MS AI School Project

: 이미지를 통한 간편 번역 프로그램

Team 5

이주형 이승현 이성규 민안세 김민정

CONTENTS

CHAPTER

1. 프로젝트 구조

CHAPTER

1. 프로젝트 구조

2개의 과정
Text detection
Text recognition

Text가 어디에 위치하는지 파악하고
그 위치의 이미지에서 텍스트화

CHAPTER

1. 프로젝트 구조

나온 텍스트 기반으로 번역

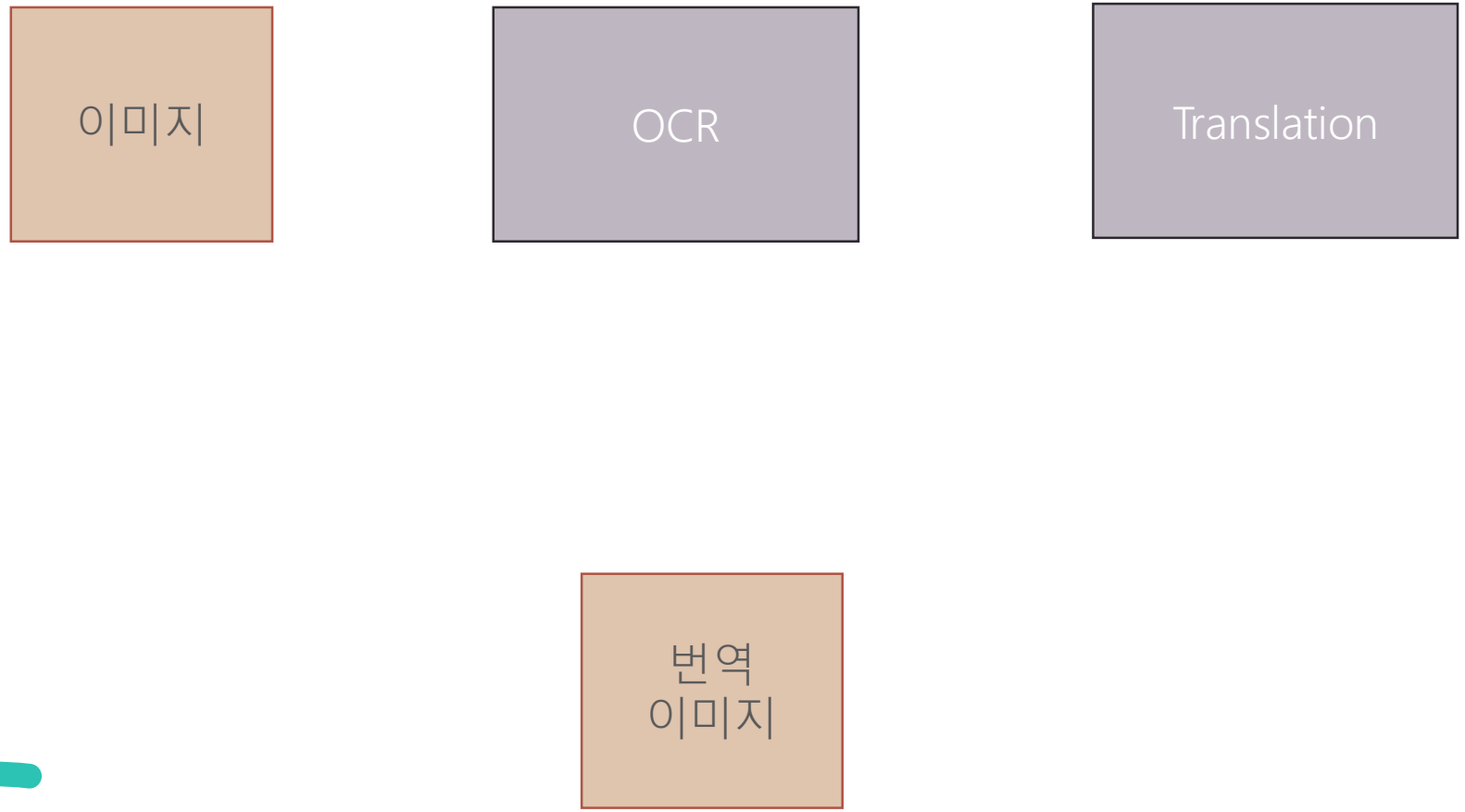
CHAPTER

1. 프로젝트 구조

번역된 텍스트를 이미지상에 적용

CHAPTER

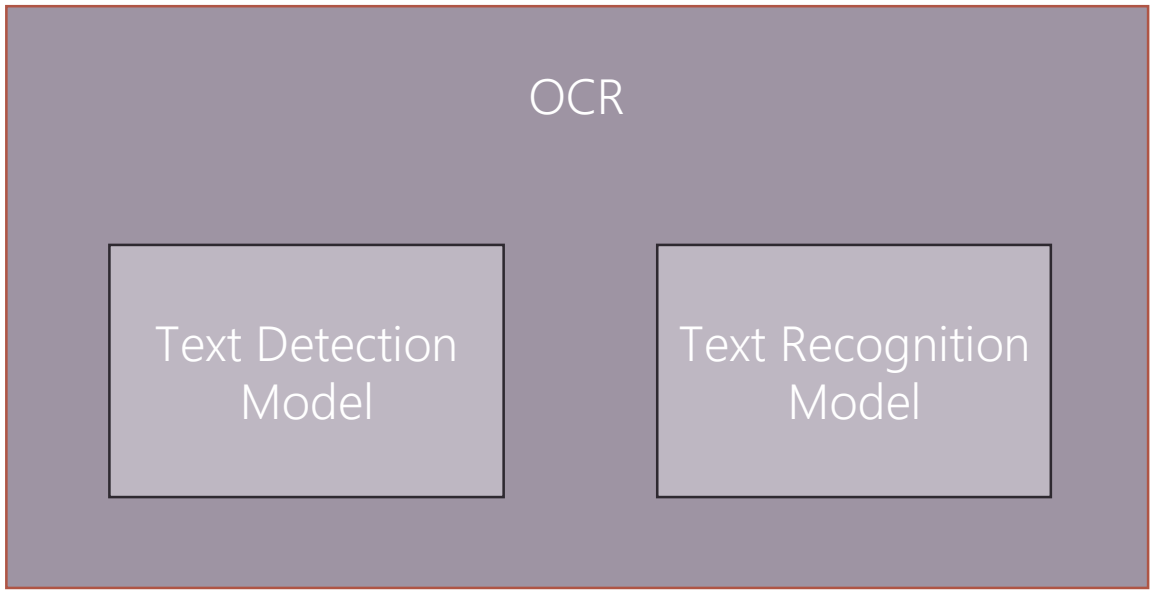
1. 프로젝트 구조



CHAPTER

1. 프로젝트 상세 정보 및 설명

OCR 모델은 사실상 2개의 모델이 합쳐진 구조.



위에서도 언급하였듯 OCR은 보통 **Text detection -> Text recognition** 이렇게 두 개의 모델을 거친 과정이 일반적입니다.

그러나 현재 Detection과 Recognition을 동시에 해내는 End-to-End OCR 모델도 연구 중 이라고 합니다.

CHAPTER

1. 프로젝트 상세 정보 및 설명

Text Detection Model

Text Detection (Vision)
: 이미지를 입력하면, 그 이미지 상에서 텍스트가 위치한 부분을 찾아내는 모델.

저희는 네이버 클로바 AI의 CRAFT 모델을 채용하였습니다.
다른 Detection 모델은 수평한 글자만 잡는데에 비해 CRAFT 모델은 굴곡진 텍스트도 탐지를 잘해냈기 때문입니다.
(바운딩 박스가 아닌 세그멘테이션)

색상점등을 낮춰서 특이점 추출 ->
단순히 object detection 기법 뿐 아니라 Segmentation 기법도 동원되며,
문자가 가지는 독특한 특성까지 고려하여 디텍트

CHAPTER

1. 프로젝트 상세 정보 및 설명

Text Detection Model

CRAFT 모델은 VGG-16 신경망을 기반으로 하며,
이미지에서 텍스트의 특징점을 찾아내어 텍스트가 위치한 위치를 찾습니다.

Final output

CHAPTER

1. 프로젝트 상세 정보 및 설명

Text Recognition Model

Text Recognition (Natural Language)
: 이미지가 입력되면 그 이미지에서 Text 를 추출해 내는 모델.

저희는 네이버 클로바 AI의
DTRB(deep-text-recognition-benchmark) 모델을 채용하였습니다.

Available

SHARESHACK

Londen

CHAPTER

1. 프로젝트 상세 정보 및 설명

Text Recognition Model

DTRB(deep-text-recognition-benchmark)

Text Recognition 모델은 이미지의 해당 텍스트가 어떤 글자인지 파악하여 추출해줍니다.

처음에는 그저 이미지상의 글자를 Classification 하는 Vision쪽 모델인줄 알았습니다. 검출된 영역의 문자가 무엇인지 인식, 한개의 문자문자별로 인식하고 다시 결합 하는 단순한 형태인 줄 알았으나,

```
character='0123456789abcdefghijklmnopqrstuvwxyz'
```

이는 단순히 One-Hot 인코딩으로 분류만 진행한 것이고, Recognition 모델의 경우는

컴퓨터가 이해할 수 있는 Text 자체로서의 변환이 이루어지기에 자연어 인공지능으로 분류되었음을 알았습니다.

CHAPTER

1. 프로젝트 상세 정보 및 설명

Translation Model

Translation (Natural Language)

Kobart-trans Model 사용.

한국어에서 영어로 번역되는 모델은 상대적으로 찾기 쉬웠는데

영어에서 한국어로 번역되는 모델이 찾기 매우 힘들었습니다.
이는 번역 모델은 상업적으로 쓰이는 자원이기에 찾기 어려울 것이다 라는 선생님의 피드백을 들었습니다.

현 프로젝트에서 사용한 이 모델 또한 상업적으로 이용하기엔 조금 부족한 모델로 성능이 매우 좋지는 않습니다.

CHAPTER

2. 프로젝트 시연 (학습용)

CHAPTER

3. 추가 구현 프로젝트 설명

앞 서 프로젝트 시연에서도 볼 수 있듯,

제작된 프로그램 성능이 생각보다 이상적이지 않았습니다.

1. 학습 시켰던 Text Recognition 모델이 잘못되게 영어를 인식하는 경우가 있습니다.

2. Text Recognition : DTRB 모델이 원래 띄어쓰기를 인지 안하는 모델인 것으로 파악했습니다.

3. 앞서 모델 설명 때 이야기하였듯, 프로젝트에 사용한 번역모델(Translation Model)의 성능이 매우 좋지 않습니다.

CHAPTER

3. 추가 구현 프로젝트 설명

저희는 성능이 같은 두 프로젝트를 제작하였습니다.

앞서 보여드린 첫 프로젝트는

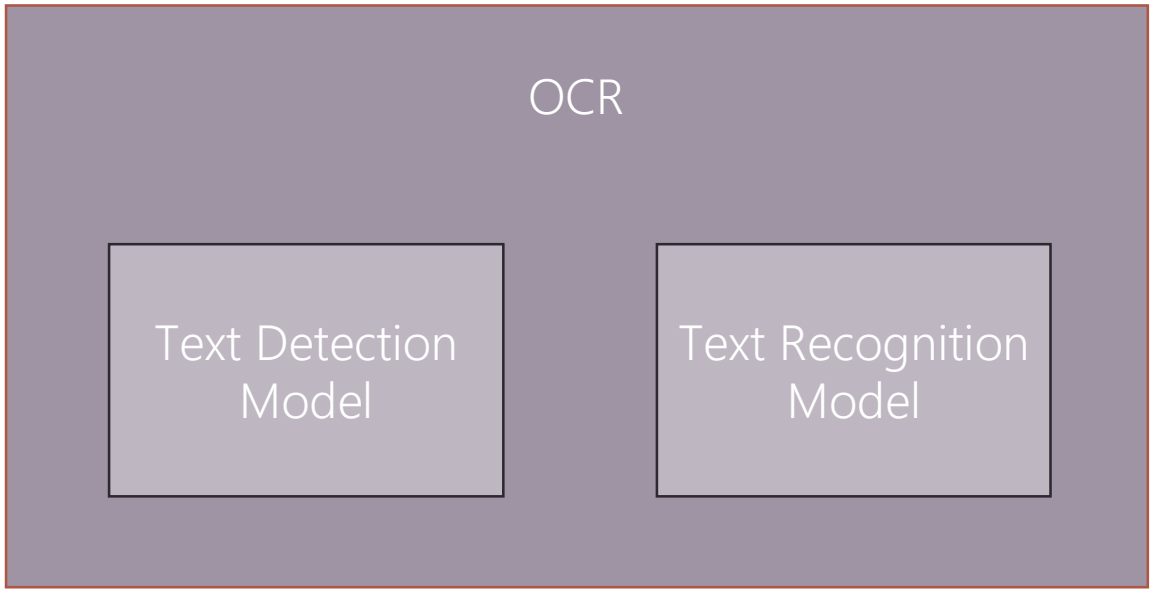
특정 모델을 코드로써 직접 제작하는 프로그램에 적용해서 그 모델을 사용해 보는 것과, 해당 모델에 원하는 데이터로 학습을 시켜보기 위해 진행한 프로젝트입니다.

다음으로 보여드릴 두번째 프로젝트는 성능에 치중하여 사용하기에 조금이라도 더 유용하도록 이미 학습된 모델과 API를 이용하여 구현하였습니다.

기능은 동일합니다.

CHAPTER

3. 추가 구현 프로젝트 설명 (모델)



Text Detection Model - CRAFT Model

Text Recognition Model - EasyOCR Recognition Model

CHAPTER

3. 추가 구현 프로젝트 설명 (모델)

Translation Model

Translation Model - 네이버 파파고 API를 발급하여 사용

CHAPTER

3. 프로젝트 시연 (성능용)

CHAPTER

4. 구현 챌린지

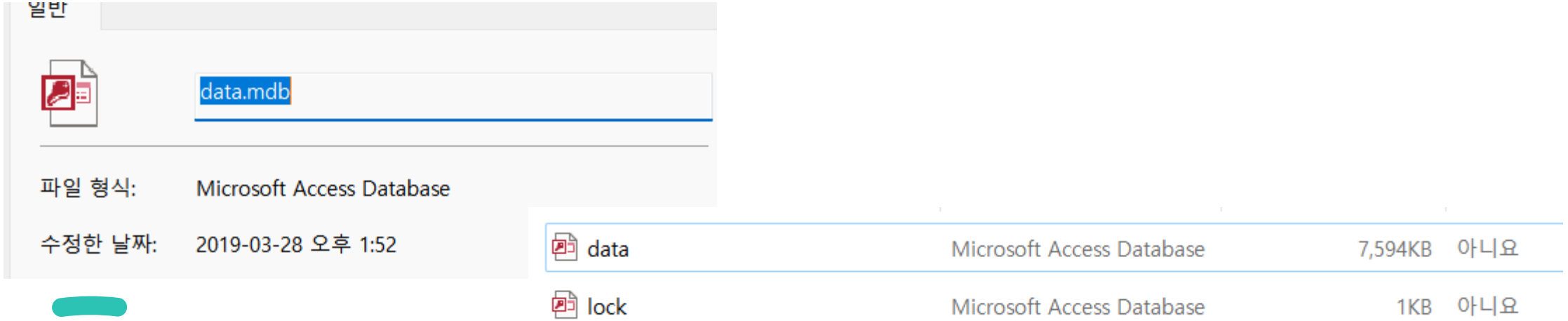
Text Recognition 모델(DTRB) 학습 데이터 관련 어려움

CHAPTER

4. 구현 첼린지

- Text Recognition 모델(DTRB) 학습 데이터 관련 어려움

사용한 DTRB 모델은 자연어 모델이므로 학습 데이터 구축에 살짝 생소하여 어려웠음.
학습 데이터를 처음보는 lmdb 양식을 통해 사용하였기에 커스텀 데이터를 넣을 때 어떻게 넣는가와 lmdb양식의 데이터를 어떻게 일반 파일화 하는지에 대해 고민을 많이 했었음.



CHAPTER

4. 구현 챌린지

- Text Recognition 모델(DTRB) 학습 데이터 관련 어려움

모델을 사용하기 위해 커스텀 데이터 로더 필요.
원래 DTRB 코드에서는 torch.utils.data.DataLoader를 통해 경로값으로 이미지를 받아 모델에 적용하였음.

하지만 우리가 하고자하는건 Text detection 모델로 부터 도출된
Numpy배열이미지를 통해 모델에 적용해야함.

즉 Numpy 배열의 이미지를 Numpy를 Tensor로 변경하고 그 변경한 Tensor들을 이용하여 TensorDataset을 만들고 이를 이용하여 DataLoader를 구축.

근데 이미지를 로드할 때 단순 imread가 아니라 여러 전처리가 함수적으로 구현이 되어있어서 커스텀하기 불편하였음

cv2 ->(차원낮추기)-> numpy -> tensor

추가적으로
Text Detection으로 cv2형식 Numpy를 가져왔는데
Recognition에선 PIL을 써서 가져온 형식을 다 변환해야햇음 커스텀 데이터를 짜며 고생

CHAPTER

4. 구현 챌린지

Text Recognition 모델 (DTRB) 학습 환경 세팅 어려움

CHAPTER

4. 구현 챌린지

- Text Recognition 모델 (DTRB) 학습 환경 세팅 어려움

data_loader의 환경문제
Exception has occurred: TypeError can't pickle Environment objects
가 떴는데 이는 실행하는곳에서의 환경이 차이나기때문이였음 (GPU가 1개여서)

데이터 로더 생성시 num_workers = 0을 선언해줌으로써 해결하였음.

```
69        self.dataloader_iter_list.append(iter(_data_loader))
```

Exception has occurred: TypeError ×
can't pickle Environment objects

```
  File "D:\deep-text-recognition-benchmark-master\dataset.py", line 69, in __init__
    self.dataloader_iter_list.append(iter(_data_loader))
  File "D:\deep-text-recognition-benchmark-master\train.py", line 31, in train
    train_dataset = Batch_Balanced_Dataset(opt)
  File "D:\deep-text-recognition-benchmark-master\train.py", line 317, in <module>
    train(opt)
TypeError: can't pickle Environment objects
```

```
parser.add_argument('--workers', type=int, help='number of data loading workers', default=0) # default=4이지만 환경맞게 0으로 수정
```

CHAPTER

4. 구현 챌린지

- Text Recognition 모델 (DTRB) 학습 환경 세팅 어려움

너무 과한 학습시간

```
TERMINAL
num_class: 38
----------------------------------------

[1/300000] Train loss: 3.65776, Valid loss: 3.61139, Elapsed_time: 37.98519
Current_accuracy : 0.000, Current_norm_ED  : 0.01
Best_accuracy    : 0.000, Best_norm_ED     : 0.01
--------------------------------------------------------------------------------
Ground Truth              | Prediction                | Confidence Score & T/F
--------------------------------------------------------------------------------
30                        | ekkkkkkkkkkkkkkkkkkkkkkk | 0.0000  False
do                        | kkkkeeeeeeeeeeeeeeeeeee | 0.0000  False
staan                     | kkkkkeeeeeeeeeeeeeeeeee | 0.0000  False
walk                      | kkkkkkkkkkkkkkkkkkkkkkk | 0.0000  False
and                       | kkkkkkkkkkkkkkkkkkkkkkk | 0.0000  False
--------------------------------------------------------------------------------
```

줄이기 위한 노력 - 배치사이즈 증가 - 동일한 시간 , 오히려 늘어난 시간

알고보니 Git 과 환경을 동일시 하기위해 낮춘 토치의 버전으로인해 GPU인식 불가,
CPU로 학습 진행이였던것
(디버깅으로 print(torch.cuda.is_available()) 에서 False가 나오는것을 확인)

기존 사용하였던 환경을 복사하여 Git환경과 무관하게 최신 모델로 학습 진행 -> 문제없이 기동,
GPU 사용 가능

CHAPTER

4. 구현 챌린지

- Text Recognition 모델 (DTRB) 학습 환경 세팅 어려움

리눅스 기반에서 구현된 모델을 윈도우에서 사용하여 생긴 에러 관련

학습을 진행하며 에러가 뜨는경우가 있었는데,
구글링을 해본 결과 리눅스에서의 코드가 윈도우일 경우 발생하는 문제로 파악.

```
87          image, text = data_loader_iter.next()

Exception has occurred: AttributeError ×
'_SingleProcessDataLoaderIter' object has no attribute 'next'
  File "D:\deep-text-recognition-benchmark-master\dataset.py", line 87, in get_batch
    image, text = data_loader_iter.next()
  File "D:\deep-text-recognition-benchmark-master\train.py", line 147, in train
    image_tensors, labels = train_dataset.get_batch()
  File "D:\deep-text-recognition-benchmark-master\train.py", line 317, in <module>
    train(opt)
AttributeError: '_SingleProcessDataLoaderIter' object has no attribute 'next'
```

이는, 사용한 모델인 DTRB가 리눅스 기반으로 구현되었기 때문이 였는데,
코드를 전체적으로 윈도우 기반으로 일일이 코드를 변경해 주어야했기때문에 조금 많이 애를 먹었음.

특히나, 학습을 진행하던 도중에 해당 코드가 실행이 되어야지만 제가 제대로 윈도우로 변경을 했는지 알 수 있었기 때문에 디버깅과정에서 너무 시간이 오래걸리고 어렵다고 느꼈었음.

CHAPTER

4.

train.py 1 | prediction.py | sequence_ | extraction.py | log_train.txt | dataset.py

dataset.py > Batch_Balanced_Dataset > get_batch

```
83      def get_batch(self):
84          balanced_batch_images = []
85          balanced_batch_texts = []
86
87          for i, data_loader_iter in enumerate(self.dataloader_iter_list):
88              try:
89                  # image, text = data_loader_iter.next() # 원본 코드 ( 리눅스 기반 )
90                  image, text = next(data_loader_iter)
91                  balanced_batch_images.append(image)
92                  balanced_batch_texts += text
93              except StopIteration:
94                  self.dataloader_iter_list[i] = iter(self.data_loader_list[i])
95                  image, text = self.dataloader_iter_list[i].next()
```

Exception has occurred: AttributeError

```
'_SingleProcessDataLoaderIter' object has no attribute 'next'
  File "D:\deep-text-recognition-benchmark-master\dataset.py", line 90, in get_batch
    image, text = next(data_loader_iter)
StopIteration:

During handling of the above exception, another exception occurred:

  File "D:\deep-text-recognition-benchmark-master\dataset.py", line 95, in get_batch
    image, text = self.dataloader_iter_list[i].next()
  File "D:\deep-text-recognition-benchmark-master\train.py", line 147, in train
```

PROBLEMS 1 | OUTPUT | TERMINAL

TERMINAL — Python Debug Console — DEBUG CONSOLE

```
of                    | of                    | 0.9994  True
com                   | com                   | 0.9994  True
--------------------------------------------------------------------------------
[16000/300000] Train loss: 0.10489, Valid loss: 0.59771, Elapsed_time: 22053.19306
Current_accuracy : 75.358, Current_norm_ED  : 0.90
Best_accuracy    : 75.572, Best_norm_ED     : 0.90
--------------------------------------------------------------------------------
Ground Truth          | Prediction            | Confidence Score & T/F
--------------------------------------------------------------------------------
engulry               | enquiry               | 0.5758  False
```

CHAPTER

4.

train.py 1 | prediction.py | sequence_ | extraction.py | log_train.txt | dataset.py

dataset.py > Batch_Balanced_Dataset > get_batch

```
83      def get_batch(self):
84          balanced_batch_images = []
85          balanced_batch_texts = []
86
87          for i, data_loader_iter in enumerate(self.dataloader_iter_list):
88              try:
89                  # image, text = data_loader_iter.next() # 원본 코드 ( 리눅스 기반 )
90                  image, text = next(data_loader_iter)
91                  balanced_batch_images.append(image)
92                  balanced_batch_texts += text
93              except StopIteration:
94                  self.dataloader_iter_list[i] = iter(self.data_loader_list[i])
95                  image, text = self.dataloader_iter_list[i].next()
```

Exception has occurred: AttributeError

```
'_SingleProcessDataLoaderIter' object has no attribute 'next'
  File "D:\deep-text-recognition-benchmark-master\dataset.py", line 90, in get_batch
    image, text = next(data_loader_iter)
StopIteration:

During handling of the above exception, another except

  File "D:\deep-text-recognition-benchmark-master\data
    image, text = self.dataloader_iter_list[i].next()
  File "D:\deep-text-recognition-benchmark-master\train.py", line 147, in train
```

절반 이상의 학습이 완료되었는데, 리눅스 윈도우 차이 미처 수정하지 못한 부분에서 디버깅오류 발생
눈물을 머금고 해당 부분 수정하여 재 학습

PROBLEMS 1 | OUTPUT | TERMINAL

TERMINAL — Python Debug Console — DEBUG CONSOLE

```
of                      | of                      | 0.9994  True
com                     | com                     | 0.9994  True
--------------------------------------------------------------------------------
[16000/300000] Train loss: 0.10489, Valid loss: 0.59771, Elapsed_time: 22053.19306
Current_accuracy : 75.358, Current_norm_ED  : 0.90
Best_accuracy    : 75.572, Best_norm_ED     : 0.90
--------------------------------------------------------------------------------
Ground Truth            | Prediction            | Confidence Score & T/F
--------------------------------------------------------------------------------
engulry                 | enquiry               | 0.5758  False
```

CHAPTER

4. 구현 챌린지

기타 이슈

CHAPTER

4. 구현 첼린지

- 이미지 상에서 번역 부분 처리 방법 고민

이미지 상에서 번역 부분에 평균 색상으로 채우고자할때

그냥 np.mean 함수 쓰면 쉬울줄 알았는데 차원이 3개인 RGB모두 고려해야해서 쉽지않았음.

mean으로 처리해버리면 그냥 Gray색으로만 평균값을 구해서 모두 회색으로만 나옴.

CHAPTER

4. 구현 첼린지

- 이미지 상에서 번역 부분 처리 방법 고민

블러처리할 이미지의 색상값 구하기.

처음 생각한 것 - 가장 쉽지만 가장 비효율적인
처음 아이디어를 따라서 단순하게 모든 차원당 평균 픽셀 값 찾는것. >> 결과가 호도하고 은근 부정확함 : 가장 주된 색상이 아닌 모든 픽셀값의 평균을 고려하기 때문에 >> 대비가 큰 이미지 (한 이미지 안에 밝고 어두운 부분이 있는경우) 로는 별로임

이를 개선한 방법 - 비지도학습인 K-Means clustering 을 통해 색상군 도출해내기.

이를 적용하였으나, 현 목적에는 맞지않아 그냥 평균값으로

CHAPTER

5. 한계점 및 개선 사항

1. 모델 3개 Process로 인한 속도

사실상 프로그램의 구조를 보면

이미지 -> 모델1 -> 모델2 -> 모델3 -> 이미지

이렇게 3개의 모델을 사용하게 된다.

이로 인해 모델 구현에 의한 속도 지연이 생기는데, 원래는 영상을 스트리밍 실시간으로 번역해주는 프로그램도 구현하려 했으나 이 이슈로 인해 구현하지 못하였다.

CHAPTER

5. 한계점 및 개선 사항

1. 모델 3개 Process로 인한 속도

나중에 현재 연구중인 Detection과 Recognition을 동시에 해내는 End-to-End OCR 모델을 사용하여 모델 시간 자체를 줄이거나,

데이터 검색 구현을 List로 저장하고 순회해서 찾도록 구현하였는데,
구조적으로 Key-value 형식으로 바꾸는 등 코드 효율적으로 시간을 줄이는 등의 방법을 통해

속도를 높여 성능 개선을 할 수 있다고 생각.

CHAPTER

5. 한계점 및 개선 사항

2. 가로 세로 글자 인식.

현 OCR모델들은 모두 수평적으로만 글자를 인식.

이에, 다른 각도의 글자를 인식하게 하고자 글자를 회전시켜서

Threshold가 가장 높은 것을 채용하도록 구현하였음.

처음에는 90, 180, 270 돌려서 가장 확률 높은 것을 사용할까 싶었으나,
결과값이 나오는 시간 단축을 위해 넓이가 높이의 1.5배 이상일 경우로
분기하여 회전하도록 구현하였음.)

CHAPTER

5. 한계점 및 개선 사항

넓이가 높이의 1.5배가 아님! 그러므로 회전!

둘 중 더 높은 Threshold를 가진 이미지를 채용

CHAPTER

5. 한계점 및 개선 사항

2. 가로 세로 글자 인식.

이에 대한 한계점으로는

90, 180, 270 으로만 변형을 시도하므로 각도가 대각적으로 존재하는 글자는 글자를 찾아내지 못할 것임.

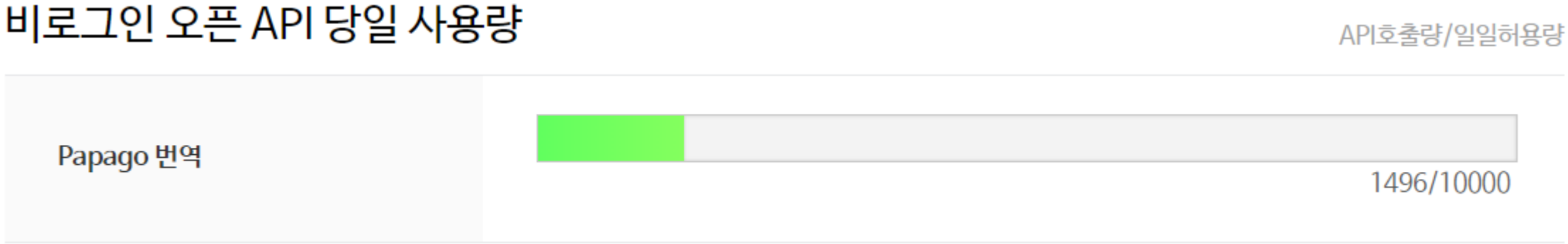
조금 더 빠르게 동작할 수 있도록 효율적으로 코드를 개선하고 대각선으로도 이미지를 변형 시도하도록 개선할 수 있다고 생각.

CHAPTER

5. 한계점 및 개선 사항

3. 파파고 API의 하루 사용 제한

성능용으로 제작한 프로젝트의 경우 파파고 API로 인해 사용량 제한이 있음.



CHAPTER

5. 한계점 및 개선 사항

3. 파파고 API의 하루 사용 제한

추후, 직접 학습을 시킨 모델을 사용하게 되거나

성능 좋고 무료인 모델을 찾음으로 이를 개선을 할 수 있다고 생각.

THANK YOU.

2023.02.20 이주형, 이승현, 이성규, 민안세, 김민정