2022-2 분석 공동세션

PyTorch와 TensorFlow

18기 분석 박규연

CONTENTS

01 PyTorch and TensorFlow

- TensorFlow and PyTorch
- 프레임워크를 사용하는 이유

02 PyTorch vs TensorFlow

- 간단 비교
- 모델 가용성
- 배포 인프라
- 생태계

03 Conclusion

- 그래서 어떤 프레임워크를 사용해야하는가?

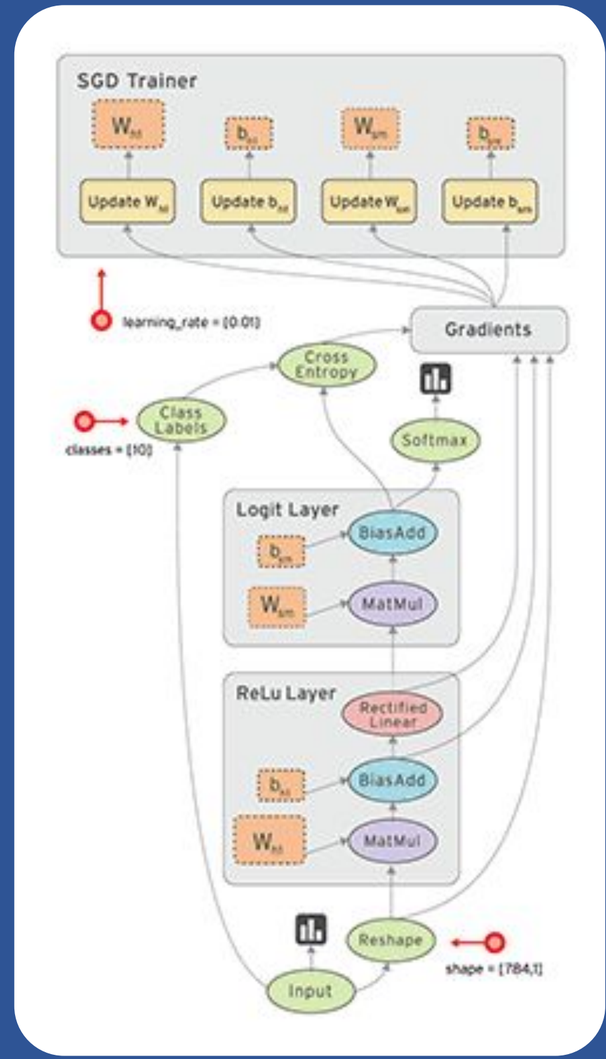
04 참고문헌

TensorFlow

Google이 개발한 오픈소스 머신러닝 라이브러리

Google

170k+ stars



- 데이터 플로우 그래프(Data Flow Graph)를 통한 풍부한 표현력

데이터 플로우 그래프 수학 계산과 데이터의 흐름을 노드(Node)와 엣지(Edge)를 사용한 방향 그래프(Directed Graph)로 표현

노드 수학적 계산 데이터 입/출력, 그리고 데이터의 읽기/저장 등의 작업 수행

엣지 노드들 간 데이터의 입출력 관계를 나타냄. 동적 사이즈의 다차원 데이터 배열(Tensor)를 실어 나름 -> TensorFlow

- 연산 구조와 목표 함수만 정의하면 자동으로 미분 계산(Automatic differentiation)

#Google #CocalCola #Airbnb #Lenovo #Qualcomm #Twitter

PyTorch

FaceBook이 개발한 오픈소스 머신러닝 라이브러리

FaceBook

60k+ stars

- NumPy와 유사하지만 GPU 상에서 실행 가능한 n-차원 텐서 (Tensor)
- 신경망을 구성하고 학습하는 과정에서의 자동 미분(Automatic differentiation)

#NVIDIA #Apple #FaceBook #Stanford

프레임워크를 사용하는 이유

프레임워크 비교 이전에

복잡한 연산 그래프를 쉽게 빌드하기 위해

그래디언트 계산을 쉽게 하기 위해

GPU를 효율적으로 사용하기 위해

Numpy

Computational Graphs

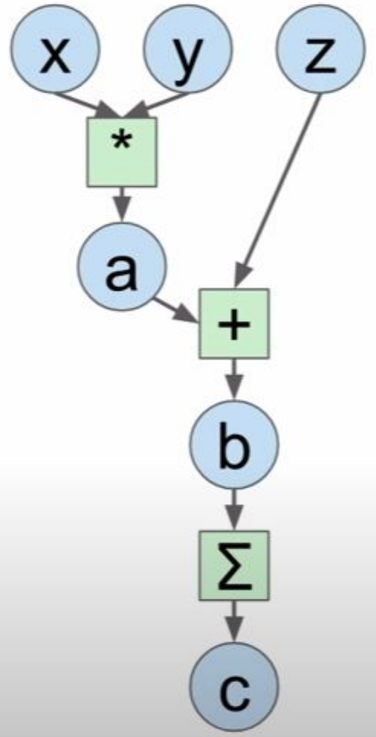
Numpy

```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)
```



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture8.pdf

Numpy

Computational Graphs

Numpy

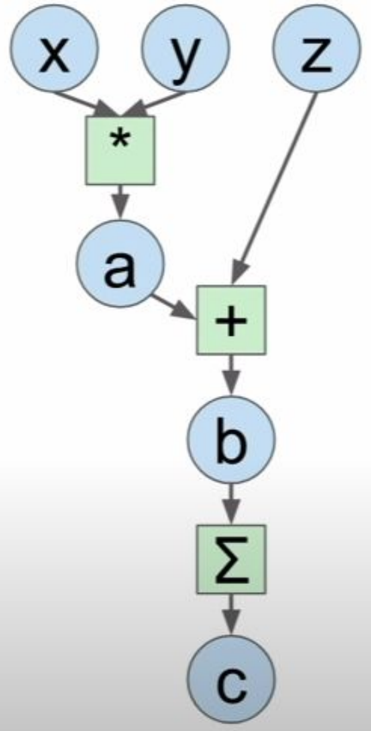
```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
grad_y = grad_a * x
```



Problems:

- Can't run on GPU
- Have to compute our own gradients

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture8.pdf

TensorFlow

Create forward computational graph

```
# Basic computational graph
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3, 4

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
z = tf.placeholder(tf.float32)

a = x * y
b = a + z
c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture8.pdf

TensorFlow

Ask TensorFlow to compute gradients

```
# Basic computational graph
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3, 4

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
z = tf.placeholder(tf.float32)

a = x * y
b = a + z
c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture8.pdf

TensorFlow

Tell TensorFlow to run on **CPU**

Tell TensorFlow to run on **GPU**

```
with tf.device('/gpu:0'):
```

```
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3000, 4000

with tf.device('/cpu:0'):
    x = tf.placeholder(tf.float32)
    y = tf.placeholder(tf.float32)
    z = tf.placeholder(tf.float32)

    a = x * y
    b = a + z
    c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture8.pdf

PyTorch

Create forward computational graph

```
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D),
             requires_grad=True)
y = Variable(torch.randn(N, D),
             requires_grad=True)
z = Variable(torch.randn(N, D),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture8.pdf

PyTorch

```
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D),
             requires_grad=True)
y = Variable(torch.randn(N, D),
             requires_grad=True)
z = Variable(torch.randn(N, D),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```

Calling c.backward() computes all gradients

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture8.pdf

PyTorch

Run on GPU by casting to .cuda()

```
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
y = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
z = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture8.pdf

간단 비교

PyTorch와 TensorFlow

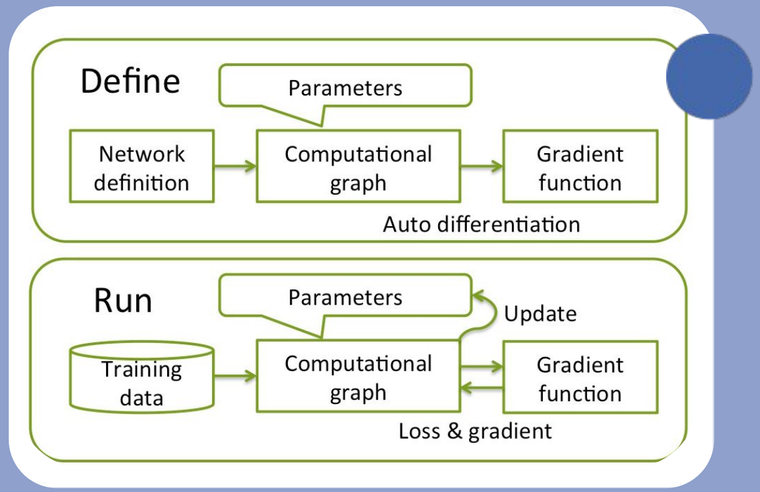
	TensorFlow	PyTorch
그래프	Static	Dynamic
디버깅	Difficult to Debug	Runtime Debugging
시각화 툴	Tensorboard	(Visidom)
사용자 규모	Rich Community	Growing Community
국내 커뮤니티	TensorFlow Korea	PyTorch Korea

Static vs Dynamic

Define and Run vs Define by Run

Define and Run

신경망 정의 이후 실행시 데이터 입력

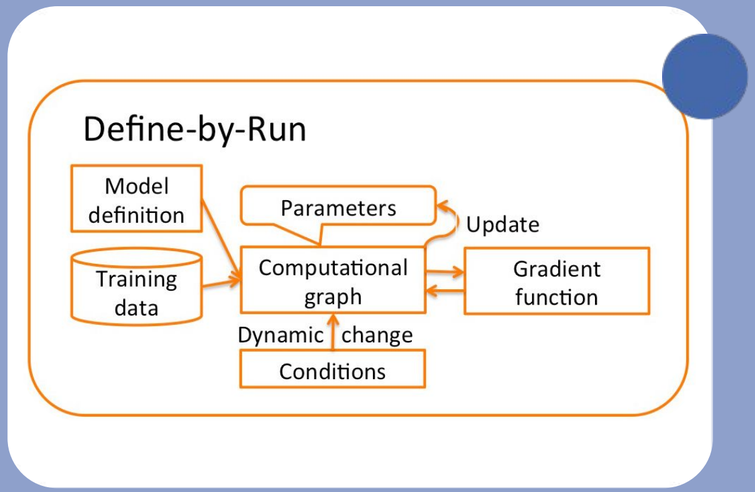



```
x = Variable(xi)
w = Variable(wi)
y = x * w

for xi, wi in data:
    eval(y, (xi, wi))
```

Define by Run

신경망 동적으로 정의




```
for xi, wi in data:
    x = Variable(xi)
    w = Variable(wi)
    y = x * w
```

BOAZ

Static vs Dynamic

TensorFlow: Build graph once, then run many times (**static**)

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.Variable(tf.random_normal((D, H)))
w2 = tf.Variable(tf.random_normal((H, D)))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))
grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])

learning_rate = 1e-5
new_w1 = w1.assign(w1 - learning_rate * grad_w1)
new_w2 = w2.assign(w2 - learning_rate * grad_w2)
updates = tf.group(new_w1, new_w2)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    values = {x: np.random.randn(N, D),
              y: np.random.randn(N, D),}
    losses = []
    for t in range(50):
        loss_val, _ = sess.run([loss, updates],
                               feed_dict=values)
```

Build graph

Run each iteration

PyTorch: Each forward pass defines a new graph (**dynamic**)

```
import torch
from torch.autograd import Variable

N, D_in, H, D_out = 64, 1000, 100, 10
x = Variable(torch.randn(N, D_in), requires_grad=False)
y = Variable(torch.randn(N, D_out), requires_grad=False)
w1 = Variable(torch.randn(D_in, H), requires_grad=True)
w2 = Variable(torch.randn(H, D_out), requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    if w1.grad: w1.grad.data.zero_()
    if w2.grad: w2.grad.data.zero_()
    loss.backward()

    w1.data -= learning_rate * w1.grad.data
    w2.data -= learning_rate * w2.grad.data
```

New graph each iteration

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture8.pdf

고려 사항

라이브러리 선택 시 고려 사항

모델 가용성 (공개된 모델을 이용 가능한지)

배포 인프라 (배포가 용이한지)

생태계 (환경/하드웨어 적합성)

모델 가용성

SOTA 모델의 복잡성이 증가하고 규모가 커짐에 따라
소규모 기업에서는 이를 직접 구현하는 것이
거의 불가능에 가까움.

모델 복잡성 증가

사전 훈련 모델 액세스

컴퓨팅 리소스 부족

- 사전 훈련된 모델에 접근할 수 있는가?
- 공개된 SOTA 모델을 활용할 수 있는가?

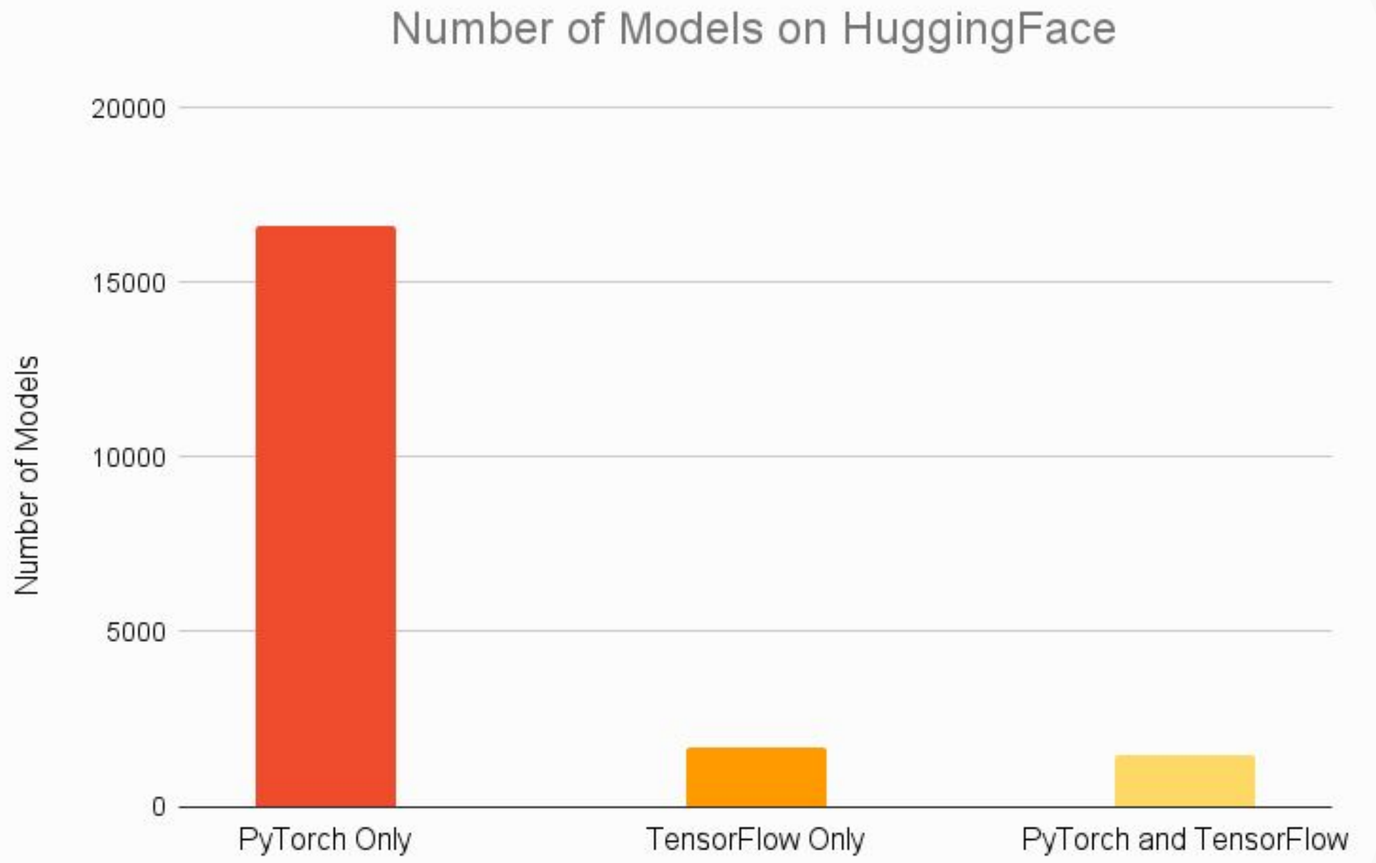
#HuggingFace #연구논문 #PapersWithCode

HuggingFace

https://huggingface.co/

HuggingFace에서 사용 가능한 모델

거의 85%의 모델이 PyTorch **독점적**이며, 독점이 아닌 모델도 PyTorch에서 사용할 수 있는
반면에 모든 모델의 약 16%만 TensorFlow에서 사용할 수 있으며 약 8%만 TensorFlow 전용



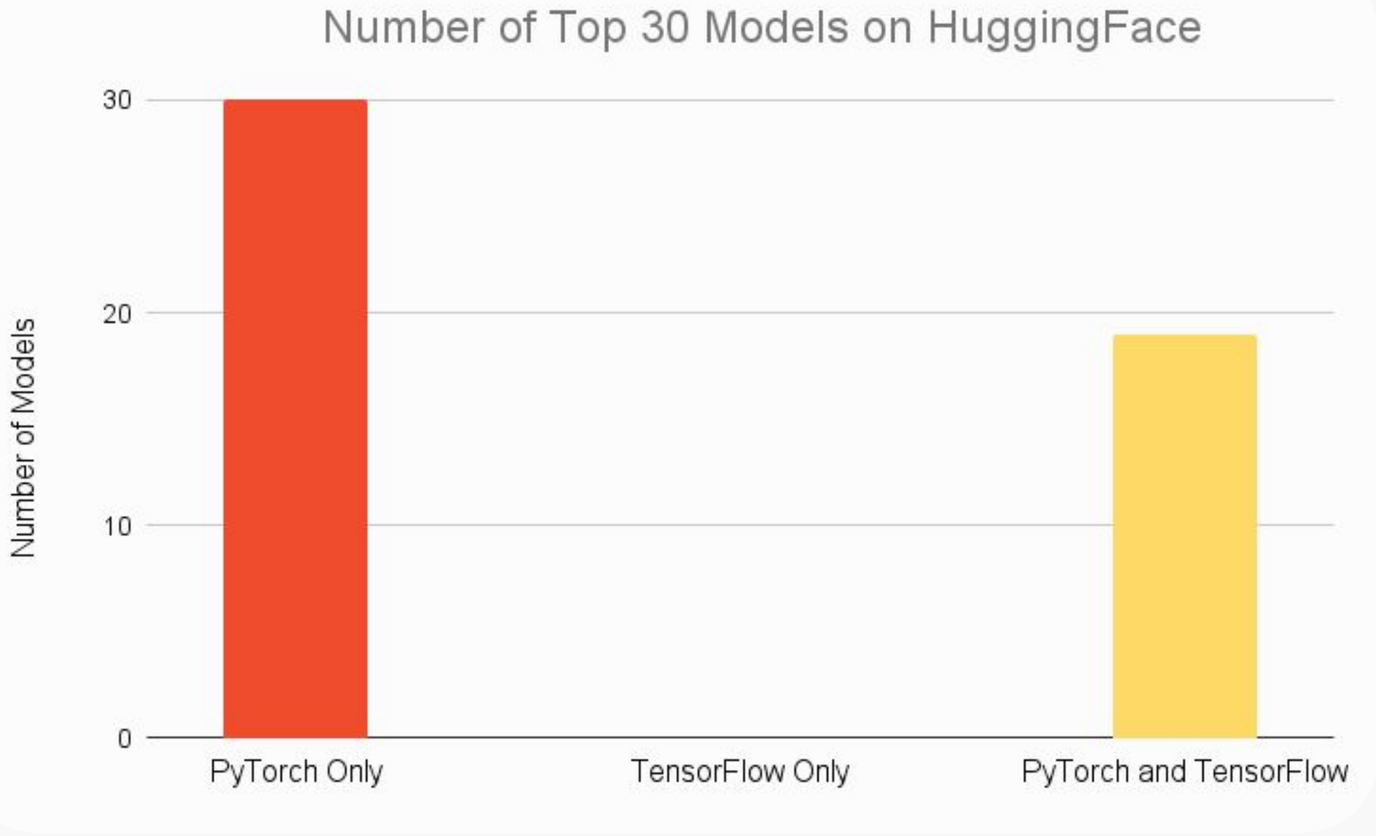
BOAZ

HuggingFace

https://huggingface.co/

HuggingFace에서 가장 인기있는 모델 Top 30

상위 30개 모델 중 2/3에 해당하는 모델들이 PyTorch0
TensorFlow 독점인 상위 30개 모델X.

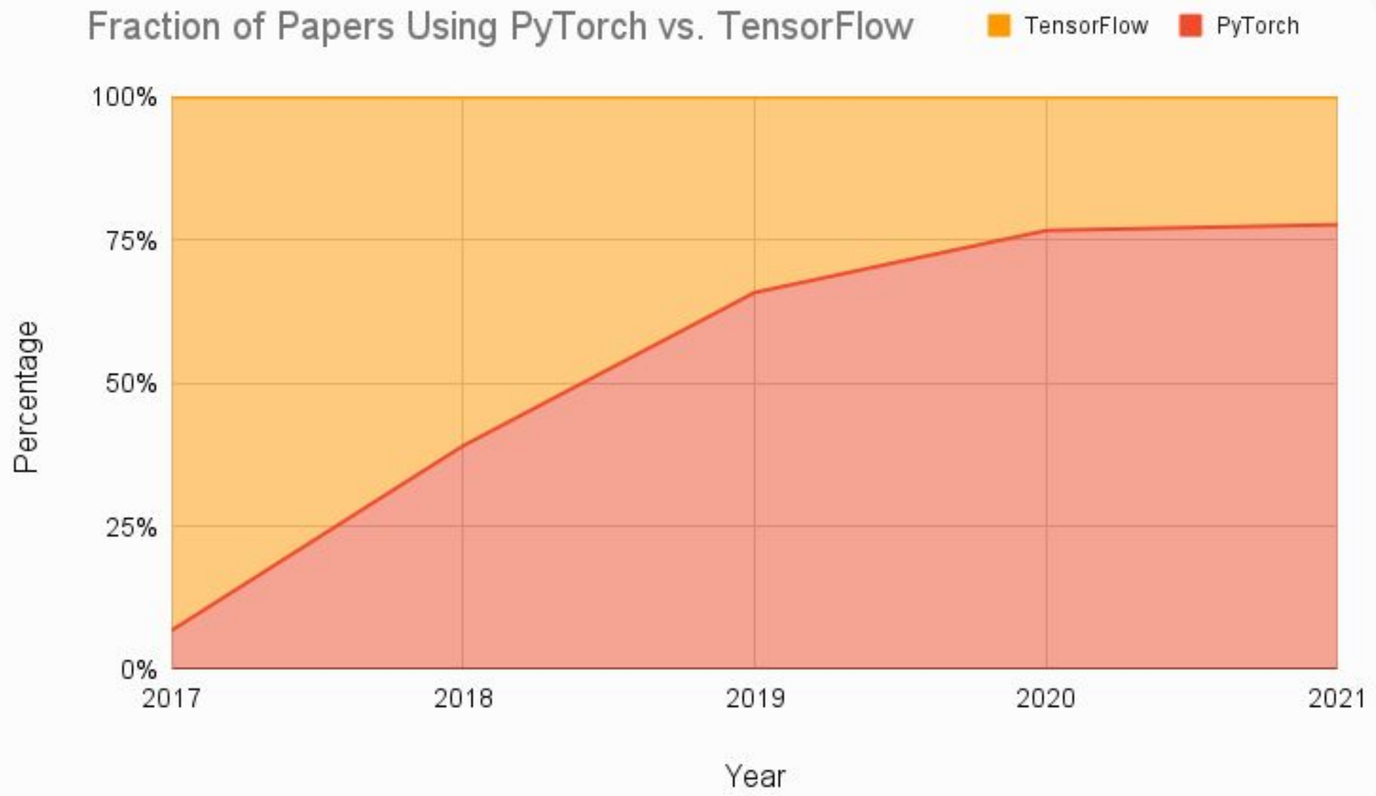


BOAZ

연구 논문 Papers

논문에서의 TensorFlow 및 PyTorch 채택 비율

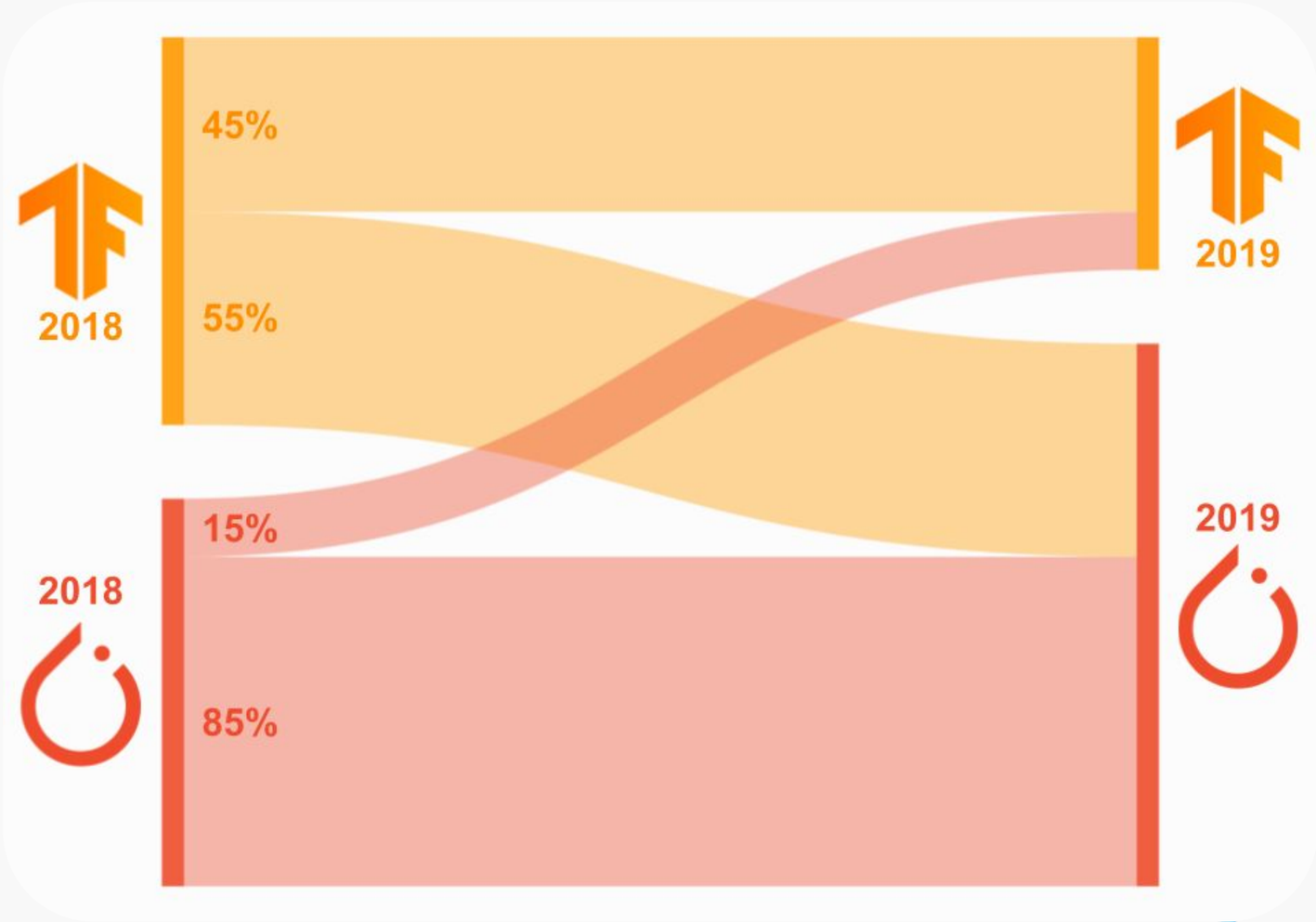
불과 몇 년 만에 PyTorch 또는 TensorFlow를 사용하는 논문에서 PyTorch의 비율이 7%에서 거의 80%까지 성장함.



연구 논문 Papers

PyTorch 채택 비율 증가 배경

TensorFlow1의 어려움 -> PyTorch를 대안으로 사용, 이후 TensorFlow2가 출시 되었지만 PyTorch의 영향력이 이미



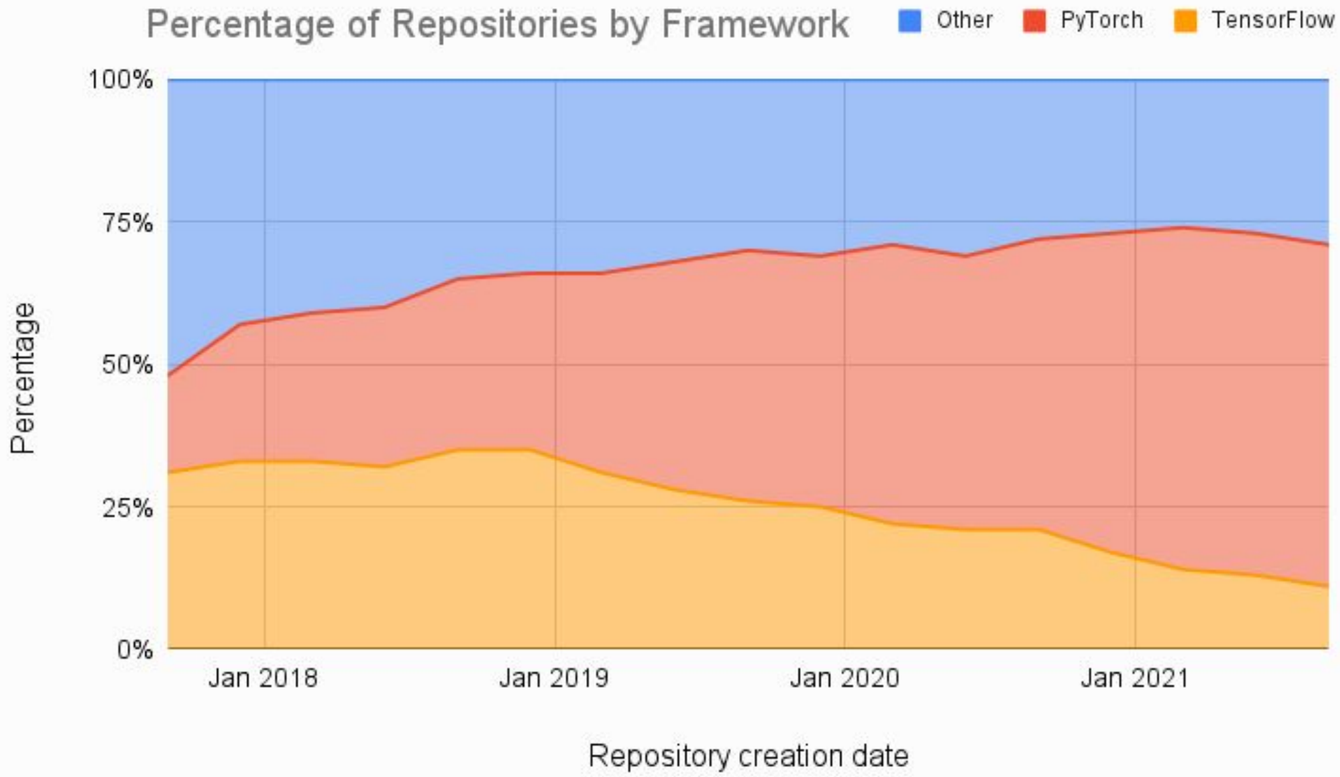
BOAZ

Papers with Code

https://paperswithcode.com/

Papers with Code에서의 프레임워크 채택 비율

PyTorch의 채택 비율이 꾸준히 성장
반면에 TensorFlow는 2019년 TensroFlow1의 단점을 보완한 ver2의 출시에도 불구하고 지속적인 감소를 보임.



BOAZ

배포 인프라

종단간 프로세스

모바일, IoT

클라우드

SOTA 모델에 액세스 하더라도 오류가 발생하기 쉬운
프로세스가 있는 경우 무의미 함.
따라서 SOTA 모델에 액세스 하는 것 외에도
각 프레임워크에서 종단간 딥러닝 프로세스를 고려하는 것이 중요함.

- 종단간 딥러닝 프로세스 고려
- 모바일, 클라우드 등에 적용 가능한가?

#TensorFlowServing #TensorFlowLite #TorchServe #PyTorchLive

TensorFlow

- 종단간 딥러닝 프로세스를 쉽고 효율적으로 만드는 도구들이 많은
- 뿐만 성능에 최적화된 정적 그래프로 확장 가능한 프로덕션 제공

TensorFlow Serving

TensorFlow 모델을 사내 또는 클라우드에 배포할 때 사용

TensorFlow Lite

TensorFlow 모델을 모바일 또는 IoT/임베디드 기기에 배포할 때 사용

장치에 대한 모델을 압축 및 최적화하고 온디바이스 AI에 대한 5가지 제약 조건
(대기 시간, 연결, 개인 정보 보호, 크기, 전력 소비)
을 보다 광범위하게 해결

PyTorch

- 이전에는 PyTorch 사용자가 Flask 또는 Django를 사용하여 모델 위에 REST API를 빌드해야 했음
- 배포 관점에서 극도로 부진했으나 최근 격차를 좁히기 위해 노력 중

Torch Serve

PyTorch 모델을 사내 또는 클라우드에 배포할 때 사용

AWS와 Facebook의 협업으로 탄생한 오픈소스 배포 프레임워크

PyTorch Live

Android, iOS 및 Linux용으로 최적화된 PyTorch 모델 배포를 위한 종단간 워크플로를 생성하도록 설계

생태계

모델링 관점에서 볼 때
PyTorch와 TensorFlow 모두 유능한 프레임워크이므로
조성되어있는 생태계가 프레임워크를 결정하는 요소

- 쉬운 배포, 관리, 분산, 교육 등을 위한 도구를 제공하는 주변 생태계가 잘 조성되어 있는가?

#Hub #Library

TensorFlow

TensorFlow Hub
사전 훈련된 기계 학습 모델의 repository

Model Garden
SOTA 모델의 소스 코드를 사용할 수 있도록 하는 저장소

Extended (TFX)
모델 배포를 위한 종단간 플랫폼

Vertex AI
Google Cloud의 통합 기계 학습 플랫폼

MediaPipe
개체 감지 등에 사용할 수 있는 기계 학습 파이프라인 구축 프레임워크

Coral
로컬 AI로 제품을 빌드하기 위한 툴킷

TensorFlow.js
기계 학습을 위한 JavaScript 라이브러리

PyTorch

PyTorch Hub
사전 훈련된 기계 학습 모델의 repository

Speech Brain
PyTorch 공식 오픈소스 음성 툴킷

TorchElastic
훈련에 영향을 주지 않고 동적으로 변경될 수 있는 컴퓨팅 노드 클러스터에서 모델을 훈련할 수 있도록 작업자 프로세스를 관리하고 재시작 동작을 조정하는 분산 훈련용 도구

TorchX
머신 러닝 애플리케이션의 빠른 구축 및 배포를 위한 SDK

Lightning
PyTorch의 모델 엔지니어링 및 교육 프로세스를 단순화하는데 유용한 도구
PyTorch의 Keras라고도 함

Conclusion

PyTorch 대 TensorFlow의 논쟁은 정답이 없으므로
상황에 맞게 선택해야 함.

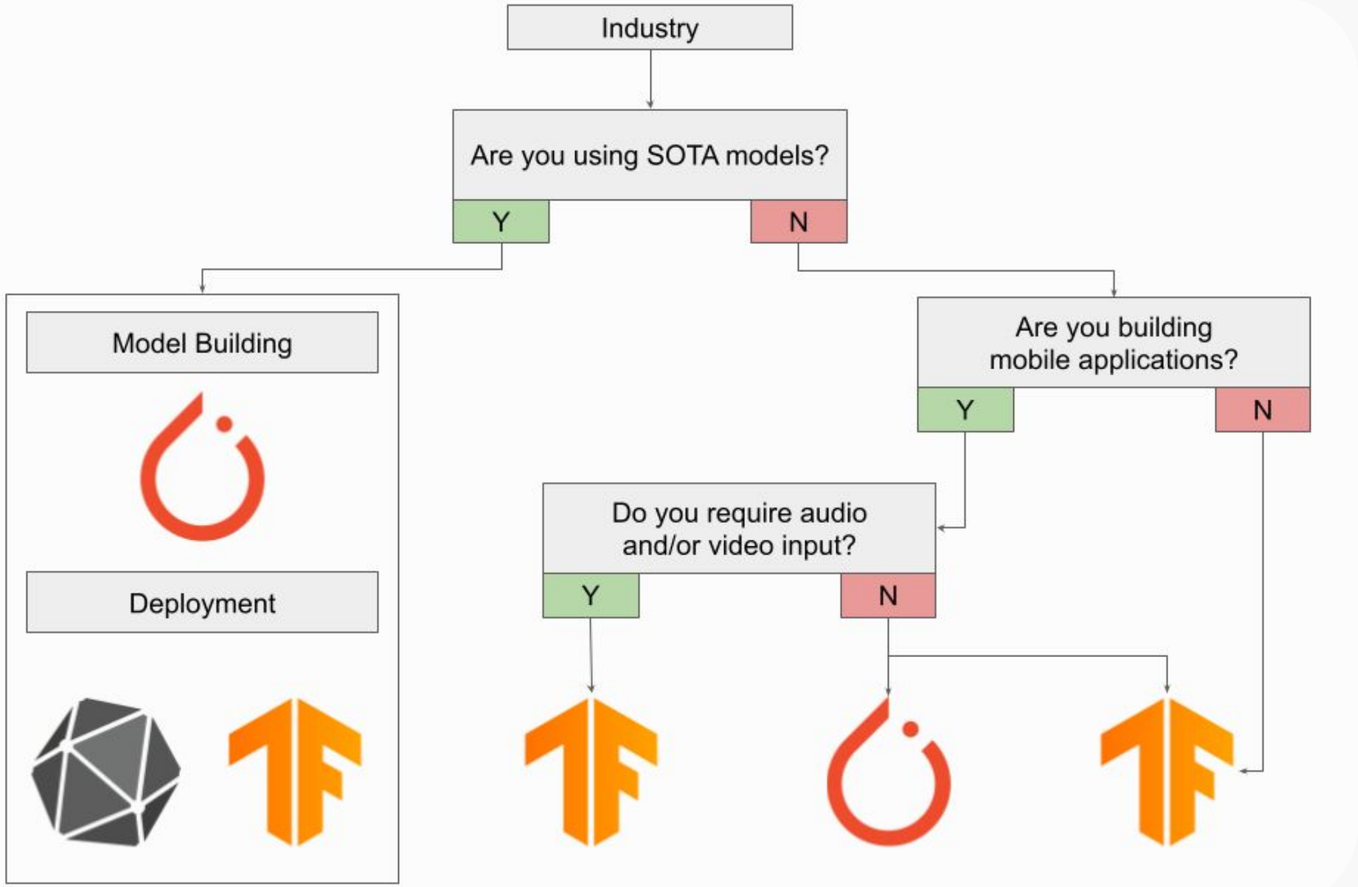
- 그래서 어떤 프레임워크를 사용해야 하는가?

#PyTorch #TensorFlow

산업

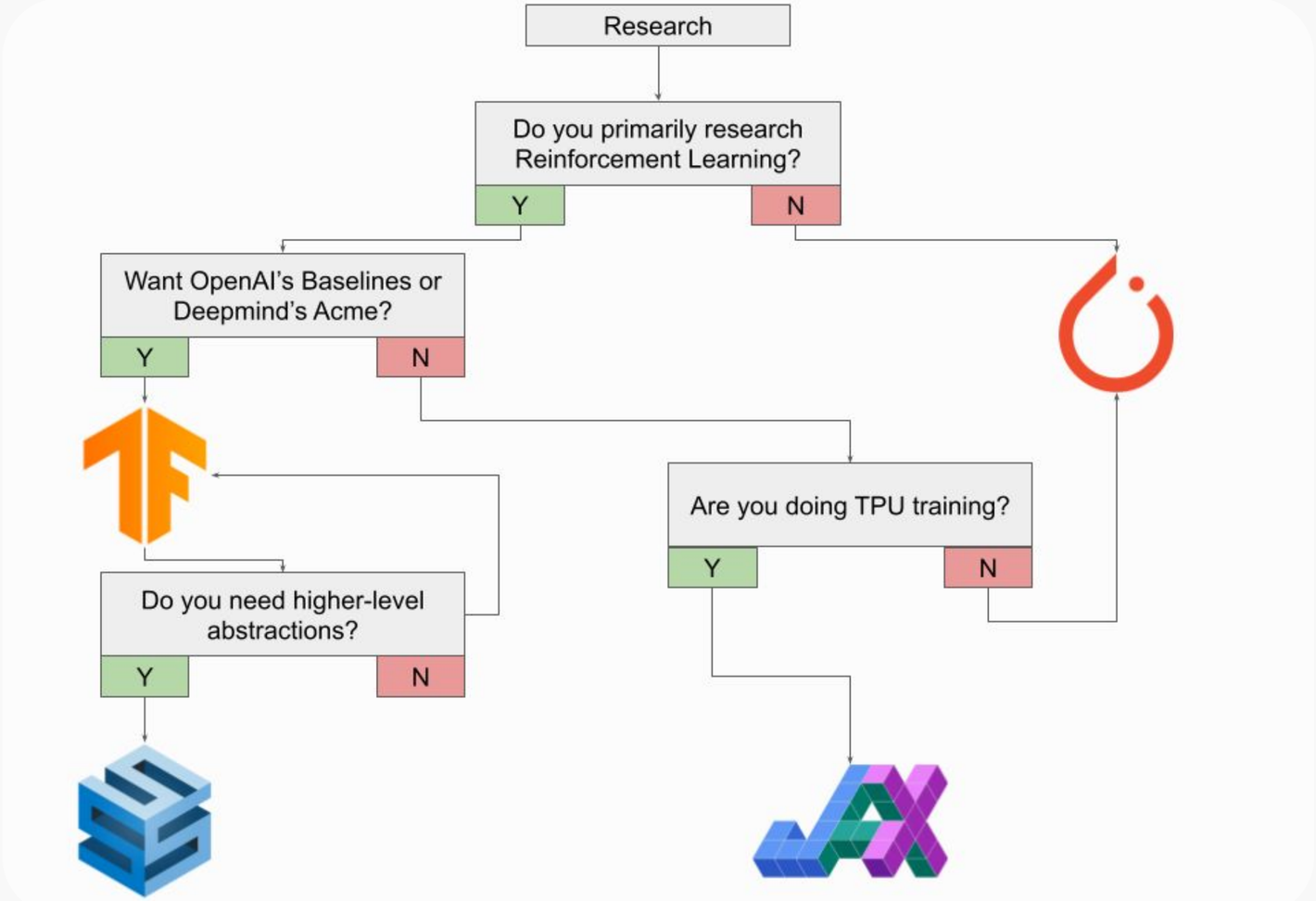
산업에서는 TensorFlow 우세

SOTA 모델 액세스가 필요한 경우, 모델 빌딩은 PyTorch로, 이후 ONNX 변환을 거쳐 TensorFlow의 배포 인프라 활용



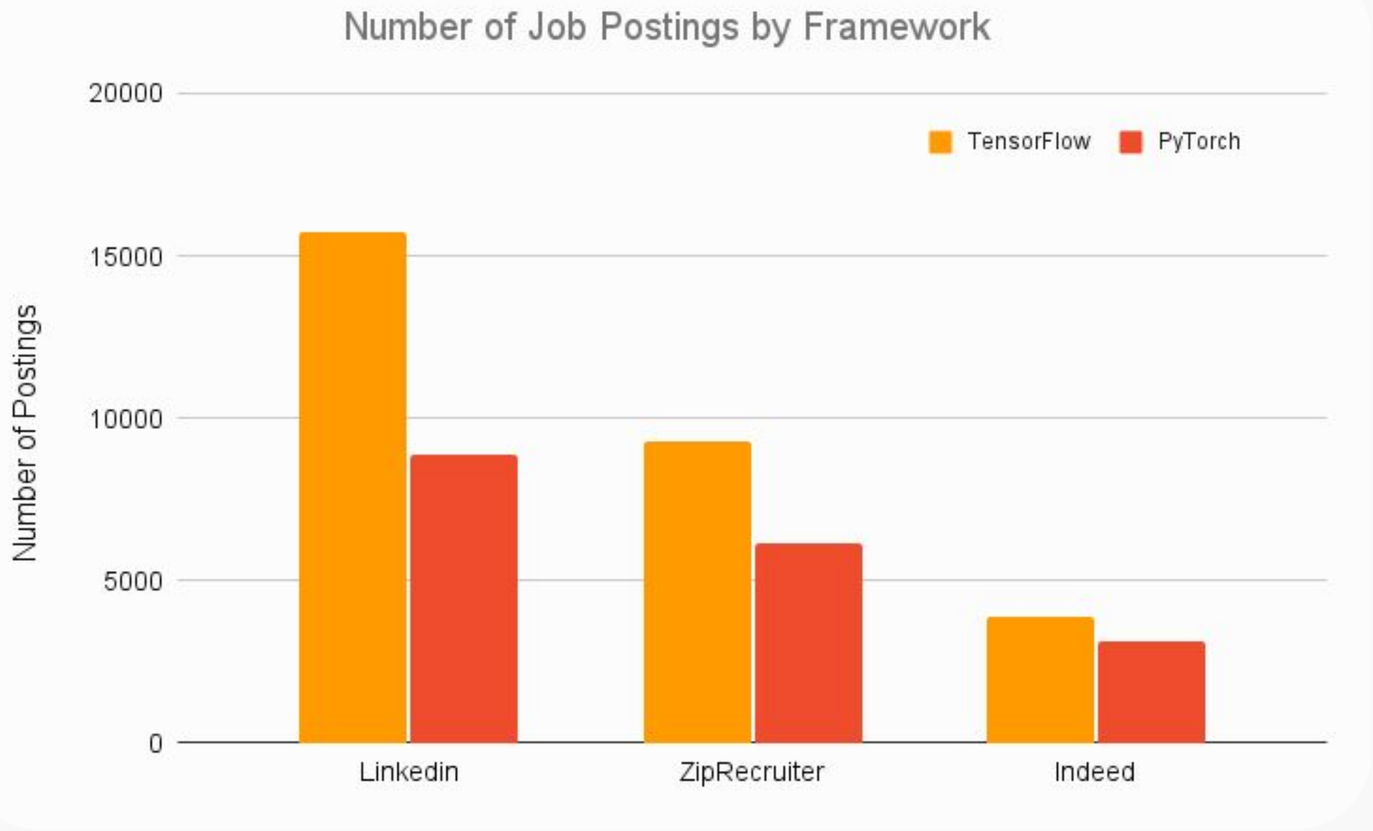
연구원

연구원의 경우 PyTorch 필수



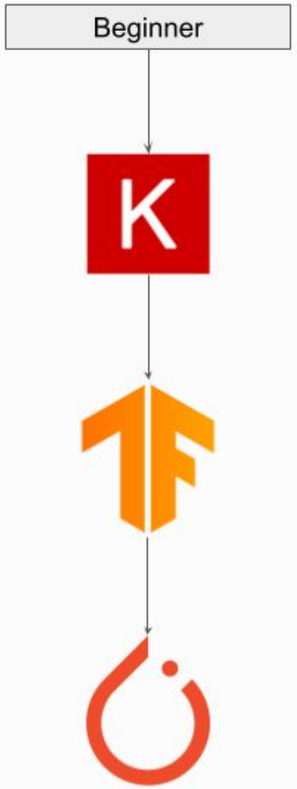
이직/취직

- 둘 다 하는 게 이상적이지만, 채용공고에서는 TensorFlow 우세



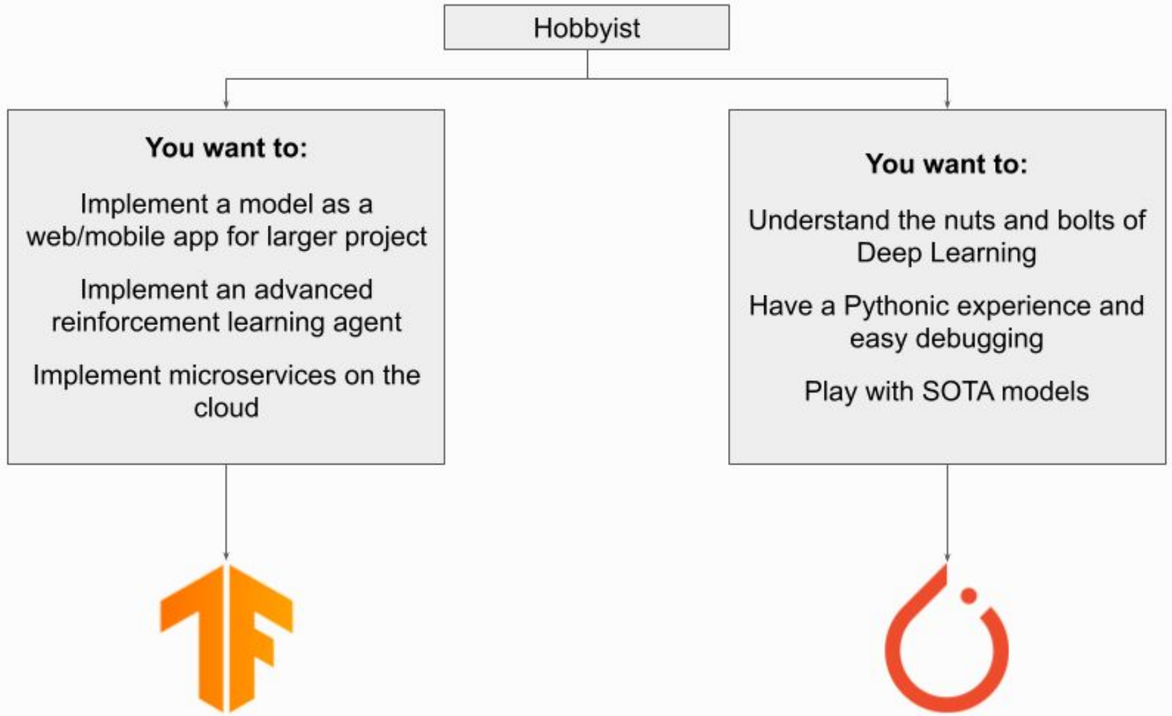
입문

입문자의 경우 Keras



취미

- 딥러닝 모델을 배포하는 것이 목적 -> TensorFlow
- 딥러닝 자체를 배우는 것이 목적 -> PyTorch



참고문헌

논문 및 기타 Document

참고 자료

참고 문헌

1 _____ https://www.youtube.com/watch?v=tNWatDufzDk
2 _____ https://tutorials.pytorch.kr/beginner/pytorch_with_examples.html
3 _____ https://dlabs.ai/resources/whitepapers/10-machine-learning-frameworks-to-try-in-2021
4 _____ https://gist.github.com/haje01/202ac276bace4b25dd3f
5 _____ https://velog.io/@ttogle918/Tensorflow-Pytorch-비교
6 _____ https://m.blog.naver.com/dsz08082/222122736953
7 _____ https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/
8 _____ CS231n

2022-2 분석 공동세션

감사합니다

PyTorch와 TensorFlow

18기 분석 박규연

BOAZ