

2020~21 겨울특강

텐서플로 기반 딥러닝

2일 1교시

동양미래대학교 컴퓨터정보공학과 강환수 교수

딥러닝 참고 사이트

- 텐서플로
 - <https://www.tensorflow.org/>
- 머신러닝 단기집중과정
 - <https://developers.google.com/machine-learning/crash-course>
- Naver D2
 - <https://d2.naver.com/home>
- Naver Tech Talks
 - <https://d2.naver.com/news/2657726>
- Naver edwith 인공지능
 - <https://www.edwith.org/search/index?categoryId=71>
- 논문으로 짚어보는 딥러닝의 맥
 - <https://www.edwith.org/deeplearningchoi>
- 모두를 위한 머신러닝/딥러닝(성김 교수)
 - <https://hunkim.github.io/ml/>
- 모두를 위한 딥러닝 시즌 2
 - <https://deeplearningzerotoall.github.io/season2/>



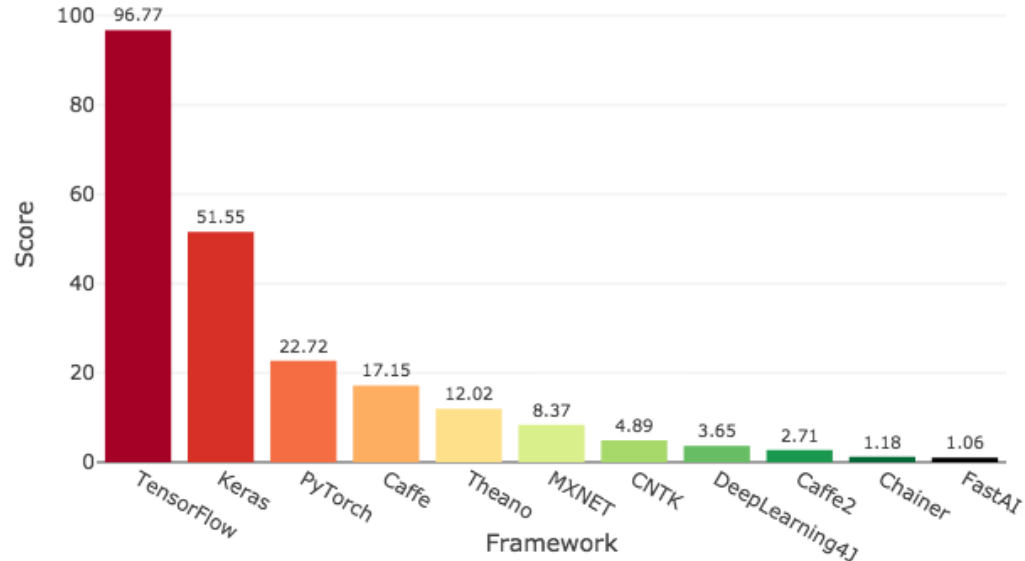
텐서플로 개요

참고 <https://excelsior-cjh.tistory.com/148> [EXCELSIOR]

딥러닝 라이브러리(플랫폼) 개요

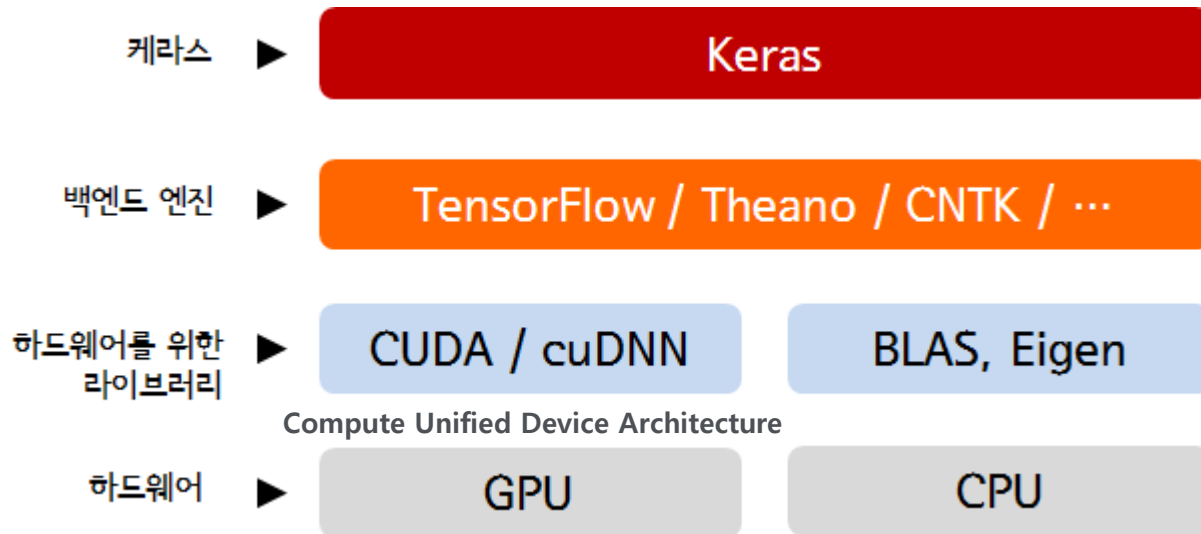
- 딥러닝 구현을 위한 클래스 및 함수 제공
- 다양한 라이브러리 활용
 - 텐서플로, 케라스, 파이토치
 - 가장 적합한 언어는 파이썬

Deep Learning Framework Power Scores 2018



케라스의 개요

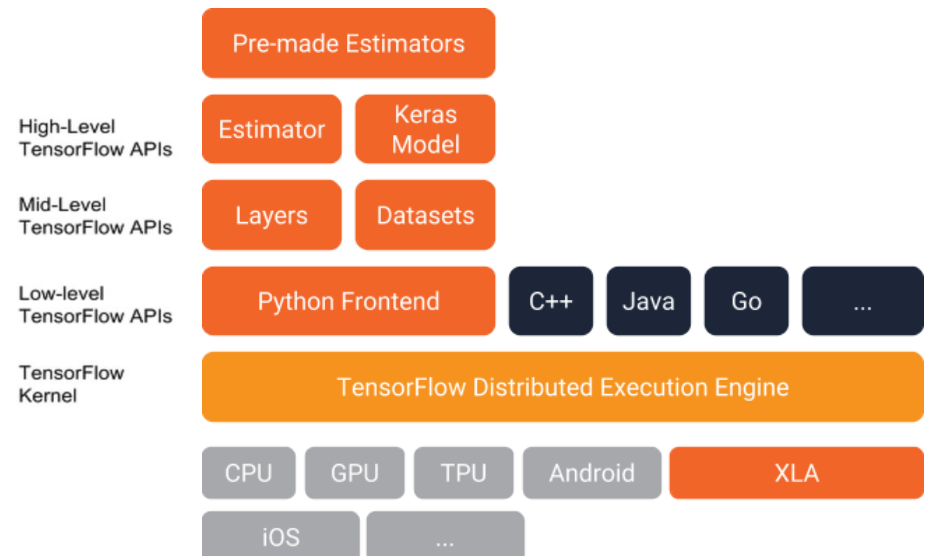
- **원래는 독자적인 고수준 라이브러리**
 - 엔진으로 텐서플로, 씨아노, CNTK 등을 사용
- **현재는 Tensorflow의 고수준 API로도 사용**
 - 동일한 코드로 CPU와 GPU에서 실행 가능
 - 사용하기 쉬운 API를 가지고 있어 딥러닝 모델의 프로토타입을 빠르게 생성



Basic Linear Algebra Subprograms
 Compute Unified Device Architecture
 CUDA Deep Neural Network library

텐서플로(TensorFlow) 개요

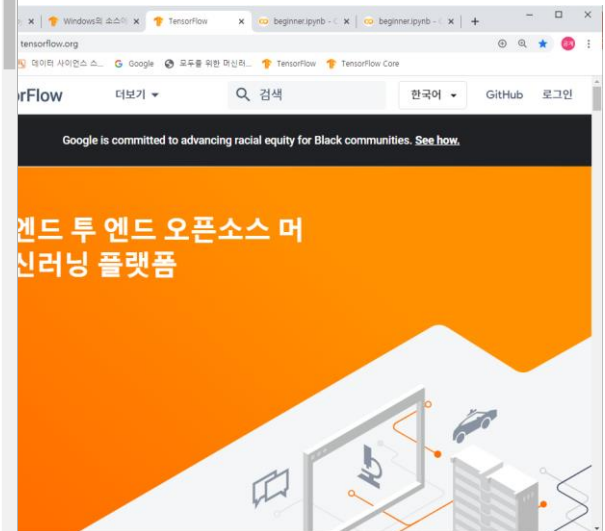
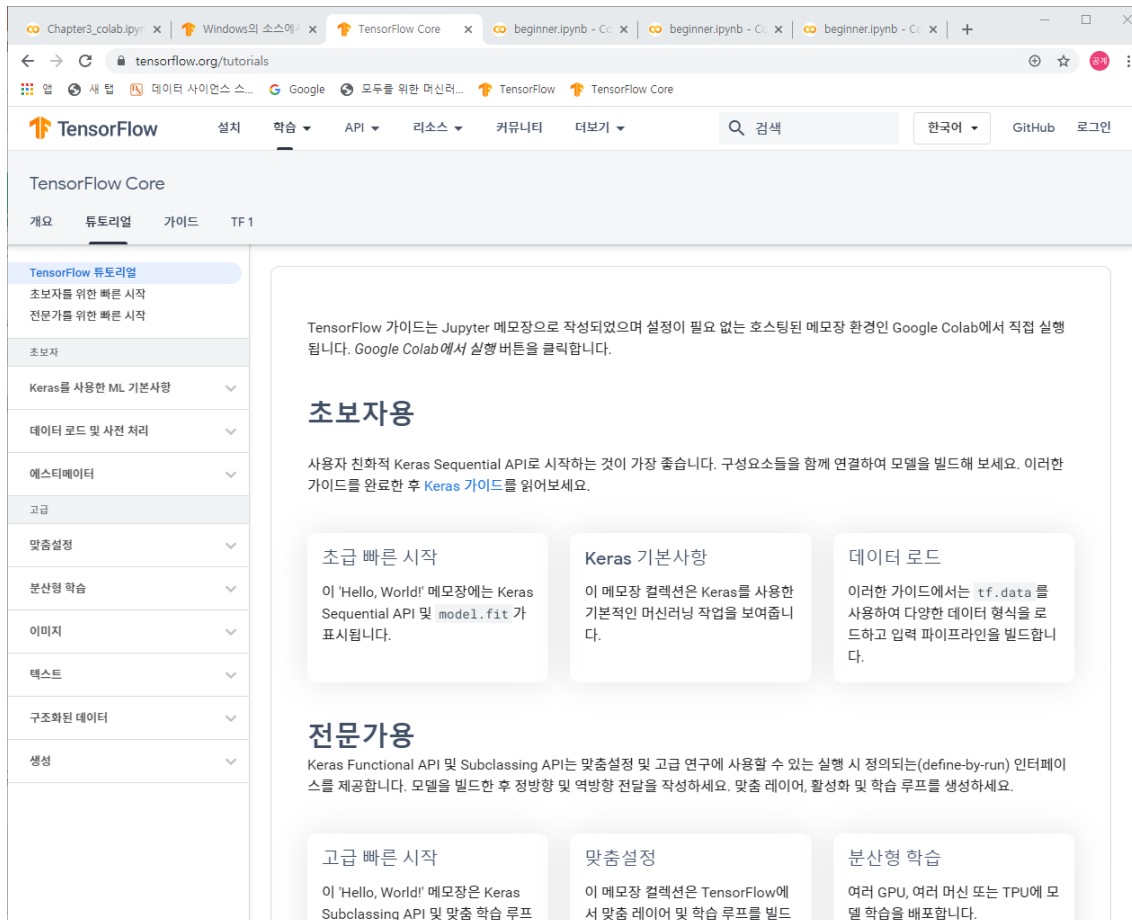
- 구글(Google)에서 만든 라이브러리
 - 연구 및 프로덕션용 오픈소스 딥러닝 라이브러리
 - www.tensorflow.org
 - 딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공
 - 데스크톱, 모바일, 웹, 클라우드 개발용 API를 제공
 - 구현 및 사용
 - Python, Java, Go 등 다양한 언어를 지원
 - 텐서플로 자체는 기본적으로 C++로 구현
 - 파이썬을 최우선으로 지원
 - 대부분의 편한 기능들이 파이썬 라이브러리로만 구현
 - Python에서 개발하는 것이 편함



텐서플로 홈페이지

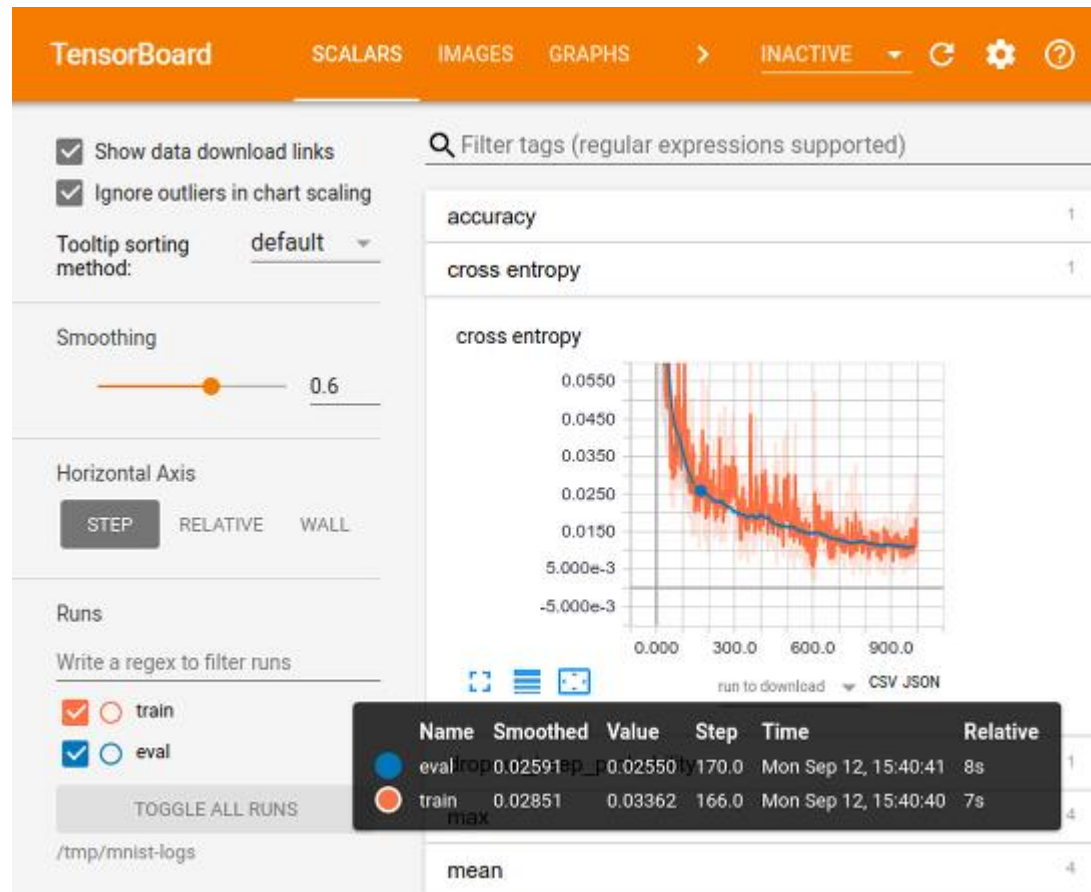
• 튜토리얼

— <https://www.tensorflow.org/tutorials>



텐서보드(TensorBoard)

- 브라우저에서 실행 가능한 시각화 도우미
 - 딥러닝 학습 과정을 추적하는데 유용하게 사용



텐서 개요

- **Tensor(텐서): 모든 데이터**
 - 딥러닝에서 데이터를 표현하는 방식
 - 0-D 텐서 : 스칼라
 - 1-D 텐서 : 벡터
 - 2-D 텐서 : 행렬 등
 - n차원 행렬(배열)
 - 텐서는 행렬로 표현할 수 있는 n차원 형태의 배열을 높은 차원으로 확장

Scalar Vector Matrix Tensor

1

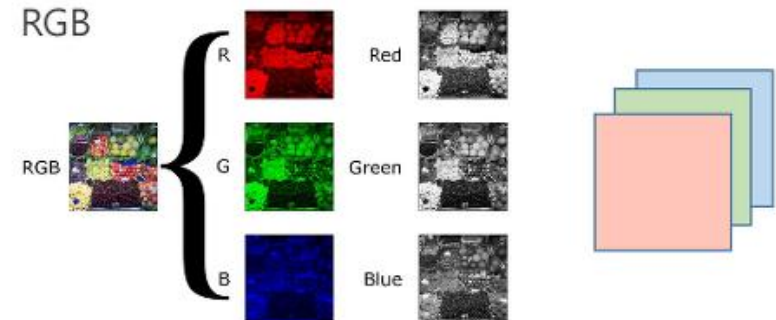
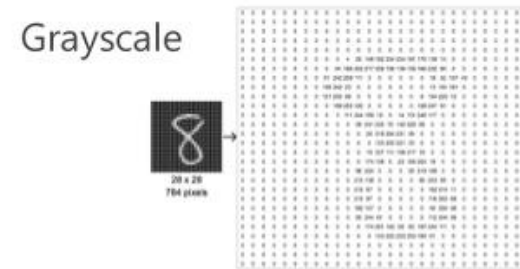
$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 2 \end{bmatrix} \\ \begin{bmatrix} 1 & 7 \end{bmatrix} & \begin{bmatrix} 5 & 4 \end{bmatrix} \end{bmatrix}$$

텐서의 사례

- 스칼라: 차원이 없는 텐서
 - 10
- 벡터 값: 1차원 텐서
 - [10, 20, 30]
- 2차원 행렬: 2차원 텐서
 - 회색조(grayscale) 이미지
 - 하나의 채널(channel)에 2차원 행렬(배열)로 표현
 - [[1, 2, 3], [4, 5, 6]]
- 텐서: n차원 행렬
 - 텐서의 차원을 텐서의 rank(순위)라 함
 - RGB 이미지
 - R(ed), G(reen), B(lue) 각 3개의 채널마다 2차원 행렬(배열)로 표현하는데, 이를 텐서(3차원의 값을 가지는 배열)로 표현
 - [[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]]



TensorFlow 계산 과정

- TensorFlow에서 텐서(Tensor) 계산 과정

- 모두 그래프(Graph)라고 부르는 객체 내에 저장되어 실행
- 그래프를 계산하려면 외부 컴퓨터에 이 그래프 정보를 전달하고 그 결과값을 받아야 함

- Session

- 이 통신과정을 담당하는 것이 세션(Session)이라고 부르는 객체
- 생성, 사용, 종료 과정이 필요
- 세션 생성
 - Session 객체 생성
- 세션 사용
 - run 메서드에 그래프를 입력하면 출력 값을 계산하여 반환
- 세션 종료
 - close 메서드
 - with 문을 사용하면 명시적으로 호출 불필요

```
x = tf.constant(3)
y = x**2
```

```
sess = tf.Session()
print(sess.run(x))
print(sess.run(y))
sess.close()
```

데이터 흐름 그래프(dataflow graph)

TensorFlow에서 계산

- 데이터 흐름 그래프 (dataflow graph)로 이루어짐
- 텐서 형태의 데이터들이 딥러닝 모델을 구성하는 연산들의 그래프를 따라 흐르면서 연산이 일어남

Tensor + DataFlow

- 딥러닝에서 데이터를 의미하는 Tensor 와 DataFlow Graph를 따라 연산이 수행되는 형태 (Flow)를 합쳐 TensorFlow란 이름이 나오게 됨

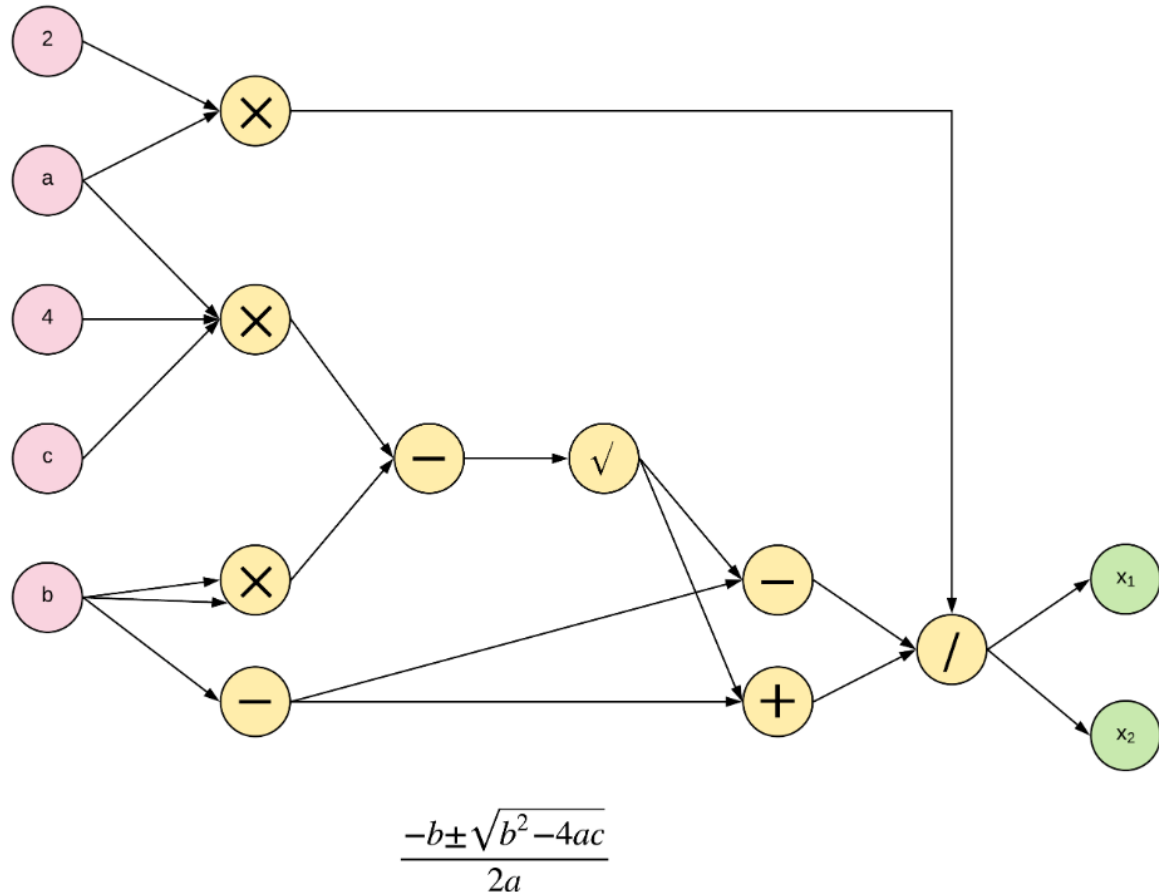


그림 1: 2차 공식을 사용하여 2차 표현식의 근을 계산하기 위한 계산 그래프.

TensorFlow API 탐색

TensorFlow API 계층

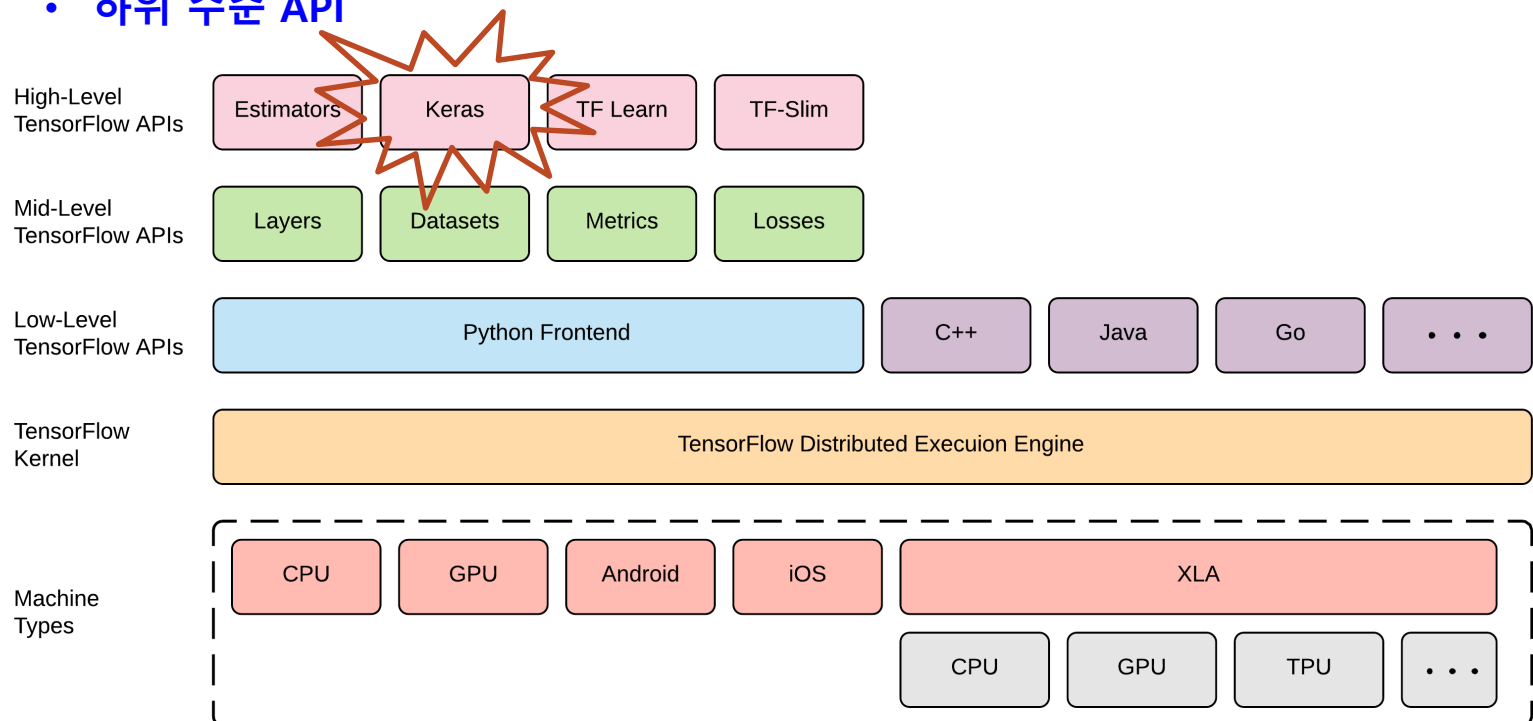
– TensorFlow 딥 러닝 모델 구축 작업은 서로 다른 API 수준을 사용하여 해결

• 고급 API

– Keras나 TF-Slim 과 같은 추상화 라이브러리를 제공하여 저수준 텐서플로 라이브러리에 대해 손쉽게 고수준 접근이 가능하게 해줌

• 중급 API

• 하위 수준 API



텐서플로 vs 케라스

- 케라스, 텐서플로 뭐가 좋아요?

If you're asking "*Keras or TensorFlow?*"



VS.



Then you're asking the *wrong* question (and here's why...)

텐서플로 API

- API

- https://www.tensorflow.org/api_docs/python/tf

Modules

`audio` module: Public API for tf.audio namespace.

`autodiff` module: Public API for tf.autodiff namespace.

`autograph` module: Conversion of plain Python into TensorFlow graph code.

`bitwise` module: Operations for manipulating the binary representations of integers.

`compat` module: Compatibility functions.

`config` module: Public API for tf.config namespace.

`data` module: `tf.data.Dataset` API for input pipelines.

`debugging` module: Public API for tf.debugging namespace.

`distribute` module: Library for running a computation across multiple devices.

`dtypes` module: Public API for tf.dtypes namespace.

`errors` module: Exception types for TensorFlow errors.

`estimator` module: Estimator: High level tools for working with models.

`experimental` module: Public API for tf.experimental namespace.

`feature_column` module: Public API for tf.feature_column namespace.

텐서플로 class와 function

Classes

`class AggregationMethod` : A class listing aggregation methods used to combine gradients.

`class CriticalSection` : Critical section.

`class DType` : Represents the type of the elements in a `Tensor`.

`class DeviceSpec` : Represents a (possibly partial) specification for a TensorFlow device.

`class GradientTape` : Record operations for automatic differentiation.

`class Graph` : A TensorFlow computation, represented as a dataflow graph.

`class IndexedSlices` : A sparse representation of a set of tensor slices at given indices.

`class IndexedSlicesSpec` : Type specification for a `tf.IndexedSlices`.

`class Module` : Base neural network module class.

`class Operation` : Represents a graph node that performs computation on tensors.

`class OptionalSpec` : Represents an optional potentially containing a structured value.

`class RaggedTensor` : Represents a ragged tensor.

`class RaggedTensorSpec` : Type specification for a `tf.RaggedTensor`.

Functions

`Assert(...)` : Asserts that the given condition is true.

`abs(...)` : Computes the absolute value of a tensor.

`acos(...)` : Computes acos of x element-wise.

`acosh(...)` : Computes inverse hyperbolic cosine of x element-wise.

`add(...)` : Returns x + y element-wise.

`add_n(...)` : Adds all input tensors element-wise.

`argmax(...)` : Returns the index with the largest value across axes of a tensor.

`argmin(...)` : Returns the index with the smallest value across axes of a tensor.

`argsort(...)` : Returns the indices of a tensor that give its sorted order along an axis.

`as_dtype(...)` : Converts the given `type_value` to a `DType`.

`as_string(...)` : Converts each entry in the given tensor to strings.

`asin(...)` : Computes the trigonometric inverse sine of x element-wise.

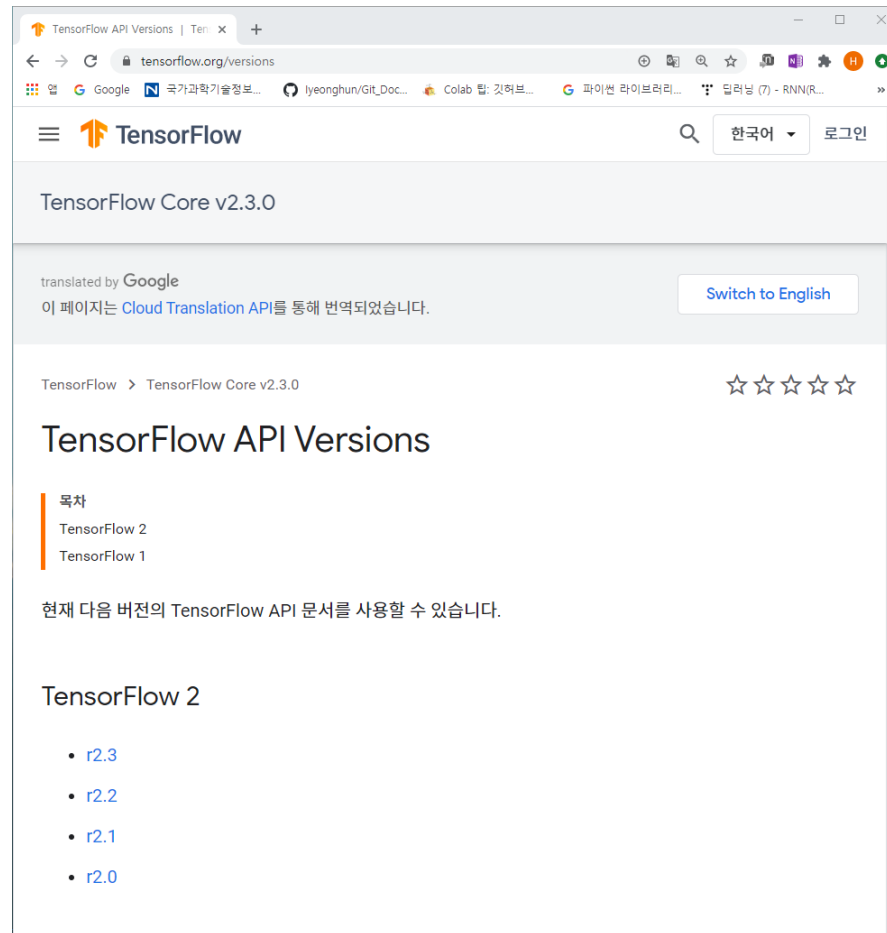
`asinh(...)` : Computes inverse hyperbolic sine of x element-wise.

`assert_equal(...)` : Assert the condition `x == y` holds element-wise.

`assert_greater(...)` : Assert the condition `x > y` holds element-wise.

버전

- <https://www.tensorflow.org/versions>



2020~21 겨울특강

텐서플로 기반 딥러닝

2일 2교시

동양미래대학교 컴퓨터정보공학과 강환수 교수

개발환경

동양미래대학교 컴퓨터정보공학과 강환수 교수

주 개발 환경

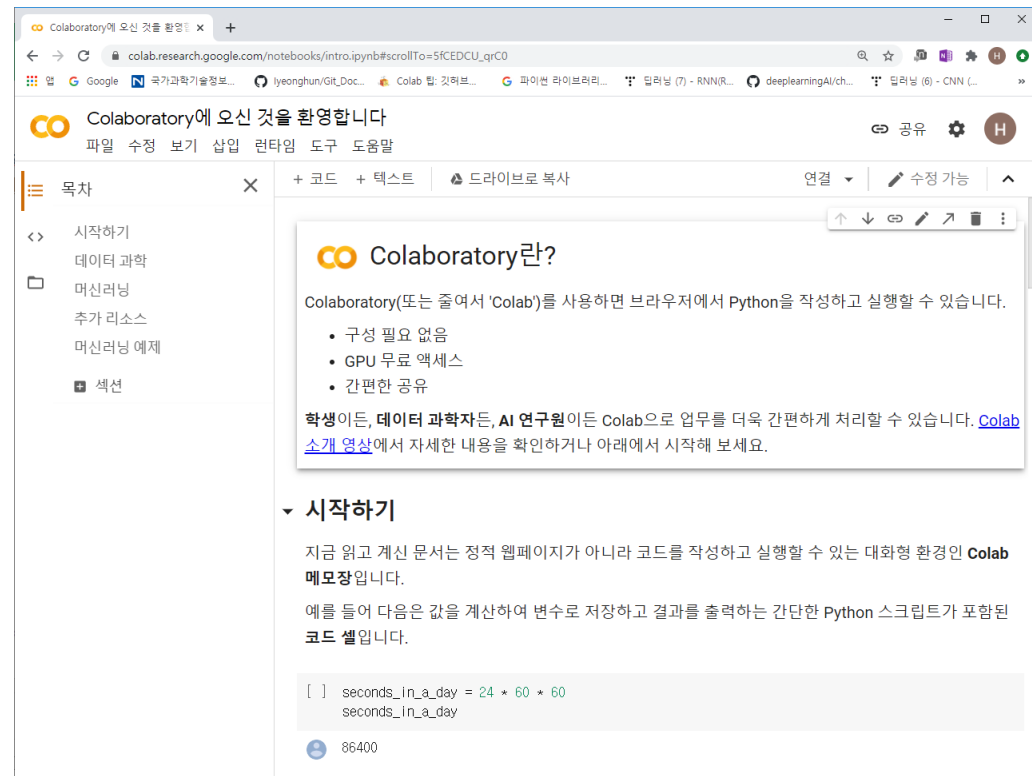
• 구글의 Colab

- 파이썬과 머신러닝, 딥러닝 개발 클라우드 서비스

- <https://colab.research.google.com/>

• 구글 계정 필요

- 구글 드라이브를 기본 저장소로 사용



구글 코랩 Google Colaboratory

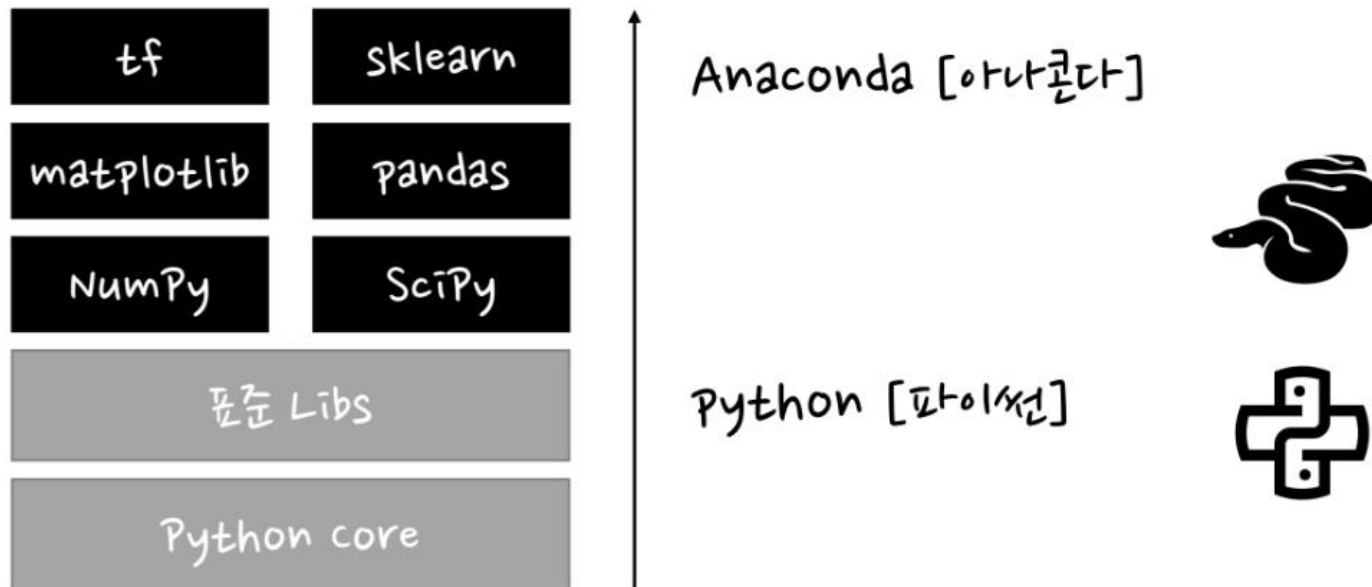
- **클라우드 기반의 무료 Jupyter 노트북 개발 환경**
 - 주피터 노트북을 지원하는 머신러닝, 딥러닝 클라우드 개발환경
 - 파이썬 뿐만 아니라 판다스, 멧플롯리브의 시각화 및 텐서플로우나 케라스 등 딥러닝 라이브러리도 쉽게 사용
 - <https://colab.research.google.com>
- **Google Drive + Jupyter Notebook**
 - 구글 계정 전용의 가상 머신 지원 – GPU, TPU 지원
 - Google drive 문서와 같이 링크만으로 접근 / 협업 가능
 - 구글 계정 필요
- **장점**
 - 구글 드라이브와 연계
 - 기본적으로 폴더 Colab Notebooks과 연결
 - 깃허브와 연계
 - 깃허브 소스를 바로 코딩 가능
 - *.ipynb

코랩 서버의 사양

- 일반 개인 PC보다 성능이 우수
 - CPU: Intel Xeon 2.2 GHz
 - RAM: 13GB
 - 저장공간: 33GB
 - 90분간 미사용 시 중지
 - 최대 12시간 연속 사용 가능
- CPU 사용
 - 생각보다 많이 빠르지는 않음
- 딥러닝은 GPU, TPU를 사용, 상당히 좋음
 - Graphics Processing Unit
 - Tensor Processing Unit

다양한 개발환경

- 자신의 PC에 설치해 활용 가능
 - 아나콘다



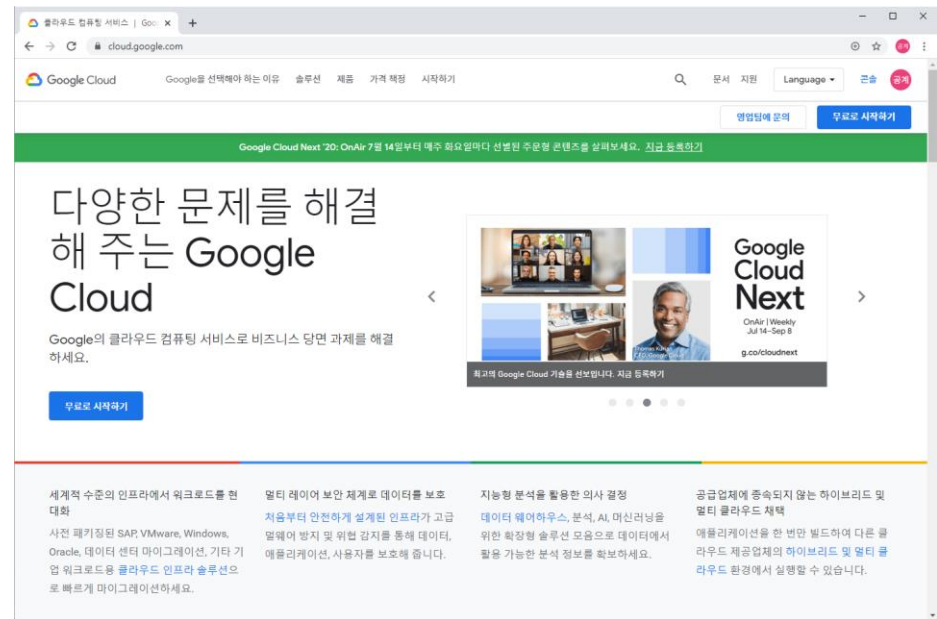
코랩으로 시작하는 텐서플로 기초 프로그래밍

텐서플로


- **Tensorflow**
 - 머신러닝을 위한 오픈소스 플랫폼
 - 가장 널리 쓰이는 딥러닝 프레임워크
 - 지원 언어
 - 파이썬, C++, 자바스크립트, 자바, Go, Swift
- **버전 변화**
 - 2017, 1월: 1.0 알파
 - 2019, 3월: 2.0 알파
 - 2019, 9월: 2.0 정식
 - 즉시 실행 모드(eager execution)가 기본
 - 세션 대신 함수 사용
 - 정식으로 TPU 지원
 - 최신 버전 2.4
- **홈페이지**
 - <http://tensorflow.org>


구글의 여러 서비스


- 텐서플로
- 구글 코랩
 - 노트북 ipynb의 자유로운 공유
- 구글 클라우드
 - <https://cloud.google.com/>
 - BigQuery
 - AutoML





추천 제품


 **Compute Engine**
Google의 데이터 센터에서 실행되는 가상 머신


 **Cloud Storage**
안전하고 내구성과 확장성이 뛰어난 객체 스토리지


 **Cloud SDK**
Google Cloud용 명령줄 도구 및 라이브러리


 **Cloud SQL**
MySQL, PostgreSQL, SQL Server용 관계형 데이터베이스 서비스


 **Google Kubernetes Engine**
컨테이너식 앱 실행을 위한 관리형 환경


 **BigQuery**
비즈니스 민첩성과 통찰력을 위한 데이터 웨어하우스

 **Cloud CDN**
웹 및 동영상 전송을 위한 콘텐츠 전송 네트워크

 **Dataflow**
스트림 및 일괄 처리를 위한 스트리밍 분석

 **작업**
모니터링, 로깅, 애플리케이션 성능 제품군

 **Cloud Run**
컨테이너식 앱 실행을 위한 완전 관리형 환경

 **Cloud Functions**
클라우드 서비스 및 앱을 위한 이벤트 기반 컴퓨팅 플랫폼

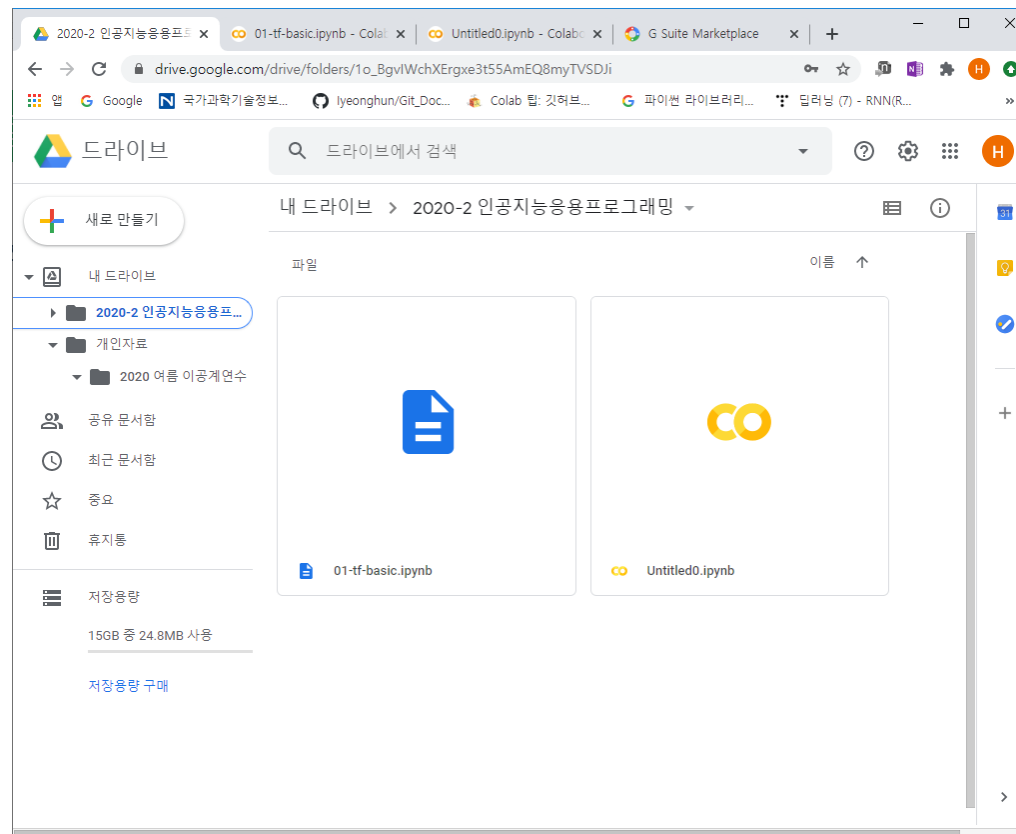
원하는 사항을 찾을 수 없으신가요?
[모든 제품 보기\(100개 이상\)](#)

코랩 드라이브에서 노트북 파일 열기

- 자신의 구글 드라이브

- 폴더 "2020-21 겨울특강 텐서플로 기반 딥러닝" 하부에 작성

- 파일 01-tf-basic.ipynb



구글 코랩, 매직 명령어로 tensorflow 불러오기

- 코랩에서 쉽게 버전 사용방법

- %tensorflow_version 1.x
- %tensorflow_version 2.x

```
%tensorflow_version 1.x
```

- Import 하기 전

- 위 매직 명령어 사용
- 사용 중에 바꾸려면 '런타임 다시 시작' 후 바로

- %tensorflow_version 1.x
- %tensorflow_version 2.x

```
# 텐서플로 1.0 버전 선택
```

```
try:
    %tensorflow_version 1.x
except Exception:
    pass
```

- 2.2 사용 중에 1.x으로 변경

- 1. 메뉴, 런타임 | 런타임 다시 시작
 - 단축키: ctrl+M .
- 2. 바로 실행
 - %tensorflow_version 1.x

```
import tensorflow as tf
tf.__version__
```

코랩으로 텐서플로 1.x 코딩

- 코랩에서 1.x 사용 지정

- Hello World에서 시작
- Session

- 그래프를 실행시키는 객체

- 만들어진 그래프에 실제 값의 흐름을 수행해 결과가 나오도록 하는 객체

```
%tensorflow_version 1.x

import tensorflow as tf

hello = tf.constant('Hello World!')

sess = tf.Session()
print(sess.run(hello))
sess.close()

a = 10
b = 10
print(tf.add(a, b))

sess = tf.Session()
print(sess.run(tf.add(a, b)))
sess.close()
```

2020~21 겨울특강

텐서플로 기반 딥러닝

2일 3교시

동양미래대학교 컴퓨터정보공학과 강환수 교수

텐서플로 2.0으로 실행

- 이미 1.x을 사용 중이라면
 - 메뉴, 런타임 | 런타임 다시 시작
 - 단축키: **ctrl+M** .
- 이후 그대로 **import**
- **텐서 출력**
 - 값만 보려면
 - 메소드 **numpy()**

▼ 메뉴, 런타임 | 런타임 다시 시작

단축키: **ctrl+M** .

```
[1] import tensorflow as tf
    tf.__version__
```

↪ '2.3.0'

```
[2] a = tf.constant(5)
    b = tf.constant(3)
    print(a+b)
```

↪ tf.Tensor(8, shape=(), dtype=int32)

```
[3] c = tf.constant('Hello, world!')
    print(c)
    print(c.numpy())
```

↪ tf.Tensor(b'Hello, world!', shape=(), dtype=string)
b'Hello, world!'

Tensor Ranks, Shapes, and Types(1)

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

| Data type | Python type | Description |
|-----------|-------------------------|-------------------------|
| DT_FLOAT | <code>tf.float32</code> | 32 bits floating point. |
| DT_DOUBLE | <code>tf.float64</code> | 64 bits floating point. |
| DT_INT8 | <code>tf.int8</code> | 8 bits signed integer. |
| DT_INT16 | <code>tf.int16</code> | 16 bits signed integer. |
| DT_INT32 | <code>tf.int32</code> | 32 bits signed integer. |
| DT_INT64 | <code>tf.int64</code> | |

```
[3] a = tf.constant(3)
    a
```

```
> <tf.Tensor: shape=(), dtype=int32, numpy=3>
```

```
[4] a.shape
```

```
> TensorShape([])
```

```
[6] a.dtype
```

```
> tf.int32
```

```
[7] a.numpy()
```

```
> 3
```

```
[8] b = tf.constant(3.)
    b
```

```
> <tf.Tensor: shape=(), dtype=float32, numpy=3.0>
```

```
[9] b.shape
```

```
> TensorShape([])
```

```
[10] b.dtype
```

```
> tf.float32
```

```
[11] b.numpy()
```

```
> 3.0
```


Tensor Shapes, type

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

| Rank | Math entity | Python example |
|------|----------------------------------|---|
| 0 | Scalar (magnitude only) | <code>s = 483</code> |
| 1 | Vector (magnitude and direction) | <code>v = [1.1, 2.2, 3.3]</code> |
| 2 | Matrix (table of numbers) | <code>m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]</code> |
| 3 | 3-Tensor (cube of numbers) | <code>t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]</code> |
| n | n-Tensor (you get the idea) | <code>....</code> |

Tensor Shapes, type code

▼ Tensor

```
[12] a = tf.constant([1, 2, 3])
      a.shape
```

```
↳ TensorShape([3])
```

```
[13] a = tf.constant([[1, 2, 3], [4, 5, 6]])
      a.shape
```

```
↳ TensorShape([2, 3])
```

```
[14] a = tf.constant([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
      a.shape
```

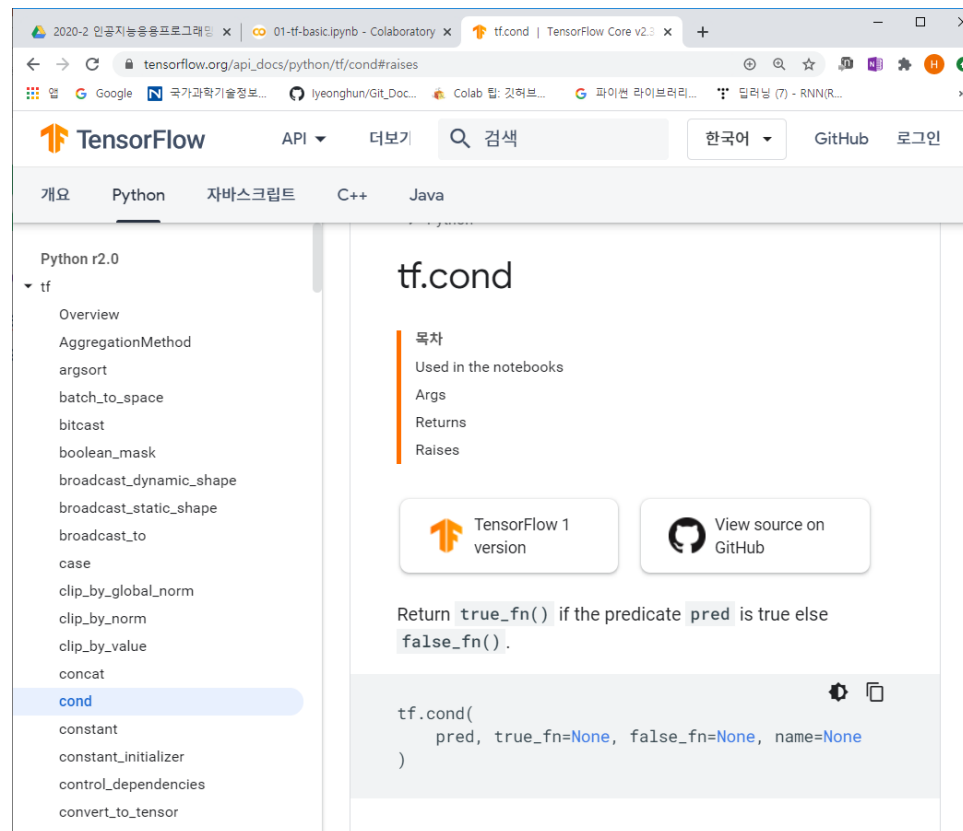
```
↳ TensorShape([2, 2, 3])
```

```
[20] a
```

```
↳ <tf.Tensor: shape=(2, 2, 3), dtype=int32, numpy=
  array([[[1, 2, 3],
          [4, 5, 6]],
        [[1, 2, 3],
          [4, 5, 6]]], dtype=int32)>
```

조건 연산 `tf.cond()`

- `tf.cond(pred, true_fn=None, false_fn=None, name=None)`
 - `pred`를 검사해 참이면 `true_fn` 반환
 - `pred`를 검사해 거짓이면 `false_fn` 반환



조건 연산 `tf.cond()` 코드

`tf.cond`

```
[6] x = tf.constant(1.)
    bool = tf.constant(True)
    res = tf.cond(bool, lambda: tf.add(x, 1.), lambda: tf.add(x, 10.))

    print(res)
    print(res.numpy())
```

```
↳ tf.Tensor(2.0, shape=(), dtype=float32)
   2.0
```

https://www.tensorflow.org/api_docs/python/tf/cond

```
[7] x = tf.constant(2)
    y = tf.constant(5)
    def f1(): return tf.multiply(x, 17)
    def f2(): return tf.add(y, 23)
    r = tf.cond(tf.less(x, y), f1, f2)

    r.numpy()
```

```
↳ 34
```

1차원 배열 텐서

```
[14] # 1차원 배열 텐서
      t = tf.constant([1, 2, 3])
      t
```

```
↳ <tf.Tensor: shape=(3,), dtype=int32, numpy=array([1, 2, 3], dtype=int32)>
```

```
[15] x = tf.constant([1, 2, 3])
      y = tf.constant([5, 6, 7])

      print((x+y).numpy())
```

```
↳ [ 6  8 10]
```

```
[16] a = tf.constant([5], dtype=tf.float32)
      b = tf.constant([10], dtype=tf.float32)
      c = tf.constant([2], dtype=tf.float32)
      print(a.numpy())

      d = a * b + c

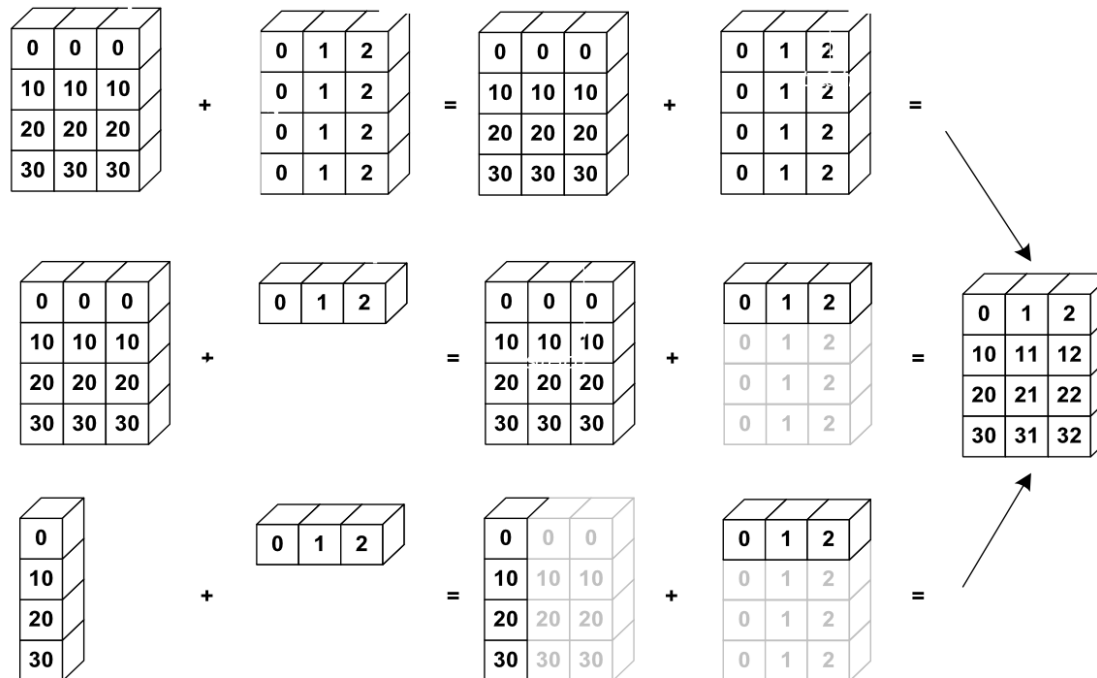
      print(d)
      print(d.numpy())
```

```
↳ [5.]
   tf.Tensor([52.], shape=(1,), dtype=float32)
   [52.]
```

배열 텐서 연산

• 텐서의 브로드캐스팅

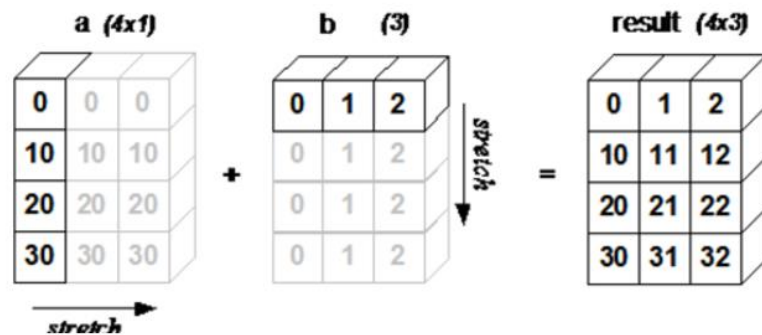
- Shape이 다르더라도 연산이 가능하도록
 - 가지고 있는 값을 이용하여 Shape을 맞춤



브로드캐스팅 코드 1

• Numpy

- np.arange()



```
[17] x = tf.constant([[0], [10], [20], [30]])
      y = tf.constant([0, 1, 2])
```

```
print((x+y).numpy())
```

```
[[ 0  1  2]
 [10 11 12]
 [20 21 22]
 [30 31 32]]
```

```
[44] import numpy as np
```

```
print(np.arange(3))
print(np.ones((3, 3)))
print()
```

```
x = tf.constant((np.arange(3)))
y = tf.constant([5], dtype=tf.int64)
print(x)
print(y)
print(x+y)
```

```
[[0 1 2]
 [[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
```

```
tf.Tensor([0 1 2], shape=(3,), dtype=int64)
tf.Tensor([5], shape=(1,), dtype=int64)
tf.Tensor([5 6 7], shape=(3,), dtype=int64)
```

브로드캐스팅 코드 2

```
[45] x = tf.constant((np.arange(3)))
      y = tf.constant([5], dtype=tf.int64)
      print((x+y).numpy())

x = tf.constant((np.ones((3, 3))))
y = tf.constant(np.arange(3), dtype=tf.double)
print((x+y).numpy())

x = tf.constant(np.arange(3).reshape(3, 1))
y = tf.constant(np.arange(3))
print((x+y).numpy())
```

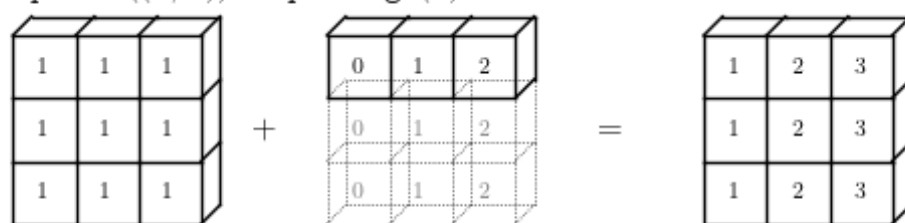
↪

```
[5 6 7]
[[1. 2. 3.]
 [1. 2. 3.]
 [1. 2. 3.]]
[[0 1 2]
 [1 2 3]
 [2 3 4]]
```

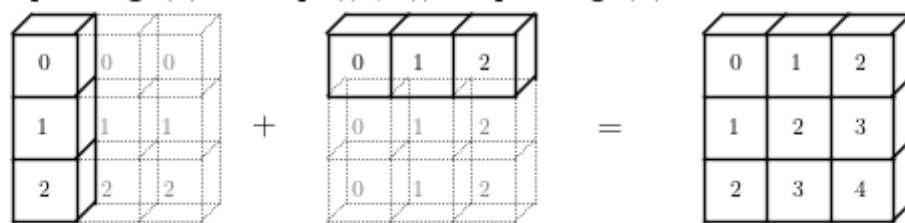
`np.arange(3) + 5`



`np.ones((3, 3)) + np.arange(3)`



`np.arange(3).reshape((3, 1)) + np.arange(3)`



텐서플로 연산

- tf.add()**

```
[46] a = 2
      b = 3
      c = tf.add(a, b)
      print(c.numpy())
```

➞ 5

```
[47] x = 2
      y = 3
      add_op = tf.add(x, y)
      mul_op = tf.multiply(x, y)
      pow_op = tf.pow(add_op, mul_op)

      print(pow_op.numpy())
```

➞ 15625

```
[48] a = tf.constant(2.)
      b = tf.constant(3.)
      c = tf.constant(5.)

      # Some more operations.
      mean = tf.reduce_mean([a, b, c])
      sum = tf.reduce_sum([a, b, c])

      print("mean = ", mean.numpy())
      print("sum = ", sum.numpy())
```

➞ mean = 3.3333333
sum = 10.0

행렬 곱셈

• 행렬 곱(내적)

- Numpy
 - `np.dot(a, b)`
 - `a.dot(b)`
- Tf
 - `tf.matmul`

$$\begin{matrix} \mathbf{A} & \mathbf{B} & \mathbf{A} * \mathbf{B} \end{matrix}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 6 & 3 \\ 5 & 2 \\ 4 & 1 \end{pmatrix} = \begin{pmatrix} 1*6 + 2*5 + 3*4 & 1*3 + 2*2 + 3*1 \\ 4*6 + 5*5 + 6*4 & 4*3 + 5*2 + 6*1 \end{pmatrix}$$

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

$$C_{ij} = \sum_k A_{ik} B_{kj} = A_{ik} B_{kj}$$

행렬 곱셈

- **tf.matmul()**

```
[50] # Matrix multiplications 1
      matrix1 = tf.constant([[1., 2.], [3., 4.]])
      matrix2 = tf.constant([[2., 0.], [1., 2.]])

      gop = tf.matmul(matrix1, matrix2)
      print(gop.numpy())

      # Matrix multiplications 2
      gop = tf.matmul(matrix2, matrix1)
      print(gop.numpy())
```

```
↳ [[ 4.  4.]
    [10.  8.]
    [[ 2.  4.]
     [ 7. 10.]
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 10 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 7 & 10 \end{bmatrix}$$