## 강의 목표

1. TensorFlow 2.0을 소개합니다.
2. TensorFlow 라이브러리의 장점을 살펴봅니다.

# 텐서플로(TensorFlow)


- **텐서플로**TensorFlow는 구글Google에서 개발하여 공개한 딥러닝/머신러닝을 위한 오픈소스 라이브러리입니다. 구글에서 내부 연구와 개발을 위해 사용하다가 2015년 11월 9일에 오픈소스로 대중에게 공개하였습니다. 텐서플로 라이브러리는 C++, JAVA, GO 등 다양한 언어를 지원하지만 기본적으로 파이썬Python 환경에 최적화되어 있습니다. 따라서 이 책에서도 파이썬 환경을 기준으로 설명합니다.
- 텐서플로가 유일한 딥러닝 라이브러리는 아닙니다. 페이스북Facebook이 주도적으로 개발하여 공개한 Lua 언어용 딥러닝 라이브러리인 토치Torch와 토치의 파이썬 버전인 파이토치PyTorch, 마이크로소프트Microsoft에서 공개한 CNTKCogNitive ToolKit 라이브러리 등 경쟁관계에 있는 라이브러리가 다수 존재합니다.

## TensorFlow의 장점

- 경쟁 라이브러리와 비교했을 때 TensorFlow 장점은 다음과 같습니다.

① 손쉬운 딥러닝 모델 구현을 가능하게하는 Python API 제공

② Mobile Device부터 멀티 GPU 클러스터까지 지원하는 폭넓은 Portability

③ 강력한 시각화를 지원하는 TensorBoard 제공

④ 전세계적으로 폭넓은 사용자 Community

⑤ Google의 강력한 지원과 발빠른 신기능 업데이트

# TensorFlow를 사용하는 기업들

- 전세계의 수많은 기업들에서 TensorFlow를 사용해서 딥러닝 알고리즘을 구현하고 있습니다.

## TensorFlow의 장점

- 특히 **활발한 커뮤니티**는 텐서플로가 지금까지 가장 인기 있는 딥러닝 라이브러리의 위치를 차지하고 있고, 앞으로도 그럴 확률을 높여주는 가장 큰 이유입니다. 이미 구글Google, 딥마인드DeepMind, 우버Uber, 스냅챗Snapchat 등 많은 글로벌 기업들에서 텐서플로를 활발히 사용하고 있고, 많은 연구자들이 새로운 알고리즘을 구현할 때 텐서플로를 이용해서 코드를 작성하고 있습니다.
- 이런 이유 때문에 텐서플로는 앞으로도 가장 인기있는 딥러닝 라이브러리의 위치를 차지할 확률이 높습니다. 따라서 새롭게 딥러닝을 공부하시는 분들은 큰 고민없이 텐서플로를 이용해서 딥러닝을 학습하는 것을 추천드립니다.

## Tensor = n차원 행렬

- Tensor는 n차원 행렬을 지칭하는 용어입니다.

0-d tensor : scalar

1-d tensor : vector

2-d tensor : matrix

## TensorFlow 2.0 Release

- 2019-09-30에 **TensorFlow 2.0** 버전이 릴리즈되었습니다.
- 그동안의 많은 개발사항을 반영한 메이저 업데이트가 이루어진만큼 API 구조가 대대적으로 변경되었습니다.

## TensorFlow 주요변화사항

- 2019-09-30에 TensorFlow 2.0 버전이 릴리즈되었습니다.
- 그동안의 많은 개발사항을 반영한 메이저 업데이트가 이루어진만큼 API 구조가 대대적으로 변경되었습니다.

① tf.Session 삭제 & Eager Execution이 기본적으로 적용됨

② tf.placeholder 삭제

③ 전역적으로 처리되던 부분이 삭제 → tf.global_variable_initializer() 삭제

④ Function을 이용한 Programming

⑤ Keras API 강화

# TensorFlow 1.x code example

```
in_a = tf.placeholder(dtype=tf.float32, shape=(2))
in_b = tf.placeholder(dtype=tf.float32, shape=(2))

def forward(x):
  with tf.variable_scope("matmul", reuse=tf.AUTO_REUSE):
    W = tf.get_variable("W", initializer=tf.ones(shape=(2,2)),
                        regularizer=tf.contrib.layers.l2_regularizer(0.04))
    b = tf.get_variable("b", initializer=tf.zeros(shape=(2)))
    return W * x + b

out_a = forward(in_a)
out_b = forward(in_b)

reg_loss = tf.losses.get_regularization_loss(scope="matmul")

with tf.Session() as sess:
  sess.run(tf.global_variables_initializer())
  outs = sess.run([out_a, out_b, reg_loss],
                  feed_dict={in_a: [1, 0], in_b: [0, 1]})
```

# TensorFlow 2.x code example

```
W = tf.Variable(tf.ones(shape=(2,2)), name="W")
b = tf.Variable(tf.zeros(shape=(2)), name="b")

@tf.function
def forward(x):
  return W * x + b

out_a = forward([1,0])
print(out_a)
```

# Thank you!