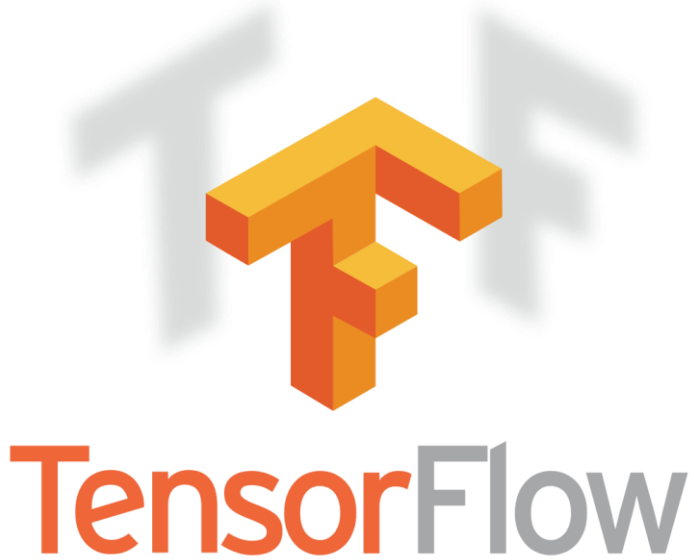
텐서플로우 소개

OSI Lab, KAIST
윤세영 교수님

조교: 김창환, 배상민

텐서플로우란?

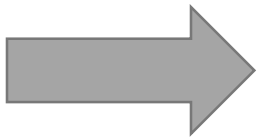
- 기계학습을 위한 오픈소스 라이브러리
  - 구글에서 개발하여 2015년 11월에 공개
- 지원 플랫폼
  - 리눅스, macOS, 윈도우, 안드로이드, 자바스크립트
- 지원 프로그래밍 언어: **Python**
  - Java, C, Go를 위한 API도 제공
- 지원 하드웨어
  - x86-64 CPU (인텔, AMD), ARM CPU (라즈베리 파이)
  - NVIDIA GPU (CUDA)
  - TPU (Google의 Tensor 처리 가속화 접적회로)



텐서플로우의 필요성

- 고차원 데이터를 표현하고자 하는 **텐서(Tensor)의 추상화**

```
# 3-D tensor using Python list (3 x 3 x 3)
tensor = [[[0, 0, 0],[0, 0, 0], [0, 0, 0]],
          [[0, 0, 0], [0, 0, 0], [0, 0, 0]],
          [[0, 0, 0], [0, 0, 0], [0, 0, 0]]]
```



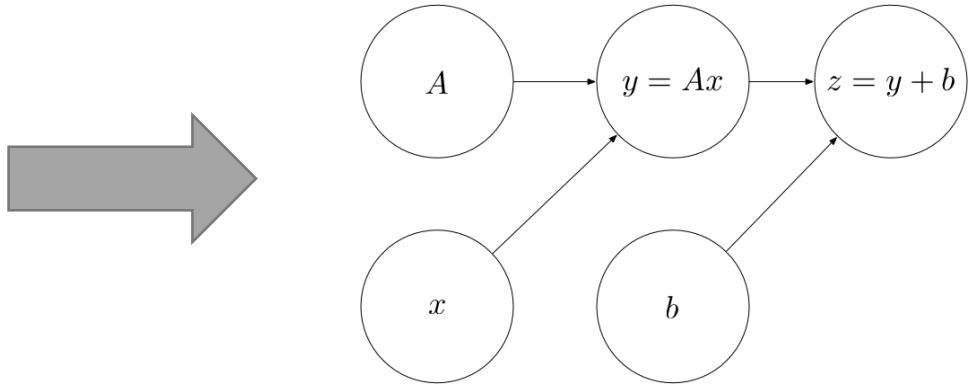
```
# 3-D tensor using tensorflow
tensor = tf.zeros([3, 3, 3])
```

- 벡터, 행렬, 텐서 간의 **연산을 위한 API** 제공
  - 사칙연산
  - 선형 대수학 연산 (transpose, determinant, eigenvalue, eigenvector, etc.)
  - Convolution
  - 기타 등등 매우 광범위한 연산 함수 제공

텐서플로우의 필요성

- **Computational Graph**의 생성 및 관리
  - 기계학습, 특히 딥러닝에서 인공 신경망의 동작을 표현하기 위한 그래프

$$z(A, x, b) = Ax + b$$



- **Automatic Differentiation**
  - Computaional graph가 필요한 **핵심** 이유
  - 기계학습이 이루어지기 위해서 모델의 **weight 업데이트**가 필요
  - 그를 위해선 최종 손실(loss)함수를 모델의 각 weight으로 **미분**해야 함
  - 보통 딥러닝 모델의 weight이 **수천만개**이기 때문에 효율적인 미분을 위해 Computational graph를 이용하여 **Automatic Differentiation**을 수행

다른 머신러닝 라이브러리

- PyTorch
  - Tensorflow와 유사하게 Python을 이용한 기계학습 라이브러리
  - Facebook에서 주로 개발, 2016년에 최초 공개
  - 마찬가지로 리눅스, macOS, 윈도우 지원 및 CUDA를 통해 NVIDIA GPU에서도 사용 가능

- Caffe
  - Tensorflow와 Pytorch 이전에 개발된 기계학습 라이브러리
  - C++를 주언어로 사용. Python 대비 성능상의 이점이 존재하나 Python의 생산성에 밀려 현재 활발히 사용되지는 않음
  - 마찬가지로 CUDA를 통한 NVIDIA GPU 가속이 가능. C++를 사용하는 만큼 직접 CUDA 코드를 작성해서 개발할 수도 있음

Tensorflow 1

- **Session**
  - 텐서플로우 1의 핵심 특징으로, Computational graph를 실행하기 위한 클래스
  - Python을 쓰면서도 Python 처럼 interactive한 환경을 지원하지 못하게 한 핵심 요인

- **Eager Execution**
  - Session이 가지고 있는 최대 단점인 **Interactive** 환경 미지원을 해결
  - Python 코드를 돌리듯이 텐서플로우 함수들을 하나씩 실행해볼 수 있음

```python
x = tf.placeholder(tf.float32, shape=[1, 1])
m = tf.matmul(x, x)

print(m)
# Tensor("MatMul:0", shape=(1, 1),
dtype=float32)

with tf.Session() as sess:
  m_out = sess.run(m, feed_dict={x: [[2.]]})
print(m_out)
# [[4.]]
```

Tensorflow session

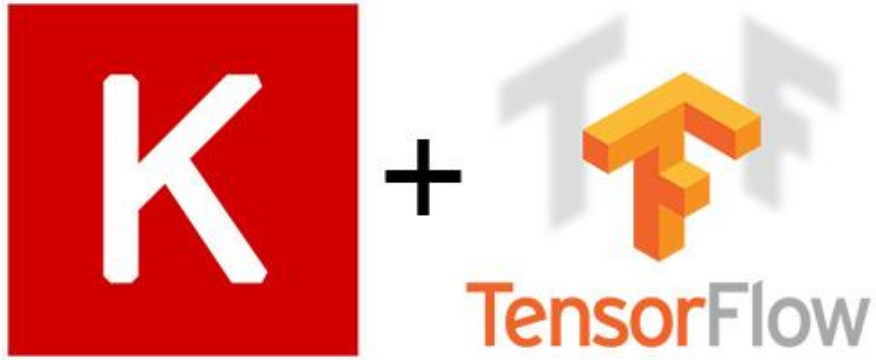
```python
x = [[2.]]
m = tf.matmul(x, x)

print(m)
# tf.Tensor([[4.]], dtype=float32,
shape=(1,1))
```

Eager execution

Keras

- 텐서플로우 API를 **추상화**한 API 라이브러리
  - 기존 텐서플로우 1의 난해한 API를 정리
  - 텐서플로우 1의 실행 모드를 자동으로 Eager Execution 환경으로 전환
- 사용자에게 친숙한 API로 모델의 빠른 개발과 실험을 촉진



Keras

```
sess = tf.InteractiveSession()
sess.run(init)
global_step = 0
# Number of training iterations in each epoch
num_tr_iter = int(len(y_train) / batch_size)
for epoch in range(epochs):
    print('Training epoch: {}'.format(epoch + 1))
    x_train, y_train = randomize(x_train, y_train)
    for iteration in range(num_tr_iter):
        global_step += 1
        start = iteration * batch_size
        end = (iteration + 1) * batch_size
        x_batch, y_batch = get_next_batch(x_train, y_train, start, end)

        # Run optimization op (backprop)
        feed_dict_batch = {x: x_batch, y: y_batch}
        sess.run(optimizer, feed_dict=feed_dict_batch)

        if iteration % display_freq == 0:
            # Calculate and display the batch loss and accuracy
            loss_batch, acc_batch = sess.run([loss, accuracy],
                                             feed_dict=feed_dict_batch)

            print("iter {0:3d}:\t Loss={1:.2f},\tTraining Accuracy={2:.01%}".
                  format(iteration, loss_batch, acc_batch))

    # Run validation after every epoch
    feed_dict_valid = {x: x_valid[:1000], y: y_valid[:1000]}
    loss_valid, acc_valid = sess.run([loss, accuracy], feed_dict=feed_dict_valid)
    print('---------------------------------------------------------')
    print("Epoch: {0}, validation loss: {1:.2f}, validation accuracy: {2:.01%}".
          format(epoch + 1, loss_valid, acc_valid))
    print('---------------------------------------------------------')
```

Tensorflow 1

```
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)
```

Tensorflow with Keras

Tensorflow 2

- 텐서플로우 1 버전의 불편함을 대거 수정한 메이저 업데이트
  - 2019년 9월에 공식적으로 업데이트

- 핵심 변경 사항
  - **Eager Execution을 기본값으로 활성화**
  - **Keras API를 내장하여 API 간소화**

- 텐서플로우 1을 하위 호환으로 지원하나, **2버전의 코드 스타일로 사용할 것을 권장**
  - Python을 처음 배울 때 Python 2가 아닌 Python 3로 배우는 것과 유사