IT시스템설계

과제4: 이미지에서 글자 인식

전자전기공학부
B415156 이승호

과제 4: 이미지에서 글자 인식

1. 설계 계획과 목표

처음 영어 알파벳/숫자를 인식할 수 있는 Network를 여러번 학습시켜 보았으나 알파벳과 숫자 사이에 유사성이 너무 많아 성능이 좋지 않았고 알파벳 간에도 대문자와 소문자가 같은 모양을 하고 있는 알파벳이 많아서 정확도 0.87 이상의 성능이 나지 않았고, 실제로 내가 작성한 손글씨를 인식시켰을 때는 인식률이 더 떨어졌다.

*유사한 알파벳/숫자들

l	l	l	i	
o	o	o		
c	c			
k	k			
w	3	E	m	

위와 같이 유사한 알파벳/숫자 모양으로 인해 성능이 떨어져서 한글/숫자 인식으로 계획을 변경했다.

한글은 영어 알파벳과 달리 조합문자이기 때문에 구분해야할 class의 수가 비교가 되지 않을 정도로 많았다. 하지만 실제로 자주 사용되는 문자와 자주 사용되지 않는 문자가 확연히 구분되기 때문에, 나는 인터넷에서 구한 980 class 짜리 dataset을 이용하여 980 class로 구분하기로 하였다.

처음엔 숫자/한글 2 class로 먼저 구분하고 각각 숫자와 한글을 인식하는 network로 인식하려 하였으나 숫자와 한글은 그 특징이 확연히 구분되기 때문에 class를 합쳐 990개의 class를 구분하는 network를 학습하여도 괜찮은 성능을 얻을 수 있을 것이라고 판단했다.

이 과제에서 무엇보다 중요한 점은 이미지 파일에서 글자를 구분해내는 과정이다. 그 과정은 opencv를 이용하기로 하였다. 먼저 불러온 이미지 파일을 grayscale로 바꾸고 자잘한 노이즈를 줄이기 위해 median filter로 blur 처리를 하고, 색을 반전시켜 morphology를 이용해 한 음절이 하나의 contour를 형성하도록 preprocessing을 해주려고 한다.

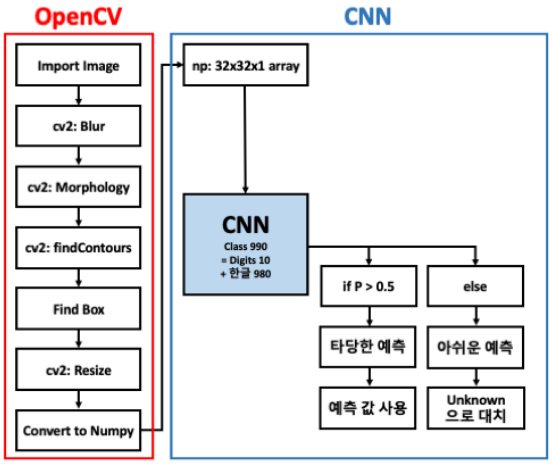
예상되는 문제점은 한글 조합 중에서도 글자 모양이 흡사한 경우가 있어서 그런 경우 인식률이 떨어질 것이라고 생각했다.

*모양이 비슷한 한글 문자들

띵	명
팔	괄
빙	넹

그리고 미리 학습시킨 980개의 한글 중 어느것에도 해당하지 않는 문자가 입력으로 들어올 경우 어떻게 처리할 것이냐는 문제이다. 이상적으로는 학습된 문자 중에 없음을 인식하여 unknown으로 표시할 수 있다면 좋을 것이다.

마지막 문제점은 초성, 중성, 종성을 하나의 contour로 만들기 위해 morphology 처리를 하면서 각각의 음절이 띄어쓰기가 되어 있지 않으면 하나의 단위로 판단할 것이라는 점이다. 아래는 대략적인 system의 diagram이다.



우리는 글을 읽으면서 잘 읽히지 않는 문자가 있어도 앞뒤 문맥을 기반으로 문자를 예측하면서 읽기 때문에, 단순히 글자의 모양만을 판단하여 결정하는 neural network로는 한국인 수준의 글자 인식을 할 수 없다고 생각하였다. 그렇기 때문에 다소 모호한 목표지만, 한글을 갓 배운 외국인이 뜻은 몰라도 발음 정도는 할 수 있는 수준 정도의 성능을 갖추는 것을 목표로 하였다.

2. opencv를 이용한 음절 분리

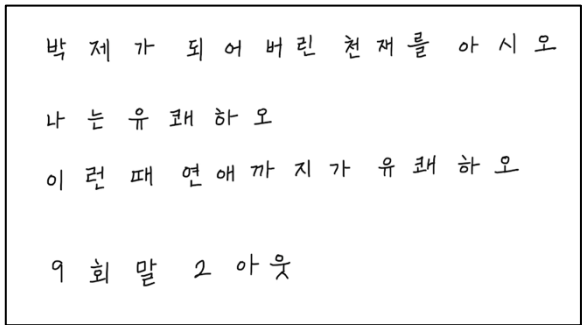
우선 테스트를 위해 사용한 문장은 다음과 같다.

> 박제가 되어버린 천재를 아시오?
> 나는 유쾌하오.
> 이런 때 연애까지가 유쾌하오.
>
> 9 회말 2 아웃

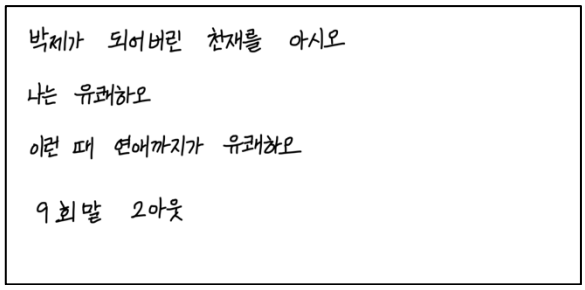
학습 데이터의 980개의 문자에 "를"은 없었기 때문에 "를"을 네트워크에서 어떻게 인식할 것인지도 살펴보기로 했다.

iPad에 디지타이저를 이용해서 1) 또박또박 모든 음절에 띄어쓰기를 한 이미지 2) 또박또박 평범하게 띄어쓰기를 한 이미지 3) 흘겨써 모든 음절에 띄어쓰기를 한 이미지 3가지 버젼으로 글을 작성하였다. 손글씨의 raw data는 다음과 같다.

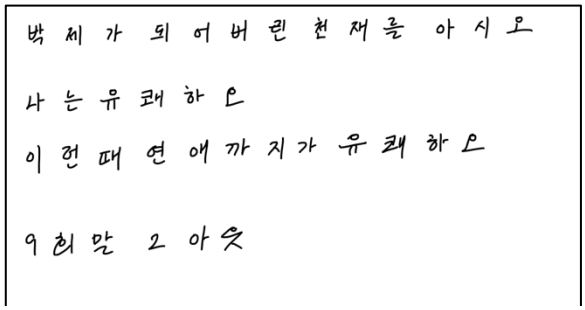
1) 또박또박 + 모든 음절 띄어쓰기



2) 또박또박 + 평범하게 띄어쓰기



3) 흘겨쓰기 + 모든 음절 띄어쓰기

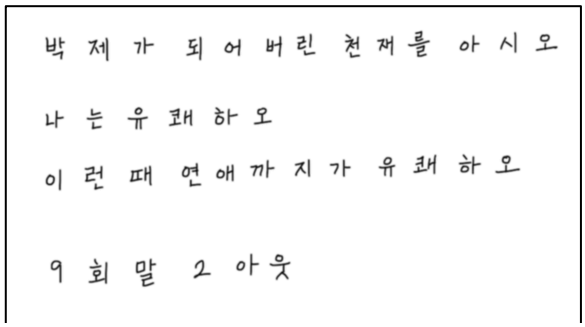


먼저 가장 음절을 찾아낼 확률이 높은 1) 또박또박 모든 음절에 띄어쓰기를 한 이미지에 대해서 preprocessing을 진행해보았다.

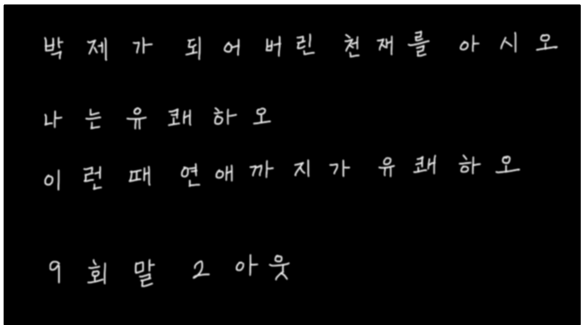
**Preprocessing 과정*

- Blur

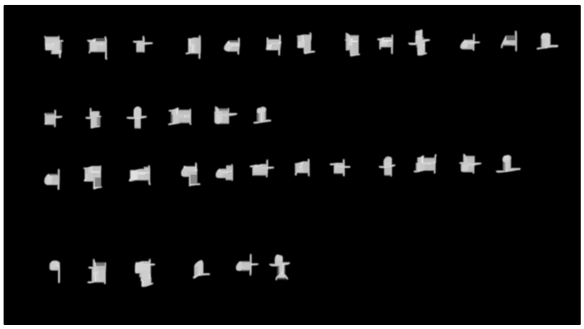
손글씨를 쓰는 과정에서 생기는 자잘한 노이즈를 줄이기 위해 (3,3) size의 median filter로 blur 처리를 해주었다.



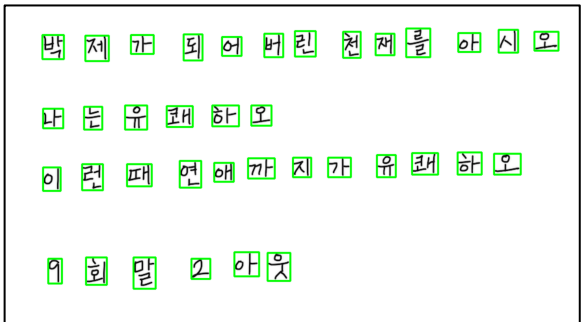
- Invert



- Morphology (Closed) (kernel size = (13, 13))



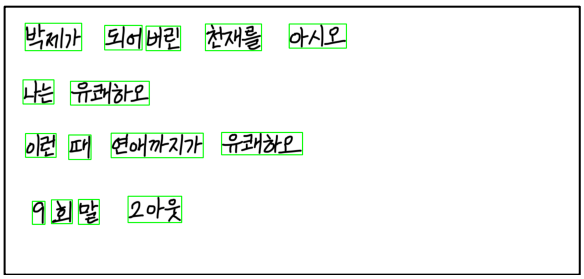
- findContours + Rectangular



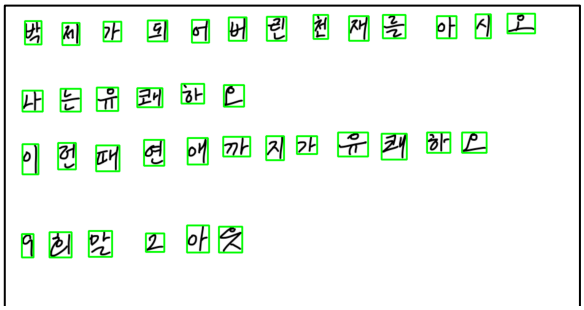
정상적으로 모든 음절을 나누는 것을 확인할 수 있었다. 같은 알고리즘으로 2), 3)의 이미지도 테스트해보았다.

- findContours + Rectangular

2)



3)



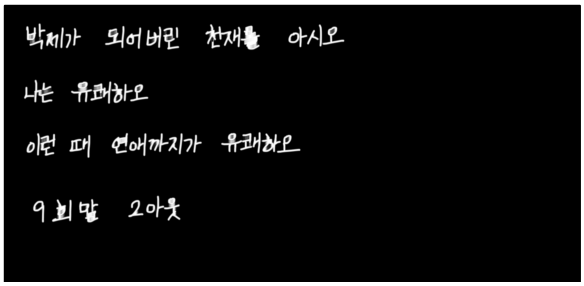
글을 흘겨쓴 것은 후에 classification 과정에서 문제가 될 수는 있겠지만 음절을 분리해내는 과정에서는 문제가 없었다. 예상했던 대로 의도적으로 모든 음절마다 넓게 띄어쓰기를 하지 않으면 음절별로 분리가 되지 않았다.

2)의 경우를 극복하기 위한 방법으로 생각한 첫번째 아이디어는 다음과 같다. 한글은 기본적으로 대략 정사각형 정도의 box 안에 들어가는 글자이다. 그러므로 대략 정사각형 정도의 a x a box를 찾게 된다면 1음절일 가능성이 높고 2a x a 의 크기 box를 찾는다면 2음절일 가능성이 높다. 그러므로 찾아낸 box의 크기를 통해 몇 음절인지 판단하고 box를 가로로 해당 음절만큼 나누어 각각의 음절로 나눌 수 있다. 이런 아이디어는 글을 쓰는 사람이 이상적으로 모든 글자를 정사각형의 규격으로 써야하고, 글자의 크기도 모두 일정하게 써야만 효과적일 것으로 생각한다. 즉, robust하지 않을 것으로 예상된다.

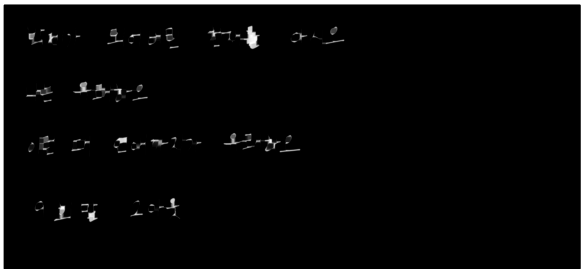
두번째 아이디어는 morphology 연산을 여러번 활용하여 세로축으로 closed morphology 필터를 적용하고, 그 후 erode morphology 필터를 가로축으로 적용하여 가로축선에서 음절이 서로 연결된 부분을 최소화 해보려는 방법이다.

2) 또박또박 + 평범하게 띄어쓰기

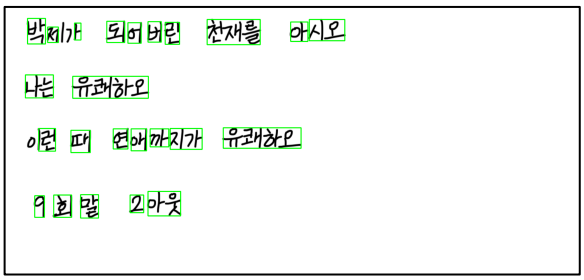
- Morphology (Closed) (kernel size = (9,3))



- Morphology (Erode) (kernel size = (3,7))



- findContour + Rectangular

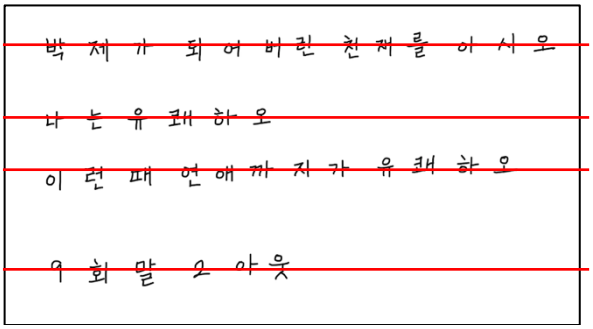


처음 시도했던 단순한 closed morphology 연산에 비해서 음절을 좀 더 분리하여 인식하긴 했으나, 그럼에도 유난히 붙혀쓴 음절의 경우 인식하지 못했다.

다음 단계를 테스트하기 위해 음절을 의도적으로 띄어쓴 1), 2)로 다음 단계를 진행하기로 하였다.

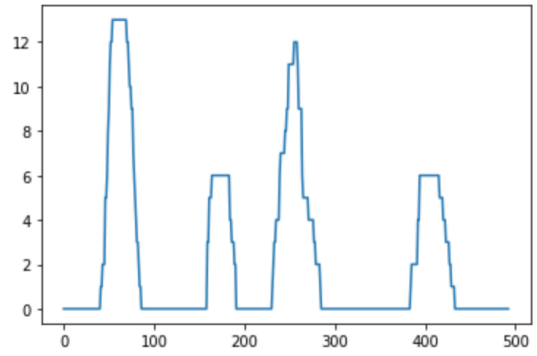
3. 정렬

37개의 음절별로 구분된 box를 찾아냈지만 사람이 글을 읽는 순서인, 위에서 아래로, 좌에서 우로 정렬 되어있지는 않다. 그러므로 글을 읽을 수 있는 순서대로 정렬을 해주어야 한다. 우선 세로 축을 searching 하여 하나의 가로 선 안에 들어오는 문자들 끼리 분리하고, 거기서 가로 축 좌표가 작은 순서대로 다시 정렬하려고 한다.



적용하려는 알고리즘은 다음과 같다. 세로축을 따라 픽셀을 이동하면서 가로 선 안에 몇개의 음절 박스가 들어오는지 모두 센다. 그리고 그 중 양의 극점이 있는 곳을 문장이 있는 선의 위치로 판단하고 그 선이 있는 곳의 박스들을 가로축 순으로 정렬한다.

```
image_size = (493, 889)
syllable_length = 37
```



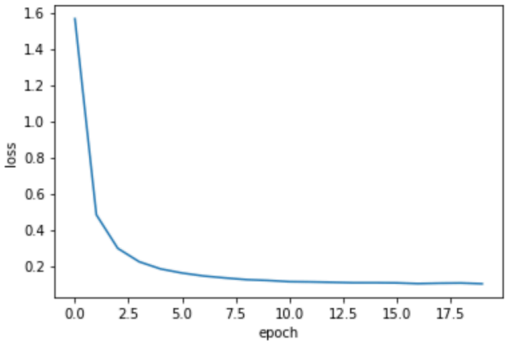
세로축으로 1픽셀 씩 훑어내려오며 가로축 수평선에 박스(음절)이 포함된 개수를 센 것이다. 이 중 local max값을 가지는 4개의 index를 선택하고 그 가로선을 중심으로 다시 정렬한다. 이 방법은 글을 이미지의 가로축과 평행하게 써야한다는 단점이 있다. 삐딱하게 글씨를 썼을 경우 행을 인식하지 못했다.

4. Resizing

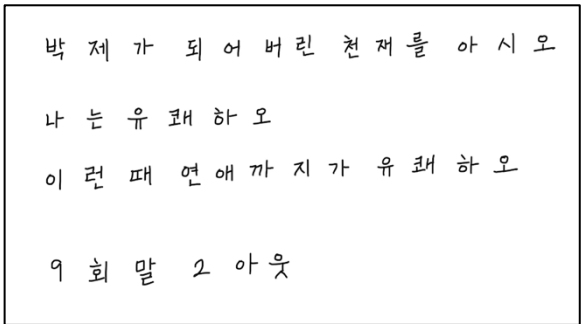
그렇게 정렬한 박스들을 network의 input size에 맞게 resizing 해주어야 한다. input size는 32x32로 정사각형의 규격을 가지고 있기 때문에 무작정 resizing 하다가는 원래의 이미지가 가지고 있는 가로, 세로 ratio를 손상시킬 수 있다. 가로 또는 세로 중 더 길이가 긴 쪽을 먼저 32 이하의 pixel로 맞추고 남는 공간은 zero-padding을 해주는 방식으로 비율의 문제를 해결했다. 적당히 상하좌우에 padding이 들어가야 인식률이 더 좋았다.

5. Neural Network

네트워크는 CNN을 사용하여 980개의 한글 이미지 + 10개의 숫자 이미지를 학습시켰다. 원본 데이터와 45도로 random rotation되고 1.3배로 random scaling된 이미지를 더해 202120개의 데이터를 학습시켰다. 20 epoch 간의 loss function의 변화는 아래와 같다.

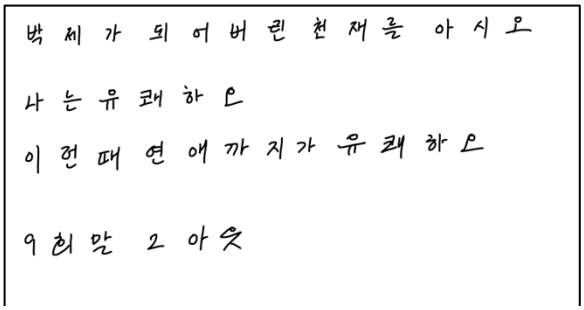


1)번 이미지를 음절 단위로 분리하여 네트워크에 넣은 결과물은 다음과 같다.



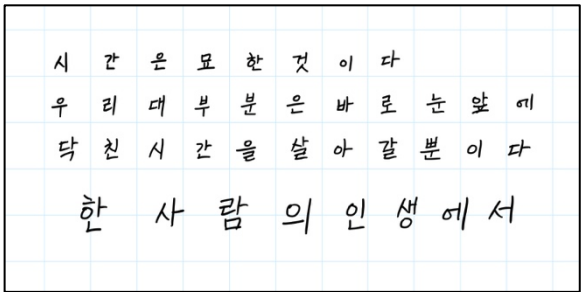
박제가되어버린천져클아시오
나는유쾌하노
이런따연애까지가유쾌하와
이회말2아웃

학습 데이터에 없었던 '를' 같은 문자는 비슷한 문자인 '클'로 인식하였다. 이렇듯 비슷하게 생긴 문자 사이에 오류가 많이 발생함을 알 수 있었다. 3)번 이미지는 아래와 같이 인식되었다.



박씨가9어버린천젓을아시와
나눈규쾌라2
이런때연에까지가육쾌하2
이히갈2아엿

그 외의 이미지에 대해서 한글 판독을 테스트해보았다. 다음의 이미지는 원고지에 글을 쓴 것을 생각하며 작성한 것이다. 글씨의 크기를 다르게 하며 테스트해보았다.



시간은묘한것이다
우리대부분은바로눈앞7
닥친시간을살아갈분이다
한사람의인생에서

전자신문
1우82던발행한
정분기술관련
일간신문
컴퓨터화면7
떠올려읽을4임게만든

6. 고찰 및 한계

음절 단위로 분리된 글자만 인식하도록 네트워크를 구성했기 때문에 인식 시키는 이미지가 의도적으로 매 음절마다 띄어쓰기된 형식이 아니면 인식할 수 없었다. 지금 상태로는 원고지에 쓴 글이 아닌 이상 인식할 수 없을 것이다. 궁극적으로 practical한 ocr 프로그램을 만들려면 음절 단위가 아닌 마디 별로 인식할 수 있는 방식을 고안해야할 것 같다.

네트워크에 넣기 전 과정을 모두 수동적인 filtering으로 진행했다. 그렇기에 사진의 상태에 따라 성능이 매번 달라졌고 적당한 사진을 기준으로 잡아 성능을 가장 끌어올리는 방향으로 만들 수 밖에 없었다. preprocessing에도 머신러닝 기술을 적용한다면 좀 더 robust하게 작동할 수 있지 않을까 생각한다. 특히 배경에서 글자를 분리해내는 방법을 사용하거나, 또는 깔끔하게 문자만 쓰인 학습 데이터에 인위적으로 배경을 합성하여 학습시키는 방법도 고려할 수 있을 것이다.

특히 한글은 조합문자이기 때문에 자주 사용하는 문자를 학습시키는 것은 한계가 있다. 문자를 무작정 많이 학습시키면 class가 증가되어 정확도 면에서 손해가 있을것이고 비슷하게 생긴 문자의 경우 잘못 인식하기도 하였다. 한글의 특성을 고려한다면 완성된 형태의 문자를 학습시키기보다 초성, 중성, 종성을 분리하여 인식하는게 더 정확할 것이다. 다만 그렇게까지 손글씨를 분리할 수 있는지에 대해서는 아직 방법을 모르겠다.

그리고 테스트 데이터를 내가 직접 작성한 손글씨로 했기 때문에 프로그램의 성능을 객관적으로 나타낼 수 있는 정확도 측정을 하지 못했다. 이 부분은 웹 크롤링을 통해 웹상에서 무작위 손글씨를 수집하여 테스트하는 방향으로 개선시킬 수 있을 것 같다.