**Pixel Link Pytorch Implementation**

**201421553**

**소프트웨어학과**

**김 한**

**1.Introduction to functions**

|  |
| --- |
| **Main.py**  **Net.py**  **Config.py**  **Datasets.py** |

**2. Dataset 구성하기**

|  |  |
| --- | --- |
|  | **<explanation>**  Import가 되는 순간 사용자정의 클래스인 net.py, datasets.py, criterion.py 등이 호출 된다. 단, 내부의 정의된 함수와 init이나 getitem 등은 실행이 되지 않고 class의 껍데기만 호출이 된다.  <사용하는 라이브러리>  Numpy : 각종 array 연산  Torch: CNN 신경망 구성  Torch.utils.data : DataLoader를 통해 raw한 이미지들을 dataset으로 구성.  Cv2 : 각종 이미지에 대한 전처리 |

|  |
| --- |
|  |
| **<explanation>**  Argparse는 **“Command Line”에** 사용자로부터 argument를 입력받아서 프로그램을 시킬 때 각 argument를 분리(pasring)해주는 것을 도와주는 module이다. 따라서 argparse.ArgumentParser함수를 통해 parser가 먼저 생성된다. 다음으로 add\_argument를 이용하여 입력받고자 하는 인자의 조건을 설정한다. **위의 코드에선 –-train 혹은 –retrain**이다. 다음으로 parser.parse\_args함수는 인자들을 parsing하여 args에 저장한다. |

|  |
| --- |
|  |
| **<explanation>**  Args 의 namespace에 train : true / retrain : false와 같이 미리 정의된 argument와 type의 값이 저장되어 있다. 따라서 위의 코드를 만나게 되면 **입력된 type 값에 맞게 해당함수가 호출**된다. **Python main.py –train True라고 입력할 경우** elif 문 안에 main() 함수를 호출하게 된다. |

|  |
| --- |
|  |
| **<explanation>**  다음으로 main함수가 호출이되면 datasets이라는 py의 함수인 PixelLinkIC15Dataset를 호출하게 되는데 argument로 config.py의 train\_image\_dir 와 train\_labels\_dir에서 값을 사용하게 된다.  두번째 코드는 config.py의 train\_image\_dir와 train\_labels\_dir 인데, 변수의 이름처럼 **학습을 위한 이미지와 정답 label에 해당하는 Ground Truth의 경로**가 적혀있다. |

|  |
| --- |
|  |
| **<explanation>**  pixelLinkIC15Dataset 클래스의 객체가 생성되는 동시에 init이라는 생성자를 호출하게 된다. 다시 **super를 통해 상속받는 parent 클래스인 ICDAR15Dataset 클래스의 생성자를 부르게 된다**. |

|  |
| --- |
|  |
| **<explanation>**  Parent 클래스인 ICDAR15Dataset의 생성자가 불릴 때, image\_dir 과 label\_dir을 저장하게 된다. 위에서 설명했듯, 여기에는 학습을 위한 이미지와 label이 담겨있는 경로이다. 이것을 argument로 하여 read\_labels을 부르게 된다. Read\_label 함수의 argument로 label\_dir와 config.all\_trains을 사용한다. 그 결과를 all\_label에 저장하게 된다.  두번째 코드는 config.py에 들어있는 all\_trains라는 변수인데 학습을 위한 이미지의 총 개수인 1000이 저장되어있다. |

|  |
| --- |
|  |
| **<explanation>**  우선, res에 위와 같은 코드를 실행하게 되면, 2차원 배열안에 num개수만큼의 1차원 배열이 생성되게 된다. 따라서 num이라는 파라미터로 all\_trains라는 이미지 1000개의 개수가 passing 되므로 1000개의 1차원 배열을 생성하게 된다. |

|  |
| --- |
|  |
| **<explanation>**  기본 파일 open으로 ground\_truth가 열리지 않고 오류가 생기기 때문에 codecs.open 함수와 utf-8 방식읭 인코딩을 통해 정상적으로 파일을 open한다. argument로는 label이 저장된 경로인 train\_images/ground\_truth/gt\_img\_1.txt 라는 최종경로에서 for문을 통해 하나씩 불러오게된다.  다음으로 tmp라는 dictionary를 선언하고 coordinate(text영역의 각 꼭짓점 4쌍의 좌표), content(들어있는 text의 내용), ignore(무시할만한 text), area(text 영역의 넓이)라는 key를 선언한다.    Ground\_Truch의 raw한 data값은 위와 같다. 총 8개의 coordinate 값과 text 값으로 이루어져 있다. 따라서 먼저 이를 “,” 기준으로 split하여 list로 담은 후에 reshape(4,2)를 이용하여 [[377,117],[463,117],[465,130],[378,130]](첫번째 줄 예시)와 같이 coordinate 값을 알맞게 재구성한다. 다음으로 tmp[‘content’]에는 인식이 된 text 문자열을 담는다. 또한 무시되어도 좋을만한 ###(ignore) 이라면 True로 그 값을 저장하고 마땅히 인식되어야 할 값이라면 False로 값을 저장한다.  마지막으로 tmp[‘area’]는 cv2의 contourArea에 coor(좌표값)을 argument로 넘겨서 넓이를 구한 값을 저장한다.  따라서, read\_label이라는 함수를 거치고 나면 ground\_truth의 txt 파일은 아래과 같은 구조로 res에 담기게 된다.    즉, **1000개의 이미지 파일에 대한 ground\_truth로써 좌표값, text 내용, 무시여부, text 넓이**를 담은 dictionary 1000개가 담겨있다. |

|  |
| --- |
|  |
| **<explanation>**  생성된 현재 dataset 객체에는 ground\_truth label에 대한 전처리가 완료된 상태로 남아있다.  다음으로 WeightedRandomSampler를 이용하여 총 1000개의 이미지 중에서 **config.batch\_size(24), replacement(중복)=True로 하여 24개의 이미지를 임의로 선택**하게 된다.  다음으로 dataloader를 통해 앞으로 학습을 진행할 data를 load할 수 있도록 완성한다.  여기까지가 **기존의 raw한 이미지 data를 활용하기위한 dataset 구성방식**이다. |

**3.Network 이해하기**

|  |
| --- |
|  |
| **<explanation>**  Dataloader가 완성되면 다음으로 net.py 의 class net의 객체를 생성하고 초기화를 진행한다. |

|  |
| --- |
|  |
| Class net은 torch 의 nn에 들어있는 module을 상속받아서 신경망 관련 함수들을 overriding하여 이용한다. 위에 보이는 network 의 구조는 pixellink에서 언급된 VGG Network의 기본프레임을 변형하여 이용하였다. 먼저 convolution stage1(conv1\_1 + relu + conv1\_2 + relu + pool)로 정의되었고 각 파라미터에서 stride와 padding은 1로 한다. 최초 input channel은 RGB이므로 3개이며, kernel(filter)의 사이즈는 3x3으로 VGG와 같다. Output channel 64는 hyperparameter로써 convolution stage를 거칠때마다 64->128->256->512로 늘어나며 따라서 최초의 input channel 의 개수를 제외하고는 모든 imput channel의 개수는 당연히 이전 output channel의 개수와 동일하다. Pooling 같은 경우는 maxpooling 방식과 stride 2로 정의되어있다. 기본 kernel size는 2x2이며, stride도 2이다. 따라서 이미지의 크기는 pooling을 통과하면서 줄어든다. 마지막 convolution stage5의 pool에서는 kernel 의 크기를 3x3으로 하며 stride와 padding을 1로 정의한다. |