**End-to-end OCR 시스템**

최지호

**Abstract**

Keras ocr과 CRNN 모델을 사용하여 입력한 문자가 포함된 이미지에서 문자 영역을 인식하고, 문자 자체를 추가적으로 인식하여 text 형식의 결과를 출력하는 시스템을 제작하였습니다. 해당 시스템은 이미지 내의 다양한 크기와 형태의 문자를 인식하여 text형식으로 출력할 수 있습니다. 이를 통해 다양한 문자가 포함된 이미지에서 문자 인식을 수행할 수 있습니다. 하지만 그 한계점도 명확한데, 영어 알파벳만을 인식할 수 있으며 너무 긴 문자, 방향이 가로가 아닌 문자, 너무 해상도가 낮은 이미지의 문자 등은 인식하는 데 어려움이 있습니다. 따라서 더 다양한 형태의 데이터셋을 사용한다면 문자 인식률이 더욱 향상시킬 여지가 존재합니다.

**1. Introduction**

이 프로젝트는 문자를 포함한 이미지가 주어질 시, 문자를 인식하여 Text 형식으로 예측을 출력하는 시스템을 제작하는 것이 목표입니다.

CRAFT라는 알고리즘을 기반으로 만들어진 Keras ocr은 이미지에서 가우시안 분포를 사용하여 문자가 포함된 영역을 단어 단위로 인식할 수 있습니다. 또한 CRNN은 입력된 문자열 이미지를 text로 추론할 수 있도록 학습할 수 있는 모델입니다. 이번 프로젝트에서는 이 두 가지 기술을 접목시켜 아래 Figure 1의 프로젝트 루브릭을 만족하는 end-to-end OCR 시스템을 만드는 것을 목표로 하였습니다.

Figure 1. 프로젝트 루브릭

Keras OCR, CRNN은 이미 코드가 존재하기 때문에 이를 사용하였습니다. 따라서 본 논문에서는 데이터셋의 전처리 과정, 학습과정, 결과를 확인하기 위한 함수 제작 부분을 집중적으로 다루겠습니다.

**2. Method**

**2.1. 데이터셋 전처리**

데이터셋은 MJSynth를 사용하였습니다. MJSynth는 데이터셋은 9만장의 이미지, 900만 개의 단어가 포함되어 있는 합성 데이터셋으로, 90개 이상의 글꼴과 9개의 서로 다른 형태의 문장 구조를 사용하여 합성되었습니다. 이 합성된 이미지에서 텍스트 영역과 바운딩 박스가 추출되어 데이터셋에 포함됩니다.

MJSynth 데이터셋에서 이미지와 레이블을 각각 전처리 하였는데, 이미지는 RGB(100, 32, 3)크기로, 레이블은 모델에서 처리하기 용이하도록 모두 대문자로 변경하고 알파벳 대문자와 숫자에 포함되지 않는 문자는 전부 제거한 뒤 레이블의 입력 길이를 22 이하로 제한하였습니다. 이러한 전처리 모두 입력 데이터의 크기를 일정하게 유지하여 모델을 학습시키기 위해 실시하였습니다.

그 후 추가적으로 레이블은 스칼라 값으로 인코딩 하였는데, 중복되는 문자 사이에는 ‘-‘(인코딩하면 0)를 추가하여 중복 문자를 따로 인식할 수 있도록 하였습니다.

**2.2. Recognition모델 생성**

CRNN 모델을 사용하였습니다. CNN에서 수행한 특징 추출 결과물이 생성되고, RNN에서 이 결과물을 바탕으로 추측이 이루어집니다. 입력 이미지 데이터의 형태는 (100, 32, 3)의 형태로 입력되며, 입력 레이블 데이터의 형태는 입력되는 문자의 길이+2의 크기로 지정하였습니다. (공백 문자와 ‘-‘문자가 포함되기 때문에)

Figure 2. CRNN 모델



**2.3. 모델 훈련**

Loss function으로는 CTC loss를, optimizer로는 clipnorm=5로 설정한 Adadelta를 사용하였습니다. 첫 학습에선 학습률은 0.05, 에포크는 100으로 학습을 진행하였습니다.

Figure 3. 학습 그래프1

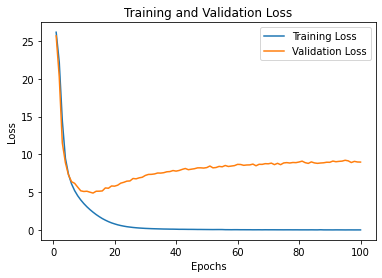
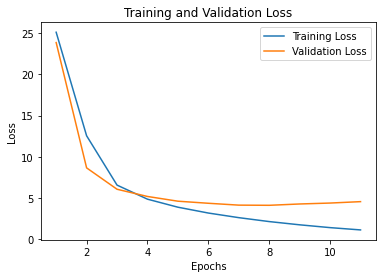


Figure 3을 보면 알 수 있듯이 10에포크 언저리에서 과적합이 일어남을 확인할 수 있었습니다.

이후 두 번째 학습에선 학습률을 0.05에서 0.1로 수정한 후, Early Stopping을 사용하여 모델을 처음부터 재학습하였습니다.

Figure 4. 학습 그래프2



학습이 안정적으로 진행된 것을 Figure 4의 그래프를 통해서 확인할 수 있습니다.



Figure 5. 모델이 생성한 prediction

Figure 5는 모델이 생성한 prediction입니다. 눈으로 보기에도 대부분 결과가 알맞게 출력되는 것을 확인할 수 있습니다. 그러나 RWNDE(Rwanda), HALBO(halloo)처럼 인식이 올바르게 되지 않는 문자도 있는 것을 확인할 수 있었습니다. 하지만 이번 프로젝트의 목적은 단순히 성능이 높은 문자인식 시스템을 만드는 것이 아닌 end to end로 문자인식 시스템을 구현하는 것이기 때문에, 이 정도 성능이면 충분히 만족스럽다고 할 수 있습니다.

**2.4. 이미지에서 문자 영역 검출**

3.3절까지는 주어진 데이터를 기반으로 학습하여 예측 레이블을 출력하는 CRNN 모델을 학습하였습니다. 하지만 실제로 이 시스템에 주어질 이미지는 학습 데이터와 같이 잘 정제된 형식의 문자만 잘려있는 데이터가 아닙니다. 따라서 모델이 잘 인식할 수 있도록 문자를 포함한 이미지에서 문자가 포함된 부분만 정확하게 잘라서 입력할 필요가 있습니다.

Keras ocr을 사용하여 이 문제를 해결하였는데, keras ocr은 CRAFT 텍스트 감지 모델을 사용합니다. 이는 글자 자체의 중앙값의 가우시안 분포와 글자 사이의 자간에도 가우시안 분포를 사용하여 한 글자의 영역이 어디까지인지, 한 단어의 영역이 어디까지인지를 탐지하는데 매우 용이합니다.





Figure 6. Keras ocr 결과1

위의 Figure 6는 keras ocr을 통해 입력 이미지에서 텍스트 영역을 감지하여 해당 영역만 잘라낸 이미지입니다. 앞서 모델 훈련에서 사용한 MJSynth 데이터의 형태와 비슷한 형태로 이미지가 잘리게 됩니다. 여기에 추가적으로 이미지 크기를 (100, 32, 3)의 형태로 전처리하여 모델의 입력으로 사용할 준비를 마칩니다.

**2.5. 검출된 이미지를 모델을 통해 inference**

앞서 모델은 학습을 위해 이미지 데이터와 레이블 데이터를 입력으로 받도록 설계했었습니다. 하지만 inference 과정에선 이미지만이 입력 데이터로 필요하기 때문에, inference용 모델을 따로 선언하여 이를 사용했습니다. Inference용 모델은 전처리된 (100, 32, 3)크기의 이미지를 입력으로 받아서 결과 이미지와 결과 레이블을 출력합니다.

**3. Result**

앞서 완성한 전처리 함수, 모델, inference 함수를 전부 활용해 실험을 진행하였습니다. Figure 6의 이미지의 결과는 다음과 같습니다.

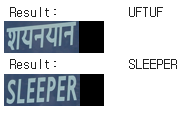


Figure 7. inference 결과1

영어가 아닌 문자는 다른 문자가 출력되지만, SLEEPER의 경우에는 올바르게 SLEEPER가 결과로 출력된 것을 확인할 수 있었습니다. 이를 통해 우선 입력된 이미지에서 문자 영역이 올바르게 인식되고, 인식된 문자 이미지가 추론 모델을 통해 올바르게 텍스트 형식의 레이블을 결과로 출력하는 것을 확인할 수 있습니다.

다른 이미지로 테스트 해보겠습니다.

Figure 8. Keras ocr 결과2

반투명하게 있는 워터마크의 문자까지 문자 영역으로 keras ocr이 올바르게 수행된 것을 확인할 수 있습니다.

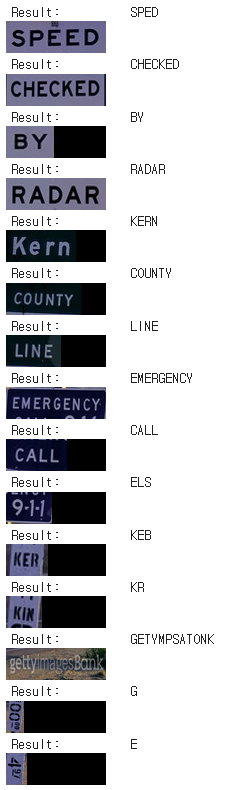


Figure 9. inference 결과2

Inference 결과입니다. 영어 알파벳으로 이루어진 텍스트들은 SPEED를 제외하면 전부 눈으로 보기에도 올바른 결과가 나옴을 확인할 수 있습니다. 하지만 숫자로 이루어진 텍스트, 방향이 가로방향이 아닌 텍스트, 너무 긴 텍스트, 해상도가 낮아 알아보기 어려운 텍스트들은 올바르게 인식이 이루어지지 않았음을 확인할 수 있습니다. 이는 학습한 MJSynth 데이터셋이 아무리 다양한 형태의 문자 이미지를 데이터로 가지고 있다고 하더라도 9-1-1처럼 특수문자와 숫자, 옆으로 회전된 문자 등은 학습하지 않았기 때문에 일어나는 현상입니다.

‘gettyimagesBank’처럼 너무 긴 문자를 (100, 32)로 압축시켜 세로로 길쭉해진 문자 또한 학습된 데이터에 세로로 길쭉한 문자가 없어서 그럴 것입니다. 그렇다면 반대로 가로로 늘어난 문자를 입력해도 결과가 안 좋게 나올 것입니다. 이를 증명하기 위해 다른 실험 결과를 첨부하겠습니다.

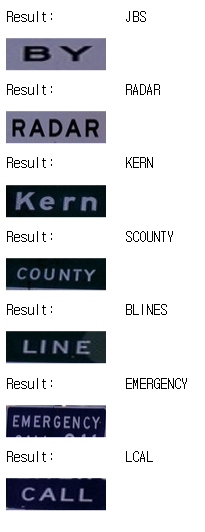


Figure 10. inference 결과3

Figure 10은 가로로 padding을 넣는 전처리를 수행하지 않고 inference를 수행한 결과입니다. 짧은 단어들이 (100, 32)의 크기에 맞게 가로로 이미지가 늘어나 글자도 가로로 길쭉해진 것을 확인할 수 있습니다. 같은 inference 모델인데 Figure 9와는 달리 인식률이 매우 좋지 않은 것을 확인할 수 있습니다. 따라서 이 모델이 너무 긴 문자를 인식하지 못하는 것은 학습된 글씨에 비해 너무 글씨가 세로로 길게 압축됐기 때문임을 알 수 있습니다.

**4. Discussion, Conclusion**

Keras ocr, MJSynth를 학습한 CRNN 모델, inference 모델을 이용해 문자를 포함한 이미지에서 문자 영역을 올바르게 인식하고 인식된 영역에서 올바르게 문자를 인식하여 text 형태로 출력하는 것을 확인할 수 있었습니다. 이는 단순히 문자 이미지를 넣어서 label값을 출력하는 CRNN에 CRAFT 기반 문자 인식 알고리즘을 사용하는 keras ocr을 접목시켜 다양한 이미지 상황에서 다양한 문자를 인식할 수 있게 되었습니다.

하지만 정확도 면에서 상용화하기엔 부족한 수준의 정확도(CTC loss 0.49)와 너무 긴 문자, 숫자, 해상도가 너무 낮은 문자 등은 인식할 수 없는 문제점 또한 안고 있습니다.

이는 추가적으로 더 다양한 형태의 데이터셋을 수집하여 학습을 한다면 해결될 여지가 있을 것입니다. 또한 방향이 가로로 쓰여지지 않은 문자의 경우에는 추가적인 데이터셋 수집 외에 다른 방식으로 해결을 해야 할 것입니다.