REPORT

딥러닝을 이용한 한글 필기 인식

국민대학교	
학 부	컴퓨터공학부
과 목	비주얼 컴퓨팅 최신기술
담 당 교 수	김준호 교수님
제 출 일	2017년 5월 19일
학 번	20143390
이 름	김선영

1. Project 목표

텐서플로우(Tensorflow)를 기반으로 합성곱 신경망을 구성하고 학습 데이터로 학습시켜 필기 인식기를 구현하는 것을 목표로 한다. '가/나/다/… /파/하'의 총 14자에 대한 필기 데이터를 활용하여 총 14글자를 인식하는 필기 인식기를 구현하고 학습된 필기 인식기로 테스트 데이터를 인식하여 성능을 테스트한다

본 프로젝트의 최종 목표는 학습된 필기 인식기로 테스트 데이터를 인식하였을 때 90% 이상의 인식률을 달성하는 것이다. 이를 달성하기 위해 여러 가지 방법을 시도하고 구현한 인식기의 성능에 대하여 분석해본다.

2. Project 수행 내용

1) Data

(1) Training data

28*28 사이즈의 정제된 학습 데이터와 테스트 데이터를 사용한다.

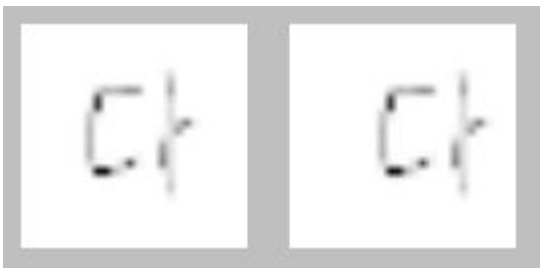
- resized_kalph_train.hf와 resized_kalph_test.hf 사용

(2) Data augmentation

인식률을 높이기 위해 주어지 training data에 영상 처리를 하여 새로운 영상들을 생성하였다.

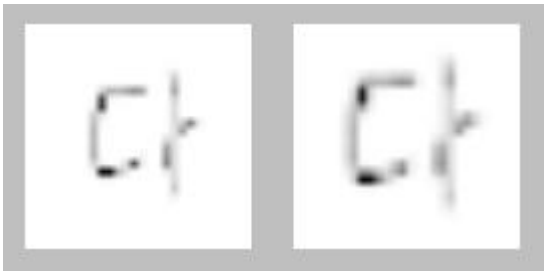
① 영상 변환

입력 영상에 회전, 이동, 크기 변환을 적용해 새로운 영상들을 생성하였다.



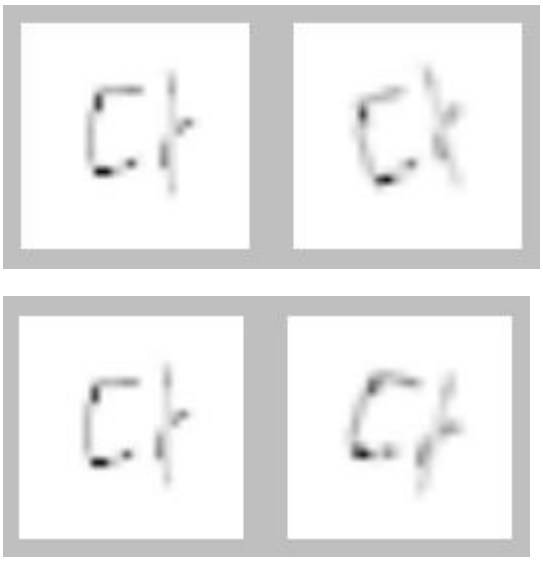
```
# translate
translatedd_images = np.zeros((19600,28,28), dtype=np.float32)
for i in range(19600):
    M = np.float32([[1,0,3],[0,1,random.randint(-3,3)]])
    translatedd_images[i] = cv.warpAffine(images[i], M,(cols, rows))
```

오른손 잡이가 많기 때문에 입력된 데이터가 중앙이 아닌 왼쪽에 치우쳐 있을 것이라고 예상하였다. 테스트 데이터에 오른쪽에 치우쳐진 데이터가 있는 경우 인식률이 떨어질 것이라고 생각하여 전체적 오른쪽으로 이동시켰고 -3에서 3사이의 임의의 값으로 위, 아래 이동시켰다



```
# magnify 1.2
magnified_images = np.zeros((19600,28,28), dtype=np.float32)
for i in range(19600):
    M = cv.getRotationMatrix2D((cols/2, rows/2), 0, 1.2)
    magnified_images[i] = cv.warpAffine(images[i], M,(cols, rows))
```

글씨를 수집할 때 학생들은 많은 양의 글씨를 썼어야 했으므로 글씨를 작게 썼을 것이라 예상하고 원본 글씨 크기의 1.2배로 확대하였다.

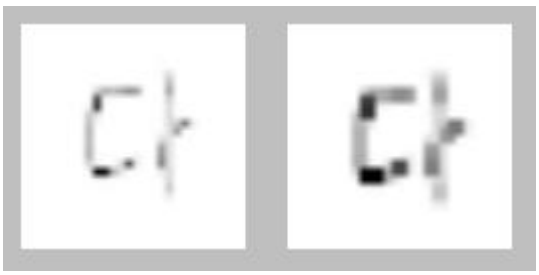


```
# plus 15 dgree rotate
rotscaled_p15_imgs = np.zeros((19600,28,28), dtype=np.float32)
for i in range(19600):
    M = cv.getRotationMatrix2D((cols/2, rows/2), 15, 1)
    rotscaled_p15_imgs[i] = cv.warpAffine(images[i], M,(cols, rows))

# minus 15 dgree rotate
rotscaled_m15_imgs = np.zeros((19600,28,28), dtype=np.float32)
for i in range(19600):
    # M = cv.getRotationMatrix2D((cols/2, rows/2), -15, 1)
    rotscaled_m15_imgs[i] = cv.warpAffine(images[i], M,(cols, rows))
```

오른손 잡이와 왼손잡이의 글씨 쓰는 방향을 고려하여 예상되는 각도로 회전시켰다.

② 모폴로지 연산



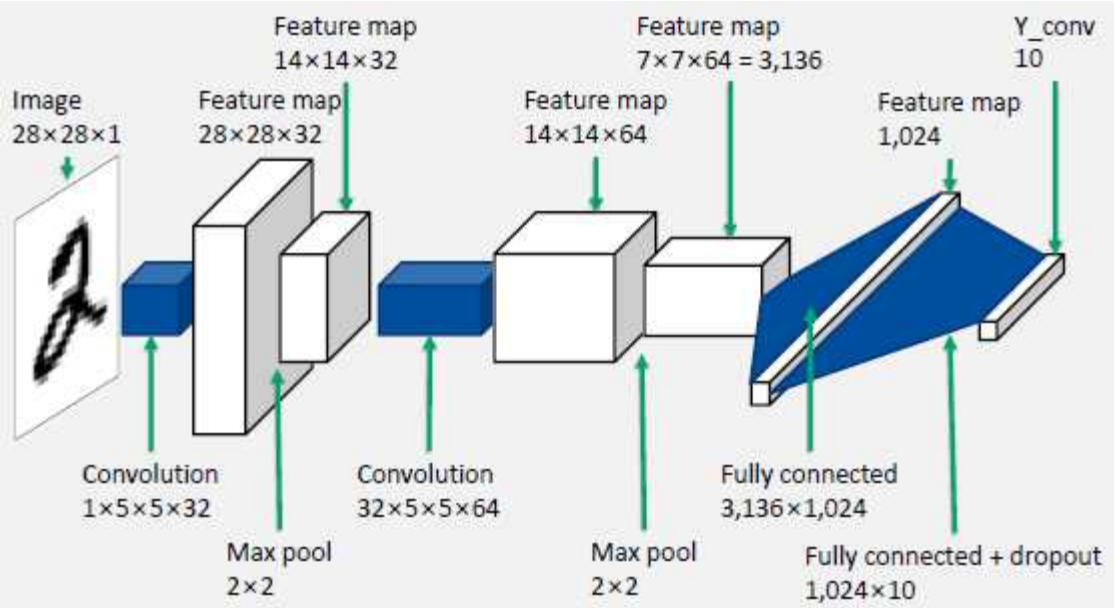
```
# dilation
dilated_imgs = np.zeros((19600,28,28), dtype=np.float32)
index=0
for img in images_to_dilate:
    temp_img = img
    dilated_imgs[index] = cv.dilate(temp_img,
                                    kernel=cv.getStructuringElement(cv.MORPH_RECT, (2, 2)), iterations=1)

    index += 1
```

모폴로지 연산에는 침식, 팽창, 닫힘, 열림 연산이 있다. 이 중에서 팽창 연산을 사용하였다. 각 픽셀에 필터를 적용하여 대상을 확장하면 경계가 부드러워지고 글씨 두께가 두꺼워지는 것을 확인하여 여러 가지 도구(샤프, 볼펜 등)로 작성한 데이터를 인식할 수 있을 것이라고 예상했다.

위와 같은 연산을 통하여 총 117600개의 데이터를 만들었다.

2) Network model & architecture and Learning method




```
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_input, logits=y_conv))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_input,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

Deeper MNIST에서 Network model과 architecture, learning method를 가져와 사용하였다.

3. Project 결과 및 분석

```
step 29300 training accuracy 0.98
step 29400 training accuracy 1.0
step 29500 training accuracy 1.0
step 29600 training accuracy 1.0
step 29700 training accuracy 0.96
step 29800 training accuracy 0.94
step 29900 training accuracy 1.0
test accuracy 0.840816

Process finished with exit code 0
```

배치 크기를 50으로 잡고 30000번 학습한 결과 위와 같은 결과가 나왔다.

인식기의 인식률을 높이기 위해서 여러 방법이 있지만 데이터 증강과 전이 학습이 기초가 된다. 구현한 인식기에서 인식률을 높이기 위해서는 기존에 있는 데이터에서 잡음 추가, 영상 변환, 모폴로지 연산을 통해 더 많이 데이터를 생성하거나 이미 기존에 있는 데이터로는 인식할 수 없는 데이터를 판별하기 위해서 더 많은 사람들의 글씨체를 수집할 필요가 있다.

또한 MNSIT가 0~9의 글씨 인식률이 높기 때문에 가/나/다/…/파/하와 같은 글씨도 잘 학습

할 것이라 생각하여 network mode과 architecture, learning method를 가져다 사용하였지만 전이 학습을 사용하면 이미 특징 정보를 추출하는 레이어가 잘 학습되어 있기 때문에 이를 잘 결합하여 최종 정보를 추출하는 레이어들만 학습시키면 된다. 때문에 적은 수의 데이터로 학습할 수 있고 학습시간을 단축 시킬 수 있으므로 더 효율적인 인식기를 구현할 수 있을 것이다.

같은 MNIST의 network mode과 architecture, learning method를 사용하더라도 레이어를 더 추가, filter크기 변경, activation map의 개수 조절, dropout할 노드 개수 조절 등 여러 변수를 잘 조절했으면 더 좋은 인식기를 구현할 수 있을 것이다.