다.

회원 가입 후 로그인하면 그림 2.12와 같은 화면이 나타납니다. “I Agree To...”를 눌러서 사용권 계약에 동의하면 다운로드할 수 있는 cuDNN 버전의 리스트가 보입니다. 7.6.0 버전은 “Archived cuDNN Releases”를 선택하면 표시됩니다.

Home

cuDNN Download

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

I Agree To the Terms of the cuDNN Software License Agreement

Note: Please refer to the [Installation Guide](#) for release prerequisites, including support
For more information, refer to the [cuDNN Developer Guide](#), [Installation Guide](#) and [Release Notes](#).

[Download cuDNN v7.6.4 \[September 27, 2019\], for CUDA 10.1](#)

[Download cuDNN v7.6.4 \[September 27, 2019\], for CUDA 10.0](#)

[Download cuDNN v7.6.4 \[September 27, 2019\], for CUDA 9.2](#)

[Download cuDNN v7.6.4 \[September 27, 2019\], for CUDA 9.0](#)

[Archived cuDNN Releases](#)

그림 2.12 로그인 후 표시된 cuDNN 다운로드 링크

수많은 버전 중 CUDA 버전과 맞는 파일을 내려받습니다. 여기서는 cuDNN 7.6.0, CUDA 10.0 버전입니다.



그림 2.13 cuDNN 버전의 다운로드 링크 선택

그림에서 표시된 영역을 클릭하면 운영체제별로 설치 파일이 나눠져 있습니다. 이 책에서는 윈도우 10에 설치하는 것을 가정하고 있기 때문에 윈도우 10 설치 파일을 선택해서 내려받습니다.

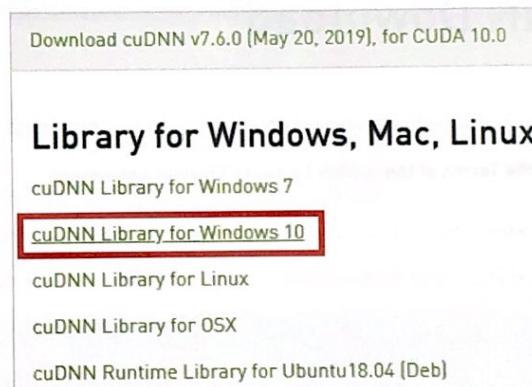


그림 2.14 cuDNN 윈도우 10 버전 다운로드

내려받은 압축파일 안에는 cuda 폴더가 있고, cuda 폴더 안에는 그림 2.15처럼 bin, include, lib라는 3개의 폴더가 있습니다.

내 PC > 다운로드 > cudnn-10.0-windows10-x64-v7.6.0.64.zip > cuda	
이름	유형
bin	파일 폴더
include	파일 폴더
lib	파일 폴더
NVIDIA_SLA_cuDNN_Support.txt	텍스트 문서

그림 2.15 cuDNN 압축 파일의 내용

이 폴더들을 복사해서 CUDA 설치 폴더에 붙여넣어야 합니다. CUDA 설치 폴더는 C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0 경로에서 찾아볼 수 있습니다. 이곳으로 이동한 후 bin, include, lib 폴더를 붙여넣고 겹치는 내용을 덮어씁니다.

내 PC > 로컬 디스크 (C:) > Program Files > NVIDIA GPU Computing Toolkit > CUDA > v10.0			
이름	수정한 날짜	유형	크기
bin	2019-10-25 오후...	파일 폴더	
doc	2019-10-25 오후...	파일 폴더	
extras	2019-10-25 오후...	파일 폴더	
include	2019-10-25 오후...	파일 폴더	
jre	2019-10-25 오후...	파일 폴더	
lib	2019-10-25 오후...	파일 폴더	
libnvv	2019-10-25 오후...	파일 폴더	
nvml	2019-10-25 오후...	파일 폴더	
nvvm	2019-10-25 오후...	파일 폴더	
src	2019-10-25 오후...	파일 폴더	
tools	2019-10-25 오후...	파일 폴더	
CUDA_Toolkit_Release_Notes.txt	2018-08-26 오후...	텍스트 문서	9KB
EULA.txt	2018-08-26 오후...	텍스트 문서	63KB
NVIDIA_SLA_cuDNN_Support.txt	2018-11-16 오후...	텍스트 문서	39KB
version.txt	2018-08-26 오후...	텍스트 문서	1KB

그림 2.16 cuDNN을 CUDA 설치 폴더에 붙여넣기

이로써 CUDA와 cuDNN 설치가 모두 끝났습니다.

2.1.3 아나콘다 설치

아나콘다(Anaconda)는 파이썬과 R 등의 언어 패키지 관리를 편리하게 해주는 통합 오픈소스 플랫폼입니다. 아나콘다를 설치하고 나면 아나콘다 내에서 conda, pip 등의 패키지 관리 툴로 각 개발 환경을 분리해서 관리할 수 있습니다. 이미 아나콘다를 사용 중인 분은 이번 절을 넘어가셔도 좋습니다.

아나콘다는 이 URL(<http://bit.ly/2qjrNm1>)에서 내려받을 수 있습니다. 텐서플로 윈도우 버전은 파이썬 3 버전으로, 64비트만 지원하기 때문에 이 파일을 다운로드하면 됩니다. 기본 다운로드 버튼을 눌러도 64비트 설치 파일이 다운로드됩니다.

Windows | macOS | Linux

Anaconda 2019.10 for Windows Installer



그림 2.17 아나콘다 설치 파일 다운로드

설치 파일을 실행한 후 “다음”, “동의” 버튼을 차례로 눌러서 설치 경로를 지정하는 화면으로 이동합니다. 설치 경로는 단순하게 설정하는 편이 나중에 찾기 좋습니다. 그리고 나중에 시스템 환경 변수 등에 파이썬 경로를 지정할 때 예러가 생기는 것을 방지하기 위해 한글이 들어가지 않는 것이 좋습니다.

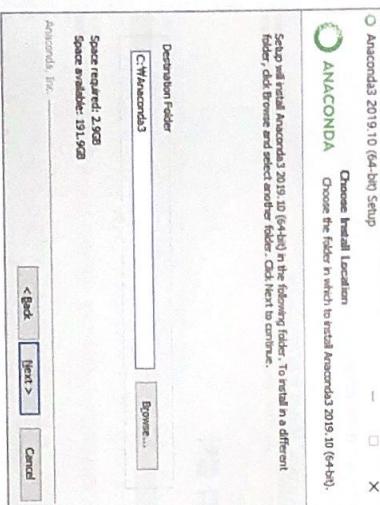


그림 2.18 아나콘다 설치 경로 지정

“다음”을 누르면 두 개의 체크박스가 나오는데, 첫 번째 것은 지정하지 않는 것이 좋습니다. 첫 번째 체크박스를 체크하면 윈도우의 환경변수(PATH)에 아나콘다의 경로를 지정하게 되는데, 파이썬 버전이 바뀌거나 할 때 아나콘다를 다시 설치하는 경우가 있고 그때 환경변수에 이전 경로가 남아 있으면 문제가 발생합니다.

두 번째 체크박스는 기존에 설치된 파이썬이 없을 경우에는 체크하고, 그렇지 않은 경우나 여러 개의 아나콘다(파이썬 3과 파이썬 2를 동시에 사용)를 사용하는 경우에는 체크를 해제하는 것이 좋습니다. 저는 보통 채크하지만 체크하지 않아도 되어서 아나콘다 프로그램으로 아나콘다를 실행한 다음 파이썬을 실행하는 데는 문제가 없습니다.

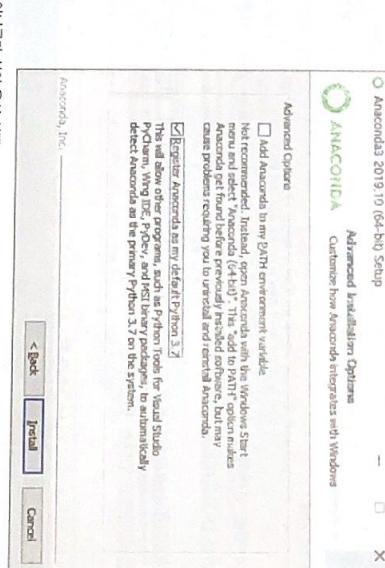


그림 2.19 아나콘다 설치 옵션 선택

설치가 끝나면 윈도우 시작 메뉴에서 “아나콘다 프롬프트(Avacaonda Prompt)”를 선택합니다.

텐서플로 2.0을 설치하기 전에 마지막으로 한 가지 더 해야 하는 작업이 있습니다. 바로 텐서플로에서 팔로로 하는 DLL 파일을 포함하고 있는 Microsoft Visual C++ 2015 Redistributable Update 3 x64를 설치하는 것입니다. 이 URL(<http://bit.ly/2WQat4l>)에서 내려받을 수 있습니다.

X64로 이동한 뒤 스크롤을 아래까지 내려서 “제비포 가능 패키지 x64 버전을 다운로드합니다.”의 제비포 가능 패키지 x64 버전을 다운로드합니다.

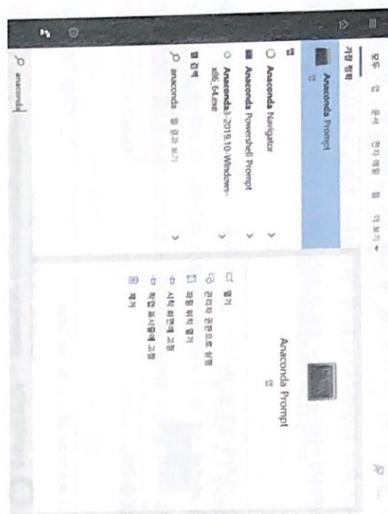


그림 2.20 아니콘다 프롬프트 선택

Anaconda Prompt 앱을 선택하면 아니콘다가 실행됩니다. 아니콘다가 시작되는 경로는 사용자 폴더입니다. 시작 후 폴더 D 드라이브 등으로 바꿔도 상관없습니다.

아니콘다가 정상적으로 설치됐다면 프롬프트 왼쪽에 (base) 표시가 나타납니다. 이는 아니콘다의 기본 환경에서 프로그램을 실행하고 있다는 뜻입니다.



그림 2.21 아니콘다 실행 확인

2.1.4 텐서플로 2.0 설치

아니콘다는 개발환경을 분리해서 관리할 수 있습니다. 여기서는 텐서플로 2.0을 위한 전용 환경을 만들겠습니다. 아니콘다에서 다음과 같은 명령으로 새로운 환경을 만들 수 있습니다.

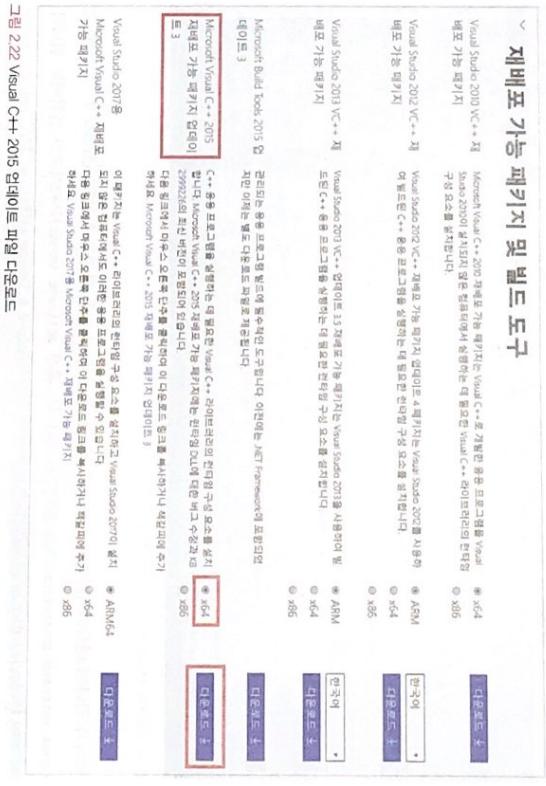


그림 2.22 Visual C++ 2015 업데이트 파일 다운로드

이 끝에는 “`tensorflow2`”라는 이름을 가진 파이썬 3.7 환경(`env`)을 만듭니다. 새로운 환경을 만들기 위해 저는 여러 개의 의존성 라이브러리를 새롭게 내려받아야 하기 때문에 그림 2.23에서 볼 수 있듯이 추가로 라이브러리를 내려받을 때 묻는 창이 나옵니다. ‘Y’를 입력해서 모두 설치합니다.

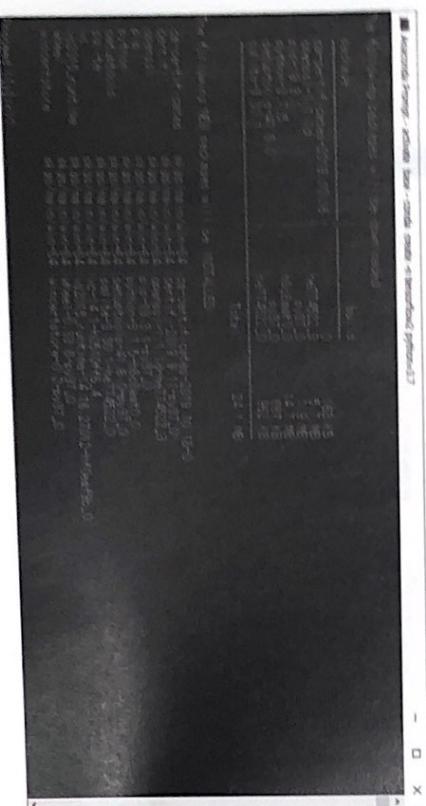


그림 2.23 라이브러리 추가 대화창의 예제 확인

설치가 끝나면 다음과 같은 안내 문구가 표시됩니다. 방금 만든 환경을 활성화(activate)하기 위해서는 `conda activate tensorflow2`라고 입력하면 됩니다. 혹은 `activate tensorflow2`라고만 입력해도 됩니다. 4장 맨 앞 줄 하나로 환경을 활성화합니다.

```
pip install tensorflow-gpu
```

환경이 활성화되면 프롬프트 왼쪽에 붙는 괄호 안에 활성화된 환경의 이름이 표시됩니다. 이제 이 환경에 텐서플로 2.0을 설치하겠습니다. CPU를 사용하는 텐서플로의 라이브러리 이름은 `tensorflow`이고, GPU 기속을 사용하는 버전의 이름은 `tensorflow-gpu`입니다. 따라서 다음과 같은 명령을 입력하면 GPU 가속을 사용하는 텐서플로를 설치할 수 있습니다.

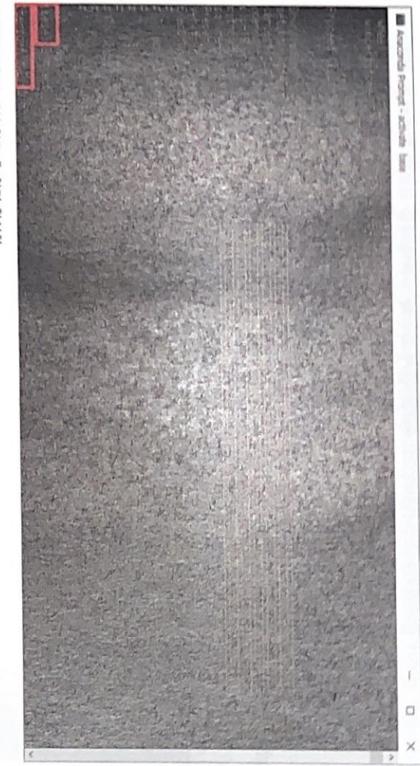


그림 2.25 텐서플로 2.0 설치

텐서플로 설치가 끝나면 GPU가 정상적으로 동작하는지 확인하기 위해 다음 코드를 입력해봅니다.

```
python -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
```

```
[  anaconda Prompt - activate base ]  
$ python -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"  
441.00000000000005  
$
```

그림 2.26 정상 출력 결과 확인

그 밖에 구글 코랩의 주피터 노트북(jupyter notebook)³ 등을 설치하면 코랩처럼 셀 단위로 코드를 편집하고 실행할 수 있습니다. 주피터 노트북은 아래 명령으로 설치합니다.

```
pip install jupyter matplotlib seaborn
```

주피터 노트북 외에 matplotlib과 seaborn 라이브러리 등 여러 개를 동시에 설치할 수도 있습니다. matplotlib과 seaborn은 데이터 시각화에 주로 쓰이는 라이브러리입니다. 이어지는 장에서 자세한 사용법을 배울 것입니다.

지금까지 윈도우 환경에서 텐서플로 2.0을 설치하는 방법을 알아보았습니다.

2.2 macOS

2019년 11월 현재, 맥북 프로와 아이맥 프로 등 애플의 데스크톱 라인은 엔비디아 그래픽 카드 대신 AMD의 라데온(Radeon) 그래픽 카드를 사용합니다. 현재 텐서플로 홈페이지에서는 공식적으로 macOS에서 GPU를 지원하지 않는 것을 명시하고 있습니다.⁴ 따라서 여기서는 CPU를 사용하는 텐서플로의 설치에 대해서만 다루겠습니다.⁵

2.2.1 아나콘다 설치

macOS에서 아나콘다를 설치하려면 2.1.3절의 URL에서 상단의 macOS를 선택해서 전용 설치 파일을 내려받을 수 있습니다. 여기서는 파이썬 3.7 버전의 그래픽 설치 파일을 다운로드하겠습니다.

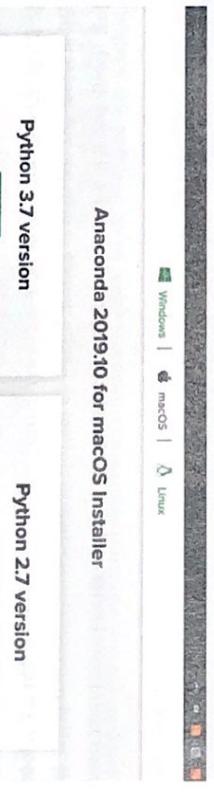


그림 2.27 macOS용 아나콘다 설치 파일 다운로드

³ 파이썬의 코드를 템 터미널에서 연번 터미널로 쉽게 실행할 수 있게 하는 개발 환경으로, 구글 코랩은 바로 이 주피터 노트북을 기본으로 만든 것입니다.

⁴ (<https://anaconda.org>)

⁵ 이 설치 가이드는 macOS 모하비(Mojave) 버전 10.14.6을 기준으로 작성되었습니다.

다운로드가 끝나면 설치 파일을 열어서 설치를 시작합니다. 설치 프로그램의 디자인은 윈도우 버전에 비해 깔끔한 편입니다. 사용권 계약에 동의한 다음 적당한 위치를 지정해서 아나콘다를 설치합니다. 보통 기본 디스크에 설치하면 문제가 없습니다.



그림 2.28 아나콘다 설치 위치 선택

macOS에서는 아나콘다를 설치할 때 아나콘다의 위치를 환경 변수(PATH)에 직접 쓰기 원니다. 동일한 이름으로 macos에서 사용하는 기본 파이썬이 아나콘다의 파이썬으로 바뀝니다. [기존 파이썬 버전은 파이썬 2.7 버전이기 때문에 텐서플로를 주력으로 사용하려면 3.7로 바꾸는 것이 좋습니다. 하지만 기존의 2.7 버전을 지웠다가 문제가 생길 수도 있기 때문에 굳이 자을 필요는 없습니다.]

아나콘다 설치가 끝나면 터미널을 실행합니다. 터미널을 실행하기 위해 Command + 스페이스 바[#]를 눌러서 면자 Spotlight를 실행합니다. 여기서 'terminal'을 입력하면 터미널을 찾을 수 있습니다.



그림 2.29 Spotlight에서 터미널 찾기

터미널에서 커서 앞에 (base) 표시가 보이면 아나콘다의 기본 환경에서 실행 중이라는 뜻으로, 아나콘다가 정상적으로 설치된 것입니다.

⁶ Finder 창에서 Spotlight 검색을 수행하면서 Command + Option + 스페이스 바를 누릅니다. 현재 실행 중인 프로그램에 따라 단축키가 충돌할 수도 있습니다. 예전 악티브 오른쪽 상단 모서리에 있는 풀사이드 아이콘을 클릭해도 Spotlight를 활성화할 수 있습니다.

그림 2.30 아나콘다 설치 확인

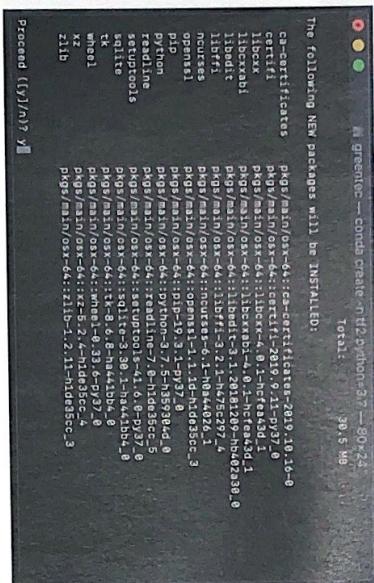
이제 텐서플로 2.0을 설치합니다.

2.2.2 텐서플로 2.0 설치

아나콘다는 개별 환경을 분리해서 관리할 수 있습니다. 여기서는 텐서플로 2.0을 위한 전용 환경을 만들 예보겠습니다. 아나콘다에서 다음과 같은 명령으로 새로운 환경을 만들 수 있습니다.

```
conda create -n tf2 python=3.7
```

이 명령은 “tf2”라는 이름을 가진 파이썬 3.7 환경(env)을 만듭니다. 새로운 환경을 만들기 위해서는 여러분의 의존성 라이브러리를 새롭게 내려받아야 하기 때문에 그림 2.31에서 볼 수 있듯이 추가로 라이브러리를 내려받을지 묻는 창이 나옵니다. Y를 입력해서 모두 설치합니다.



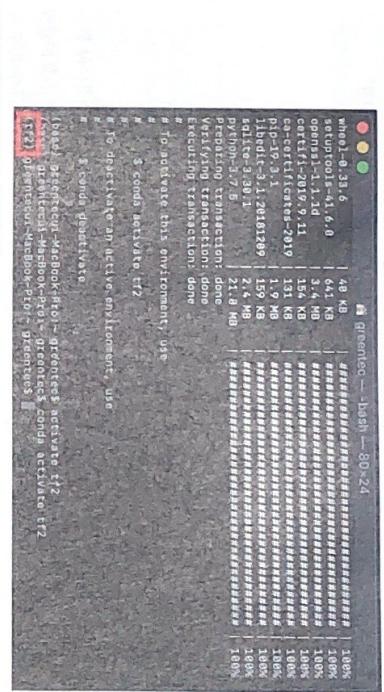
```
conda create -n tf2 python=3.7 -y
```

The following NEW packages will be INSTALLED:

- certifi
- idna
- libcurl
- libffi
- libgcc
- libgccabi
- libgcc_s
- libstdc++
- libxml2
- libxslt
- lz4
- ncurses
- openssl
- pcre
- pycurl
- pygments
- pytz
- python
- readline
- setuptools
- sqlite
- tk
- wheel
- xz
- yaml
- zlib

그림 2.31 라이브러리 추가 다운로드 여부 확인

설치가 끝나면 봄금 만든 환경을 활성화합니다. 윈도우 버전은 activate tf2로 활성화할 수 있었지만 macOS에서는 앞에 conda를 붙여서 conda activate tf2라고 입력해야 합니다.



```
conda activate tf2
```

```
(base) greenetc01-MacBook-Pro:~ greenetc$ pip install tensorflow
```

```
Collecting tensorflow
  Downloading https://files.pythonhosted.org/packages/fe/72/5bb2081f4d78e336fc/tensorflow-2.0.0-py37-cp37m-macosx_10_6.whl (142.7MB)
```

그림 2.32 환경 생성 완료 후 환경 활성화

환경이 활성화되면 프롬프트 왼쪽에 붙는 괄호 안에 활성화된 환경의 이름이 표시됩니다.

이제 이 환경에 텐서플로 2.0을 설치하겠습니다. CPU를 사용하는 텐서플로의 라이브러리 이름은 tensorflow입니다. 따라서 다음과 같은 명령을 입력하면 텐서플로를 설치할 수 있습니다.

```
pip install tensorflow
```

tensorflow 디렉토리 tensorflow==2.0.0-alpha3로처럼 구체적인 버전을 지정할 수도 있습니다. 버전을 입력하지 않으면 자동으로 인강적인 버전을 찾아서 설치를 시작합니다.



```
pip install tensorflow
```

```
Collecting tensorflow
  Downloading https://files.pythonhosted.org/packages/fe/72/5bb2081f4d78e336fc/tensorflow-2.0.0-py37-cp37m-macosx_10_6.whl (142.7MB)
```

그림 2.33 텐서플로 2.0 설치 시작

텐서플로 설치가 끝나면 정상적으로 동작하는지 확인하기 위해 다음 코드를 입력해봅니다.

텐서플로 2.0 시작하기

정상적으로 실행되면 (1000, 1000) 크기의 정규분포 랜덤값의 총합을 반환합니다.

```
python -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
```

```
(ipython3) greentea@MacBook-Pro: ~ Greentea
```

그림 2.34 정상 출력 결과 확인

그 밖에 구글 코랩의 원조적인 주피터 노트북 등을 설치하면 코랩처럼 셀 단위로 코드를 편집하고 실행 할 수 있습니다. 주피터 노트북은 아래 명령으로 설치합니다.

```
pip install jupyter matplotlib seaborn
```

주피터 노트북 외에 matplotlib과 seaborn 라이브러리 등 여러 개를 동시에 설치할 수도 있습니다.

matplotlib과 seaborn은 데이터 시각화에 주로 쓰이는 라이브러리입니다. 이어지는 장에서 자세한 사용법을 배울 것입니다.

지금까지 macOS 환경에서 텐서플로 2.0을 설치하는 방법을 알아왔습니다.

3.1 Hello World

텐서플로는 C++와 파이썬 언어를 기본으로 합니다. 다른 프로그래밍 언어도 사용할 수 있지만 일반적으로는 파이썬이 가장 많이 쓰이고 구글 코랩도 파이썬을 기본적으로 지원하기 때문에 이 책에서는 파이썬에서 실행되는 예제를 공부할 것입니다.

Hello World는 화면에 "Hello, World!"를 출력하는 프로그램입니다. 많은 프로그래밍 책에서 처음 소개하는 예제로, 원하는 내용을 화면에 출력할 수 있는지 테스트하는 간단한 연습용 프로그램입니다.

예제 3.1 Hello World 프로그램

```
[IN]
# 3.1 Hello World 프로그램
print("Hello, World!")
```

이미지

Hello, World!

우리 코드를 확인하기 위해서는 먼저 구글 코랩에 접속해야 합니다. 이 페이지¹로 들어가면 해당 코드를 확인할 수 있습니다. 또한 깃허브 저장소²에서도 확인할 수 있습니다.

구글 코랩에 접속한 뒤 위의 코드를 실행하기 위해서는 먼저 가장 첫 번째 셀³을 클릭해서 커서를 위치시킵니다.



그다음 셀의 좌측에 생긴 플레이 버튼을 누르거나, 단축키로 Ctrl + Enter를 누르면 셀이 실행됩니다.

3.1 Hello World

```
# 3.1 Hello World 프로그램
print("Hello, World!")
```

그림 3.1 구글 코랩에서의 실행 화면

파이썬을 처음 접하는 분들은 위해 위 코드에 대해 간단히 설명하자면 첫 줄의 '#'는 다음에 나오는 내용이 주석으로 처리됨을 의미합니다. 즉, 어떤 내용을 적더라도 '#' 다음의 내용은 실행되지 않습니다. '#'을 자우면 아래와 같이 에러가 발생할 것입니다.

```
[IN]
3.1 Hello World
print("Hello, World!")
```

예제 3.2 Hello World 프로그램에서 첫 줄의 '#'를 지울 때 에러가 나는 모습

```
[IN]
File <ipython-input-2-8c1e222b3f>, line 1
    ^
SyntaxError: invalid syntax
```

주석은 프로그램의 일부로 실행되지는 않지만 코드를 읽는 사람이 알아두면 좋은 유용한 정보를 적어놓을 때 필요합니다. 여기서는 이 예제의 이름이 '3.1 Hello World'라는 것을 알려주기 위해 주석을 사용했습니다. 이 책의 예제 코드에는 독자 여러분의 이해를 돋우기 위한 주석이 많이 포함될 것입니다.

코드 세 줄의 부분은 명령의 내용을 실행했을 때의 출력 결과를 나타냅니다. 이 책에서는 출력 결과를 생략하는 경우도 있겠지만 금지 이해를 돋우기 위해 출력 결과를 코드 본문과 함께 표시하겠습니다.

명령을 실행하는 방법은 명령을 입력한 셀에 커서가 위치한 상태에서 단축키 Ctrl + Enter를 누르거나, 구글 코랩의 상단 메뉴에서 [런타임] → [초기 및 취진 셀 실행]을 차례로 선택하면 됩니다.

예제 3.2의 두 번째 줄에 있는 print는 print 키에 나오는 괄호 안의 내용을 화면에 출력하는 함수입니다. 여기서는 "Hello, World!"를 화면에 출력하고 있습니다. 큰따옴표("")는 문자열 데이터를 감싸는 기호로 서 대부분의 프로그래밍 언어에서 사용되는 기호입니다. 파이썬에서는 작은따옴표('')도 문자열 데이터를 감싸는 용도로 쓸 수 있습니다. 따라서 예제 3.3도 위와 동일하게 동작합니다.

예제 3.3 Hello World 프로그램의 문자열에 사용된 큰따옴표(")를 작은따옴표(')로 바꿈

```
[IN]
print('Hello, World!')
```

1 <https://colab.research.google.com>
2 <https://github.com/AMMAG>
3 코드나 이미지를 저장할 수 있는 코랩의 기본 단위입니다. 코드의 경우 출력 결과가 함께 저장됩니다.

한국에서 활동하는 외국인 노동자들은 대부분 노동자로 활동하면서 동시에 노동자로서의 권리와 책임을 갖는다. 그러나 노동자로서의 권리와 책임은 노동자로서의 권리와 책임을 갖는다.

3.2 Hello 웹서버 2.0

이제 반드시 청진기에서도 100%, 빠르게 진단할 수 있는 예상치 외에는 모든 진단을 청진기로 확정하는 데에 대한 확신이 있다. 예전에는 청진기로 진단하는 경우가 많았지만 최근에는 청진기로 진단하는 경우가 적어졌다.

여행의 길도, 여행의 일도, 여행의 주제도, 여행의 희망도, 여행의 일상도 그렇듯 여행의 철학도 여행의 철학입니다. 여행의 철학은 여행을 통해 얻을 수 있는 철학입니다.

卷之三

즉, 총 4개가 불는다는 점입니다. 출력 결과를 토대로 2.0.0 버전이 설치된 것을 확인할 수 있습니다. 여기서 실수하기 쉬운 점 중 하나는 tensorflow를 tf로 줄였기 때문에 원래 이름인 tensorflow를 사용해서 print(tensorflow._version_)로 실행해보면 에러가 발생한다는 것입니다.

예제 3.6 텐서플로 불러오기 버전 확인 에러

```
[IN]
import tensorflow as tf
print(tensorflow._version_)

[OUT]
NameError: name 'tensorflow' is not defined
```

이렇게 해서 텐서플로의 버전을 확인하는 부분까지 살펴봤습니다. 이제는 텐서플로에서 할 수 있는 기초적인 연산을 배우고, 간단한 신경망 레이어와 네트워크를 만들어 보겠습니다.

3.3 텐서플로 기초

인공지능은 현재 과학계에서 가장 각광받는 분야지만을 지금처럼 좋은 시절만 있던 것은 아니었습니다. '인공지능의 계절(Al winter)'이라고 불리는 시기가 두 차례가 있었는데, 그중 첫 번째는 신경망을 구성하는 뉴런의 원조인 퍼셉트론(perceptron)의 한계를 지적한 같은 이름의 책, 〈퍼셉트론〉의 발간이 어느 정도 영향을 끼쳤습니다. 퍼셉트론의 한계를 지적하는 데 사용했던 AND, OR, XOR 연산을 할 수 있는 신경망 네트워크를 함께 만들어보며 이를 구체적으로 알아보겠습니다.

3.3.1 난수 생성

먼저 난수(Random Number)를 만들어보겠습니다. 랜덤은 신경망에서 꼭 필요한 기능입니다.

신경망을 쉽게 정의해보면 많은 숫자로 구성된 행렬이라고 할 수 있습니다. 이 행렬에 어떤 입력을 넣으면 출력을 얻게 되고, 잘 작동할 경우 원하는 출력에 점점 가까워지게 됩니다.

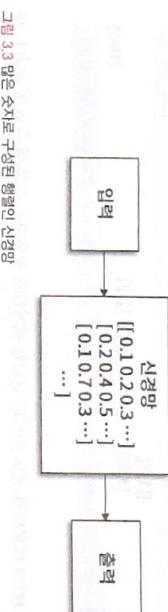


그림 3.3 맵은 숫자로 구성된 행렬인 신경망

그런데 여기서 행렬을 구성하는 숫자는 어떻게 구할 수 있을까요? 처음에는 이 숫자들을 랜덤한 값으로 지정해 줄 수밖에 없습니다. 처음에 신경망의 초기값을 지정해주는 것을 초기화(Initialization)라고 하며, 이에 관련해 세도 많은 논문들이 나와 있습니다. 현재 가장 많이 쓰이는 방법은 Xavier 초기화(Xavier Initialization)와 He 초기화(He Initialization)인데, 이 방법들은 랜덤하지만 어느 정도 규칙성이 있는 범위 내에서 난수를 지정합니다. 여기서는 이 방법들보다 단순하게 난수를 지정해보겠습니다.

텐서플로에서 난수를 생성하는 방법은 아래와 같습니다.

```
[IN]
rand = tf.random.uniform([1], 0, 1)
print(rand)

[OUT]
tf.Tensor([0.4458922], shape=(1,), dtype=float32)
```

예제 3.7 난수 인가(균일 분포)

tf.random.uniform 함수를 불러오면 균일 분포(uniform distribution)의 난수를 얻을 수 있습니다. 균일 분포란 최솟값과 최댓값 사이의 모든 수가 나올 확률이 동일한 분포에서 수가 뽑는다는 뜻입니다. 처음에 나오는 [1]은 걸맞은 Shape을 의미합니다. Shape이란 행렬을 구성하는 행, 열 등 차원의 수를 나타내는 값입니다.

```
[5] → shape=[1]
[[2], [3]] → shape=[2,1]
[[1,2], [3,6]] → shape=[2,2]
```

그림 3.4 행렬 Shape의 예

Shape을 확인하려면 print 명령어를 써도 되지만 간단한 데이터의 경우에는 직접 셀 수도 있습니다. 세는 방법은 바깥쪽부터 인쪽으로 원소의 개수를 세는 것입니다. 앞의 [[2], [3]]의 경우 가장 바깥쪽은 원소가 [2], [3]으로 2개이고, 인쪽으로 들어가면 2, 3이 각각 하나씩 있기 때문에 shape=[2,1]이 됩니다.⁶

tf.random.uniform의 두 번째와 세 번째 인수인 0, 1은 각각 최솟값과 최댓값을 의미합니다. 즉, 여기서는 최솟값 0과 최댓값 1 사이에서 모든 수가 나올 확률이 동일한 분포에서 난수 하나를 뽑는 것입니다. 여기서는 tf.Tensor 바로 다음에 나오는 대괄호([]) 안의 0 4458922라는 값을 얻었습니다.

Shape은 1인 것을 확인할 수 있습니다. 파이썬에서는 번하지 않는 값에 대해 대괄호 대신 소괄호(())로 묶는 투플(tuple) 자료형을 사용합니다. 이때 원소의 개수가 한 개인 경우에도 앞에 서치처럼 (1,)로 원소 뒤에 쉼표를 찌워서 이 값이 투플이라는 것을 나타냅니다. 이 같은 입력할 때의 [1]과 동일한 값입니다.

마지막의 dtype은 자료형(data type)을 의미합니다. float32는 32비트의 부동소수점 수⁷라는 뜻입니다. 렌서플로 2.0에는 16비트 부동소수점 수인 float16과 64비트 부동소수점 수인 float64도 존재합니다. 특별한 설정이 없다면 앞에서 확인할 수 있도록 렌서플로는 float32 값을 tf.random.uniform의 출력으로 반환합니다.

그런데 여기서는 1 이상의 값도 나오고 음수도 나오고 있습니다. tf.random.normal의 두 번째와 세 번째 인수는 앞에 나온 tf.random.uniform과 다릅니다. 여기서 두 번째의 0은 정규 분포의 평균(mean), 세 번째의 1은 정규 분포의 표준편차(standard deviation)를 의미합니다. 평균이 0이고 표준편차가 1일 때의 정규 분포를 표준정규분포(standard normal distribution)라고 합니다. 두 번째, 세 번째 인수가 0, 1로 같을 때 균일 분포와 정규 분포의 차이를 그래프로 나타내면 다음 페이지의 그림 3.5와 같습니다.

예제 3.9 난수 얻기(정규 분포)

```
[IN] rand = tf.random.uniform([4], 0, 1)
print(rand)

[OUT]
tf.Tensor([0.4000026  0.02589869  0.97795693  0.37491727], shape=(4,), dtype=float32)
```

6 만약 [[2], [3]]처럼 특정 단계에서 원소의 개수가 다른 경우에는 shape를 특성을 수 없습니다.

7 부동소수점이라는 말은 소수점 뒷자리나는 일로 계산된 비트 인덱스 대신 수를 표현하기 위해 소수점 뒷자리는 것입니다. 비트 수가 커질수록 표현할 수 있는 숫자의 범위는 넓어지지만 예외를 막기 위해 계산 속도가 느린 단점도 있습니다. 부동소수점 수는 제한된 비트수에서 실제적인 값을 표현하기 위해 예전에 정해진 수치가 아닌 근사를 씁니다.



그림 3.5 군일 분포와 정규 분포의 차이

군일 분포와 정규 분포에 대해 천만 개의 샘플을 각각 구해서 히스토그램으로 나타내면 군일 분포는 좌우 대칭과 최댓값 사이에서 일정한 확률의 난수가 나타나게 되지만 정규 분포는 주어진 평균과 표준편차를 만족하는 중형곡선(bell curve)을 그리게 됩니다. 정규 분포는 학생들의 시험 성적이나 카, 농작물의 무게 등 실생활에서 자주 볼 수 있는 분포 형태입니다. 앞에서 소개한 Xavier 초기화나 He 초기화는 군일 분포와 정규 분포 중 하나를 선택해서 신경망의 초기값을 초기화합니다.

3.3.2 뉴런 만들기

이제 신경망의 가장 기본적인 구성요소인 뉴런을 만들어보겠습니다. 이전에는 뉴런을 퍼셉트론이라고도 불렀으며, 입력을 받아서 계산 후 출력을 반환하는 단순한 구조입니다(그림 3.6).

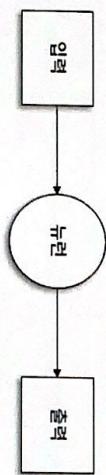


그림 3.6 초상화한 뉴런의 구조

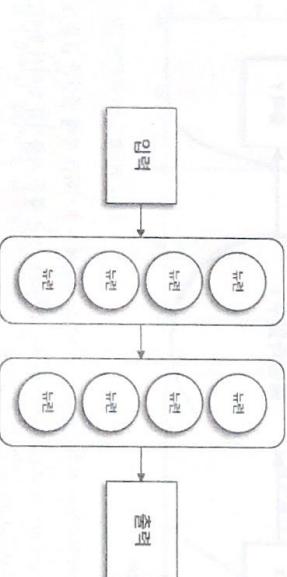


그림 3.7 뉴런과 레이어로 구조화한 신경망의 구조

뉴런은 입력, 가중치, 활성화함수, 출력으로 구성됩니다.



그림 3.8 뉴런의 구성요소

입력, 가중치, 출력은 보통 정수(integer)나 앞에서 살펴본 float이 많이 쓰입니다. 활성화함수는 뉴런의 출력값을 정하는 함수입니다. 가장 간단한 형태의 뉴런은 입력에 가중치를 곱한 뒤(행렬의 곱셈입니다) 활성화함수를 취하면 출력을 얻을 수 있습니다.

사실 신경망은 뉴런이 여러 개 모여 레이어(layer)를 구성한 후, 이 레이어가 다시 모여 구성된 형태입니다. 참고로 뉴런과 레이어를 우리말로 각각 '신경 세포'와 '층'이라고 할 수도 있겠지만 이 책에서는 의미상의 혼란을 막기 위해 뉴런, 레이어라고 표기하겠습니다.

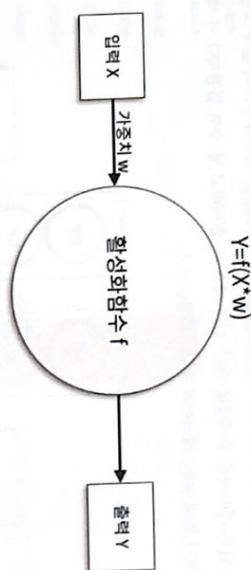


그림 3.9 뉴런의 출력 계산식

뉴런에서 학습할 때 변하는 것은 가중치입니다. 가중치는 처음에는 초기화를 통해 랜덤한 값을 넣고, 학습 과정에서 점차 일정한 값으로 수렴합니다. 학습이 잘 된다는 것은 좋은 가중치를 얻어서 원하는 출력에 점점 가까운 값을 얻을 것이라고 할 수 있습니다.

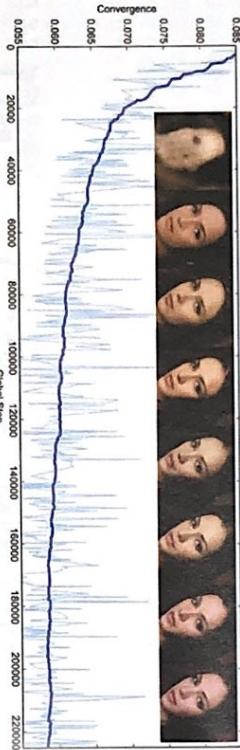


그림 3.10 딥러닝 학습을 이용한 가상 인물의 생성 시례⁸

활성화함수로는 시그모이드(sigmoid), ReLU 등을 주로 쓰게 됩니다. 시그모이드는 S자 형태의 극선이라는 뜻입니다. ReLU는 Rectified Linear Unit의 약자로, 정류된(rectified) 선형 함수(linear unit)라는 뜻입니다. 딥러닝에서 선형 함수는 $y=x$ 라는 식으로 정의할 수 있는 입력과 출력이 동일한 함수를 의미합니다. 이 함수를 정류해서 음수 값을 0으로 만든 것이 ReLU입니다. 그레프로 나비내면 다음과 같습니다.

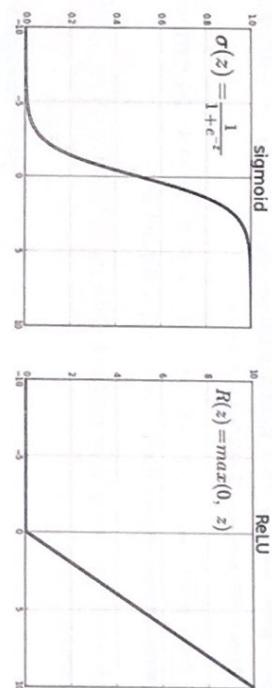


그림 3.11 시그모이드와 ReLU 함수의 그래프

신경망 초창기에는 시그모이드가 주로 쓰였지만 은닉층⁹을 다수 사용하는 딥러닝 시대가 되면서 ReLU 가 더 많이 쓰이고 있습니다. 딥러닝에서 오류를 역전파(backpropagation)할 때 시그모이드 험수가 값 을 점점 작아지게 하는 문제를 토대로 대학교의 비노드 네어(Vinod Nair)와 제프리 힌튼(Geoffrey Hinton) 교수가 시작하며, ReLU를 대안으로 제시한 2010년 논문¹⁰에 이에 대한 자세한 내용이 나와 있습니다. 앞의 그레프에서도 시그모이드는 출력값을 0~1 사이로만 제한하게 되지만 ReLU는 양수를 그대로 반환하기 때문에 값의 외곡이 적어집니다.

여기서는 시그모이드 함수를 활성화함수로 사용해보겠습니다. 출력을 0~1 사이로 제한해서 뒤에서 살펴볼 AND, OR, XOR 연산을 다루는 데 적합하기 때문입니다.

먼저 시그모이드 함수를 파이썬으로 구현해 보겠습니다.

예제 3.10 시그모이드 함수

```
import math
def sigmoid(x):
    return 1 / (1 + math.exp(-x))
```

⁹ 신경망에서 입력층과 출력층을 제외한 나머지 뉴런은 은닉층이라고 합니다. 은닉층은 가중치와 활성화함수로 구성되며, 입력에 대한 양은 출력을 학습하는 신경망의 핵심 구성요소입니다. 같은 기준으로 입력 뉴런은 3회 이상 사용되는 딥러닝으로 간주합니다.

¹⁰ V. Nair and G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in Proc. 27th International Conference on Machine Learning, 2010.

첫 줄에서는 앞에서 tensorflow 모듈을 불러왔던 것처럼 `math` 모듈을 불러옵니다. `math` 모듈은 상수 e 의

두 번째 줄에서 정의한 sigmoid() 함수는 x 를 입력으로 받고 return 명령을 통해 출력값을 반환합니다. 여기서 반환하는 $1/(1+math.exp(-x))$ 가 x 에 시그모이드 함수를 취한 값이 됩니다.

정사(강법) 효과를 발휘하는지 코드로 확인해보겠습니다.

예제 3.12 경사 허강법을 이용한 뉴런의 학습

그럼 이제 뉴런의 입력과 출력을 정의해 보겠습니다. 입력이 1일 때 $g_{\text{input}}(1)$ 이 0이 되는 뉴런을 만드는 데 어떤 조건을 걸까요?

예제 3.11 뉴런의 입력과 출력 정의

$$\begin{aligned} \text{primitivo} &= \text{outlet} = \text{sumidoir} \times (\mathbf{A}) \\ &= \text{radio_narrow} \left(\left[\begin{array}{c} 1 \\ 0 \end{array} \right] \right) \\ &= \theta = y \\ &L = x \end{aligned} \quad [\text{N}]$$

[OUT]

함수에 입력과 출력을 같은 값을 넣어서 계산합니다.

실제 출력으로 나온 0.43007과 기대 출력인 0의 차이인 $0 - 0.43007 = -0.43007$ 을 에러(error)라고 합니다. 뉴런의 학습은 이 에러가 0에 가까워지게 해서 출력으로 기댓값에 기울기 값을 얻는 것입니다.

여기서 뉴리어린 결국 ∇ 같았습니다. 이제 ∇ 를 변화시켜야 하는데, 어떤 알고리즘을 사용하느냐에 따라서 차이가 있습니다. 경사 하강법(Gradient Descent)이라는 방법을 사용하면 되는데, 이것은 ∇ 에 일련의 학습률(α)과 이례로 같은 학습률을 더해주는 것입니다.¹² 학습률은 α 를 업데이트하는 정도로, 큰 값으로 설정하면 학습이 빨라지지만 과도한 학습으로 적정한 수치를 벗어날 수가 있고, 너무 작은 값으로 설정하면 학습 속도가 너무 느려질 수 있습니다. 여기서는 $\alpha=1.0$ 로 설정하겠습니다.

11. **개인화된 유익한 노하우를 기록하는** 문집으로 정답에 해당합니다. 실무적인 노하우나 개인의 실력을 기록하는 글에서는 같았습니다. 학습 시스템이 전래 기록방법과 실무성을 고려해 개인화된 노하우를 기록하는 문집입니다.
12. **제가 알거나** 책이나 책에서 알게 된 내용이나 기록방법을 기록하는 문집입니다. 예전에는 노트로 기록해 두었지만 최근에는 노트에 대체로 디지털 기록방법으로 전환되었습니다. 그래서 노트에 기록한 내용을 디지털로 기록하는 문집입니다. 예전에는 노트에 기록한 내용을 디지털로 기록하는 문집입니다. 예전에는 노트에 기록한 내용을 디지털로 기록하는 문집입니다.

44 시장세계로 텐서플로 2.0 프로그래밍: 기초 이론부터 실전 예제까지

```
[IN] for i in range(1000):  
    output = sigmoid(x * w  
    error = y - output  
    w = w + x * 0.1 * error
```

```
if i % 100 == 99:  
    print(i, error, output)
```

[out]
99 -0.07284613375703874 0.07284613375703874 199 -0.02218122340039822 0.02218122340039822 299 -0.016153670688397104 0.0116137067658397104 399 -0.00547949044093222 0.00547949044093222 499 -0.004243580712586666 0.004243580712586666 599 -0.00343469043384763 0.00343469043384763 799 -0.002858305114466369 0.002858305114466369 899 -0.00245197953421888 0.00245197953421888 999 -0.002134366548119732 0.002134366548119732

for는 반복문을 나타내는 명령어입니다. i는 반복문에서 사용할 변수이고, range(1000)은 [0, 1, 2, 3, ..., 998, 999] 형태의 리스트를 자동으로 생성하는 명령어입니다. 정리하면 이 반복문은 $i=0$ 부터 $i=999$ 까지 1,000번 동안 for 문 안의 내용을 반복하게 됩니다.

출력을 구하는 부분은 위와 동일하고 error를 구하는 식은 기대출력인 `output`에서 실제출력인 `output`을 뺀다. 이렇게 구한 error를 정사-하강법의 업데이트 식에 넣어 `w` 값을 다시 계산합니다. if 문은 프로그램을 제어하는 명령어이로 if 다음에 나오는 부분이 참(true) 값일 때 블록 안의 내용을 실행하는 것입니다. %는 나머지를 구하는 연산자입니다. i=99, i=99, ..., i=999 일 때 if 문 안의 블록이 실행되어 i, error, output 값을 출력합니다. error 값은 0에 점점 가까워지고, output도 기대출력인 0에 가까워지는 것을 확인할 수 있습니다.