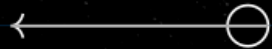


내 자식 하나쯤은 있어야죠

#나만의_실험환경 #베이스라인 #아이고내새끼 #캠퍼님들화이팅

고지형



반갑습니다!



고지형

- 부스트백수캠퍼
- '🔥각공' 윤대혁 캠퍼님 오른팔
- DST, FreshTomato!
- 천천히, 꾸준히 🐢

Contents

- Motivation
- Growth
- Tips

Motivation



Motivation

'실험 환경 구축'
발표를 준비하게 된 이유는 말이죠...

Motivation

때는 바야흐로 U 스테이지 3주차...

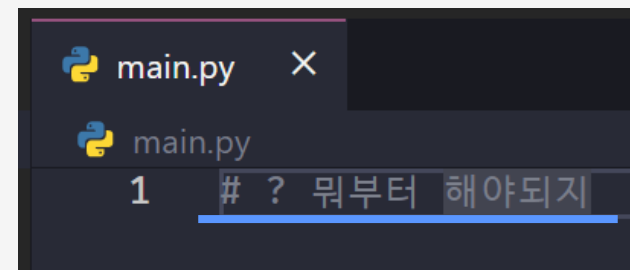
```

36 # MNIST dataset
37 def get_mnist(BATCH_SIZE: int):
38     mnist_train = datasets.MNIST(
39         root="data/", train=True, transform=transforms.ToTensor(), download=True
40     )
41     mnist_test = datasets.MNIST(
42         root="data/", train=False, transform=transforms.ToTensor(), download=True
43     )
44
45     train_iter = torch.utils.data.DataLoader(
46         mnist_train, batch_size=BATCH_SIZE, shuffle=True, num_workers=1
47     )
48     test_iter = torch.utils.data.DataLoader(
49         mnist_test, batch_size=BATCH_SIZE, shuffle=True, num_workers=1
50     )
51
52     return train_iter, test_iter
53
54
55 # Defining Model
56 def get_network(LEARNING_RATE: float, device: str):
57     network = Model().to(device)
58     criterion = nn.CrossEntropyLoss()
59     optimizer = optim.Adam(network.parameters(), lr=LEARNING_RATE)
60
61     return network, criterion, optimizer
62
63
64 # Print Model Info
65 def print_model_info(model: nn.Module):
66     total_params = 0
  
```

U 스테이지 3주차 DLBasic [데이터셋 다루기]

“데이터 클래스 만들고 직접 학습 파이프라인 짜서
구동해보시는거예요. **이거 생각보다 되게 어렵습니다.**”

- 최성철 마스터



Motivation

“실험 환경 구축, 필요할까?”

Q. 실험 환경만들 줄도 모르는데, AI 엔지니어 할 수 있겠어?

A. 다른 사람 코드 참고하면 직접 만들 필요는 없지 않을까

Q. 아무도 시도하지 않은 새로운 도메인에서도 개발할 수 있겠어?

A. 말이 심하잖아

“실험 환경 구축, 어떤 도움이 될까?”

AI 엔지니어 - 'AI를 실현하는 개발자'

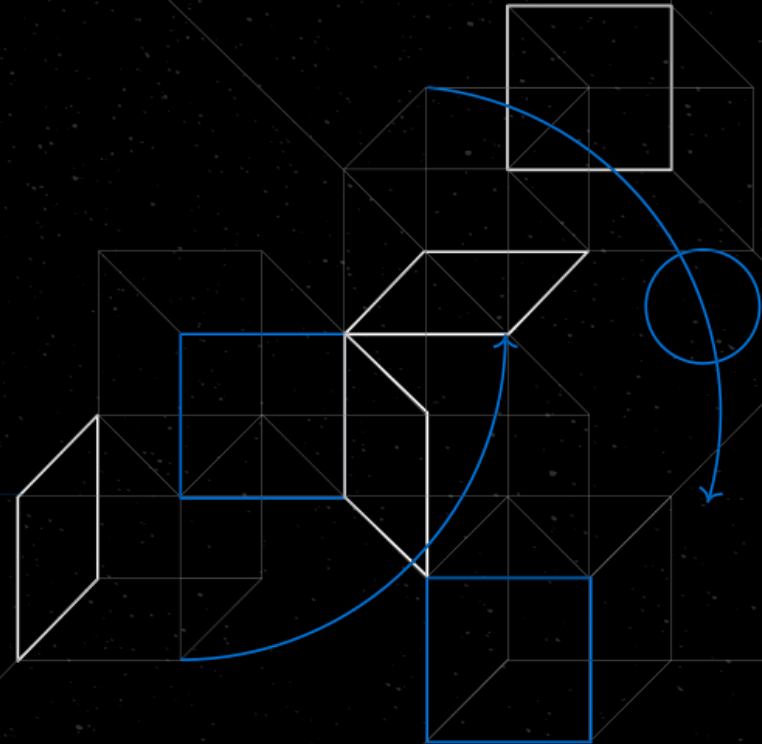
- 주요 역량 (1) AI에 대한 이론적 이해
- 주요 역량 (2) AI 개발 역량

=> 실험 환경을 구축하다 보면 두 역량의 기본기를 한번에 다질 수 있겠다!



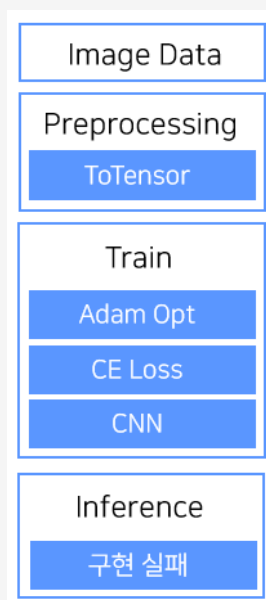
DO!

Growth

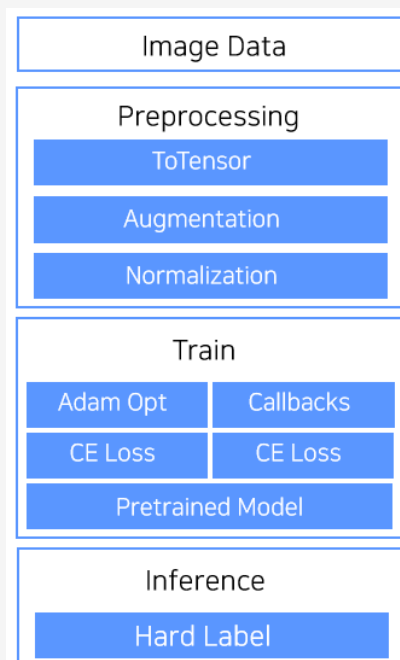


Growth

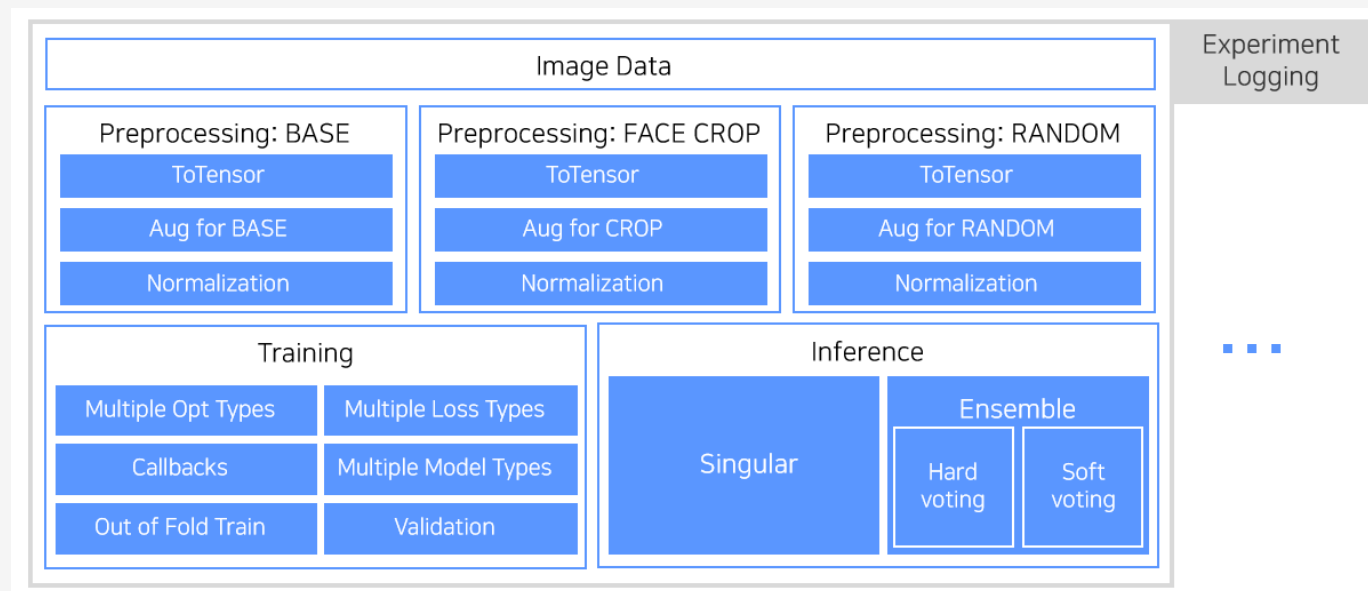
여덟 번의 실험 환경 구축



U스테이지 3주차
[데이터셋 다루기]



[데이콘 CV 경진 대회]



P 스테이지 마스크 상태 분류

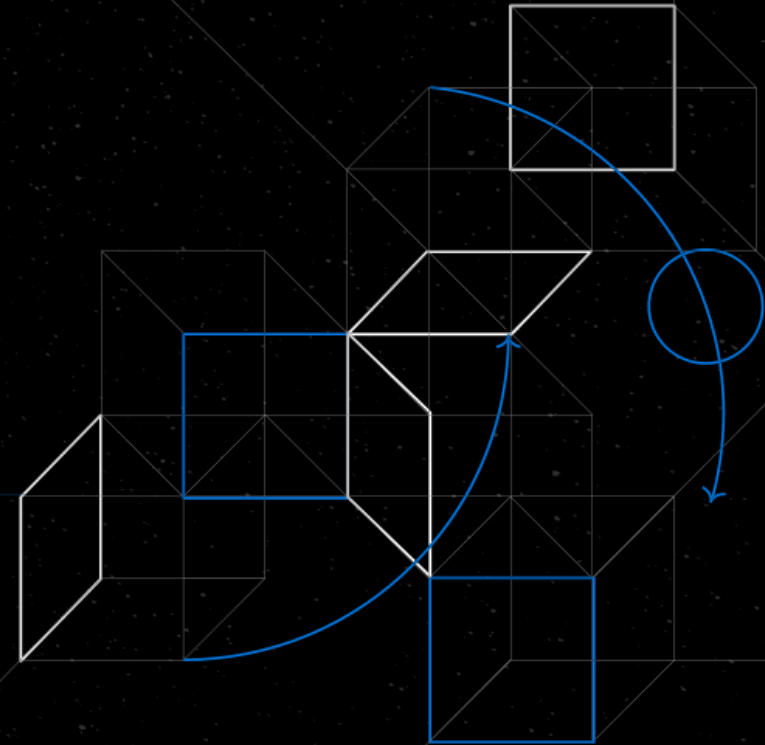
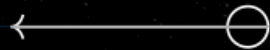
Growth

얻은 것

- 다양한 실험을 염두에 둔 **유연한 실험 환경을 고민/설계**하는 태도
- **타인의 코드를 비판적으로** 바라보는 태도
- 닥친 **문제 해결을 위해 구체적으로 고민**하는 태도
- 조XX 캠퍼님: "실험에 필요한 여러 함수들을 직접 작성하니, **커스터마이징** 하기 좋더라구요."
- 김XX 캠퍼님: "Task에 맞게 실험 환경을 꾸리는 등, **실험 자유도**가 높아졌어요."
- 박XX 캠퍼님: "CPU/GPU 연산을 확인해보면서 **실험 효율성**도 높일 수 있었어요."
- 송XX 캠퍼님: 늦참했는데 상품 개이득이네요;

➡ 한번쯤 도전할 가치가 있습니다 😊

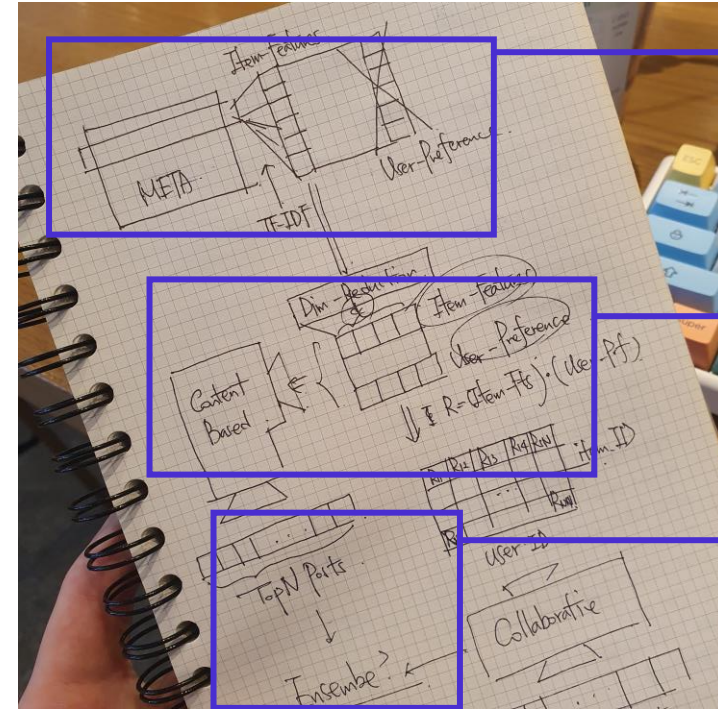
Tips



Tips - 밑그림 그리기

종이는 참 친숙하고 유용한 도구입니다

- 큰 그림을 직접 그려보며 구현 방향을 구체적으로 고민하는 과정
- 부분별 **입출력에 집중**하여 생각해보기
- 어떤 부분에서 **까다로운 작업이 요구될지 예상 가능**
- **시작 단계의 모호함을 바로잡기** 좋음



데이터
처리 방식

모델
구현 방식

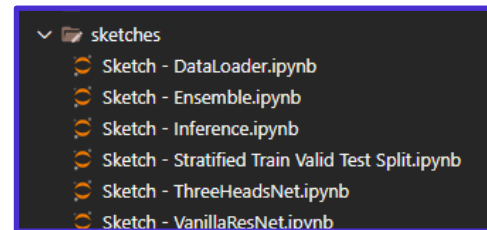
추론
방식

추천시스템 스터디
[컨텐츠 기반 추천 시스템] 스케치

Tips - 밑그림 그리기

주피터 노트북으로 속-삭!

- 주피터 노트북 - 입출력을 즉각 확인할 수 있는 장점
- 처음부터 프로젝트 파일(.py)에 작성하기 어렵다면,
주피터 노트북으로 기능을 우선 구현 후, 관련 모듈에 추가
하면 좀더 수월한 작업 가능
- 해당 기능의 전후 입출력에 집중하여 작성하는 것이 좋음



[P 스테이지 마스크 상태 분류]
스케치를 위해 마련한 작업 폴더

```
# Sketch - TestDataset for Inference
class TestDataset(Dataset):
    def __init__(self, data_root, transform=None):
        self.img_paths = glob(os.path.join(data_root, '*'))
        self.transform = transform

    def __getitem__(self, index):
        name = os.path.basename(self.img_paths[index])
        image = Image.open(self.img_paths[index])
        if self.transform:
            image = self.transform(image)
        return name, image

    def __len__(self):
        return len(self.img_paths)

> ML

# 작동 확인
data = pd.read_csv('../prediction/vanillaresnet_epoch09_transformbase_
transform = configure_transform('train', 'base')
dataset = TestDataset(data_root='../input/data/eval/images', transform=
```

[P 스테이지 마스크 상태 분류]
추론을 위한 데이터셋 스케치 노트북

Tips - 디버깅

주피터 노트북을 활용한 기능 간 흐름 확인

- 작성한 커스텀 모듈을 주피터 노트북에 직접 불러와 기능 간 흐름 확인

```
sys.path.insert(0, '커스텀모듈 디렉토리')
```

- 확인할 **상황을 동적으로 구성**할 수 있어 간편
- 예상하지 못한 문제 발견 가능

입출력이 잘못됐네

메모리 과부하

데이터 처리 속도가 너무 느린데?

```
import sys
sys.path.insert(0, '[자신이 작성한 모듈의 디렉토리 경로]')
# sys.path.append('[자신이 작성한 모듈의 디렉토리 경로]')

# 커스텀 모듈 불러와 테스트
import custom_module import CustomClass
from dataset import TestDataset
from inference import predict
from models import VanillaResNet

# 진행해볼 상황을 구성
data = pd.read_csv('../prediction/vanillaresnet_epoch09')
transform = configure_transform('train', 'base')

model = VanillaResNet(num_classes=18)
predict(model, dataset, load_state_dict='../saved_model')
dataset = TestDataset(data_root='../input/data/eval/ima

# 각 phase 별 자유로운 확인
print(dataset[0])
```

[P 스테이지 마스크 상태 분류]
TestDataset이 추론 과정에서 잘 작동하는지 확인

Tips - 디버깅

디버깅 툴을 활용한 기능 간 흐름 확인

- 작성한 기능의 주요 상황을 작성한 뒤,
- 디버깅 툴을 통해 해당 상황을 실현
- 각 함수 내부 instance까지 확인 가능
- 예상하지 못한 문제 발견 가능
- 레퍼런스 코드 이해에도 큰 도움

```
28
29 | def load_data(path: str, drop_id: bool = True, encode_label: bool = True) -> T:
30 |     data = pd.read_csv(path, sep="\t", header=None, names=COLUMNS)
31
32 |     # test data have no labels
33 |     if path == Config.Test:
34 |         data.drop("label", axis=1, inplace=True)
35
36 |     # drop 'id' column
37 |     if drop_id:
38 |         data.drop("id", axis=1, inplace=True)
39
40 |     # encode label from string to integer
41 |     if encode_label and path != Config.Test:
42 |         enc = LabelEncoder()
43 |         data["label"] = data["label"].apply(lambda x: enc.transform(x))
44
45 |     dataset = data.drop('label', axis=1)
46 |     labels = data['label'].tolist()
47
48 |     return dataset, labels
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

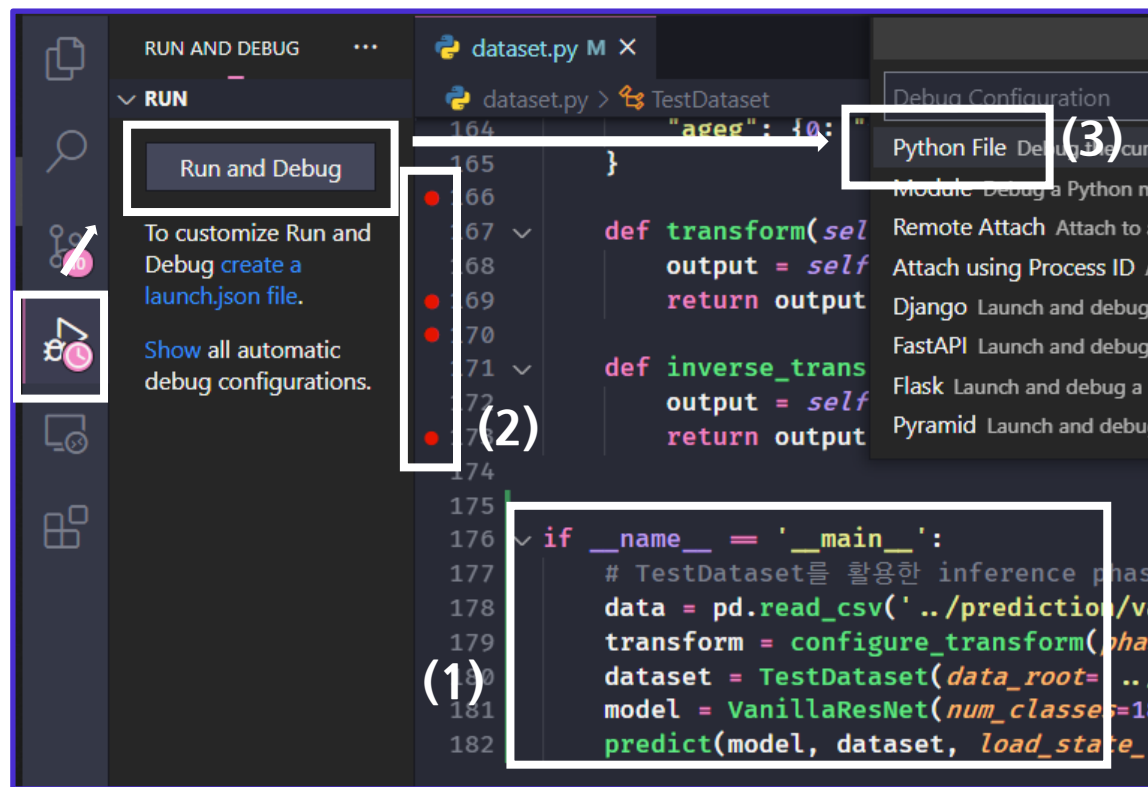
```
0 wikipedia-24896-25-30-33-19-21 중국에서 사용되는 스포츠 유틸리티 자동차의 브랜드로
> 1 wikipedia-12728-224-5-7-42-44 선거에서 민주당은 해산 전 의석인 230석에 한참 못
2 wikipedia-28460-3-0-7-9-12 유럽 축구 연맹(UEFA) 집행위원회는 2014년 1월 24일
3 wikipedia-11479-37-24-26-3-5 용병 공격수 차디의 부진과 시즌 초 활약한 강수일의
4 wikipedia-15581-6-0-2-32-40 람캄행 왕은 1237년에서 1247년 사이 수코타이의 왕
-> data.head()
relation_state e1 e1_start e1_end
0 영국에서 사용되는 스포츠 유틸리티 자동차의 브랜드로는 랜드로버(Land Rover)...
> 1 선거에서 민주당은 해산 전 의석인 230석에 한참 못 미치는 57석(지역구 27석,...
2 유럽 축구 연맹(UEFA) 집행위원회는 2014년 1월 24일에 열린 회의를 통해 ... 유럽 축구
3 용병 공격수 차디의 부진과 시즌 초 활약한 강수일의 침체, 시즌 중반에 영입한 세르...
4 람캄행 왕은 1237년에서 1247년 사이 수코타이의 왕 퍼곤 씨 인트라릿과 쓰엿 ...
```

[P 스테이지 문장 내 개체 간 관계 추출]
작성한 'load_data' 함수가 잘 작동하는지 확인

Tips - 디버깅

디버깅 툴을 활용한
기능 간 흐름 확인

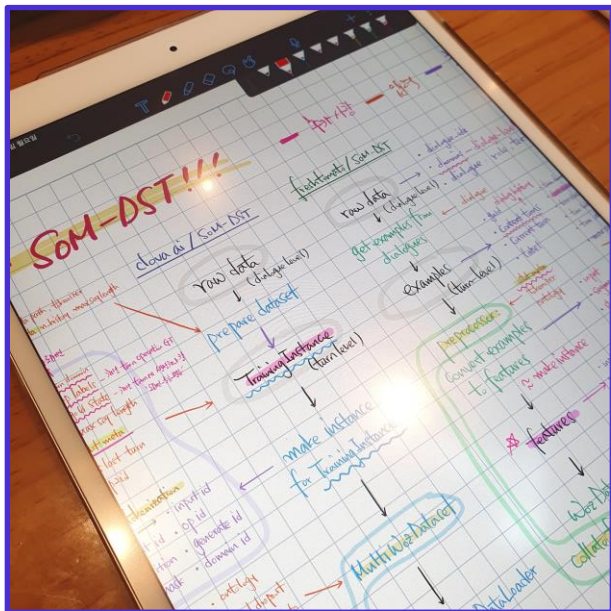
- (1) 확인이 필요한 가상의 상황 구성
- (2) 상황 진행 과정에서, 확인할 부분 체크
- (3) 디버깅 실행



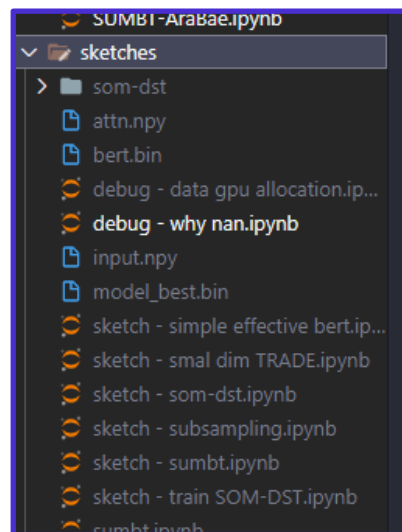
[P 스테이지 마스크 상태 분류]
TestDataset이 추론 과정에서 잘 작동하는지 확인

Tips - 지금도..

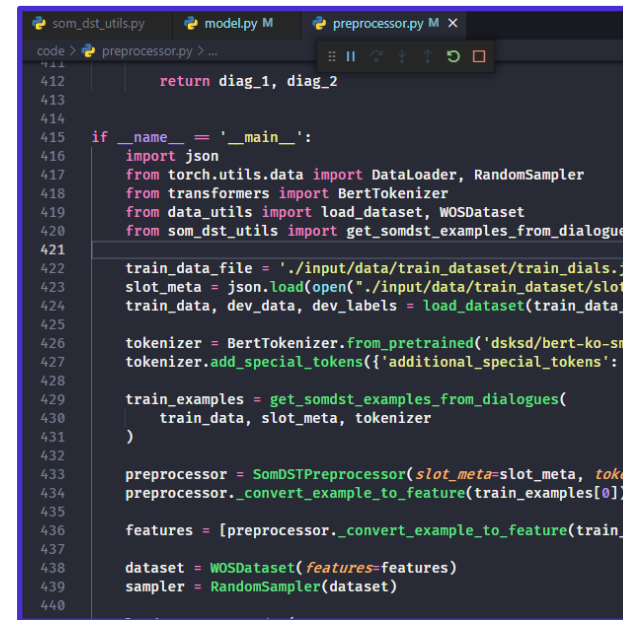
P 스테이지 3 - DST



스케치



적용 및 구현



디버깅

Tips – 태도

이런 태도가 도움됐어요

내 코드에 고립되지 않기

‘내 손으로 처음부터 끝까지 다 해보자!’

- 실력 향상에 도움되나, Task가 복잡해질 수록 자칫 비효율적일 수 있음

자신의 코드를 객관적으로 바라보고, 필요한 레퍼런스 코드는 적극 이해/수용

- 단, 레퍼런스 코드‘만’ 찾아다니는 건 별로였어요

‘작동한다’에만 머물지 않기

실험 환경 조성의 근본적 목표 - ‘AI 엔지니어로서 기본기 다지기’

비효율적인 부분을 개선하거나 기능을 섬세하게 추가한 경험은 기본기 향상으로 연결

Tips – 기타

PyTorch Lightning을 활용한 효율적인 실험 환경 조성

우종빈 (미라클모닝 멤버, 친절왕)

아까 말씀드리긴 했는데 pytorch lightning 한번 보셔도 좋을거 같아요! (아직 써본지 일주일도 안 됐지만) 베이스라인짤때 고려해야되는 추가적인 부분을 엄청 쉽게 처리해줘서 편하더라고요

오후 4:22

- Mixed precision training, logging(W&B, ...) 등 실험 환경에 필요한 추가적인 요소들을 쉽게 처리할 수 있음

```
# models
encoder = nn.Sequential(nn.Linear(28 * 28, 64), nn.ReLU(), nn.Linear(64, 3))
decoder = nn.Sequential(nn.Linear(28 * 28, 64), nn.Linear(3, 64), nn.ReLU(), nn.Linear(64, 28 * 28))

encoder.cuda(0)
decoder.cuda(0)

# download on rank 0 only
if global_rank == 0:
    mnist_train = MNIST(os.getcwd(), train=True, download=True)

# download on rank 0 only
transform=transforms.Compose([transforms.ToTensor(), transforms.Normalize(0.5, 0.5)])
mnist_train = MNIST(os.getcwd(), train=True, download=True, transform=transform)

# train (55,000 images), val split (5,000 images)
mnist_train, mnist_val = random_split(mnist_train, [55000, 5000])

# The dataloaders handle shuffling, batching, etc...
mnist_train = DataLoader(mnist_train, batch_size=64)
mnist_val = DataLoader(mnist_val, batch_size=64)

# optimizer
params = [encoder.parameters(), decoder.parameters()]
optimizer = torch.optim.Adam(params, lr=1e-3)

# TRAIN LOOP
model.train()
num_epochs = 1
for epoch in range(num_epochs):
    for train_batch in mnist_train:
        x, y = train_batch
        x = x.cuda(0)
        x = x.view(x.size(0), -1)
        z = encoder(x)
        x_hat = decoder(z)
        loss = F.mse_loss(x_hat, x)
        print('train loss: ', loss.item())

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

# EVAL LOOP
model.eval()
with torch.no_grad():
    val_loss = []
```

PyTorch Lightning으로 간소화

```
# Train
model = LitAutoEncoder()
trainer = pl.Trainer()
trainer.fit(model, mnist_train, mnist_val)
```

Q&A

감사합니다