

[과제연구] 최종 TTS 코드

```
import tensorflow as tf
import os
import cv2
import numpy as np
import re
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from numpy import array
from gtts import gTTS

##cnn이미지 변환은 '모두의 머신러닝'을 강의하신 김성훈 교수님의 코드를 참고함 (https://goo.gl/CjHapo)
###학습 데이터 가공
#학습데이터 가져올 위치
TRAIN_DIR = 'D:/deeplearning/programming/fingerlanguage/traindata'
train_folder_list = array(os.listdir(TRAIN_DIR))

#데이터값들은 input 에, 정답은 label에
train_input = []
train_label = []

output_nodes = 31

label_encoder = LabelEncoder()
#폴더들을 인식, 그것들의 이름을 순차적으로 숫자형 데이터로 전환하기
integer_encoded = label_encoder.fit_transform(train_folder_list)

onehot_encoder = OneHotEncoder(sparse=False)
#onehotencoder 쓰기 위해서 integer_encoded 를 2차원배열으로 전환한다.
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
#해당 값을 가지는 index만 값을 1로 하고 나머지는 다 0을 가지는 배열으로 전환한다.
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)

#img를 변환시키는 과정
for index in range(len(train_folder_list)):
    path = os.path.join(TRAIN_DIR, train_folder_list[index])
    path = path + '/'
    #경로에 / 를 추가한다.
    img_list = os.listdir(path)
    for img in img_list:
```

[과제연구] 최종 TTS 코드

```
# 해당 경로 내 폴더들에 대해서 이미지를 하나씩 읽어준다.
img_path = os.path.join(path, img)
#print(img_path)
#이미지를 읽어온다.
img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
#print(img)
#이미지를 numpy형 배열으로 변환해서 train_input에 넣는다.
train_input.append([np.array(img)])
#라벨 (정답) 도 마찬가지로 train_label에 넣는다.
train_label.append([np.array(onehot_encoded[index])])

#input 데이터를 784개 (28x28) 가 있는 배열으로 변환
#-1은 데이터 개수 정확히 모를 때 사용하는 거
#label 데이터도 마찬가지로
train_input = np.reshape(train_input, (-1, 784))
train_label = np.reshape(train_label, (-1, output_nodes))
#최종 변환
train_input = np.array(train_input).astype(np.float32)
train_label = np.array(train_label).astype(np.float32)
#print(train_label)
#이거를 npy 파일로 저장 (실제로 해당폴더에 저장됨)

###테스트 데이터 가공
#학습 데이터와 같은 원리 (설명 생략)
TEST_DIR = 'D:/deeplearning/programming/fingerlanguage/testdata'
test_folder_list = array(os.listdir(TEST_DIR))

test_input = []
test_label = []

label_encoder2 = LabelEncoder()
integer_encoded2 = label_encoder.fit_transform(test_folder_list)

onehot_encoder2 = OneHotEncoder(sparse=False)
integer_encoded2 = integer_encoded2.reshape(len(integer_encoded2), 1)
onehot_encoded2 = onehot_encoder2.fit_transform(integer_encoded2)

path = ""
for index in range(len(test_folder_list)):
```

[과제연구] 최종 TTS 코드

```
path = os.path.join(TEST_DIR, test_folder_list[index])
path = path + '/'
img_list = os.listdir(path)
for img in img_list:
    img_path = os.path.join(path, img)
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    test_input.append([np.array(img)])
    test_label.append([np.array(onehot_encoded2[index])])

test_input = np.reshape(test_input, (-1, 784))
test_label = np.reshape(test_label, (-1, output_nodes))
test_input = np.array(test_input).astype(np.float32)
test_label = np.array(test_label).astype(np.float32)

#실제학습 시작
X = tf.placeholder(tf.float32, [None, 784])
X_img = tf.reshape(X, [-1, 28, 28, 1])
Y = tf.placeholder(tf.float32, [None, output_nodes]) #출력값
#추후 드롭아웃 시 사용할 변수 keep_prob
keep_prob = tf.placeholder(tf.float32)

#학습률 설정
learning_rate = 0.001

#컨볼루션 계층 생성하기(3x3) (내장함수사용)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01)) #3x3 크기 커널 가진 계층, 오른쪽/아래쪽으로 1칸씩 움직이며, 32개 커널
#랜덤으로 가중치를 넣는다.
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME') #padding SAME : 이미지 외각 (테두리) 까지도 조금 더 명확히 평가가능
L1 = tf.nn.relu(L1) #활성화함수로 relu 적용

#풀링 계층 (내장함수사용) (맥스풀링사용)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

#두 번째 계층 역시 같은 방법으로
W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01)) #L1 풀링 맵을 기본으로 (32개) -> #이번에는 3x3 의 커널 64개로 구성됨
#랜덤으로 가중치를 넣는다.
```

[과제연구] 최종 TTS 코드

```
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

#10개의 분류를 만들어내는 계층
W3 = tf.Variable(tf.random_normal([7*7*64, 256], stddev = 0.01))
L3 = tf.reshape(L2, [-1, 7*7*64]) #직전 풀링 계층 크기가 7x7x64, 이를 1차원으로 만든다.
L3 = tf.matmul(L3, W3)
L3 = tf.nn.relu(L3) #활성화함수 적용
L3 = tf.nn.dropout(L3, keep_prob) #과적합 방지
#편향은 일단제거
#b = tf.Variable(tf.random_normal([4]))

W4 = tf.Variable(tf.random_normal([256, output_nodes], stddev = 0.01))
model = tf.matmul(L3, W4) # + b

cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=model, labels=Y))
#평균 엔트로피 함수
optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)
#optimizer = tf.train.RMSPropOptimizer(0.001, 0.9).minimize(cost) 이거로도 가능..

init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # 초기화 실행시키기

batch_size = 100 # 미니배치 사용 (100개)
total_batch = int(len(train_input) / batch_size)
all_epoch = 55

print('학습 시작!')
for epoch in range(all_epoch):
    total_cost = 0
    for i in range(total_batch):
        start = ((i+1) * batch_size) - batch_size
        end = ((i+1) * batch_size) # 학습할 데이터를 배치 크기만큼 가져온 뒤,
        batch_xs = train_input[start:end]
        batch_ys = train_label[start:end]
        #batch_xs, batch_ys 정의해주기
        # 입력값 = batch_xs, 출력값 = batch_ys
```

[과제연구] 최종 TTS 코드

```
_ , cost_val = sess.run([optimizer, cost], feed_dict={X: batch_xs, Y: batch_ys, keep_prob : 0.57})
# 최적화+손실값 저장, 입력값X와 평가할 때 쓸 실제값 Y를 넣어준다. 과적합방지 비율 = 57%
total_cost = total_cost + cost_val
#오차값 구하기

print('Epoch:', '%04d' % (epoch + 1), 'Avg. cost = ', '{:3f}'.format(total_cost / total_batch))
print('학습 완료!')
```

#결과확인

```
is_correct = tf.equal(tf.argmax(model, 1), tf.argmax(Y, 1)) #1번 인덱스의 차원 중 최댓값의 인덱스를 뽑아낸다! equal
로 같은지 비교!
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32)) #cast를 통해서 0과 1로 변환
print('정확도:', sess.run(accuracy, feed_dict = {X: test_input, Y: test_label, keep_prob : 1}))
result = sess.run(model, feed_dict = {X: test_input, Y: test_label, keep_prob : 1})
#print(test_input)
#각각 model 에서 얻은 결론값에 해당하는 한글 조성을 출력한다.
#현재 index 0 = ㄱ 1 = ㄴ 2 = ㄷ 3 = ㄹ 4 = ㅁ 5 = ㅂ 6 = ㅅ 7 = ㅇ 8 = ㅈ 9 = ㅊ 10 = ㅋ
11 = ㅌ 12 = ㅍ 13 = ㅎ 14 = ㅊ 15 = ㅊ 16 = ㅊ 17 = ㅊ 18 = ㅊ
19 = ㅊ 20 = ㅊ 21 = ㅊ 22 = ㅊ 23 = ㅊ 24 = ㅊ 25 = ㅊ 26 = ㅊ 27 = ㅊ 28 = ㅊ
29 = ㅊ 30 = ㅊ 31 = ㅊ 32 = ㅊ 33 = ㅊ 34 = ㅊ 35 = ㅊ 36 = ㅊ
```

```
for i in range(len(result)):
    #print(result[i])
    maxi = -100
    ind = 0
    for j in range(0, output_nodes):
        if result[i][j] > maxi:
            maxi = result[i][j]
            ind = j
    print(char[ind])
```

###TTS데이터

#학습 데이터와 같은 원리

```
TTS_DIR = 'D:/deeplearning/programming/fingerlanguage/ttss'
```

```
tts_input = []
img_list = os.listdir(TTS_DIR)
TTS_DIR = TTS_DIR + '/'
#경로에 / 를 추가한다.
```

[과제연구] 최종 TTS 코드

```
img_list = os.listdir(TTS_DIR)
for img in img_list:
    img_path = os.path.join(TTS_DIR, img)
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    tts_input.append([np.array(img)])
tts_input = np.reshape(tts_input, (-1, 784))
tts_input = np.array(tts_input).astype(np.float32)

han_result = []
#결과출력
#print(tts_input)
ttsresult = sess.run(model, feed_dict = {X: tts_input, keep_prob : 1})

#각각 model 에서 얻은 결론값에 해당하는 한글 초성을 출력한다.
for i in range(len(ttsresult)):
    maxi = -1000
    ind = 0
    for j in range(0, len(ttsresult[i])):
        if ttsresult[i][j] > maxi:
            maxi = ttsresult[i][j]
            ind = j
    ##인위적 변환
    #완료
    if ind == 6 :
        ind = ind - 5
        print(char[ind])
        han_result.append(char[ind])
        continue
    #완료
    if 25 <= ind and ind <= 30 :
        ind = ind - 22
        print(char[ind])
        han_result.append(char[ind])
        continue
    #완료
    if 1 <= ind and ind <= 5:
        ind = ind + 8
        print(char[ind])
        han_result.append(char[ind])
```

[과제연구] 최종 TTS 코드

continue

#완료

```
if 7 <= ind and ind <= 16:
```

$$\text{ind} = \text{ind} + 7$$

```
print(char[ind])
```

```
han_result.append(char[ind])
```

continue

#완료

```
if ind == 17 :
```

```
ind = ind - 15
```

```
print(char[ind])
```

```
han_result.append(char[ind])
```

continue

#완료

```
if 18 <= ind and ind <= 19:
```

```
ind = ind + 6
```

```
print(char[ind])
```

```
han_result.append(char[ind])
```

continue

#완료

```
if 20 <= ind and ind <= 24:
```

```
ind = ind + 6
```

```
print(char[ind])
```

```
han_result.append( char[ind])
```

continue

```
print(char[ind])
```

```
han_result.append(char[ind])
```

```
print(han_result)
```

[illegible]

[과제연구] 최종 TTS 코드

[illegible]

[과제연구] 최종 TTS 코드

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
b = han_result[cntind - 1]
```

[과제연구] 최종 TTS 코드

```
c = han_result[cntind]
elif han_result[cntind] in Vowel:
    a = han_result[cntind - 1]
    b = han_result[cntind]
    c = ""
# 특수한 경우는 따로 처리
elif len(han_result) == 2:
    a = han_result[0]
    b = han_result[1]
    c = ""
    flag = True
elif len(han_result) == 3:
    a = han_result[0]
    b = han_result[1]
    c = han_result[2]
    flag = True
# 1. 모음-자음-자음 순서일 때 : 자음에서 바꾸기
elif cntind >= 1 and han_result[cntind - 1] in Vowel and han_result[cntind] in Z and han_result[cntind + 1] in
Z:
    a = han_result[cntind - 2]
    b = han_result[cntind - 1]
    c = han_result[cntind]
# 2. 모음-자음-모음순일때 : 모음에서 바꾸기
elif cntind >= 1 and han_result[cntind - 1] in Vowel and han_result[cntind] in Z and han_result[
cntind + 1] in Vowel:
    a = han_result[cntind - 2]
    b = han_result[cntind - 1]
    c = ""
    # cntind = cntind + 1

# 합성하기.
if a != "" and b != "":
    letter = [a, b, c]
    # 입력한 글자의 초성보다 앞에 있는 초성 수 세기
    # 한 초성당 588개의 글자 존재함. 따라서 그 초성 수에 588을 곱해 cnt에 더한다.
    for i in range(len(Consonant)):
        if letter[0] in Consonant[i]:

            for x in range(0, i):
```

[과제연구] 최종 TTS 코드

```
cnt += Consonant_number[x] * 588

for j in range(len(Consonant[i])):
    if letter[0] == Consonant[i][j]:
        cnt += j * 588

# 입력한 글자의 중성보다 앞에 있는 중성 수 세기
# 초성이 결정된 상태에서 한 중성 당 28개의 글자 존재함. 따라서 그 중성 수에 28을 곱해 cnt에 더
한다.

for k in range(len(Vowel)):
    if letter[1] == Vowel[k]:
        cnt += (k) * 28

# 입력한 글자의 중성보다 앞에 있는 중성 수 세기
# 이 수까지 cnt에 더하면, 입력한 글자보다 앞에 있는 모든 글자의 수가 cnt가 된다.

for l in range(0, len(Consonants)):
    if letter[2] in Consonants[l]:
        for m in range(0, l):
            cnt += Consonants_number[m]
        for n in range(0, len(Consonants[l])):
            if letter[2] == Consonants[l][n]:
                cnt += n

# 배열에 추가하기
char_result.append(M[int(cnt)])

cntind = cntind + 1
# print(char_result)

# 하나의 단어로 합치기
total_char = ""
for i in range(len(char_result)):
    total_char = total_char + char_result[i]
print(total_char)

# (최종)TTS변환
tts_file = gTTS(text=total_char, lang='ko')
tts_file.save("FinalTTSFile.mp3")
print('저장 완료!')
```