Design Homework Report

Subject : 시각장애인을 위한 점자 to 음성 번역기

Lecture	인간컴퓨터 상호작용	
Lecturer	**Prof. Jin-Woo Jung**	
Student	**Student ID Number**	**Student Name**
	2018112003	이승현
	2018112033	이준탁
	2017112072	윤광식
	2018112000	최강희

Abstract

Subject	시각장애인을 위한 점자 to 음성 번역기
Keywords (2~7 words)	Canny Edge Detection, Geometric Transform, CNN, TTS

1. Research Purpose

본 주제는 OpenCV와 CNN 모델을 사용하여 점자를 음성으로 번역해주는 프로그램으로, 시각장애인들 중 점자를 읽지 못하여 점자책이나 안내 문구를 읽기 어려운 사람들을 도와주기 위해 고안되었다. 점자책의 사진을 찍어 input으로 입력하면, CNN 모델이 점자를 text로 변환하고, 그 결과를 TTS API를 이용하여 음성 파일로 출력해줌으로써 시각 장애인들이 점자책을 읽는 데 불편함을 덜어주는 것이 목적이다.

2. Research Method

모델 training 과정에서는 training image를 load하고, Canny Edge Detection을 통한 전처리를 거친 후, model의 정확도를 향상시키기 위해 geometric transform을 거쳐 데이터를 확장시킨다. 이후, 생성된 training data를 이용하여 CNN 모델을 생성한다.

사용자는 직접 촬영한 점자 이미지 및 글자수를 입력한다. 이후, 입력받은 이미지를 Canny Edge Detection을 통해 전처리하고, 입력받은 글자수에 맞춰 이미지를 divide한다.

이후, divide된 이미지를 CNN Model의 input으로 넣어 predict하고, 그 결과를 배열에 저장한다. 배열에 저장된 값을 TTS API를 이용하여 음성 파일로 출력한다.

3. Research Result

〈 input image 〉

〈 predict 결과 〉

```
predicted = predict(model, '1.jpg', (1, 10))
✓ 0.8s
```

```
Result: abcdefghij
```

〈 TTS 변환 〉

4. Discussion

본 프로젝트를 통해 사용자는 점자를 읽을 필요 없이 사진을 찍는 것만으로도 해당 문장을 소리로 들을 수 있다. 하지만, 점자의 정확한 위치를 알지 못해, 입력해야하는 영상이 비교적 까다롭다는 점과 글자 수를 직접 입력해야 한다는 단점이 존재한다. 이러한 부분은 추가적인 개선이 필요로 한다.

현재의 알고리즘은 높은 정확도를 보임과 동시에 음성으로 자동으로 변역되어 재생된다. 하지만 점자의 시작과 끝이 정확하게 잘려진 형태의 입력만을 받을 수 있다. 만약 시작과 끝 부분에 큰 여백이 존재할 경우, 잘못된 인식으로 인하여 전혀 다른 결과가 도출될 수 있다. 잘못된 점자 글자 수를 입력하는 경우 또한 잘못된 결과로 이어질 수 있다. 이 두가지 요소 모두 점자의 정확한 위치를 알지 못해 발생하는 문제점들이다.

이러한 단점들은 사용자가 실제로 사용하기에는 큰 무리가 있다고 판단되었다. 따라서 추후 개선사항으로 점자의 위치 및 글자 수를 자동으로 인식하고 인식하는 과정이 추가될 필요성이 있다.

1. Introduction

1.1 Background

(+includes needs and trend of related industry/study field)

- 시각장애인 점자 해독 여부를 확인해 보니 점자 문맹률의 심각성을 확인할 수 있었다.

시각장애인의 점자 해독 여부

(단위: %)

구분	남자	여자	전체
가능하다	8.3	4.9	6.9
배우는 중이다	2.8	2.5	2.7
불가능하다	88.9	92.7	90.4
계	100	100	100

출처: 2020년 장애인 실태조사(보건복지부)

[출처] 2020년 장애인 실태조사(보건복지부)

- 지식 정보화 시대에 정보 습득은 개인의 능력을 개발하는 데 필수 요소로서, 사회의 구성원 으로 문화를 누리며 주체적으로 살아가기 위해 갖추어야 할 경쟁력이다.
  정보 접근에 어려움 을 겪는 시각장애인에게 있어 점자는 문해 매체 그 이상의 의미로 자신감과 독립심을 갖게 하 며 능숙한 점자 사용 능력은 사회에서 직업을 갖는데 유리한 위치를 갖게 한다.

[출처] Ryles,1996

1.2 Previous study

(+includes related knowledge or theory for this project)

- Canny Edge Detection
  경계선 검출 방식에서 가장 많이 사용하는 알고리즘으로, 잘못된 경계선을 계산하는것을 방지하기 위해 개발된 알고리즘이다. 다섯 단계로 진행되는데, 먼저 가우시안 필터링을 통해 노이즈를 제거한다. 이후 그래디언트를 계산한다. 그리고 비최대 억제를 통해 에지가 두껍게 되는 현상을 방지한다. 그리고 더블 쓰레쉬 홀드 과정을 거친 후 히스테리시스 에지 트래킹을 한다.
- Geometric Transform
  사용자가 원하는대로 확대, 축소, 위치 변경, 회전, 왜곡 등을 하는 이미지 변환하는 것을

의미한다. 즉 영상을 구성하는 픽셀의 위치들을 재배치하고 새로운 픽셀 값을 생성하여 넣는 것(interpolation)을 포함한다. 본 프로젝트에서는 데이터의 확장을 위해 Geometric Transform를 사용한다.

- CNN

  데이터로부터 직접 학습하는 딥러닝의 신경망 아키텍처다. CNN은 영상에서 객체, 클래스, 범주 인식을 위한 패턴을 찾을 때 특히 유용하다. 또한, 오디오, 시계열 및 신호 데이터를 분류하는 데도 매우 효과적이다. 본 프로젝트에서 정의된 이미지와 점자를 통해 학습을 시키고 입력된 이미지를 분석하는데 쓴다.

- TTS

  TTS는 오디오 표현을 위해 텍스트 단위를 음성 단위로 변경해야하는 자연스러운 언어 모델링 프로세스이다. 텍스트 음성 변환은 디지털 텍스트에서 오디오 출력을 렌더링하여 읽을 수 없거나 다른 종류의 용도로 사용하는 사람들을 돕기위한 기술에서 일반적이다. 본 프로젝트에서는 시각장애인이 음성으로 들을 수 있도록 번역한 결과를 음성으로 출력한다.

1.3 Research purpose

- 시각장애인들 중 점자를 읽지 못하여 점자책이나 안내 문구를 읽기 어려운 사람들이 많다.
- 따라서, 점자책의 사진을 찍어 input으로 입력하면 이미지를 text로 변환하고, 다시 text를 음성 파일로 출력해준다.
- 이를 통해 시각장애인들이 점자책을 보다 정확하고 빠르게 읽을 수 있도록 한다.

1.4 Research plan

(+includes division of task for each member using Work Breakdown Structure)

(+includes timeline scheduling of each task such as Gantt Chart

진행단계	수행내용	상세수행내용	8주차	9주차	10주차	11주차	12주차	13주차	14주차
1. 분석	주제선정	아이디어 제시							
		사용할 기술 분석							
		팀원과의 의견 논의							
		사용 점자 조사							
2. 설계	개발환경 조성	알고리즘 제시							
		제한요소 제시							
		개발환경 조성 및 시험							
3. 구현	개발	메인 프로그램							
		CNN 알고리즘							
		TTS 적용							
		전처리 확인 -Canny							
4. 검토	프로그램 확인	성능 개선							
		오류 수정							
5. 발표	프로젝트 발표 준비	보고서 작성							
		발표자료 작성							
		데모 영상 촬영							

2. Constraints

2.1 Constraints for development process (개발환경에 대한 제한요소)

- programming language: Python 3.10.11
- run environment: jupyter-notebook
- execute environment: Colab(Cloud), VS code(Local)
- 노트북에 Nvdia GPU가 존재하지 않는 팀원이 있어 Colab을 활용함

2.2 Constraints for execution environment (실행환경에 대한 제한요소)

- 알파벳 점자 이미지여야 함
- 흰 배경의 점자 이미지여야 함
- 정면에서 촬영된 이미지여야 함
- 이미지의 좌, 우, 상, 하에 여백이 존재하지 않아야 함
- predict함수 호출 시, 인식할 알파벳의 개수를 입력해야 함

2.3 Constraints for performance (시스템 성능에 대한 제한요소)

- Model Training 시, Nvidia GPU가 존재하지 않으면 training 시간이 기하급수적으로 늘어날 수 있다.

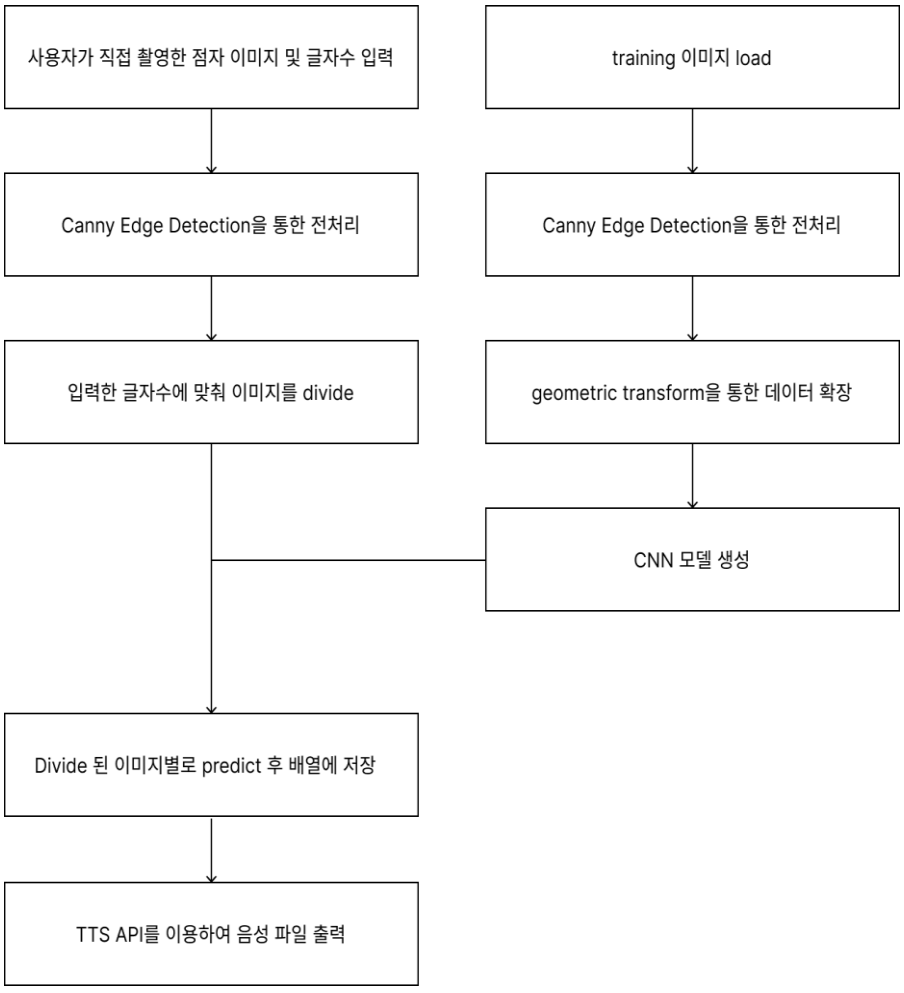
2.4 Additional constraints

- 이미지 size가 너무 작을 경우 제대로 인식되지 않음
- 빛 반사같은 환경적 요인으로 인하여 배경과 점자가 이미지상에서 구분이 불가능한 경우 인식되지 않음

3. Proposed System

3.1 Overall system description

(+includes block diagram)



3.2 Detailed explanation of each system module

〈 Model Training 과정 〉

1. 미리 준비한 a~z까지 알파벳에 대한 점자 이미지를 load한다.
2. 각 점자 이미지를 cvtColor 함수를 사용하여 grayscale 이미지로 변환한다.
3. 2에서 변환한 grayscale 이미지를 Canny()함수를 사용하여 edge를 detection한다. 이때, threshold값은 75, 150으로 한다.
4. geometric transform을 이용하여 모델의 정확도를 높이기 위해 데이터를 확장한다. 이때, 확장된 data의 rotation_range는 15도, shift_range는 width와 height 값의 15% 내로 설정한다. 또한, shear intensity는 반시계 방향으로 0.05도로 설정하며, zoom_range는 기존 이미지 크기의 99.9% ~ 100.1%로 설정한다. 데이터를 확장하는 과정에서는 각 transform을 범위 내의 임의의 값으로 변환한다.
5. input layer의 shape는 (36, 36, 3)으로 설정한다. Convolution Layer는 총 3개가 존재하며, 각 layer의 unit 개수는 64, 128, 256개로 한다. Activation function은 ReLU이며, Pooling 방식은 2x2 may pooling을 사용하였다. Dense Layer는 총 3개가 존재하며, 각 layer의 unit 개수는 256, 64, element의 개수로 한다. Activation function은 앞 두 개의 layer는 LeakyReLU이며, 마지막 layer는 Softmax function을 사용하여 모델을 만든다.
6. 모델 Compile과정에서의 Loss function은 Categorical crossentropy를, Optimizer는 Adam을 사용하였다.
7. 모델 fit 과정에서 epoch는 999회, Batch size는 128로 설정하였으며, 콜백으로는 모델을 저장하기 위한 model_ckpt, 모델의 개선이 없을 경우, Learning Rate를 조절해 모델의 개선을 유도하기 위한 reduce_lr, 모델을 더 이상 학습을 못할 경우(loss, metric등의 개선이 없을 경우), 학습 도중 미리 학습을 종료하기 위한 early_stop을 사용하였다.

〈 사용자 input 처리 과정 〉

1. 사용자가 직접 촬영한 이미지를 load한다.
2. 각 점자 이미지를 cvtColor 함수를 사용하여 grayscale 이미지로 변환한다.
3. 2에서 변환한 grayscale 이미지를 Canny()함수를 사용하여 edge를 detection한다. 이때, threshold값은 75, 150으로 한다.
4. 사용자가 지정한 알파벳의 수만큼 이미지를 divide하여 저장한다.

5. divide된 각 이미지를 model로 불러와 predict한다.
6. predict한 각 결과를 result 배열에 append한다.
7. result배열에 담긴 text를 TTS API를 이용하여 wav audio file로 변환시켜 저장한다.

3.3 Implementation method and tool

(+includes version info. of OpenCV and what OpenCV funtions are mainly used)

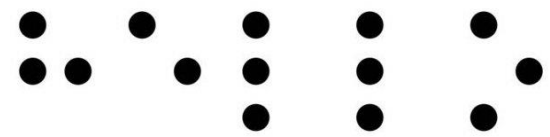
OpenCV version: 4.7.0
- cv2.cvtColor()
- cv2.Canny()

4. Experimental Result

4.1 Experimental environment

(+includes the detailed explanation of experimental environment)

< 이미지 a >
원본 글자가 hello인 점자 영상으로서 총 5개의 점자가 존재하는 영상



< 이미지 b >
원본 글자가 abcdefghij인 점자 영상으로서 총 10개의 점자가 존재하는 영상

< 이미지 c >
원본 글자가 klmnopqrst인 점자 영상으로서 총 10개의 점자가 존재하는 영상

< 이미지 d >
원본 글자가 uvwxyz인 점자 영상으로서 총 6개의 점자가 존재하는 영상

〈 이미지 e 〉
원본 문장이 This was a triumph인 점자 영상으로서 총 18개의 점자가 존재하는 영상

4.2 Performance measure

(+includes the definition of your performance measure used to prove effectiveness)

이미지 a를 사용하였을 때의 결과: ‘hello’
원본 문장: ‘hello’
정확도: 100%
실행시간: 0.3초

```
Result: hello
```

이미지 b를 사용하였을 때의 결과: ‘abcdefghij’
원본 문장: ‘abcdefghij’
정확도: 100
실행시간: 0.8초

```
Result: abcdefghij
```

이미지 c를 사용하였을 때의 결과: ‘klmnopqrst’
원본 문장: ‘klmnopqrst’
정확도: 100%
실행시간: 1.3초

```
Result: klmnopqrst
```

이미지 d를 사용하였을 때의 결과: ‘uvwxyz’
원본 문장: ‘uvwxyz;
정확도: 100%
실행시간: 0.7초

```
Result: uvwxyz
```

이미지 e를 사용하였을 때의 결과: ‘this wcs a triumph’
원본 문장: ‘this was a triumph’
정확도: 94.4%
실행시간: 2.2초

```
Result: this wcs a triumph
```

〈 기존 모델과의 비교 〉
이미지 b를 사용하였을 때, original model의 결과

```
Result: yyyyyyyyyy
```

원복 문장: ‘abcdefghij’
정확도: 0%
실행시간: 0.8초

이미지 b를 사용하였을 때, edge detection을 적용한 model의 결과

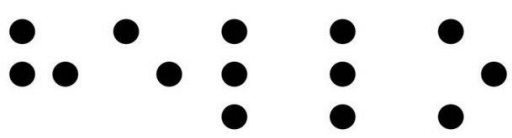
```
Result: abcdefghij
```

원복 문장: ‘abcdefghij’
정확도: 100%
실행시간: 0.8초

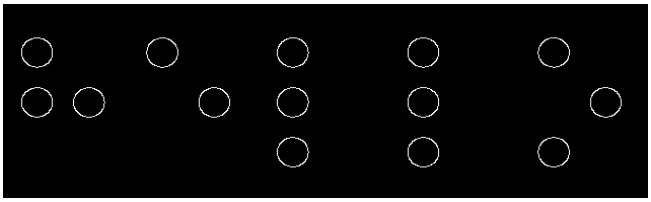
개선사항: 기존 모델은 흰 배경과 흰 점자인 경우 detection을 하지 못하지만, edge detection을 사용한 모델의 경우, 흰 배경과 흰 점자인 경우에도 정확히 detection을 해낼 수 있었다.

4.3 Experimental results

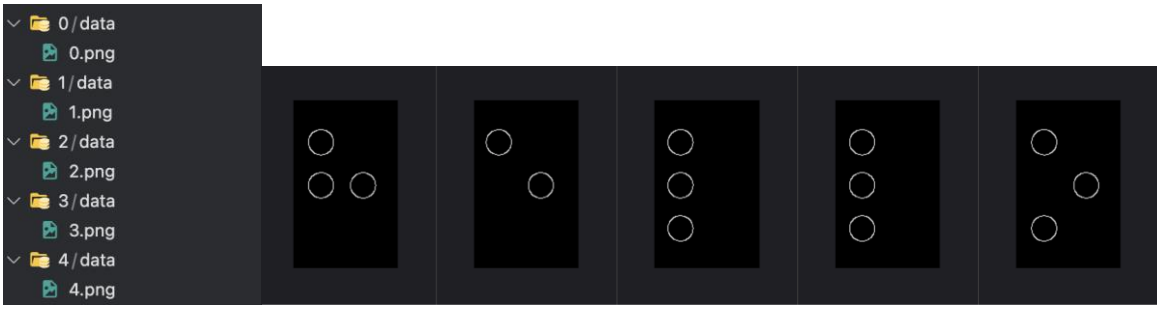
이미지 a를 기준으로 실행하였을 때의 결과이다



영상에 Canny Edge Detection을 적용한 결과이다



영상을 입력한 글자수에 맞춰 divide한 결과이다



각 글자는 아래와 같이 각 알파벳에 대한 확률을 구한 결과이다

```
[1.1183471e-12 4.7149243e-07 8.1734825e-08 1.6522462e-07 2.0584408e-05
 2.8632297e-07 6.0654193e-06 9.9996448e-01 2.4760607e-10 6.3411389e-06
 2.4858650e-13 1.3123907e-13 1.5253669e-10 1.3646353e-10 2.3418278e-11
 3.2529497e-13 2.1375799e-11 4.7519461e-07 7.3747193e-13 1.8952756e-10
 6.3593233e-09 1.1755392e-06 1.2213620e-11 2.0515401e-08 2.9433040e-12
 2.8866507e-09 9.7640274e-10]
```

모든 글자에 대하여 위와 같은 결과를 수행하고 각 글자에서 가장 높은 확률을 가지는 글자를 추출한 결과이다

```
Found 1 images belonging to 1 classes.
1/1 [==============================] - 0s 73ms/step
Best prediction: 0.9999645 h
Found 1 images belonging to 1 classes.
1/1 [==============================] - 0s 88ms/step
Best prediction: 0.99748766 e
Found 1 images belonging to 1 classes.
1/1 [==============================] - 0s 91ms/step
Best prediction: 0.55246913 l
Found 1 images belonging to 1 classes.
1/1 [==============================] - 0s 82ms/step
Best prediction: 0.8769353 l
Found 1 images belonging to 1 classes.
1/1 [==============================] - 0s 79ms/step
Best prediction: 0.90509474 o

Result: hello
```

위 결과를 음성으로 변환한 결과이다

4.4 Discussion

(1) 이미지 a, b, c, d

원본 text와 동일한 글자가 출력되었으며 이를 기반으로 TTS로 정상적으로 변환되었다.

(2) 이미지 e

원본 text와 비슷한 결과가 출력되었으나, 완벽하게 동일한 결과가 출력되지는 못하였다. 이를 기반으로 TTS로 변환한 결과 문장의 문맥을 파악하는데 다소 어려움이 있다고 판단되었다.

Best Case

점자 이미지가 알파벳 점자로 구성된 이미지여야 하며, 배경이 흰 색이여야 하고, 이미지의 좌, 우, 상, 하에 여백의 존재하지 않아야 한다. 또한, 이미지의 size가 점자가 인식되지 않을 정도로 작지 않아야 하며, 조명이 존재하는 환경에서 정면으로 촬영한 점자 이미지일 경우 가장 좋은 accuracy를 보였다.

5. Conclusion

canny edge detection으로 점자를 인식하며, 한 학기 동안 학습한 OpenCV의 기술들을 활용하여 원하는 정보를 추출하고, 얻은 정보를 활용하여 점자 해석이라는 목적에 맞는 기능을 수행하는 프로그램을 만들어 보았다.

프로젝트를 진행하면서 어떻게 이미지를 처리하고 분석해야 하는지 다양한 방법으로 접근해 보았고, 이 결과를 HCI 관점에서 해석해 보는 기회가 되었다. 프로그램을 개발하고 실행시켜 보며 이미지 처리의 방법과 어려움, 제한점들에 대해 느낄 수 있었고, 최적의 방법을 선택하여 초기에 계획한 목적에 맞게 프로그램을 발전시키는 과정을 경험하였다.

본프로젝트를 통해 점자의 이미지를 촬영해 입력한다면 점자를 해석할 수 없는 사람들이 점자를 해석할 수 있게 된다. 다만 시작과 끝 부분에 큰 여백이 존재할 경우, 잘못된 인식으로 인하여 전혀 다른 결과가 도출될 수 있으며 잘못된 점자 글자 수를 입력하는 경우 또한 잘못된 결과로 이어질 수 있다.추후 개선사항으로 점자의 위치 및 글자 수를 자동으로 인식하고 인식하는 과정의 추가가 필요하다.

HCI 관점

HCI의 사용성을 기준으로 이번 프로젝트를 평가해 보았다. 넓은 의미의 사용성으로 보았을 때, 사용자가 프로젝트의 프로그램을 사용할 경우를 생각해보면, 점자를 모르는 시각장애인 사용자에게 자신이 알고자 하는 점자를 해석하여 음성으로 제공함으로써, 사용자에게 점자 정보를 확인 시켜주는 만족감을 줄 수 있다. 해당 시스템은 정확한 번역을 해야하므로 속도보다 정확성의 초점을 맞춘 시스템이지만 대부분의 실행이 수초 이내로 실행되기 때문에 효율성에서도 큰 문제가 되지 않는다. 하지만 글자 수 많아진다면 실행시간 또한 기하급수적으로 늘어나는 단점이 있다. 유연성 관점에서는 다양한 배경과 굴곡이나 빛반사가 있을경우 인식이 어렵다는 점이 있기 때문에 유연성 면에서 아쉬운 프로그램이라 할 수 있다.
유용성의 관점에서는 OpenCV를 통해서 전처리하여 이미지를 변환하고 CNN을 통해 학습된 이미지와 대조하여 인식하고, TTS를 통해 음성으로 변환하여 사용자에게 제공하므로 사용자의 요구사항을 적절히 해결해준다. 추후 한글 또한 인식하는 등의 업데이트를 통해 더 다양한 번역이 가능하다는 점에서 본 프로젝트의 발전성에 대해 기대를 가질 수 있다.

Reference

[1] Braille Classifier (Keras)
https://www.kaggle.com/code/kwisatzhaderach/braille-classifier-keras/notebook

[2] Braille Character Dataset
https://www.kaggle.com/datasets/shanks0465/braille-character-dataset

[3] '시각장애인 10명중 9명은 점자 문맹' - 일간경기
http://www.1gan.co.kr/news/articleView.html?idxno=219771

[4]
Ryles, Ruby. "The Impact of Braille Reading Skills on Employment, Income, Education, and Reading Habits". Journal of Visual Impairment & Blindness 90, 1996, pp. 219 - 226.

[5]
Seung Hyun Kim, Seok Hyen Lee, Jong Chang Lee, Jung Sick Hwang, and Su Min Kim. "A Smart Braille Translation System". Dept. Electronics Engineering, Korea Polytechnic Univ.