15기 정규세션

ToBig's 14기 강의자

서아라

Framework

Contents

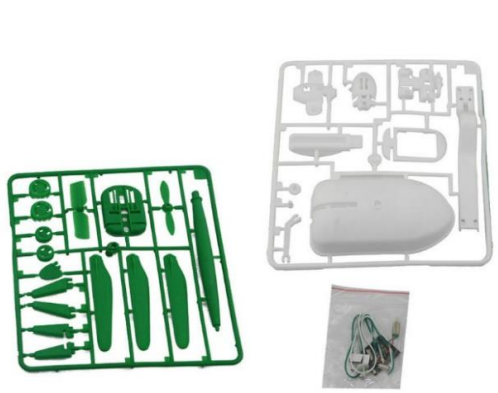
Unit 01 | Framework란?

정의 : 응용 프로그램을 개발하기 위한 여러 라이브러리나 모듈 등을 하나로 묶은 일종의 패키지

검증된 알고리즘을 사용할 때마다 구현하는 것은 비효율적,
빠르게 핵심만 구현하도록 도와주는 역할

Unit 01 | Framework란?

Frame(=틀) + Work(=일하다)
= Framework(=일정하게 짜여진 틀, 뼈대를 가지고 일하다.)



<Frame>

<Work>

<Program>

Unit 01 | Framework란?

Unit 01 | Framework란?

Caffe

Contents

Unit 02 | Tensorflow Tutorial

Tensorflow란?

TensorFlow를 사용해야 하는 이유

TensorFlow는 머신러닝을 위한 엔드 투 엔드 오픈소스 플랫폼입니다. 도구, 라이브러리, 커뮤니티 리소스로 구성된 포괄적이고 유연한 생태계를 통해 연구원들은 ML에서 첨단 기술을 구현할 수 있고 개발자들은 ML이 접목된 애플리케이션을 손쉽게 빌드 및 배포할 수 있습니다.

정보 →

손쉬운 모델 빌드

즉각적인 모델 반복 및 손쉬운 디버깅을 가능하게 하는 즉시 실행 기능이 포함된 Keras와 같은 높은 수준의 직관적인 API를 사용하여 ML 모델을 쉽게 빌드하고 학습시키세요.

어디서든 강력한 ML 제작

사용하는 언어에 상관없이 클라우드, 온프렘, 브라우저 또는 기기에서 모델을 손쉽게 학습시키고 배포하세요.

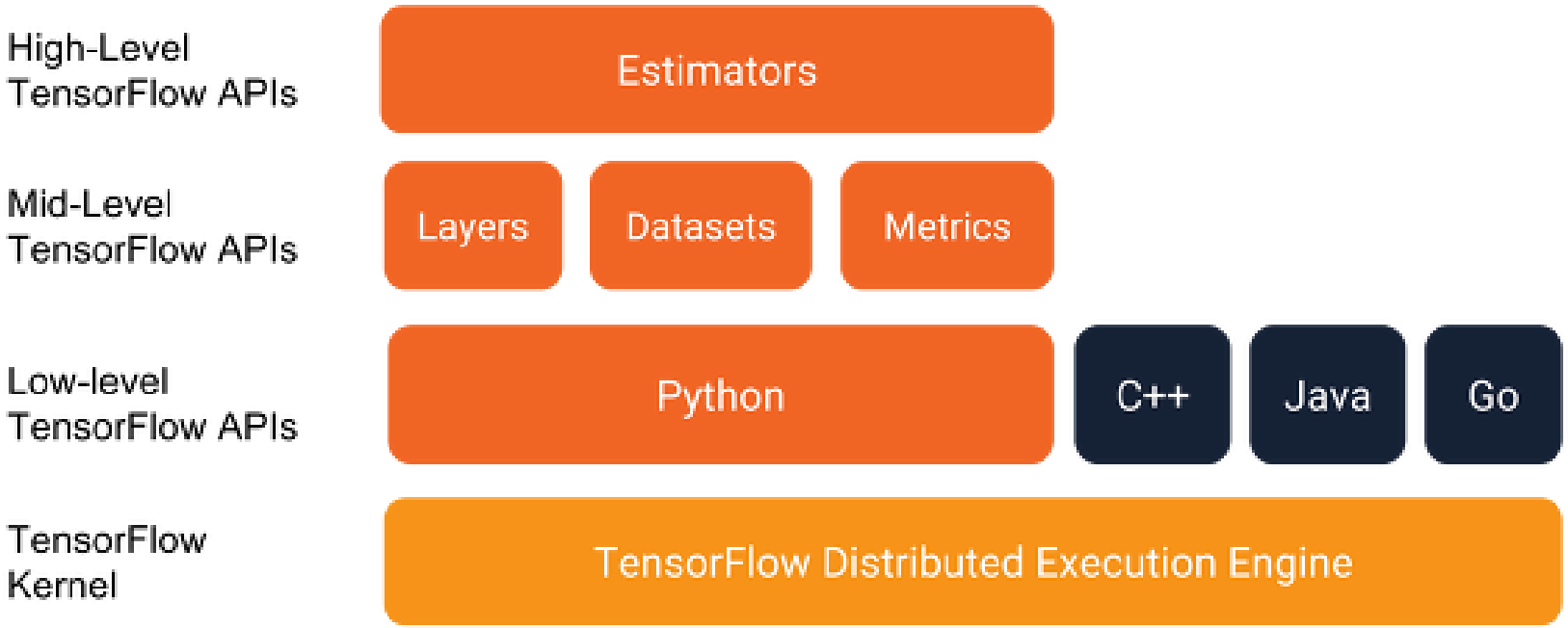
연구를 위한 강력한 실험

새로운 아이디어를 개념부터 코드, 최첨단 모델, 게시에 이르기까지 더 빠르게 발전시킬 수 있는 간단하고 유연한 아키텍처

딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공해주는 머신러닝 라이브러리로, 구글에서 만들었음.

Unit 02 | Tensorflow Tutorial

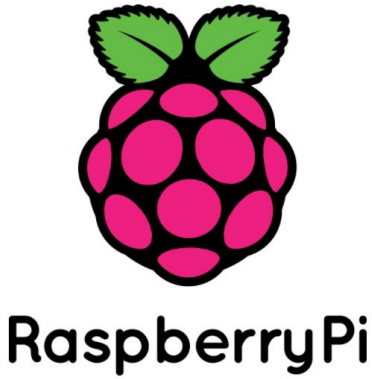
Tensorflow란?



파이썬을 최우선으로 지원하여 대다수 편의 기능이 **파이썬 라이브로리로만 구현**되어있음.

Unit 02 | Tensorflow Tutorial

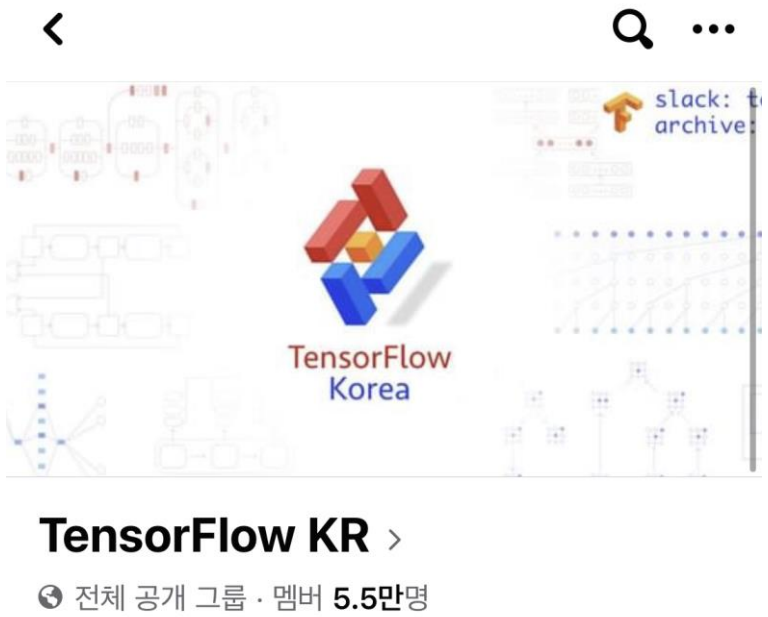
왜 Tensorflow인가?



윈도우, 맥, 리눅스 등의 **다양한 시스템에서 쉽게 사용할 수 있도록 지원**

Unit 02 | Tensorflow Tutorial

왜 Tensorflow인가?



현존하는 머신러닝 라이브러리 중 **커뮤니티가 가장 활발함**

Tensorflow를 이용한 Simple Linear Regession

$$H(x) = Wx + b$$

```
x_data = [1, 2, 3, 4, 5]
y_data = [1, 2, 3, 4, 5]

W = tf.Variable(2.0)
b = tf.Variable(0.5)

hypothesis = W * x_data + b
```

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

```
cost = tf.reduce_mean(tf.square(hypothesis - y_data))
```

Tensorflow를 이용한 Simple Linear Regession

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

```
cost = tf.reduce_mean(tf.square(hypothesis - y_data))
```

- tf.reduce_mean()

```
v =[1., 2., 3., 4.]
tf.reduce_mean(v) # 2.5
```

- tf.square()

```
tf.square(3) # 9
```

Unit 02 | Tensorflow Tutorial

Tensorflow를 이용한 Simple Linear Regession

$$\underset{W,b}{minimize}\ cost(W,b)$$

```
with tf.GradientTape() as tape:
    hypothesis = W * x_data + b
    cost = tf.reduce_mean(tf.square(hypothesis - y_data))

W_grad, b_grad = tape.gradient(cost, [W, b])
```

```
learning_rate = 0.01

W.assign_sub(learning_rate * W_grad)
b.assign_sub(learning_rate * b_grad)
```

A.assign_sub(B)

```
A = A - B
A -= B
```

Tensorflow를 이용한 Simple Linear Regession

```
W = tf.Variable(2.9)
b = tf.Variable(0.5)

for i in range(100):
    with tf.GradientTape() as tape:
        hypothesis = W * x_data + b
        cost = tf.reduce_mean(tf.square(hypothesis - y_data))
    W_grad, b_grad = tape.gradient(cost, [W, b])
    W.assign_sub(learning_rate * W_grad)
    b.assign_sub(learning_rate * b_grad)
    if i % 10 == 0:
      print("{:5}|{:10.4f}|{:10.4f}|{:10.6f}".format(i, W.numpy(), b.numpy(), cost))
```

Tensorflow를 이용한 Simple Linear Regession

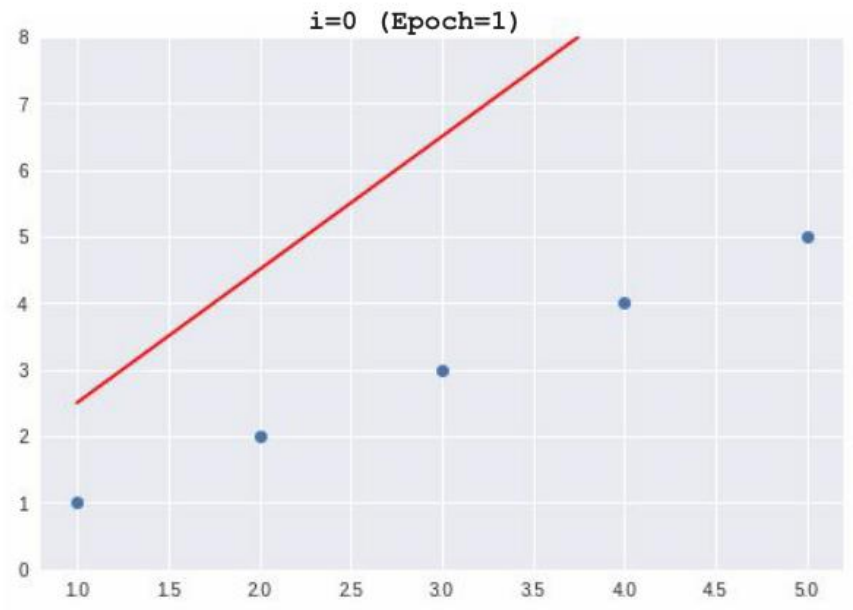
```python
import tensorflow as tf
import numpy as np

# Data
x_data = [1, 2, 3, 4, 5]
y_data = [1, 2, 3, 4, 5]

# W, b initialize
W = tf.Variable(2.9)
b = tf.Variable(0.5)

# Assign learning rate
learning_rate = 0.01

# W, b update
for i in range(100):
    # Gradient descent
    with tf.GradientTape() as tape:
        hypothesis = W * x_data + b
        cost = tf.reduce_mean(tf.square(hypothesis - y_data))
    W_grad, b_grad = tape.gradient(cost, [W, b])
    W.assign_sub(learning_rate * W_grad)
    b.assign_sub(learning_rate * b_grad)
    if i % 10 == 0:
      print("{:5}|{:10.4f}|{:10.4f}|{:10.6f}".format(i, W.numpy(), b.numpy(), cost))

print()
```

```python
# predict
print(W * 5 + b)
print(W * 2.5 + b)
```

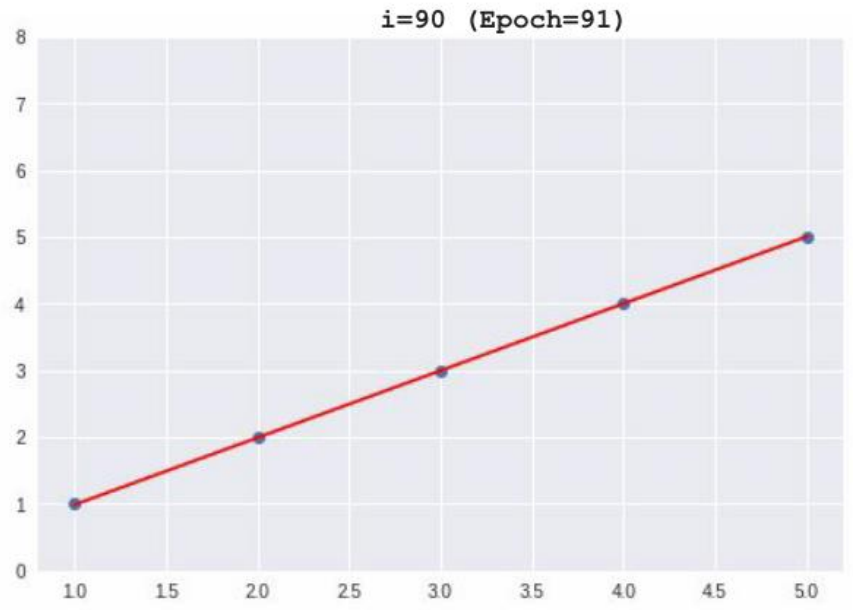
```
    0|    2.4520|    0.3760| 45.660004
   10|    1.1036|    0.0034|  0.206336
   20|    1.0128|   -0.0209|  0.001026
   30|    1.0065|   -0.0218|  0.000093
   40|    1.0059|   -0.0212|  0.000083
   50|    1.0057|   -0.0205|  0.000077
   60|    1.0055|   -0.0198|  0.000072
   70|    1.0053|   -0.0192|  0.000067
   80|    1.0051|   -0.0185|  0.000063
   90|    1.0050|   -0.0179|  0.000059

tf.Tensor(5.0066934, shape=(), dtype=float32)
tf.Tensor(2.4946523, shape=(), dtype=float32)
```

Tensorflow를 이용한 Simple Linear Regession

```
W = tf.Variable(2.9)
b = tf.Variable(0.5)
```




```
W = 2.4520
b = 0.3760
```




```
W = 1.0050
b = -0.0179
```

Tensorflow를 이용한 Simple Linear Regession

Hypothesis $H(x) = Wx + b$

```
# Data
x_data = [1, 2, 3, 4, 5]
y_data = [1, 2, 3, 4, 5]
```

```
print(W * 5 + b)
print(W * 2.5 + b)
```

```
tf.Tensor(5.0066934, shape=(), dtype=float32)
tf.Tensor(2.4946523, shape=(), dtype=float32)
```

Contents

Keras란?

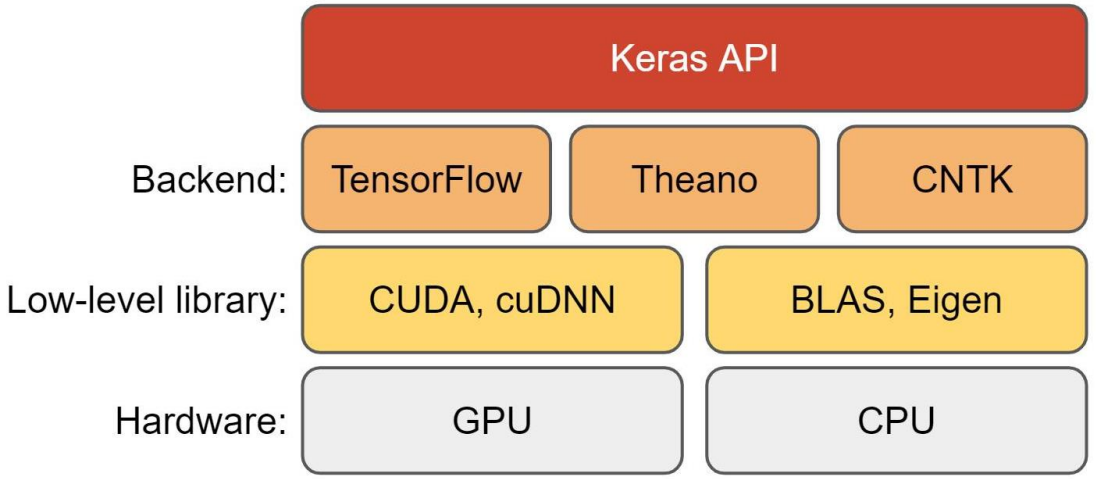
케라스: 파이썬 딥러닝 라이브러리

케라스에 오신걸 환영합니다.

케라스는 파이썬으로 작성된 고수준 신경망 API로 TensorFlow, CNTK, 혹은 Theano와 함께 사용하실 수 있습니다. 빠른 실험에 특히 중점을 두고 있습니다. *아이디어를 결과물로 최대한 빠르게 구현하는 것은 훌륭한 연구의 핵심입니다.*

Unit 03 | Keras Tutorial

Q. Tensorflow와 Keras 이 둘은 무슨 사이인가요?



Tensorflow와 마찬가지로 Keras 역시 라이브러리인데, Keras는 **Tensorflow(혹은 Theano나 CNTK) 위에서 동작하는 라이브러리**입니다.

Unit 03 | Keras Tutorial

Q. Tensorflow가 있는데 왜 그 위에서 동작하는 Keras가 필요하나요?

Tensorflow는 처음 머신러닝을 접하는 사람이라면 아직 사용하기에는 어려운 부분이 많음. 반면, Keras는 사용자 친화적으로 개발된 라이브러리라 매우 사용이 편하기 때문에 **머신러닝 초보자가 공부하기에 안성맞춤!**

Q. 그럼 Keras만 사용해도 충분할까요?

단순한 신경망을 구현하는 정도라면 Keras만으로 충분하겠지만, **Tensorflow를 사용하면 더 디테일한 조작이 가능**.
따라서 **사용 목적에 따라서 선택하는 것이 좋을 것!**

Keras를 이용한 Classifier Training

1. 텐서플로우 라이브러리 임포트

```
!pip install -q tensorflow-gpu==2.0.0-rc1
import tensorflow as tf
```

2. MNIST 데이터셋을 로드하여 준비

```
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
```

Keras를 이용한 Classifier Training

3. 층을 쌓아서 모델 객체를 만든 후, optimizer와 loss함수 정하기

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Keras를 이용한 Classifier Training

4. 모델을 훈련하고 평가하기

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

model.evaluate(x_test,  y_test, verbose=2)
```

```
60000/60000 [==============================] - 4s 68us/sample - loss: 0.2941 - accuracy: 0.9140
Epoch 2/5
60000/60000 [==============================] - 4s 62us/sample - loss: 0.1396 - accuracy: 0.9587
Epoch 3/5
60000/60000 [==============================] - 4s 62us/sample - loss: 0.1046 - accuracy: 0.9680
Epoch 4/5
60000/60000 [==============================] - 4s 62us/sample - loss: 0.0859 - accuracy: 0.9742
Epoch 5/5
60000/60000 [==============================] - 4s 62us/sample - loss: 0.0724 - accuracy: 0.9771
10000/1 - 0s - loss: 0.0345 - accuracy: 0.9788

[0.06729823819857557, 0.9788]
```

Contents

Unit 04 | Pytorch Tutorial

Pytorch란?

Deep Learning with PyTorch

PyTorch 소개

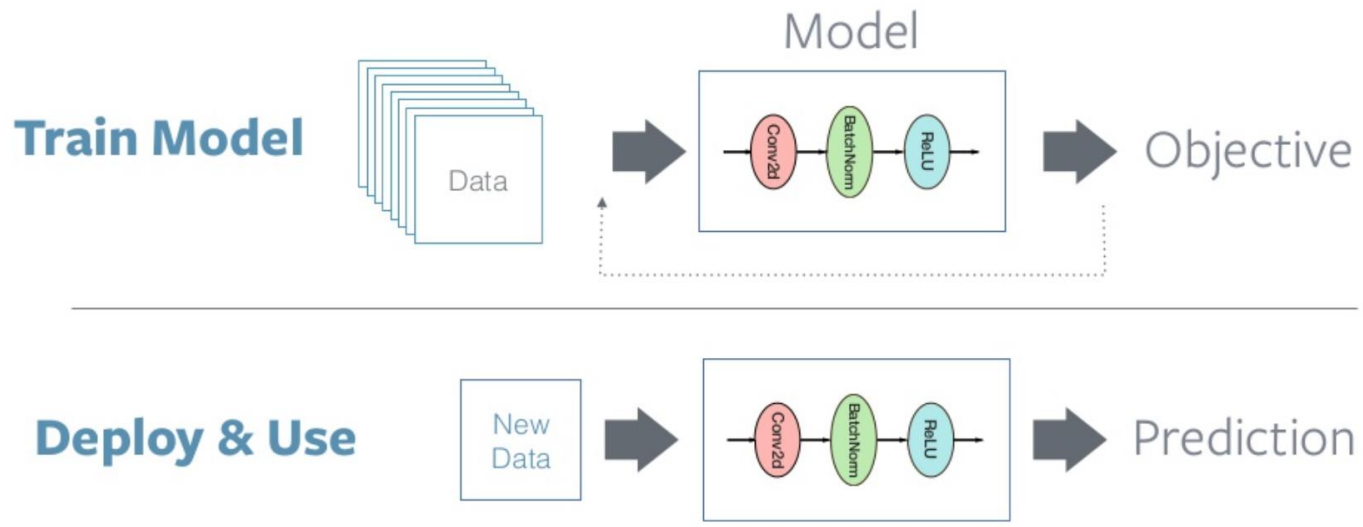
PyTorch(파이토치)는 연구용 프로토타입부터 상용 제품까지 빠르게 만들 수 있는 오픈 소스 머신러닝 프레임워크입니다.

PyTorch는 사용자 친화적인 프론트엔드(front-end)와 분산 학습, 다양한 도구와 라이브러리를 통해 빠르고 유연한 실험 및 효과적인 상용화를 가능하게 합니다.

PyTorch에 대한 더 자세한 소개는 공식 홈페이지 또는 공식 저장소에서 확인하실 수 있습니다.

2016년에 발표된 **딥러닝 구현을 위한 파이썬 기반의 오픈소스 머신러닝 라이브러리**로
facebook 인공지능 연구팀에 의해 개발되었음

Pytorch란?



현재의 딥러닝 트렌드에 맞춰 제작된 다른 프레임워크들과 다르게
pytorch는 **미래의 연구 트렌드에도 대응할 수 있도록 제작됨**

Unit 04 | Pytorch Tutorial

Pytorch란?

PyTorch KR ›

전체 공개 그룹 · 멤버 **1.1만**명

Tensorflow 못지 않게 **커뮤니티가 활발함**

Pytorch를 이용한 Classifier Training

1. 환경 설정하기

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torchvision import transforms, datasets
```

2. GPU 설정하기

```
USE_CUDA = torch.cuda.is_available()
DEVICE = torch.device("cuda" if USE_CUDA else "cpu")
```

Pytorch를 이용한 Classifier Training

3. 에폭 및 배치사이즈 설정

```
EPOCHS = 30
BATCH_SIZE = 64
```

4. 이미지를 텐서로 바꿔주기

```
transform = transforms.Compose([
    transforms.ToTensor()
])
```

Pytorch를 이용한 Classifier Training

5. 데이터셋 불러오기

```
trainset = datasets.FashionMNIST(
    root      = './.data/',
    train     = True,
    download  = True,
    transform = transform
)
testset = datasets.FashionMNIST(
    root      = './.data/',
    train     = False,
    download  = True,
    transform = transform
)

train_loader = torch.utils.data.DataLoader(
    dataset     = trainset,
    batch_size  = BATCH_SIZE,
    shuffle     = True,
)
test_loader = torch.utils.data.DataLoader(
    dataset     = testset,
    batch_size  = BATCH_SIZE,
    shuffle     = True,
)
```

Pytorch를 이용한 Classifier Training

6. 이미지 분류를 위한 인공 신경망 구현

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(784, 256)
        self.fc2 = nn.Linear(256, 128)
        self.fc3 = nn.Linear(128, 10)

    def forward(self, x):
        x = x.view(-1, 784)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Pytorch를 이용한 Classifier Training

7. GPU or CPU

```
model         = Net().to(DEVICE)
```

8. Optimizer 설정

```
optimizer     = optim.SGD(model.parameters(), lr=0.01)
```

Pytorch를 이용한 Classifier Training

9. Train

```
def train(model, train_loader, optimizer):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        # 학습 데이터를 DEVICE의 메모리로 보냄
        data, target = data.to(DEVICE), target.to(DEVICE)
        optimizer.zero_grad()
        output = model(data)
        loss = F.cross_entropy(output, target)
        loss.backward()
        optimizer.step()
```

Pytorch를 이용한 Classifier Training

10. Test

```
def evaluate(model, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(DEVICE), target.to(DEVICE)
            output = model(data)

            # 모든 오차 더하기
            test_loss += F.cross_entropy(output, target,
                                         reduction='sum').item()

            # 가장 큰 값을 가진 클래스가 모델의 예측입니다.
            # 예측과 정답을 비교하여 일치할 경우 correct에 1을 더합니다.
            pred = output.max(1, keepdim=True)[1]
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)
    test_accuracy = 100. * correct / len(test_loader.dataset)
    return test_loss, test_accuracy
```

Pytorch를 이용한 Classifier Training

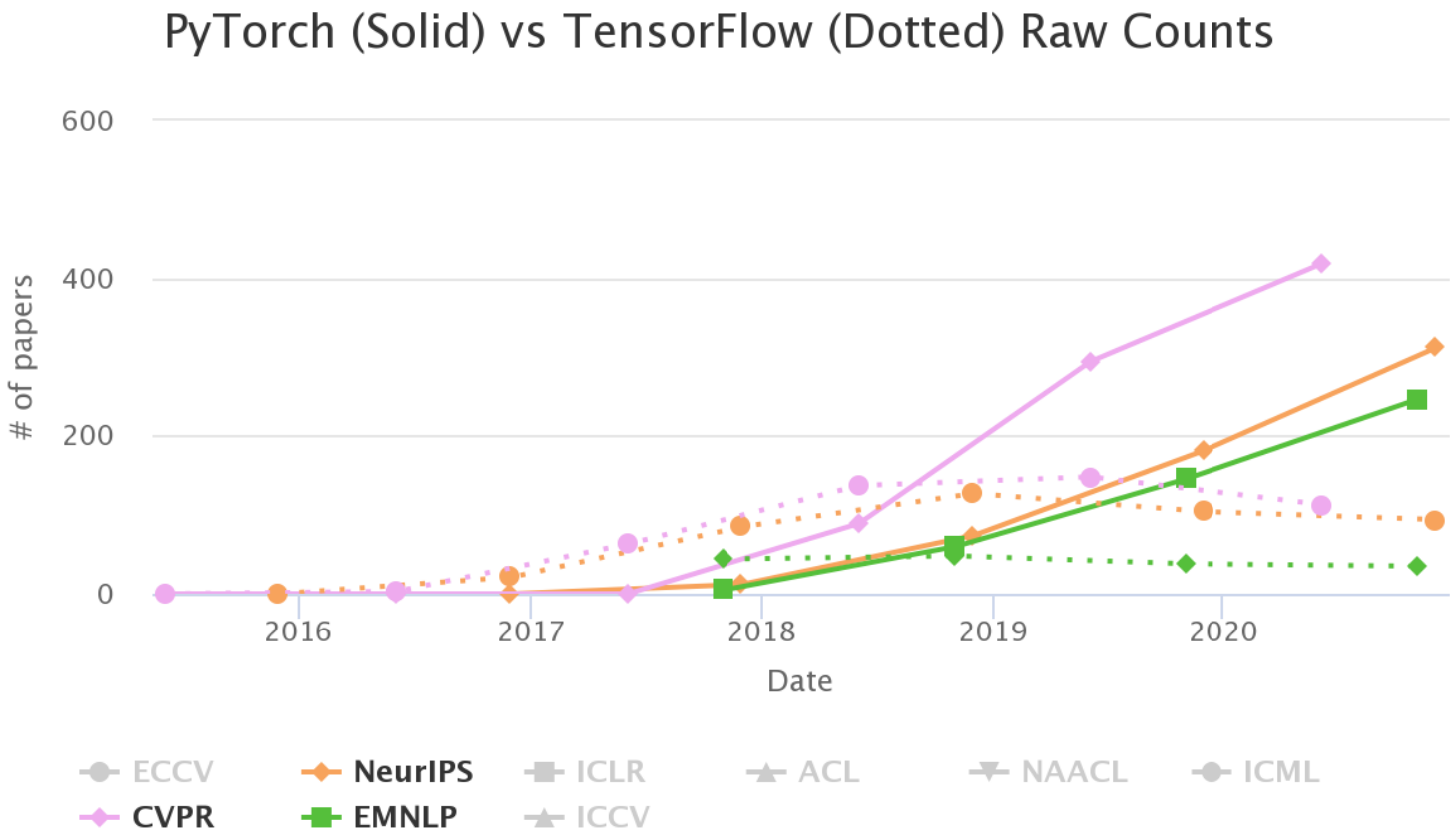
11. 코드 돌려보기

```
for epoch in range(1, EPOCHS + 1):
    train(model, train_loader, optimizer)
    test_loss, test_accuracy = evaluate(model, test_loader)

    print('[{}] Test Loss: {:.4f}, Accuracy: {:.2f}%'.format(
          epoch, test_loss, test_accuracy))
```

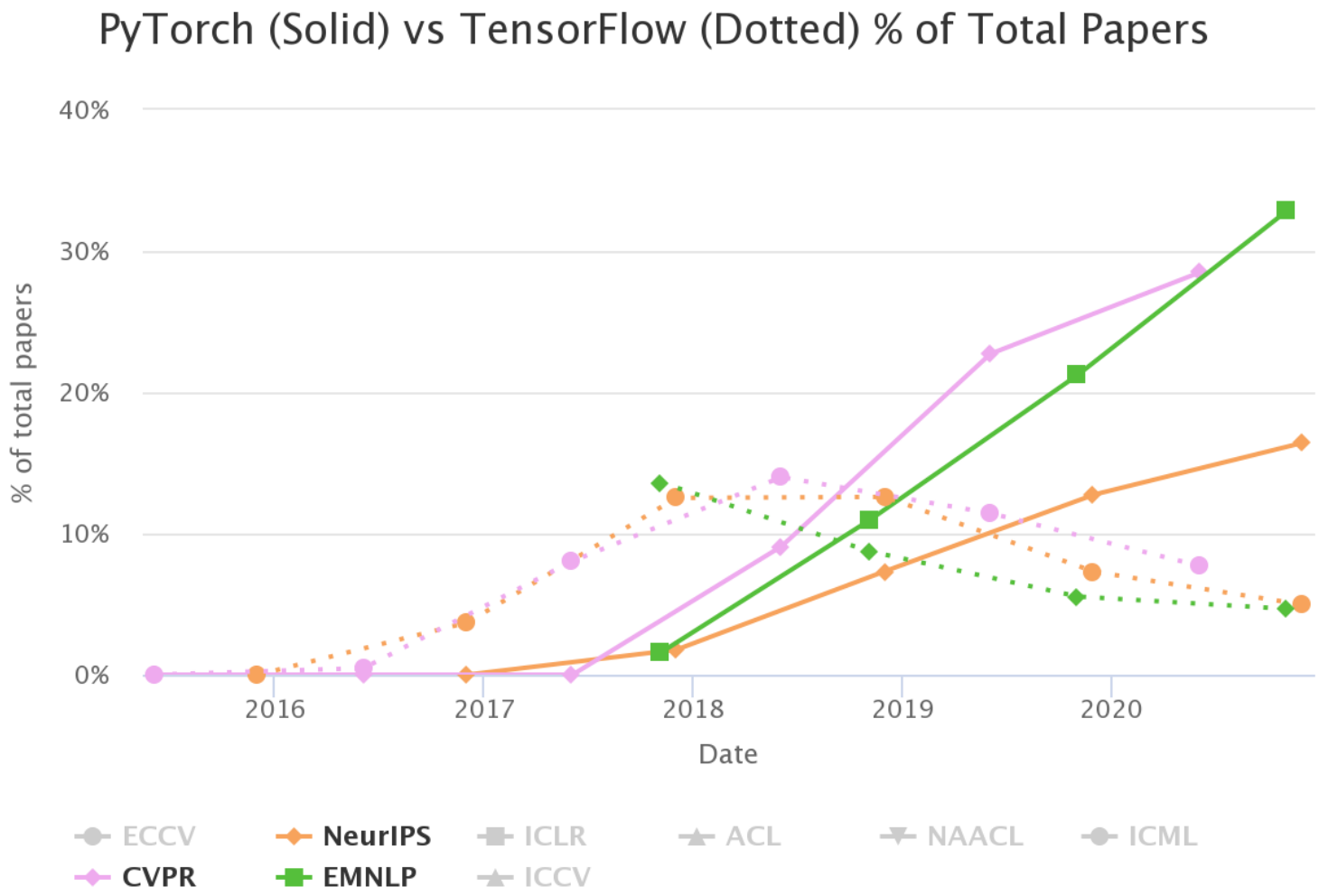
```
[1] Test Loss: 0.8419, Accuracy: 67.97%
[2] Test Loss: 0.6651, Accuracy: 76.44%
[3] Test Loss: 0.5845, Accuracy: 79.20%
[4] Test Loss: 0.5463, Accuracy: 80.69%
[5] Test Loss: 0.5213, Accuracy: 81.86%
[6] Test Loss: 0.4973, Accuracy: 82.26%
[7] Test Loss: 0.4911, Accuracy: 82.66%
[8] Test Loss: 0.5134, Accuracy: 81.31%
[9] Test Loss: 0.4628, Accuracy: 83.50%
[10] Test Loss: 0.4546, Accuracy: 83.81%
[11] Test Loss: 0.4541, Accuracy: 83.78%
[12] Test Loss: 0.4366, Accuracy: 84.45%
[13] Test Loss: 0.4486, Accuracy: 83.66%
[14] Test Loss: 0.4312, Accuracy: 84.81%
[15] Test Loss: 0.4228, Accuracy: 85.18%
[16] Test Loss: 0.4332, Accuracy: 84.61%
[17] Test Loss: 0.4132, Accuracy: 85.38%
[18] Test Loss: 0.4072, Accuracy: 85.84%
[19] Test Loss: 0.4054, Accuracy: 85.52%
[20] Test Loss: 0.4459, Accuracy: 84.59%
[21] Test Loss: 0.4092, Accuracy: 85.73%
[22] Test Loss: 0.3908, Accuracy: 86.23%
[23] Test Loss: 0.4023, Accuracy: 85.48%
[24] Test Loss: 0.3913, Accuracy: 86.12%
[25] Test Loss: 0.4025, Accuracy: 85.70%
[26] Test Loss: 0.3844, Accuracy: 86.35%
[27] Test Loss: 0.3772, Accuracy: 86.73%
[28] Test Loss: 0.3823, Accuracy: 86.34%
[29] Test Loss: 0.3743, Accuracy: 86.85%
[30] Test Loss: 0.3764, Accuracy: 86.57%
```

Contents

Unit 05 | Pytorch vs Tensorflow



Unit 05 | Pytorch vs Tensorflow



Kaggle 참가 과제

1. MNIST 대회 참가 후 리더보드 및 ipynb 파일 제출(Github)

2. NN심화에서 배운 다양한 기법 적용

3. 오토인코더, 전이 학습 등의 추가 기법 사용시 가산점

4. 자세한 주석 역시 가산점

5. 우수과제 선정시 Leaderboard 및 ipynb 파일을 종합적으로 볼 것

https://www.kaggle.com/t/e24d68edf7004da29420ecd0e8775b54

Q & A 시간

Q & A

<Reference>

- 투빅스 12기 이승현님 강의 자료
- <골빈해커의 3분 딥러닝, 텐서플로맛> (김진중 저/ 한빛미디어)
- <코딩 셰프의 3분 딥러닝, 케라스맛> (김성진 저/ 한빛미디어)
- <펭귄브로의 3분 딥러닝, 파이토치맛> (김건우, 염상준 저/ 한빛미디어)
- https://www.youtube.com/watch?v=TvNd1vNEARw&list=PLQ28Nx3M4Jrguyuwg4xe9d9t2XE639e5C&index=5
- https://www.tensorflow.org/tutorials/quickstart/beginner?hl=ko
- https://engkimbs.tistory.com/673
- https://backgomc.tistory.com/78
- https://tensorflow.blog/케라스-딥러닝/3-2-케라스-소개/
- https://www.slideshare.net/SessionsEvents/soumith-chintala-ai-research-engineer-facebook-at-mlconf-nyc-2017
- https://m.blog.naver.com/PostView.nhn?blogId=os2dr&logNo=221565409684&proxyReferer=https:%2F%2Fwww.google.com%2F
- http://horace.io/pytorch-vs-tensorflow/?fbclid=IwAR0fC_bML92TWMHUiVAWeyrIPYfc4PFwHsGmz6ibfxDZ9zWIT1v3qx7zhCs

들어주셔서 감사합니다.