230707 B팀 주간발표

Heading To the Ground

01

진행상황


SW중심대학 공동 AI 경진대회 2023
SW중심대학 | 알고리즘 | 비전 | 객체분할 | DiceScore
상금 : 총 1700만원
2023.07.03 ~ 2023.07.28 09:59
+ Google Calendar
971명
D-22

진행상황

데이터 확인
완료

Baseline
코드 실행

Data Augmentation

Train

Test

Augmentation을 이용한 데이터 증강이 중요하다고 판단 (1차 crop)

TensorFlow

VS

PyTorch

02

STUDY

Tensorflow.Keras

```
tf.debugging.set_log_device_placement(True)

class MyModel(tf.keras.Model):
    def __init__(self):
        super(MyModel,self).__init__()

        self.flatten = tf.keras.layers.Flatten(input_shape=(28,28))
        self.fc1 = tf.keras.layers.Dense(512,activation='relu')
        self.dropout = tf.keras.layers.Dropout(0.2)
        self.fc2 = tf.keras.layers.Dense(10, activation='softmax')

    def call(self,inputs):
        x = self.flatten(inputs)
        x = self.fc1(x)
        x = self.dropout(x)
        x = self.fc2(x)
        return x

model = MyModel()
```

PyTorch

```
# Get cpu or gpu device for training.
device = "cuda" if torch.cuda.is_available() else "cpu"

# Define model
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

model = NeuralNetwork().to(device)
```

STUDY

Tensorflow.keras

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

PyTorch

```
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)
```

02

STUDY

Tensorflow.keras

```
model.fit(train_images, train_labels, epochs=5)
```

PyTorch

```
def train(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    model.train()
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        # Compute prediction error
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if batch % 100 == 0:
            loss, current = loss.item(), batch * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")

def test(dataloader, model, loss_fn):
    size = len(dataloader.dataset)
    num_batches = len(dataloader)
    model.eval()
    test_loss, correct = 0, 0
    with torch.no_grad():
        for X, y in dataloader:
            X, y = X.to(device), y.to(device)
            pred = model(X)
            test_loss += loss_fn(pred, y).item()
            correct += (pred.argmax(1) == y).type(torch.float).sum().item()
    test_loss /= num_batches
    correct /= size
    print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss: {test_loss:>8f} \n")

epochs = 5
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    test(test_dataloader, model, loss_fn)
print("Done!")
```

STUDY

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

Static graph vs Dynamic graphs

STUDY

텐서플로 2.0의 즉시실행으로 인해 차이가 없어짐!

Static graph vs Dynamic graphs

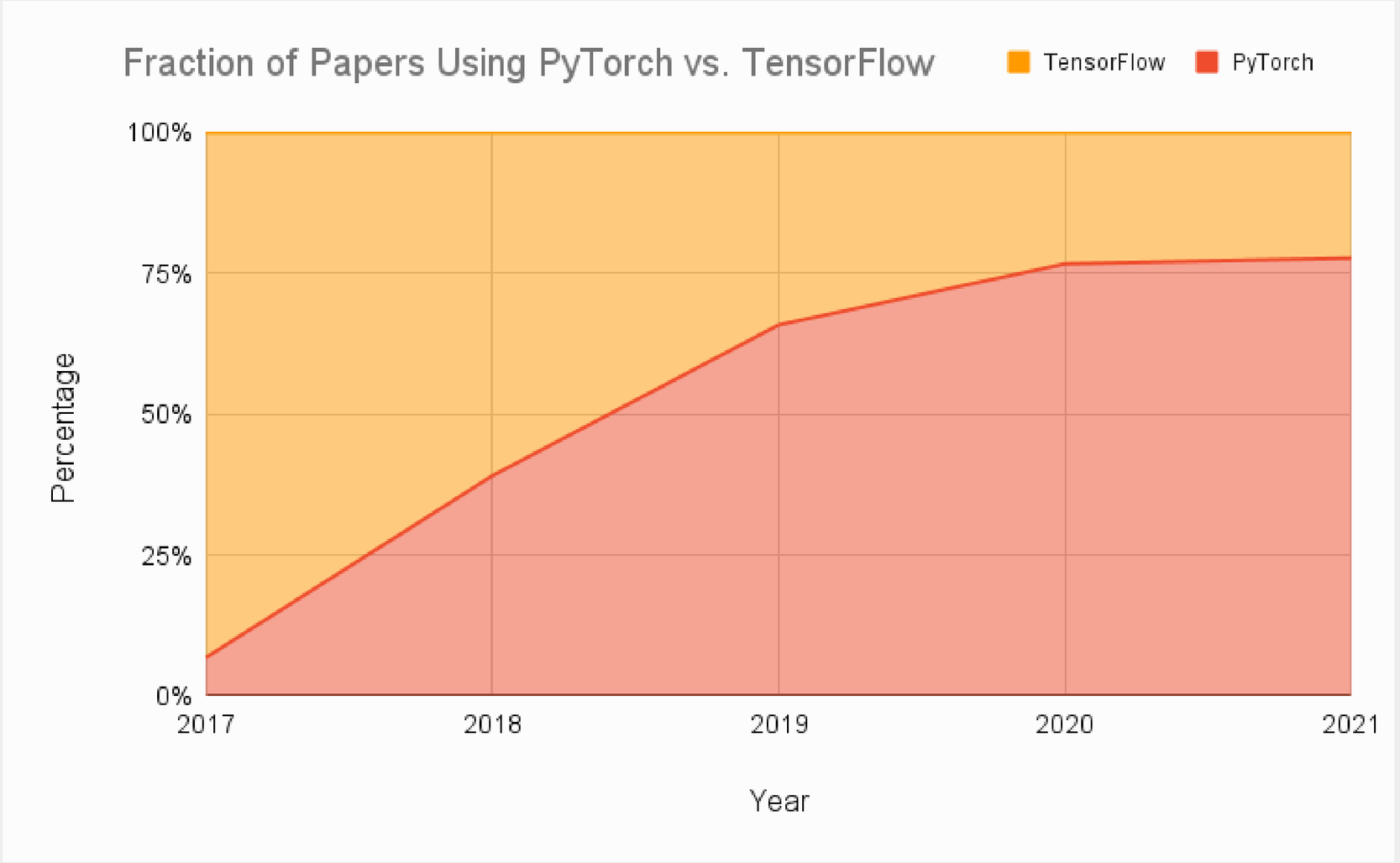
선택을 가르는 건 무엇일까?

(ver 2023)

02

STUDY

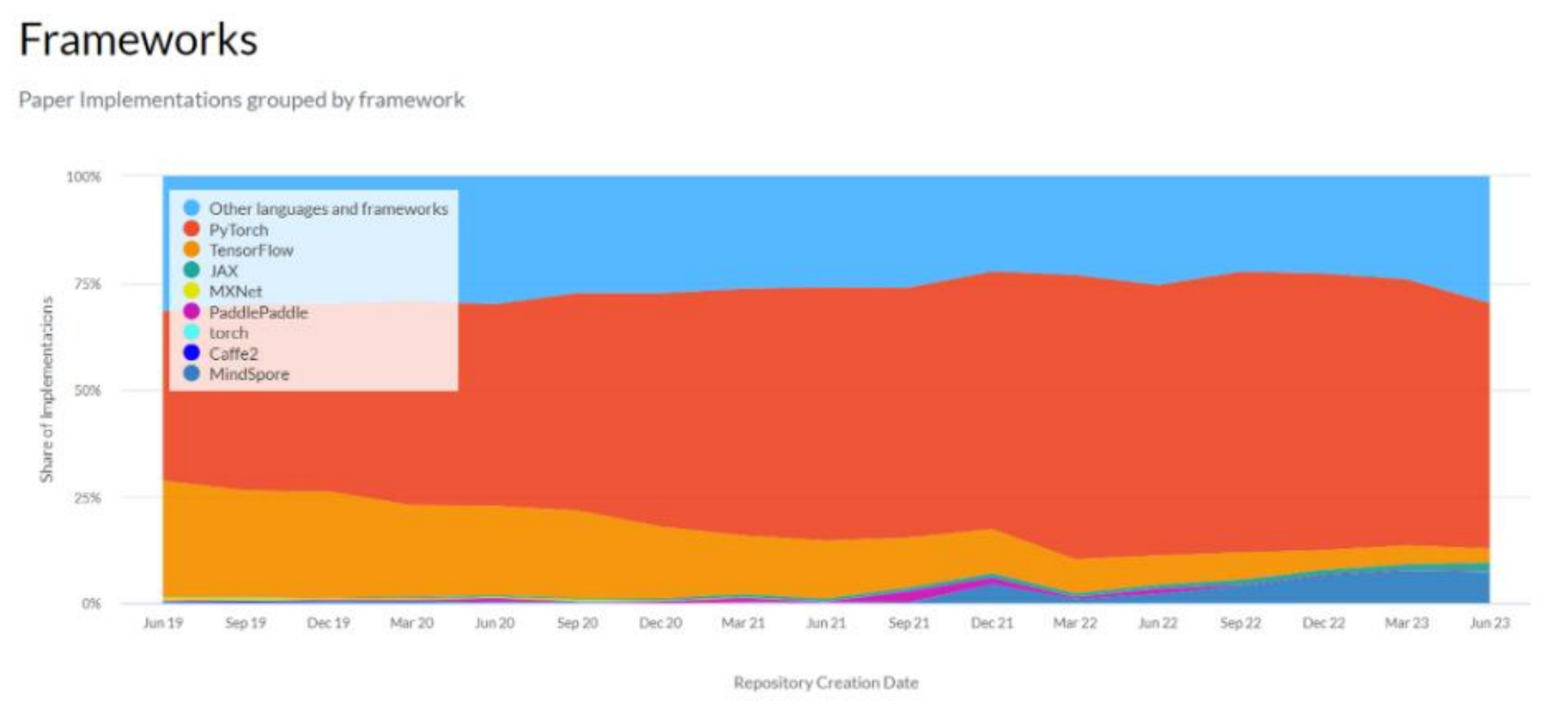
1. 점유율 차이



02

STUDY

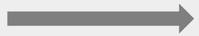
1. 점유율 차이



STUDY

TensorFlow 1

연구에 적합하지
X



PyTorch

대부분의
연구자들이 선택

TensorFlow 2.x

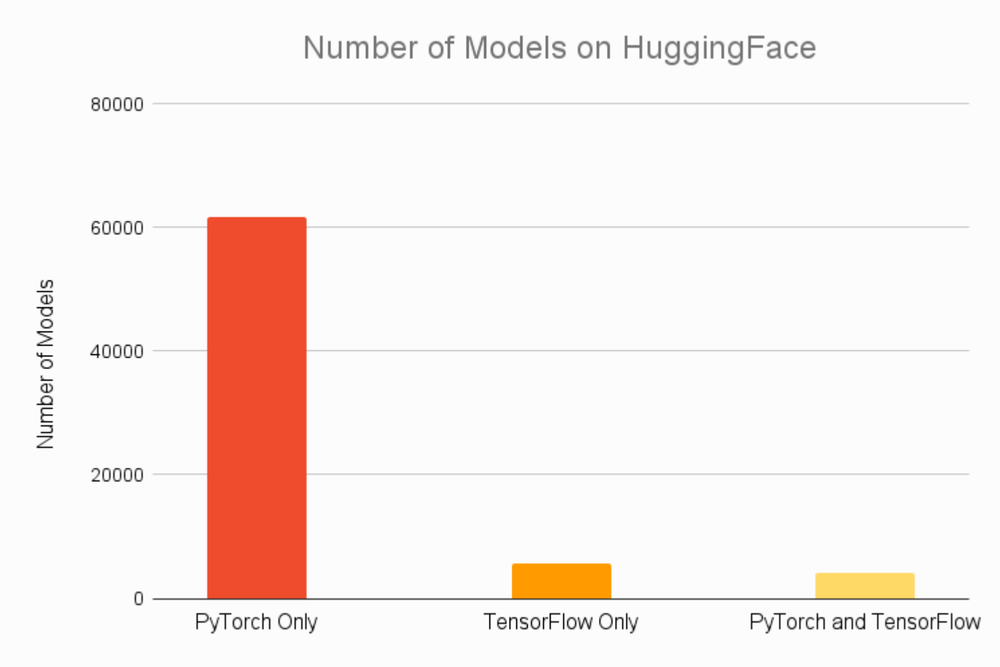
1의 많은 문제
해결!

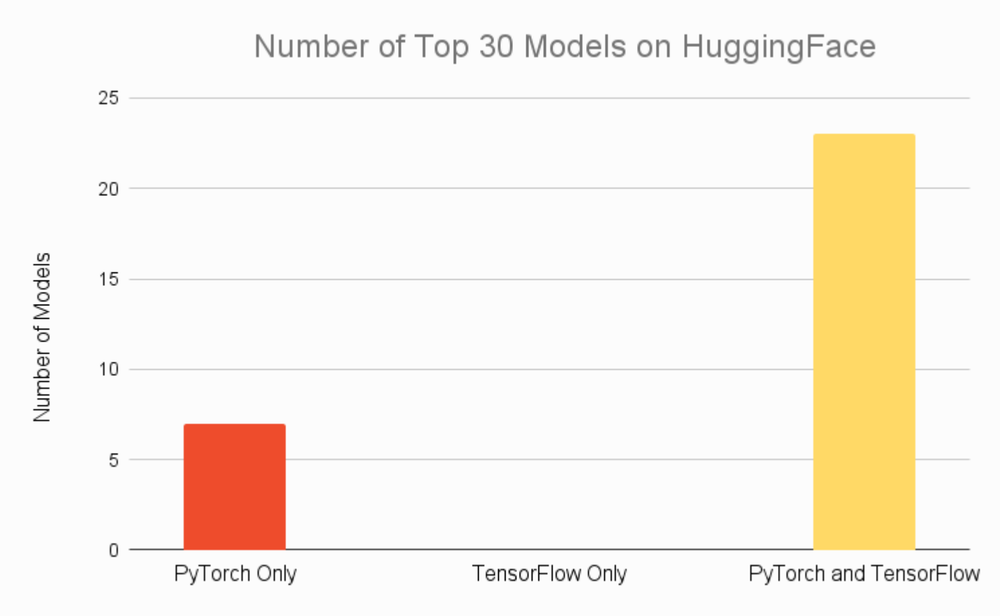
PyTorch

하지만 이미
PyTorch가
점유율에서 강세

STUDY

2. 가용성 차이





Pre-trained 모델에 대한 접근성 역시 Frame-work를 선택하는 주요 요소

3. 배포에서의 유용성 차이

TensorFlow

- TensorFlow serving
- TensorFlow lte

VS

PyTorch

- Torch serve
- PyTorch live

PyTorch는 이전부터 Tensorflow에 비하여 배포 유용성이 떨어진다는 평가가 많음

3. 시각화에서의 유용성 차이

TensorFlow vs PyTorch

그럼 어떤 게 더 좋은가?

본인 프로젝트
방향성에 따라 달라짐!

QnA

참고자료

•https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/
•https://www.projectpro.io/article/pytorch-vs-tensorflow-2021-a-head-to-head-comparison/416
•https://acdongpgm.tistory.com/231
https://github.com/KerasKorea/KEKOxTutorial/blob/master/42_keras_or_pytorch_as_your_first_deep_learning_framework.md