

# 12강: 딥러닝 프레임워크의 이해

인공지능 일반강좌 : 기계학습의 이해(L2-1)

# Tensorflow 설치 및 기본개념

## • Tensorflow 소개

- ✓ 다양한 기계학습 및 딥러닝 프로그래밍을 위한 오픈소스 라이브러리
- ✓ 현재 ver 1.15까지 배포
- ✓ ver 2.0도 정식버전으로 배포 중

The screenshot shows the official TensorFlow website at <https://www.tensorflow.org>. The page has a yellow header with the TensorFlow logo and navigation links for Install, Develop, Community, API, and Resources. A search bar and a GitHub link are also present. The main content area features a large orange background image of a 3D wireframe landscape. The text "An open source machine learning library for research and production." is displayed, along with a "시작하기" (Get Started) button. Below this, there are three sections: "속도" (Speed), "유연성" (Flexibility), and "프로덕션 대응" (Production Readiness). Each section contains a brief description of TensorFlow's capabilities in those areas.

An open source machine learning library for research and production.

시작하기

속도

기계 학습 시스템을 구축 및 배포할 때는 성능이 중요합니다. TensorFlow는 이를 위해 임베디드 프로세서, CPU, GPU, TPU 및 기타 하드웨어 플랫폼에서 TensorFlow 코드를 최대한 빠르게 실행할 수 있는 강력한 선형대수 컴파일러인 XLA를 포함합니다.

유연성

TensorFlow는 쉽게 모델을 구축하고 훈련할 수 있는 고수준 API를 제공할 뿐 아니라 유연성과 성능을 극대화하는 세부적인 컨트롤도 가능합니다.

프로덕션 대응

TensorFlow는 예비 조사에서 대규모 프로덕션 사용까지 다양한 규모에 대응합니다. 새로운 종류의 모델을 고안하는 경우든 실무에서 수백만 개의 요청을 처리하는 경우든 익숙한 TensorFlow API를 동일하게 사용할 수 있습니다.

최신 업데이트

TensorFlow 소개

# Tensorflow 설치



## Install TensorFlow

TensorFlow is tested and supported on the following 64-bit systems:

- Ubuntu 16.04 or later
- macOS 10.12.6 (Sierra) or later (no GPU support)
- Windows 7 or later
- Raspbian 9.0 or later

### Download a package

Install TensorFlow with Python's pip package manager.

Official packages available for Ubuntu, Windows, macOS, and the Raspberry Pi.

GPU packages require a CUDA®-enabled GPU card.

[READ THE PIP INSTALL GUIDE](#)

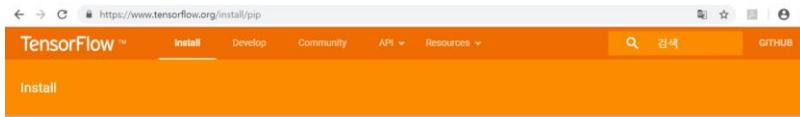
```
# Current release for CPU-only
$ pip install tensorflow

# Nightly build for CPU-only (unstable)
$ pip install tf-nightly

# GPU package for CUDA-enabled GPU cards
$ pip install tensorflow-gpu

# Nightly build with GPU support (unstable)
$ pip install tf-nightly-gpu
```

### Run a TensorFlow container



## Install TensorFlow with pip

☆☆☆☆☆

목록  
Available packages  
System requirements  
Hardware requirements  
1. Install the Python development environment on your system.  
2. Create a virtual environment (recommended)  
3. Install the TensorFlow pip package  
Package location

### Available packages

- tensorflow —Current release for CPU-only (recommended for beginners)
- tensorflow-gpu —Current release with GPU support (Ubuntu and Windows)
- tf-nightly —Nightly build for CPU-only (unstable)
- tf-nightly-gpu —Nightly build with GPU support (unstable, Ubuntu and Windows)

### System requirements

- Ubuntu 16.04 or later (64-bit)
- macOS 10.12.6 (Sierra) or later (64-bit) (no GPU support)
- Windows 7 or later (64-bit) (Python 3 only)
- Raspbian 9.0 or later

### Hardware requirements

- Starting with TensorFlow 1.6, binaries use AVX instructions which may not run on older CPUs.
- Read the [GPU support guide](#) to set up a CUDA®-enabled GPU card on Ubuntu or Windows.

- 다양한 설치방법을 제공함
- 운영체계를 다양하게 사용  
(라즈베이파이까지 사용가능)

- 윈도우 64비트에서 사용
- python 3.7사용 (ver 1.14현재)
- pip를 이용하여 설치

pip install tensorflow (CPU version)

pip install tensorflow-gpu (GPU version)

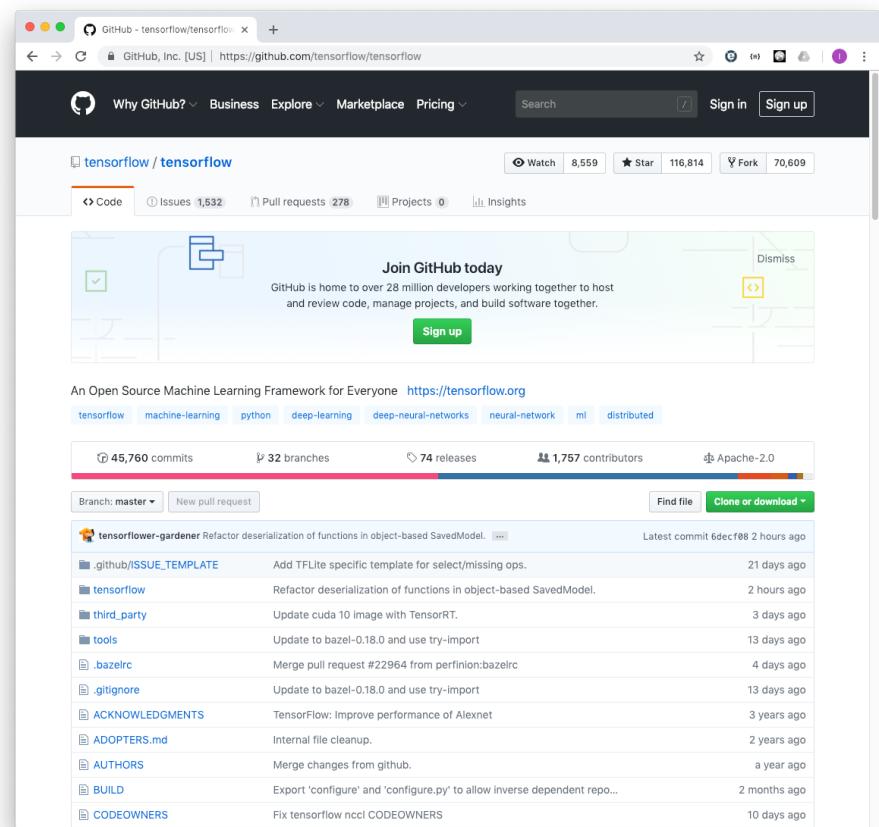
# Tensorflow 설치

The screenshot shows the TensorFlow website at <https://www.tensorflow.org/install/pip?lang=python3>. The main navigation bar includes 'Install', 'Develop', 'Community', 'API', 'Resources', and 'GITHUB'. A success message 'Success: TensorFlow is now installed. Read the tutorials to get started.' is displayed. On the left, there's a sidebar with links for 'Install TensorFlow', 'Packages' (pip, Docker), 'Additional setup' (GPU support, Problems), 'Build from source' (Linux / macOS, Windows, Raspberry Pi), 'Languages' (Java, C, Go), 'TensorFlow.js', and 'Swift for TensorFlow'. The main content area is titled 'Package location' and explains that URLs for TensorFlow Python packages depend on the Python version. It lists URLs for various Python versions (2.7 to 3.6) and operating systems (Linux and macOS). A sidebar on the right provides 'Available packages', 'System requirements', and 'Hardware requirements' (with steps 1-3 for pip installation) and a 'Package location' link.

- 각 운영체계, cpu, gpu 유무 등에 따라 각각 다운로드 가능
- windows의 경우 visual studio redistributable 2015 version 설치
- tensorflow 설치
  - ✓ cpu만 있을 경우 (nvidia graphics card 없는 경우) : python 3.7 cpu only 사용
  - ✓ gpu 사용 (nvidia graphics card) : python 3.7 gpu support version 사용  
→ gpu 사용 시 nvidia graphics driver, cuda toolkit, cudnn sdk 등을 먼저 설치해야 함

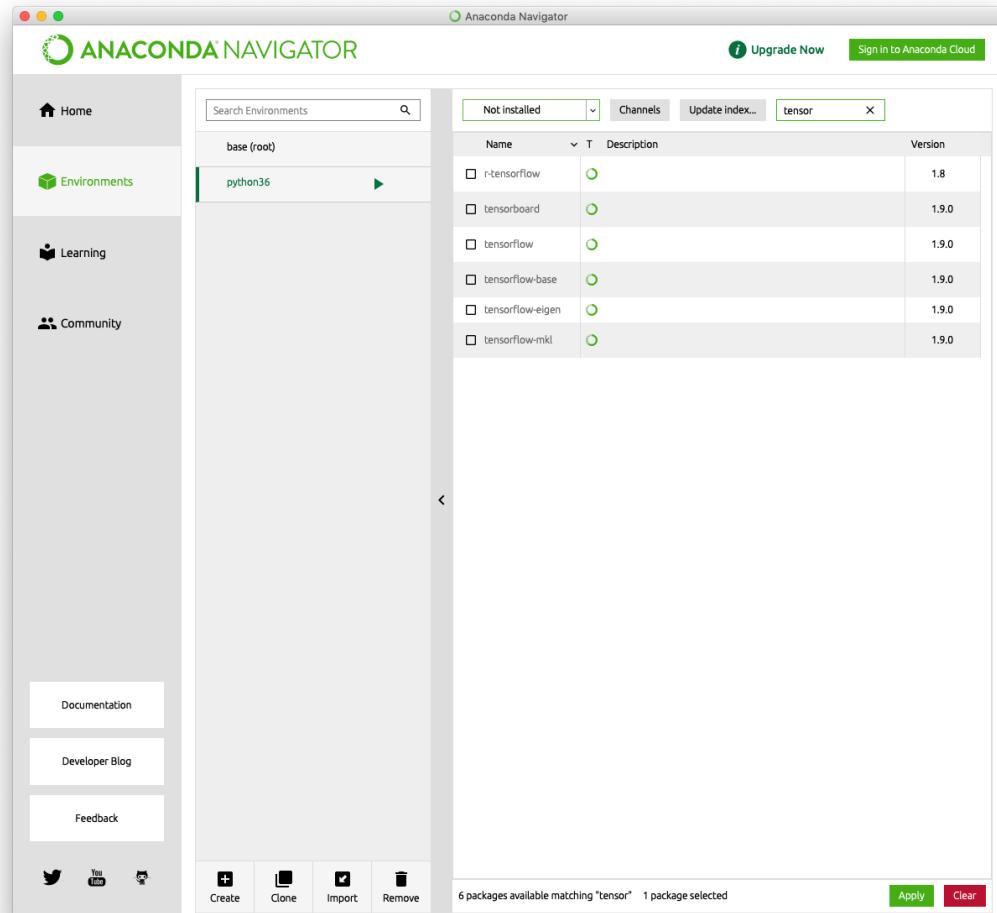
# Tensorflow 설치

- 가장 쉽게 설치하는 방법 – conda를 이용  
하면 gpu 지원도 쉽게 설치 가능  
→ conda install tensorflow-gpu
- source 를 이용하여 설치 가능  
[github.com/tensorflow/tensorflow](https://github.com/tensorflow/tensorflow)  
git clone <https://github.com/tensorflow/tensorflow>
- contrib/tensorboard (시각화 툴)



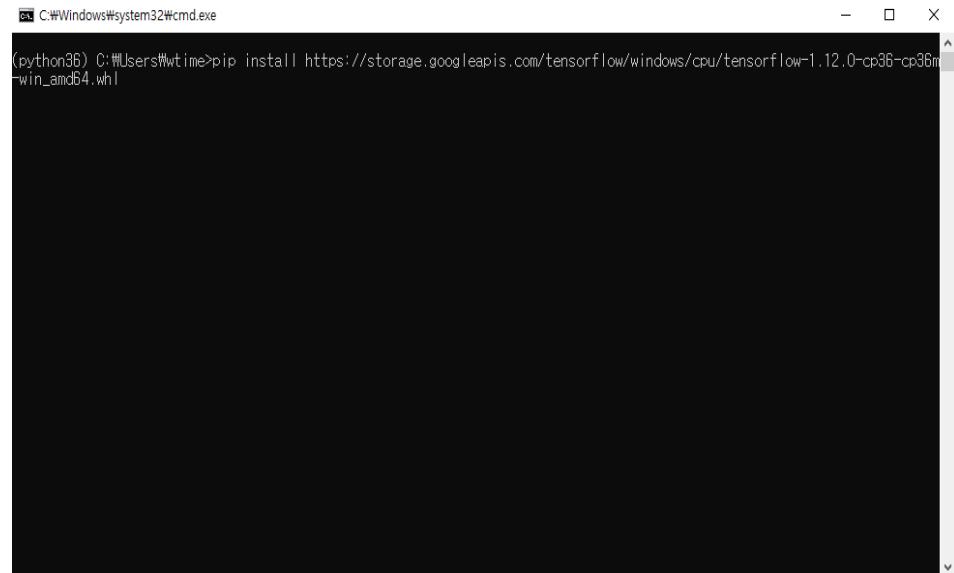
# Tensorflow 설치

- anaconda-navigator
  - ✓ anaconda-navigator 실행
  - ✓ environment – python36 선택
  - ✓ not installed 메뉴 선택
  - ✓ 검색(돋보기)에서 tensorflow 선택
  - ✓ 원하는 패키지를 선택 후 아래 apply 선택
  - ✓ tensorflow외 유용한 패키지인 tensorboard, keras 설치

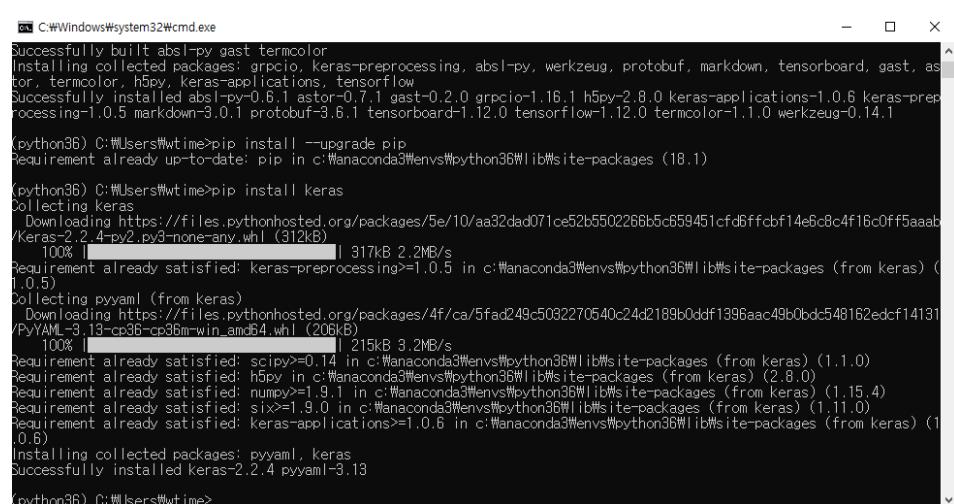


# Tensorflow 설치

- pip 를 이용한 설치
  - ✓ anaconda-navigator 실행
  - ✓ environments-python37 에서 클릭
  - ✓ open terminal 선택, 실행
  - ✓ 터미널 프롬프트에서
  - ✓ pip install –upgrade pip
  - ✓ pip install tensorflow



```
(python36) C:\Users\wttime>pip install https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow-1.12.0-cp36-cp36m-win_amd64.whl
```

```
Successfully built absl-py gast termcolor
Installing collected packages: grpcio, keras-preprocessing, absl-py, werkzeug, protobuf, markdown, tensorboard, gast, astor, termcolor, h5py, keras-applications, tensorflow
Successfully installed absl-py-0.6.1 astor-0.7.1 gast-0.2.0 grpcio-1.16.1 h5py-2.8.0 keras-applications-1.0.6 keras-preprocessing-1.0.5 markdown-3.0.1 protobuf-3.6.1 tensorboard-1.12.0 tensorflow-1.12.0 termcolor-1.1.0 werkzeug-0.14.1

(python36) C:\Users\wttime>pip install --upgrade pip
Requirement already up-to-date: pip in c:\anaconda3\envs\python36\lib\site-packages (18.1)

(python36) C:\Users\wttime>pip install keras
Collecting keras
  Downloading https://files.pythonhosted.org/packages/5e/10/aa32dad071ce52b550226b5c659451cfdbffcbf1e6c8c4f16c0ff5aaab/Keras-2.2.4-py2.py3-none-any.whl (312kB)
100% [██████████] 317kB 2.2MB/s
Requirement already satisfied: keras-preprocessing>=1.0.5 in c:\anaconda3\envs\python36\lib\site-packages (from keras) (1.0.5)
Collecting pyyaml (from keras)
  Downloading https://files.pythonhosted.org/packages/4f/ca/5fad249c5032270540c24d2189b0ddf1396aac49b0bcd548162edcf14131/PyYAML-3.13-cp36-cp36m-win_amd64.whl (206kB)
100% [██████████] 215kB 9.2MB/s
Requirement already satisfied: scipy>=0.14 in c:\anaconda3\envs\python36\lib\site-packages (from keras) (1.1.0)
Requirement already satisfied: h5py in c:\anaconda3\envs\python36\lib\site-packages (from keras) (2.8.0)
Requirement already satisfied: numpy>=1.9.1 in c:\anaconda3\envs\python36\lib\site-packages (from keras) (1.15.4)
Requirement already satisfied: six>=1.9.0 in c:\anaconda3\envs\python36\lib\site-packages (from keras) (1.11.0)
Requirement already satisfied: keras-applications>=1.0.6 in c:\anaconda3\envs\python36\lib\site-packages (from keras) (1.0.6)
Installing collected packages: pyyaml, keras
Successfully installed keras-2.2.4 pyyaml-3.13

(python36) C:\Users\wttime>
```

# Tensorflow 기본개념 실습

- **Google Colaboratory**  
(<https://colab.research.google.com/>)

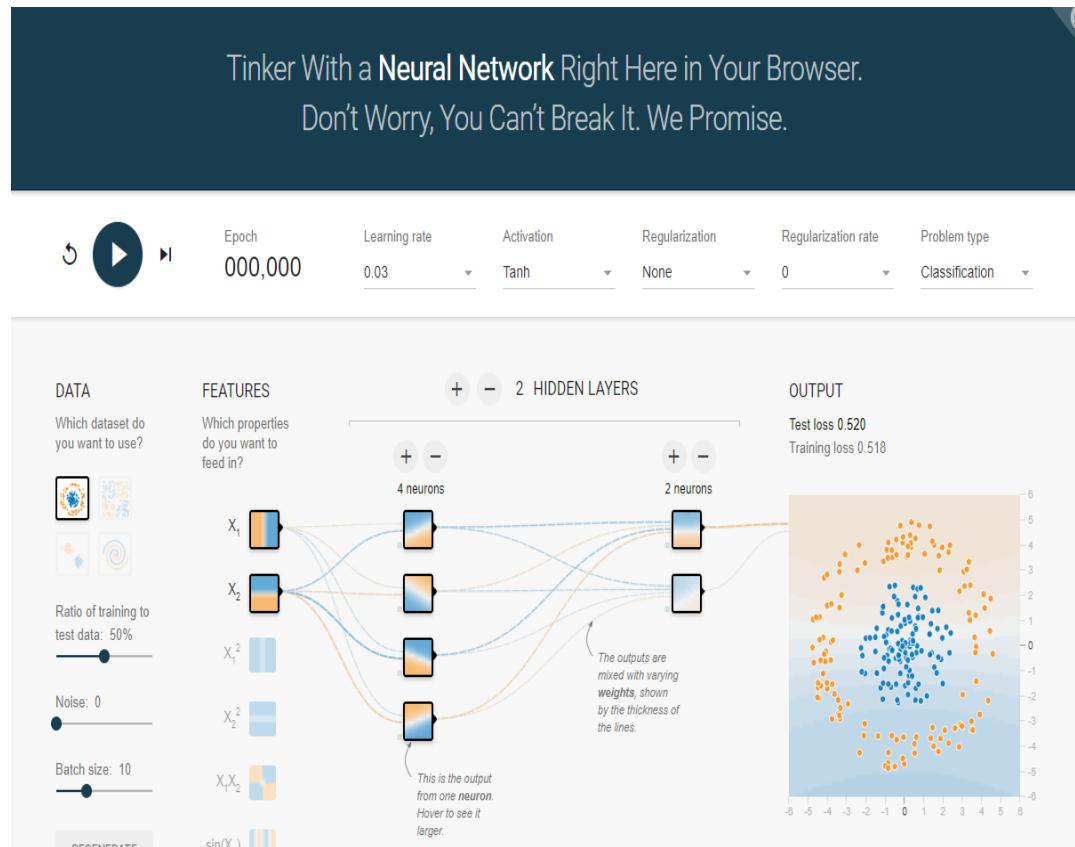
- ✓ Colaboratory는 기계학습 교육과 연구를 위한 웹상에서의 통합환경
- ✓ jupyter notebook 과 비슷한 환경으로 이루어져 있음
- ✓ google drive와 연계되어 쉽게 프로그램 개발이 가능
- ✓ 구글 계정이 있으면 누구나 무료로 사용 가능

The screenshot shows the Google Colaboratory interface. At the top, there's a navigation bar with icons for back, forward, refresh, and search, followed by the URL 'https://colab.research.google.com/notebooks/welcome.ipynb#scrollTo=9J7p406abzgl'. Below the URL is a toolbar with buttons for 'CODE', 'TEXT', 'CELL', 'COPY TO DRIVE', and other options. The main area has a sidebar on the left with sections like 'Table of contents', 'Code snippets', and 'Files'. The main content area displays a 'Welcome to Colaboratory!' message, which says: 'Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. See our [FAQ](#) for more info.' Below this, there are sections for 'Getting Started' (with a list of links), 'Highlighted Features' (with a 'Seedbank' section), and 'TensorFlow execution' (with a code example). The code example shows TensorFlow code for matrix addition:

```
[ ] import tensorflow as tf
input1 = tf.ones((2, 3))
input2 = tf.reshape(tf.range(1, 7, dtype=tf.float32), (2, 3))
output = input1 + input2
with tf.Session():
    result = output.eval()
result
```

# Tensorflow 기본개념 실습

- [playground.tensorflow.org](http://playground.tensorflow.org)
  - ✓ 인공지능망에 대한 내용을 가시화를 통해 웹상에서 바로 볼 수 있는 사이트
  - ✓ 몇개의 주어진 데이터셋을 고르고, 원하는 만큼의 뉴런을 추가(8개까지 가능), hidden layer는 6개까지 추가 가능
  - ✓ 분류 및 회귀분석에 대한 결과를 바로 가시화 해서 보여줌
  - ✓ tensorflow java version을 이용하여 제작됨



# Tensorflow 기본개념 실습

## • tensorflow 코딩 개요

- ✓ 현재 tensorflow는 keras와 결합되어 예전보다 코딩이 매우 간결하게 이루어지고 있음
- ✓ <https://www.tensorflow.org/tutorials/> 의 내용에 바로 keras를 적용한 예가 나오고 있고, tensorflow와 keras는 현재 상호보완관계로 이루어지고 있음
- ✓ 많은 코드들0| keras를 이용하여 코딩되고 있음

(참고) 한글본 문서 튜토리얼

<https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/tutorials/>

### Learn and use ML

The high-level Keras API provides building blocks to create and train deep learning models. Start with these beginner-friendly notebook examples, then read the [TensorFlow Keras guide](#).

1. [Basic classification](#)
2. [Text classification](#)
3. [Regression](#)
4. [Overfitting and underfitting](#)
5. [Save and load](#)

[READ THE KERAS GUIDE](#)

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy'
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

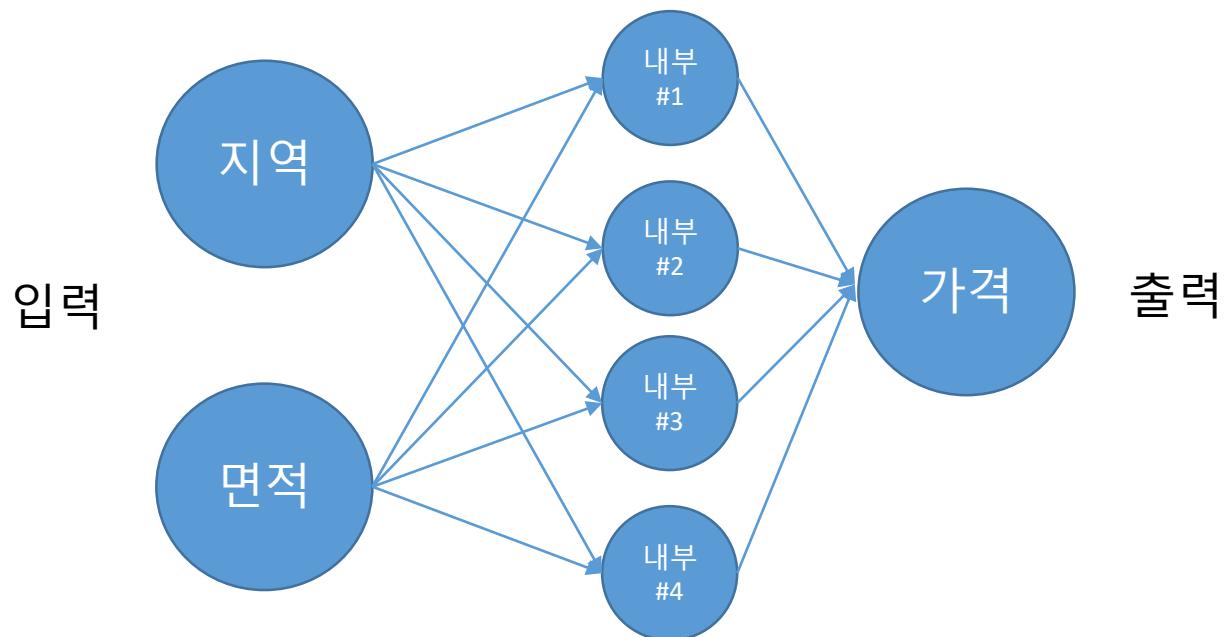
[RUN CODE NOW](#)

Try in Google's interactive notebook

# Tensorflow 기본개념 실습

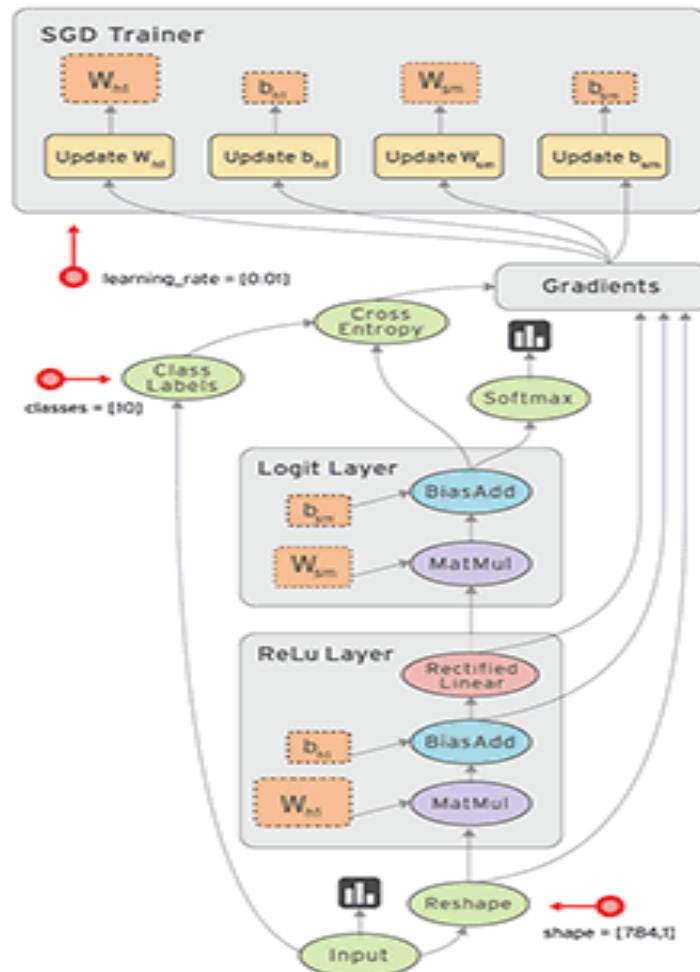
- tensorflow 동작원리

- ✓ 신경망 그림을 만든 후, 각각의 부분에 대해 코딩
- ✓ 데이터 흐름도 그래프(구조도)를 만들고, 그래프와 연결관계를 만들어두고, 이를 학습하고 이 결과를 가시화



# Tensorflow 기본개념 실습

- 연산은 graph로 표현
- graph는 점과 선, 즉 노드와 엣지로 이루어진 수학적인 구조를 의미
- graph는 session내에서 실행
- 데이터는 tensor로 표현
  - ✓ tensor는 그냥 내용을 볼 수 없음
- 구조확인을 하려면 session을 하나 열고, 이를 실행시켜서 확인
  - ✓ session.run을 통해 진행



# Tensorflow 기본개념 실습

- hello, tensorflow!

```
>>> import tensorflow as tf           ← Tensorflow 모듈 호출  
  
>>> hello = tf.constant('Hello, TensorFlow!') ← 문자열을 상수로 정의  
  
>>> sess = tf.Session()             ← 세션 정의  
>>> print sess.run(hello)          ← 세션 실행  
Hello, TensorFlow!  
  
>>> a = tf.constant(10)  
>>> b = tf.constant(32)  
  
>>> print sess.run(a+b)
```

42

# Tensorflow 기본개념 실습

- 값을 넣을때 matrix 단위로 넣음
- 행렬곱 : matmul 함수 사용
- multiply : 그냥 곱셈

```
sess.run(state)
sess.close()

import tensorflow as tf
a=tf.constant([[2, 4],[3, 2]])
b=tf.constant([[5, 2],[7, 4]])
c=tf.matmul(a,b)
d=tf.multiply(a,b)

sess=tf.Session()
sess.run([c,d])
sess.run(c)
sess.run(d)
```

```
[21, 8]]])  
In [10]: sess.run(c)  
Out[10]:  
array([[38, 20],  
       [29, 14]])  
  
In [11]: sess.run(d)  
Out[11]:  
array([[10, 8],  
       [21, 8]])  
In [12]:
```

Windows 정품  
[설정]으로 이동하여

# Tensorflow 기본개념 실습

- constant
  - ✓ constant는 python의 tuple과 비슷한 개념 (변하지 않는 값)
  - ✓ tf.constant(…)로 정의
- variables
  - ✓ python의 list, dict와 비슷한 개념
  - ✓ 변수(variable)는 여러 graph들이 작동할 때도 그 상태를 유지
  - ✓ 작업(operation)에서 데이터를 입출력 할 때 feed와 fetch를 사용
- placeholder
  - ✓ 변수(variable)과 같은 방식으로 선언
  - ✓ placeholder는 변수타입만 선언함
  - ✓ 변수를 나중에 받아 실행

# Tensorflow 기본개념 실습

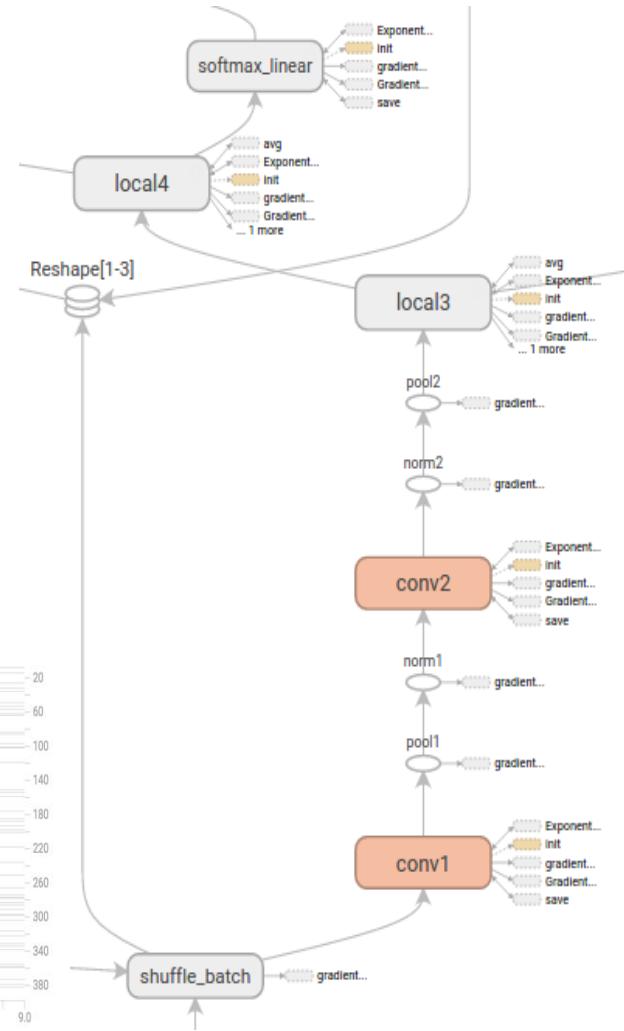
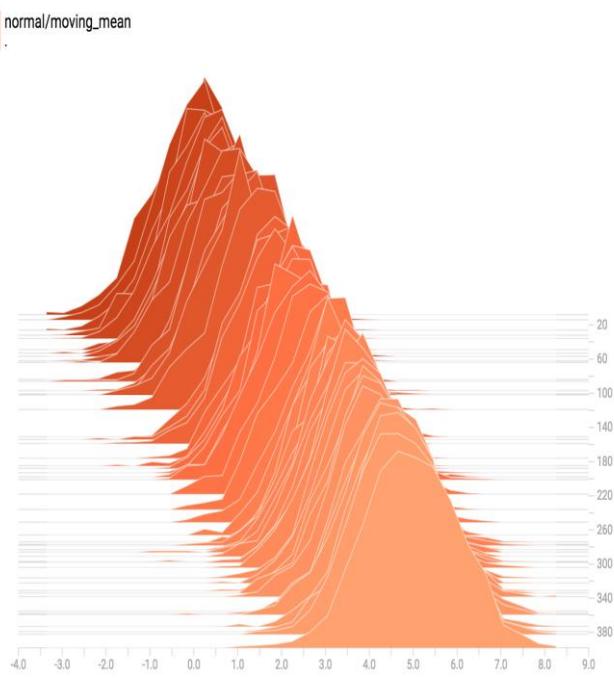
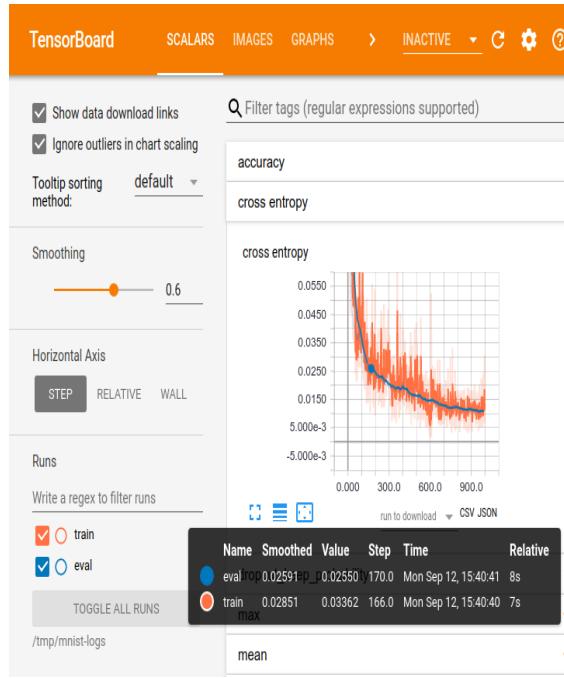
- placeholder
  - ✓ dictionary방식으로 값을 넘겨줄 수 있음 (feeding)

```
>>> import tensorflow as tf ← Tensorflow 모듈 호출
>>> x = tf.placeholder("float",None) ← float형식으로
   placeholder정의
>>> y = x * 1000 ← y라는 텐서생성 : x에
   1000을 곱하는 노드
>>> with tf.Session() as sess: ← 세션 정의 및 loop생성
>>>     z = sess.run(y, feed_dict={x : [1, 2, 3, 4]}) ← x에 dict형식으로
   값을 대입
>>>     print(z)
```

```
[1000, 2000, 3000, 4000]
```

# Tensorflow 기본개념 실습

- tensorboard
- tensorboard는 tensorflow 실행시 출력하는 데이터 및 기록을 그래프로 시각화시켜서 보여주는 도구
- 그래프, 스칼라, 이미지, … 등 저장해서 웹상에서 볼 수 있음



# Tensorflow 기본개념 실습

- tensorboard 동작을 위한 사전준비 사항
  - python 소스 코드에 tensorboard로 출력할 코드 추가

```
>>> import tensorflow as tf ← Tensorflow 모듈 호출
>>> a = tf.placeholder(tf.float32, name='a') ← float형식으로
>>> b = tf.placeholder(tf.float32, name='b') ← placeholder정의
>>> c = tf.add(a, b, name='sum') ← 합계노드생성
>>> sess = tf.Session()
>>> sess.run([c], feed_dict={a:1.0, b:2.0}) ← a, b에 dict형식으로
>>> summary_writer =                               값을 대입
tf.summary.FileWriter('./temp/1', sess.graph) ← temp/1 폴더에
                                                Tensorboard 데이터
                                                생성
```

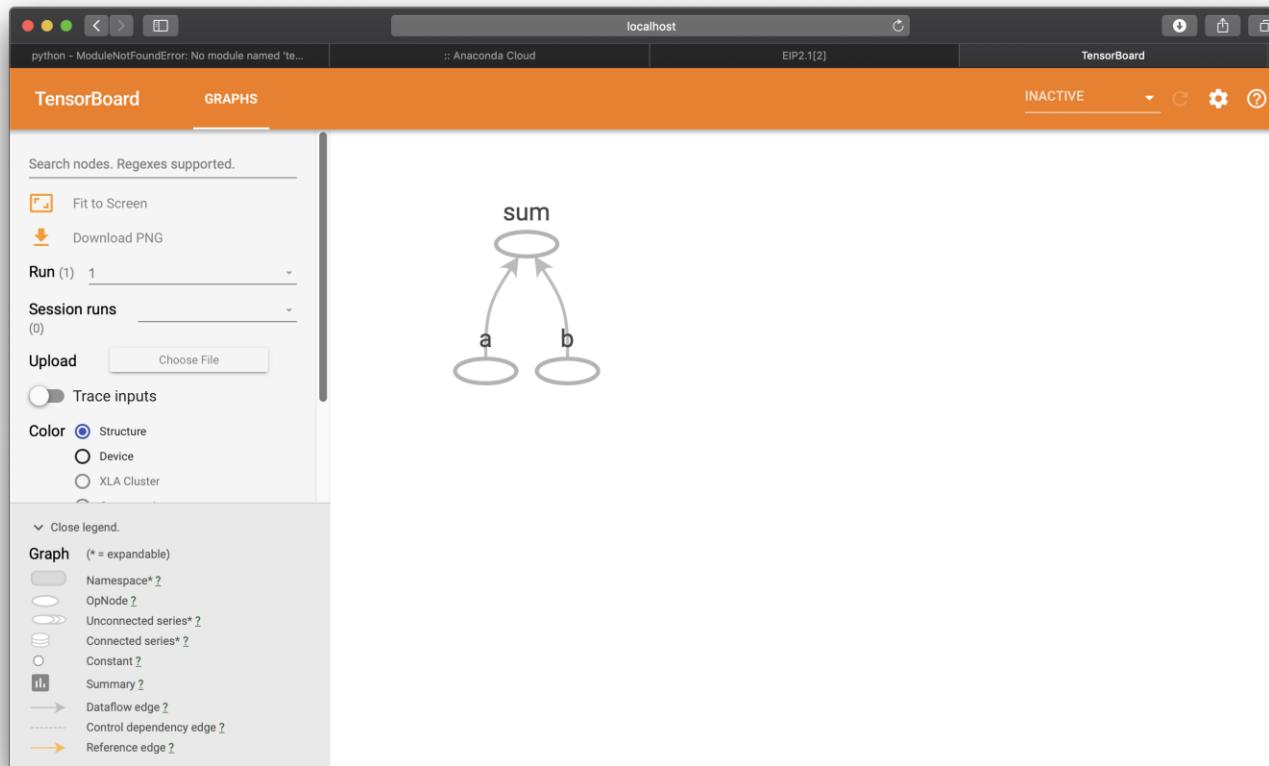
# Tensorflow 기본개념 실습

- **tensorboard 실행**

- ✓ terminal에서 다음 명령 수행

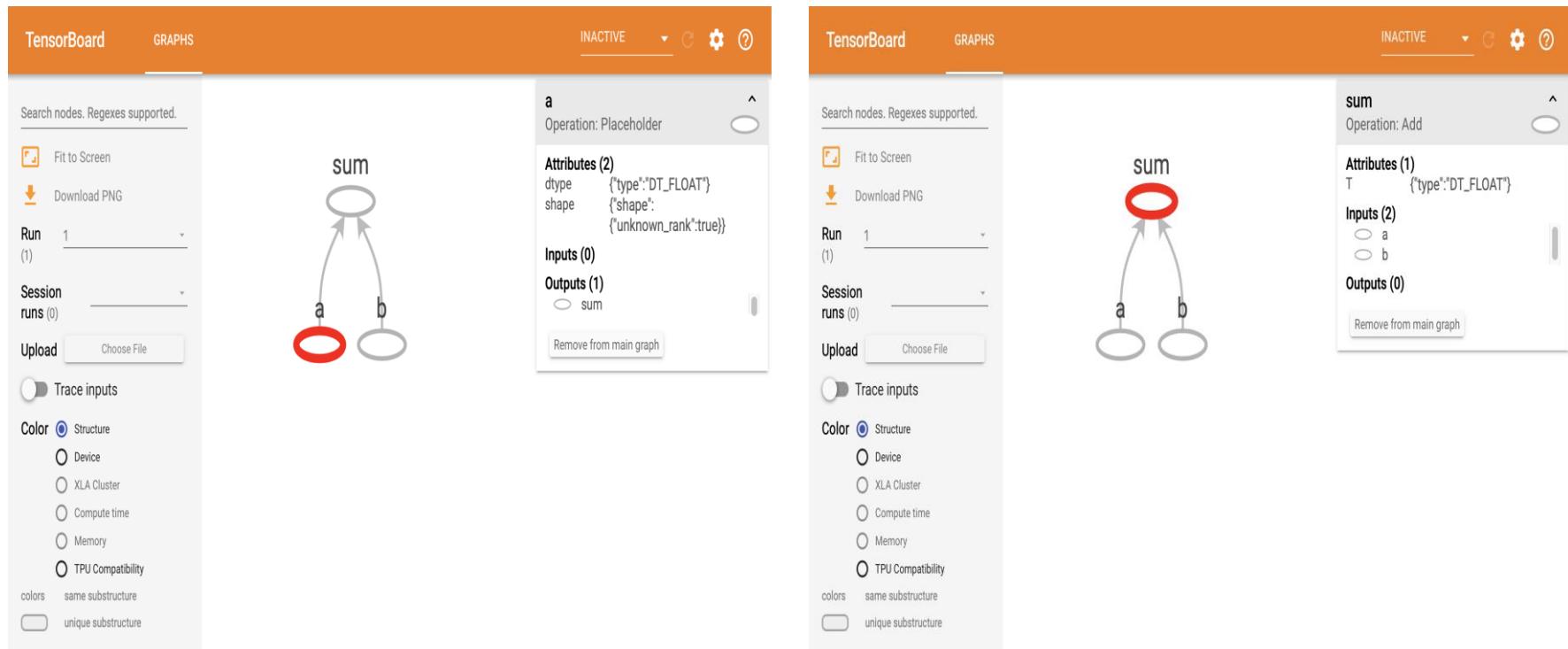
- ```
tensorboard --logdir=./temp
```

- ✓ 웹상에서 <http://localhost:6006> 으로 tensorflow 호출



# Tensorflow 기본개념 실습

- 각 노드를 클릭하면 해당노드에 대한 정보 표시
- tensorboard사용시 주의점
  - ✓ tensorboard 가 작동되어 있어야만 볼 수 있음 : 끊으면 안됨
  - ✓ 기록을 지우려면 모두 닫고 끊고 삭제 해야 함



# Tensorflow 기본개념 실습

- counter 예제

```
>>> import tensorflow as tf           Tensorflow 모듈 호출
>>> c = tf.Variable(0, name='counter')    변수 c 를 0으로 정의
>>> one = tf.constant(1)                 상수 1을 one에 정의
>>> new = tf.add(s, one)                덧셈 new 정의
>>> update = tf.assign(c, new)          Update 변수에 c, new정의
>>> init_op = tf.global_variables_initializer()    변수 초기화
>>> with tf.Session as sess:
>>>     sess.run(init_op)             초기화 실행
>>>     print(sess.run(c))         변수 출력
>>>     for _ in range(10)
>>>         sess.run(update)
>>>         print(sess.run(c))      10까지 반복
```

# Tensorflow 기본개념 실습

- counter 예제

```
>>> tf.summary.FileWriter("./temp/2", sess.graph)
```

IPython console

Console I/A

```
In [50]: with tf.Session() as sess: #with문에서만 session이 열릴 이후에는 닫힐
...:     # 'init' 작업(op)을 실행합니다.
...:     # Run the 'init' op
...:     sess.run(init_op)
...:     # 'state'의 시작값을 출력합니다.
...:     # Print the initial value of 'state'
...:     print(sess.run(state))
...:     # 'state'값을 업데이트하고 출력하는 작업(op)을 실행합니다.
...:     # Run the op that updates 'state' and print 'state'.
...:     for _ in range(10):
...:         sess.run(update)
...:         print(sess.run(state))
...:
...:
...: tf.summary.FileWriter('/tmp/2', sess.graph)
...:
```

0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

Out[50]: <tensorflow.python.summary.writer.FileWriter at 0x207fa3f61d0>

Windows 정품 인증 [설정]으로 이동하여 Windows를 정품 인증

Windows 정품 인증

Search nodes. Regexes supported.

Fit to Screen  
Download PNG

Run (1)  
Session runs (0)

Upload Choose File

Trace inputs

Color Structure  
Device  
XLA Cluster  
Compute time  
Memory

Close legend.

Graph (\* = expandable)  
Namespace\* 2  
OpNode\* 2  
Unconnected series\* 2  
Connected series\* 2  
Constant\* 2  
Summary\* 2  
Dataflow edge\* 2  
Control dependency edge\* 2

Windows 정품 인증

Windows 정품 인증

# Tensorflow 기본개념 실습

- interactive session

```
>>> import tensorflow as tf ← Tensorflow 모듈 호출
>>> sess = tf.InteractiveSession() ← Interactive 모드설정
>>> a = tf.constant(5.0) ← a에 상수 5.0 정의
>>> b = tf.constant(6.0) ← b에 상수 6.0 정의
>>> c = a * b ← 곱셈연산
>>> print(c.eval()) ← eval을 이용 c값 출력
30.0
>>> sess.close() ← 사용이 끝나면 항상
                           Close를 해 주어야 함
```

# Tensorflow 기본개념 실습

- ▶ eager execution
  - ▶ session run 을 거치지 않고 바로 실행할 수 있는 방법

```
>>> import tensorflow as tf           ← Tensorflow 모듈 호출  
>>> tf.enable_eager_execution()      ← Eager execution 시작  
>>> print(tf.add(1, 2))            ← 1 + 2  
tf.tensor(3, shape=(), dtype=int32)  
>>> print(tf.add([1, 2], [3, 4]))    ← 행렬계산  
tf.tensor([4 6], shape=(2,), dtype=int32)  
>>> print(tf.square(2) + tf.square(3)) ← 함수 연산  
tf.tensor(13, shape=(), dtype=int32)  
>>> print(reduce.sum([1, 2, 3]))     ← 타입변화 등  
tf.tensor(6, shape=(), dtype=int32)
```

# Tensorflow 기본개념 실습

- tensorflow-model-server
  - ✓ 미리 만들어 놓은 모델을 다시 사용하거나 배포할 때 사용

```
export_path_base = sys.argv[-1]
export_path = os.path.join(
    compat.as_bytes(export_path_base),
    compat.as_bytes(str(FLAGS.model_version)))
print 'Exporting trained model to', export_path
builder = tf.saved_model.builder.SavedModelBuilder(export_path)
builder.add_meta_graph_and_variables(
    sess, [tag_constants.SERVING],
    signature_def_map={
        'predict_images':
            prediction_signature,
        signature_constants.DEFAULT_SERVING_SIGNATURE_DEF_KEY:
            classification_signature,
    },
    main_op=main_op)
builder.save()
```

모델 저장경로

어떤 내용으로 저장할지 설정

모델 저장



# Keras 소개

- **케라스(keras)**

- 파이썬으로 작성된 오픈 소스 신경망 라이브러리
- mxnet, deeplearning4j, 텐서플로, microsoft cognitive toolkit 또는 theano 위에서 수행
- 딥 뉴럴넷과의 빠른 실험을 가능케 하도록 설계되었으며 최소한의 모듈 방식의 확장 가능성에 초점을 둔다.
- 구글의 엔지니어 Francois Chollet이 주 개발자이며 유지보수자
- 2017년,부터 구글의 텐서플로 팀은 텐서플로의 코어 라이브러리에 케라스를 지원
- 기계 학습 프레임워크가 아닌 인터페이스의 역할을 염두에 두어 더 높은 수준의 더 직관적인 추상화 집합을 표현
- 백엔드 과학 컴퓨팅 라이브러리임에도 불구하고 신경망을 구성하기 쉽게 작성

# Keras 소개

- keras의 주요 특징

- 모듈화 (modularity)
  - 케라스에서 제공하는 모듈은 독립적이고 설정 가능하며, 가능한 최소한의 제약사항으로 서로 연결 가능
  - 모델은 시퀀스 또는 그래프로 이러한 모듈들을 구성하며 신경망 층, 비용함수, 최적화기, 초기화기법, 활성화함수, 정규화기법 모두 독립적 모듈
  - 새로운 모델을 만들기 위해 이러한 모듈을 조합 가능
- 최소주의 (minimalism)
  - 각 모듈은 짧고 간결
  - 모든 코드는 한 번 훑어보는 것으로도 이해가능하게 구성
- 쉬운 확장성 (extendability)
  - 새로운 클래스나 함수로 모듈을 아주 쉽게 추가 가능

# Keras 소개

- **Keras의 정의**
  - ✓ Python으로 작성된 Tensorflow, CNTK, Theano와 같은 Deep Learning 라이브러리 위에서 실행할 수 있는 High-level Neural Network API
- **Keras의 특징**
  - ✓ 모듈화 (Modularity) : Keras 제공모듈은 독립적이고 설정 가능하며 시퀀스 또는 그래프로 모듈 구성이 가능
  - ✓ 최소주의 (Minimalism) : 각 모듈은 짧고 간결하며, 단면에 이해가능하게 되어 있음
  - ✓ 쉬운 확장성 : 새로운 클래스나 함수로 모듈을 쉽게 추가 가능
  - ✓ 파이썬 기반 : Caffe 처럼 별도의 모델 설정 파일이 필요없이 파이썬 코드로 모델들이 정의
- Tensorflow 1.12 버전 현재 많음 모듈들이 Keras와 연동되어 있으며, 상호 보완적으로 이루어져 운영되고 있음

# Keras 소개

- **Keras 모델 순서**

- ✓ 데이터셋 생성하기 : 원본 데이터를 불러오고 데이터로부터 훈련셋, 검증셋, 시험셋을 생성
- ✓ 모델 구성하기 : 시퀀스 모델을 생성한 뒤 필요한 레이어를 추가하여 구성, 좀 더 복잡한 모델이 필요할 때는 케라스 함수 Api를 사용
- ✓ 모델 학습과정 설정하기 : 학습하기 전 학습에 대한 설정 및 손실 함수 및 최적화 방법 정의 : Compile() 함수 사용
- ✓ 모델 학습 : Training Set을 이용 구성 모델로 학습 → Fit() 사용
- ✓ 학습 모니터링 : Training Set, Evaluation Set, 손실 및 정확도를 출력하면서 학습이 제대로 되고 있는지 모니터링
- ✓ 모델 평가 : Test Set으로 학습 모델을 평가 → Evaluate() 사용
- ✓ 모델 사용 : 임의 데이터를 넣어서 원하는 결과를 얻음 → Predict() 사용

# Keras 함수

- **Keras의 사용**

```
>>> import tensorflow as tf  
>>> Dense = tf.keras.layers.Dense
```

- **Sequential API, Functional API**

- ✓ 케라스에서는 층(layer)을 조합하여 모델(model)을 생성
- ✓ 모델은 일반적으로 층의 그래프
- ✓ Sequential API : 가장 흔한 모델 구조이며 층을 차례대로 쌓은 tf.keras.Sequential 모델
- ✓ Functional API : 다중 입력 및 다중 출력 모델, 공유 계층이 있는 모델이며 임의 연결이 있는 모델을 비롯한 복잡한 토폴로지 정의가 가능

# Keras 함수

- Sequential API 예

```
model = tf.keras.Sequential()  
model.add(layers.Dense(64, activation='relu'))  
model.add(layers.Dense(10, activation='softmax'))  
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',  
metrics=['accuracy'])  
model.fit(x_train, y_train, epochs=5)  
model.evaluate(x_test, y_test)
```

- Functional API 예

```
inputs = tf.keras.Input(shape=(32,))  
x = layers.Dense(64, activation='relu')(inputs)  
Predictions = layers.Dense(10, activation='softmax')(x)  
Model = tf.keras.Model(inputs=inputs, outputs=predictions)
```

# Keras 함수

- **Model subclassing API**

- ✓ 완전 맞춤형 모델을 빌드 가능
- ✓ 클래스 메서드 본문에 자체적인 포워드 패스를 다음과 같은 스타일로 명령형으로 정의가 가능

```
class MyModel(tf.keras.Model):

    def __init__(self):
        super(MyModel, self).__init__()
        # Define your layers here.
        self.dense_1 = layers.Dense(32, activation='relu')
        self.dense_2 = layers.Dense(num_classes, activation='sigmoid')

    def call(self, inputs):
        # Define your forward pass here,
        # using layers you previously defined in `__init__`
        x = self.dense_1(inputs)
        return self.dense_2(x)
```

# Keras 함수

- **Estimators**

- ✓ Linear Classifier, DNN Classifier, Combined DNN Linear Classifier(wide And Deep Models), Gradient Boosted Trees를 포함한 여러 가지 모델이 Premade Estimators라는 패키지로 제공
- ✓ 이러한 모델은 바로 개발 환경에서 사용 가능하고 폭넓게 배포
- ✓ Premade Estimators를 포함한 Estimator API가 Tensorflow 2.0에 포함

# Keras 예제

```
>>> from tensorflow.keras import layers  
>>> model = tf.keras.Sequential()  
>>> model.add(layers.dense(64, activation='relu'))  
>>> model.add(layers.dense(64, activation='relu'))  
>>> model.add(layers.dense(10, activation='softmax'))  
  
>>> model.compile(optimizer=tf.keras.optimizers.Adam(0.001),  
...                 loss='categorical_crossentropy',  
...                 metrics=['accuracy'])  
  
>>> import numpy as np  
>>> data = np.random.random((1000, 32))  
>>> labels = np.random.random((1000, 10))  
>>> model.fit(data, labels, epochs=10, batch_size=32)
```

64개의 유닛을 가진 완전  
연결 층을 모델에 추가

또 하나를 추가

10개의 출력 유닛을  
가진 소프트맥스 층을  
추가

# Keras 예제

```
Train on 1000 samples, validate on 100 samples Epoch 1/10 1000/1000
[=====] - 0s 232us/sample - loss: 20124.0200 -
categorical_accuracy: 0.1060 - val_loss: 28301.8253 -
val_categorical_accuracy: 0.1200 Epoch 2/10 1000/1000
[=====] - 0s 115us/sample - loss: 24420.6503 -
categorical_accuracy:
...
...
[=====] - 0s 116us/sample - loss: 61586.9824 - ca
tegorical_accuracy: 0.1020 - val_loss: 32742.4056 - val_categorical_accuracy
>>> data = np.random.random((1000, 32))
>>> labels = np.random.random((1000, 10))
>>> model.evaluate(data, labels, batch_size=32)
>>> model.evaluate(dataset, steps=30)          주어진 데이터로 추론 모드의
  손실이나 지표를 평가
1000/1 [=====
30/30 [=====] - 0s 2ms/step - loss: 178351.9229 -
categorical_accuracy: 0.1031 [178351.92291666666, 0.103125]
```

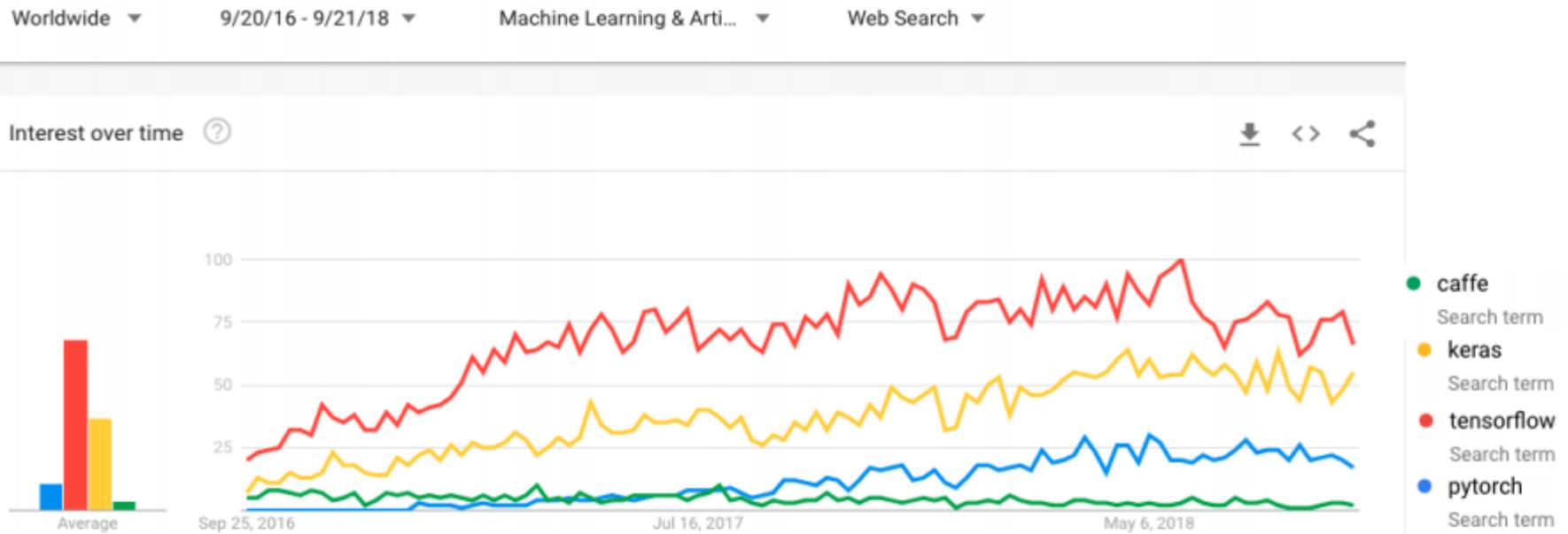
# Keras 함수

- 학습방식의 종류 (온라인, 배치)
  - ✓ 온라인방식 : Iteration 마다 업데이트 되는 방식
  - ✓ 배치방식 :
    - Batch\_size : 배치(batch)는 한 번에 처리하는 데이터 수
    - Max\_iter : 몇 개의 배치(batch)를 사용할 것인지를 의미 Max\_iter: 60000이고 Batch\_size:100 이면 한 번에 데이터 100개씩 60,000번 하겠다는 의미이므로, 즉 데이터 6,000,000번 학습
    - Test\_iter : 테스트의 Iteration이며 Max\_iter와 동일한 의미로 사용
  - ✓ Epoch : 학습용 데이터 전체를 한번 사용했을때의 단위, 만일 학습 데이터가 50,000개이고, 데이터를 6,000,000번 학습한다면, 전체 Epoch는 120이 됨

# Keras 개요

## ➊ 케라스(Keras) 개요

- 케라스는 파이썬으로 구현된 쉽고 간결한 딥 러닝 라이브러리
- 딥러닝 비전문가라도 각자 분야에서 손쉽게 딥 러닝 모델을 개발하고 활용할 수 있도록 케라스는 직관적인 API를 제공
- 내부적으로는 텐서플로우(TensorFlow), 티아노(Theano), CNTK 등의 딥 러닝 전용 엔진이 구동되지만 케라스 사용자는 복잡한 내부 엔진을 알 필요 없음
- 직관적인 API로 쉽게 다층퍼셉트론 모델, 컨볼루션 신경망 모델, 순환 신경망 모델 또는 이를 조합한 모델은 물론 다중 입력 또는 다중 출력 등 다양한 구성 가능
- <https://keras.io/>



# Keras 개요

## ❸ 케라스(Keras) 주요 특징

- 모듈화(Modularity)
  - ✓ 케라스에서 제공하는 모듈은 독립적이고 설정 가능
  - ✓ 가능한 최소한의 제약사항으로 서로 연결될 수 있음.
  - ✓ 모델은 시퀀스 또는 그래프로 이러한 모듈들을 구성한 것
  - ✓ 신경망 층, 비용함수, 최적화기법, 초기화기법, 활성화함수, 정규화기법은 모두 독립적인 모듈이며, 새로운 모델을 만들기 위해 이러한 모듈을 조합함.
- 최소주의(Minimalism)
  - ✓ 각 모듈은 짧고 간결함
  - ✓ 모든 코드는 한 번 보는 것으로도 이해 가능함
  - ✓ 단, 반복 속도와 혁신성은 다소 떨어질 수가 있음
  - ✓ <https://keras.io/>

# Keras 개요

## ❸ 케라스(Keras) 주요 특징

- 쉬운 확장성 : 새로운 클래스나 함수로 모듈을 아주 쉽게 추가할 수 있음.
- 파이썬 기반: Caffe 처럼 별도의 모델 설정 파일이 필요 없으며 파이썬 코드로 모델들이 정의
- 사용자 친화적: 배우기 쉽고 모델을 구축하기 용이
- 폭넓은 도입, 광범위한 프로덕션 배포 옵션 지원
- 최소 5개 백엔드 엔진과의 통합(텐서플로우, CNTK, 테아노, MXNet, PlaidML)
- 여러 GPU 및 분산 학습 지원
- 구글, 마이크로소프트, 아마존, 애플, 엔비디아, 우버 등 메이저 기업들의 지지
- 케라스 창시 : 프랑소와 쇼레(François Chollet)
- 케라스 유지보수 : Google

# Keras 개요

## ➊ 케라스 설치

- 설치
  - 아나콘다 프롬프트(Anaconda Prompt) 또는 명령 프롬프트를 통해서 설치

```
> pip install keras
```

```
> ipython ...
In [1]: import keras
In [2]: keras.__version__
Out[2]: '2.3.1'
```

# 딥 러닝 개발 환경

## ④ 구글 Colab을 이용한 실습 환경

- 다음 URL에 접속
  - ✓ <https://colab.research.google.com/>
- 구글 계정으로 로그인 (Sign In)

The screenshot shows the Google Colab interface with a Python code cell containing the following code:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

The output of the cell is:

```
86400
```

A tooltip for the code cell says: "위 셀의 코드를 실행하려면 셀을 클릭하여 선택한 후 코드 왼쪽의 면 코드 수정을 바로 시작할 수 있습니다. 특정 셀에서 정의한 변수를 나중에 다른 셀에서 사용할 수 있습니다".

Below the code cell, another cell contains:

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

The output of this cell is:

```
604800
```

A tooltip for this cell says: "Colab 메모장을 사용하면 실행 코드와 서식 있는 텍스트를 이미지화하면 Google 드라이브 계정에 저장됩니다. Colab 메모장을 간편하게 알아보려면 [Colab 개요](#)를 참조하세요. 새 Colab 메모장을 만들려면 [기](#)".

On the right side of the interface, there is a slide from a presentation titled "Coding TensorFlow" with the subtitle "Intro to Google Colabs". The slide includes the TensorFlow logo and two buttons: "TensorFlow 코딩하기" and "Google Colabs 소개".

# 딥 러닝 개발 환경

## ➊ 새로운 python notebook 실행

- 파일- 새 노트

The screenshot shows the Google Colaboratory interface. On the left, there's a sidebar with various options like '새 노트', '노트 열기', '노트 업로드', etc. A large blue arrow points from the '새 노트' option towards the main workspace. The main workspace shows a notebook titled 'Untitled1.ipynb'. The interface includes a toolbar at the top with file operations like 파일, 수정, 보기, 삽입, 런타임, 도구, 도움말. Below the toolbar, there are buttons for '+ 코드' and '+ 텍스트'. At the bottom of the workspace, there are several icons for file operations like download, copy, and settings.

Colaboratory에 오신 것을 환영합니다.

파일 수정 보기 삽입 런타임 도구 도움말

새 노트

노트 열기 Ctrl+N

노트 업로드

노트 다운로드

코드 셀로 전환

도화이브에 사본 저장

Github Gist로 사본 저장

Github에 사본 저장

저장 Ctrl+S

저장 파일 및 고정 Ctrl+Shift+S

설명하기, 기록

ipython 다운로드

py다운로드

초보자를 위한 가이드

언제 Ctrl+F

Untitled1.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말

+ 코드 + 텍스트

연결 ▾ 수정 가능 ▾

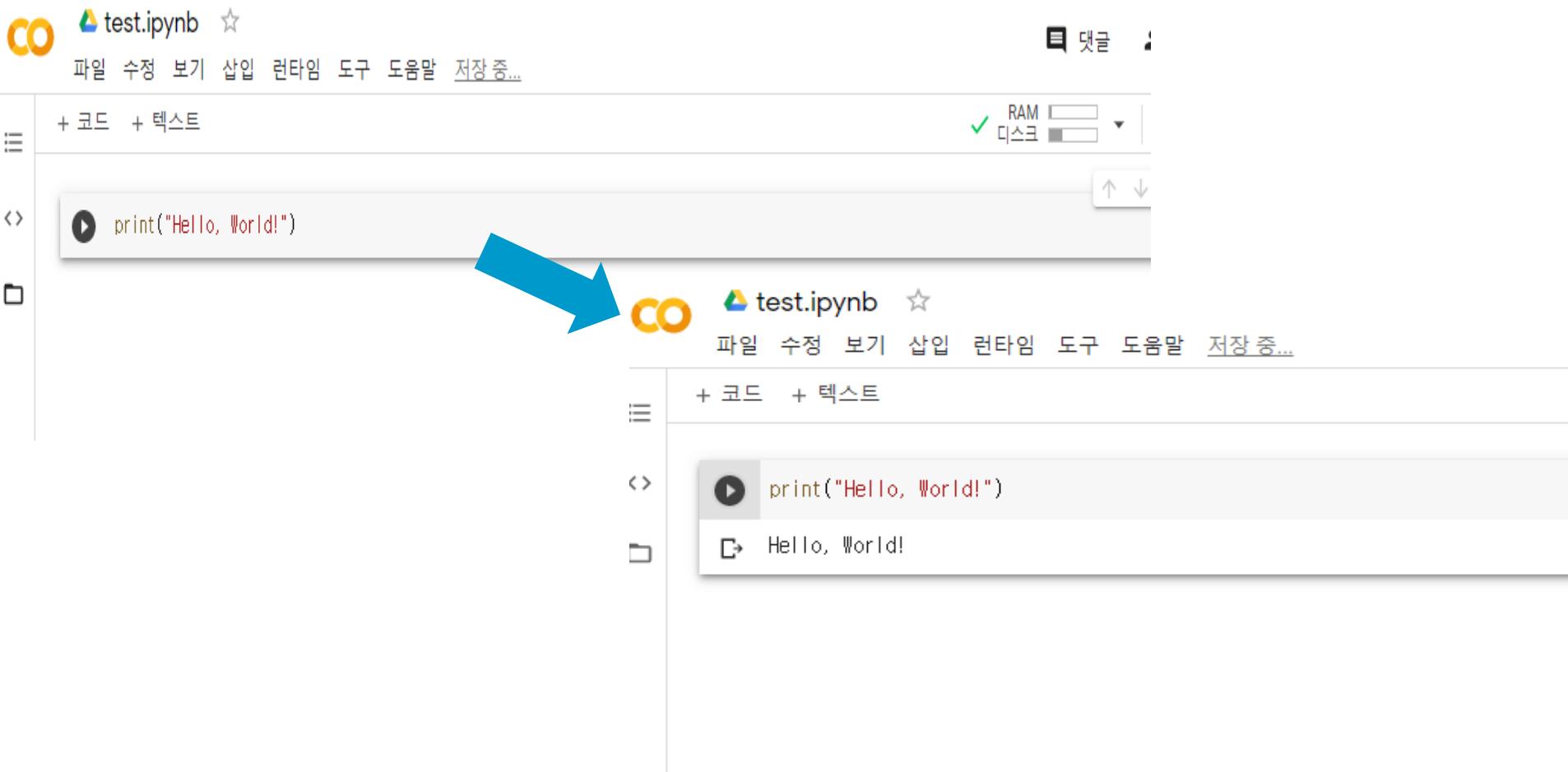
위 셀의 코드를 실행하려면 셀을 클릭하여 선택한 후 코드 왼쪽의 실행 버튼을 누르거나 단면 코드 수정을 바로 시작할 수 있습니다.

특정 셀에서 정의한 변수를 나중에 다른 셀에서 사용할 수 있습니다.

# 딥 러닝 개발 환경

## ④ 첫 python 프로그램 실행

- 첫 줄에 `print('Hello world')` 입력후 alt+enter (혹은  버튼 누르기)



The image shows two screenshots of a Jupyter Notebook interface. In the top screenshot, a cell contains the Python code `print("Hello, World!")`. A large blue arrow points from this cell to the bottom screenshot, which shows the same cell with its output: `Hello, World!`. The notebook has tabs for '코드' (Code) and '텍스트' (Text). The status bar at the bottom indicates '저장 중...' (Saving...).

# 딥 러닝 개발 환경

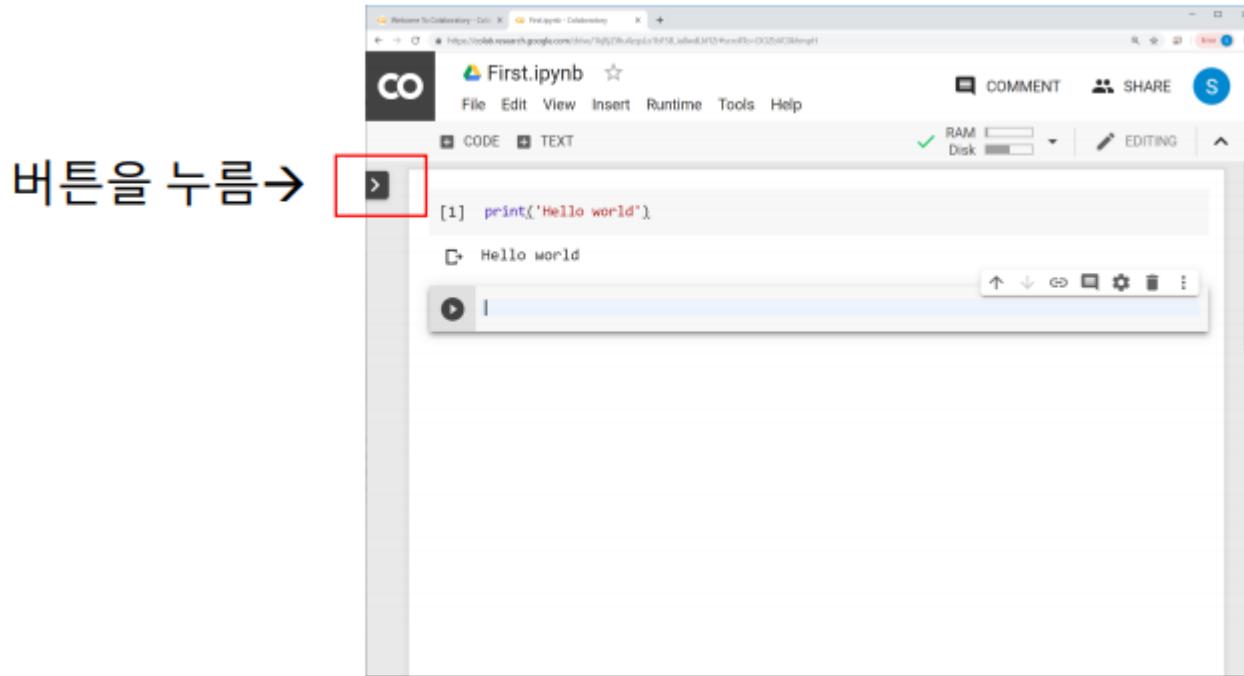
## ➊ 실습 파일 올리기

- 학습 데이터를 내 드라이브에 추가

# 딥 러닝 개발 환경

## ❸ 실습 파일 올리기

- 왼쪽 상단의 버튼을 눌러 파일을 업로드



# 딥 러닝 개발 환경

## ④ 실습 파일 올리기

- 왼쪽 상단의 버튼을 눌러 GDrive를 마운트 (인증코드 입력)



```
from google.colab import drive
drive.mount('/content/drive')

... Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6ok8odaf4n4g3ofee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=https://colab.research.google.com/drive/u/0/mount&response_type=code&scope=https://www.googleapis.com/auth/drive.readonly
Enter your authorization code:
```

The screenshot shows a Jupyter Notebook cell. It contains Python code to import the drive module from google.colab and then call drive.mount('/content/drive'). Below the code, there is a message instructing the user to go to a specific URL in a browser and enter an authorization code. The URL is highlighted with a red box. There is also a text input field for entering the authorization code.

# 딥 러닝 개발 환경 실습

## ④ 케라스를 이용한 딥러닝 모델 구현 과정

- 1. 데이터셋 생성하기
- 2. 모델 구성하기
- 3. 모델 학습과정 설정하기
- 4. 모델 학습시키기
- 5. 학습과정 살펴보기
- 6. 모델 평가하기
- 7. 모델 사용하기

# 딥 러닝 개발 환경 실습

## ④ 케라스를 이용한 딥 러닝 모델 구현 과정

- 데이터셋 생성하기
  - ✓ 원본 데이터를 불러오거나 시뮬레이션을 통해 데이터를 생성
  - ✓ 데이터로부터 학습 데이터셋, 검증 데이터셋, 평가 데이터셋을 생성
  - ✓ 딥 러닝 모델의 학습 및 평가를 위한 포맷 변환
- 모델 구성하기
  - ✓ 시퀀스 모델을 생성한 뒤 필요한 레이어를 추가하여 구성
  - ✓ 더 복잡한 모델이 필요할 때는 케라스 함수 API 사용

# 딥 러닝 개발 환경 실습

## ④ 케라스를 이용한 딥 러닝 모델 구현 과정

- 모델 학습과정 설정하기
  - 학습하기 전에 학습에 대한 설정 수행
  - 손실 함수 및 최적화 방법 정의
  - 케라스에서는 `compile()` 함수 사용
- 모델 학습시키기
  - 학습용 데이터셋을 이용하여 구성한 모델로 학습
  - 케라스에서는 `fit()` 함수 사용

# 딥 러닝 실습

## ④ 케라스를 이용한 딥 러닝 모델 구현 과정

- 학습과정 살펴보기
  - ✓ 모델 학습 시 학습 데이터셋, 검증 데이터셋의 손실 및 정확도 측정
  - ✓ 반복횟수에 따른 손실 및 정확도 추이를 보면서 학습 상황 판단
- 모델 평가하기
  - ✓ 준비된 평가 데이터셋으로 학습한 모델 평가
  - ✓ 케라스에서는 evaluate() 함수 사용
- 모델 사용하기
  - ✓ 임의의 입력으로 모델의 출력을 얻음
  - ✓ 케라스에서는 predict() 함수 사용

# 추가 실습

- 봇꽃 데이터 결정트리 시각화 실습
  - ✓ Anaconda 결정트리 시각화를 위해서는 ‘graphviz’ 필요
  - ✓ Colab에서는 추가 설치 필요 없음

# Thank You!

[www.ust.ac.kr](http://www.ust.ac.kr)