ISSN: 2508-7894 © 2017 KAIA. http://www.kjai.or.kr
Doi: http://dx.doi.org/10.24225/kjai.2017.5.2.43

A Study on the Emoticon Extraction based on Facial Expression Recognition using Deep Learning Technique

딥 러닝 기술 이용한 얼굴 표정 인식에 따른 이모티콘 추출 연구

[1]Bong-Jae Jeong(정봉재), [2]Fan Zhang(장범)

1, First Author School of Medical IT Industry, EulJi University, Korea, E-mail: bongjaejeong@naver.com

2, Corresponding Author School of Business, Shandong University of Political Science and Law, China, E-mail: zhafafa@126.com

Received: Dec 23, 2017. Revised: Jan 02, 2018. Accepted: Jan 15, 2018.

Abstract

In this paper, the pattern of extracting the same expression is proposed by using the Android intelligent device to identify the facial expression. The understanding and expression of expression are very important to human computer interaction, and the technology to identify human expressions is very popular. Instead of searching for the emoticons that users often use, you can identify facial expressions with acamera, which is a useful technique that can be used now. This thesis puts forward the technology of the third data is available on the website of the set, use the content to improve the infrastructure of the facial expression recognition accuracy, in order to improve the synthesis of neural network algorithm, making the facial expression recognition model, the user's facial expressions and similar expressions, reached 66%.It doesn't need to search for emoticons. If you use the camera to recognize the expression, itwill appear emoticons immediately. So this service is the emoticons used when people send messages to others, and it can feel a lot of convenience. In countless emoticons, there is no need to find emoticons, which is an increasing trend in deep learning. So we need to use more suitable algorithm for expression recognition, and then improve accuracy.

Keywords : Deep Learning, Tensor Flow, CNN, Facial Expression Recognition, Android Smart Phone.

1. 서론

인간의 감성이 대표적으로 드러나는 곳은 얼굴이다. 사람들은 얼굴의 표정을 통해 서로의 감성 상태를 예측한다. Darwin(1871)을 비롯한 다수의 학자들은 사람들이 어떤 근거에 의하여 특정 표정이 공통된 감성으로 인식하게 되는지 연구하였다. 메라비언의 법칙(Mehrabian, 1971)에 따르면, 한 사람이 상대방으로부터 받는 이미지는 시각이 55%, 청각이 38%, 언어가 7%에 이르며(Boyle, 1998), 여기서 시각적의사소통이 큰 비중을 차지한다. 심리학자 Ekman과 Friesen은 다른 문화적 배경에서 인간의 표정의 공통성을 연구하였고 6개의 대표적인 표정을 1972년도에 제안했으며 또한 1976년도에 FACS(Facial Action Coding System)을 개발했다(Ekman & Friesen, 1977). 사람들의 감성을 이해하기 위해 연구자들은 심리학, 생리학, 영상 처리, 패턴 인식, 기계 시각, 인공지능 등의 관점에서 탐색했으며 연구 결과가 다양한 분야에서 중요한 역할을 한다. 예를 들어 차량 운전 중에 실시간 모니터링을 통한 운전자의 피곤함을 확인하는 기술 또는 병원에서 환자의 감정 변화를 분석하여 특정 약물과 치료에 적응을 알아 볼 수 있는 기술이 있다.

본 논문에서는 이러한 연구 배경을 바탕으로 안드로이드 스마트 디바이스에서 표정 인식 기술을 이용하여 각 표정 별로 알맞은 이모티콘을 추출하는 연구를 실행했다. 표정 인식 모델을 만들기 위해 캐글 사이트에서 제공되는 fer2013 데이터를 사용했으며 이 데이터는 총 35,887 이미지 파일과 7가지 표정(분노, 혐오, 공포, 기쁨, 슬픔, 놀라움, 중립)을 가진다. 이미지 인식에 많이 사용되는 CNN(Convolutional Neural Network) 알고리즘을 이용하여 fer2013 데이터를 훈련한 모델을 만들었다.

이렇게 연구한 결과를 바탕으로 안드로이드 스마트 디바이스에 훈련된 모델을 적용하여 카메라를 이용한 표정 인식 후 그에 알맞은 이모티콘을 추출하는 연구가 된다면 안드로이드 스마트 디바이스의 불편함을 조금이나마 덜 수 있을 것이다.

2. 관련 연구

2.1. 딥 러닝 (Deep Learning)

딥 러닝(심층학습)이란 여러 비선형 변환기법의 조합을 통해 높은 수준의 추상화(abstractions, 다량의 데이터나 복잡한 자료들 속에서 핵심적인 내용 또는 기능을 요약하는 작업)를 시도하는 기계학습(machine learning) 알고리즘의 집합으로 정의되며, 큰 틀에서 사람의 사고방식을 컴퓨터에게 가르치는 기계학습의 한분야라고 이야기 할 수 있다. 어떠한 데이터가 있을 때 이를 컴퓨터가 알아들을 수 있는 형태(예를 들어 이미지의 경우는 픽셀정보를 열벡터로 표현하는 등)로 표현(representation)하고 이를 학습에 적용하기 위해 많은 연구(어떻게 하면 더 좋은 표현기법을 만들고 또 어떻게 이것들을 학습할 모델을 만들지에 대한)가 진행되고 있으며, 이러한 노력의 결과로 deep neural networks, convolutional deep neural networks, deep belief networks와 같은 다양한 딥 러닝 기법들이 컴퓨터 비젼, 음성인식, 자연어처리, 음성

/신호처리 등의 분야에 적용되어 최첨단의 결과들을 보여주고 있다.

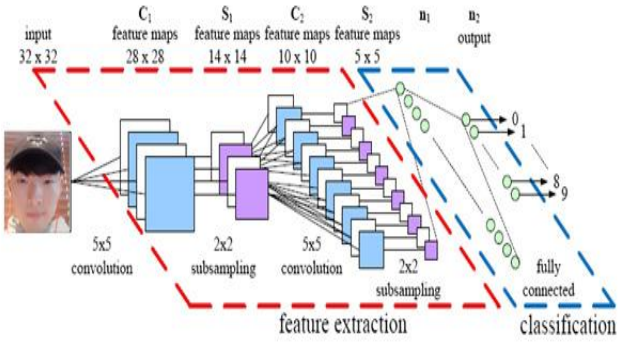
2.2. 텐서플로우(Tensorflow)

구글에서 오픈소스로 공개한 기계학습 라이브러리이다. 딥러닝과 기계학습 분야를 일반인들도 사용하기쉽도록 다양한 기능들을 제공한다. 하이 레벨 프로그래밍 언어로 알려진 파이썬을 활용하여 연산처리를 작성할 수 있다. 다른 언어들도 대부분 지원 하지만 파이썬 관련 자료가 가장 많다. 때문에 공개된지 그리오래되지 않았음에도 불구하고 텐서플로우가 다양한 분야에서 활용되고 있다.

일반 버전과 GPU 가속 버전 두 가지가 공개되어 있다. 일반 버전은 어떤 컴퓨터에서든 실행할 수 있고 GPU 가속 버전은 GPGPU를 사용해 대량 연산을 빠르게 수행하므로 훨씬 빠르게 동작하게 된다. 단, NVDIA의 GPGPU 언어인 CUDA를 사용하기 때문에 NVIDIA 그래픽카드가 없으면 사용할 수 없다.

2.3. CNN(Convolutional Neural Network) 알고리즘

합성곱 신경망(Convolutional Neural Network)은 최소한의 전처리(preprocess)를 사용하도록 설계된 다계층 퍼셉트론(multilayer perceptrons)의 한 종류이다. CNN은 하나 또는 여러 개의 합성곱 계층과 그 위에 올려진 일반적인 인공 신경망 계층들로 이루어져 있으며, 가중치와 통합계층(pooling layer)들을 추가로 활용한다. 이러한 구조 덕분에 CNN은 2차원 구조의 입력 데이터를 충분히 활용할 수 있다. 다른 딥 러닝 구조들과 비교해서, CNN은 영상, 음성 분야 모두에서 좋은 성능을 보여준다. CNN은 또한 표준 역전달을 통해 훈련될 수 있다. CNN은 다른 피드포워드 인공신경망 기법들보다 쉽게 훈련되는 편이고 적은 수의 매개변수를 사용한다는 이점이 있다.



<그림 1> CNN 알고리즘 구조

최근 딥 러닝에서는 합성곱 심층 신뢰 신경망(Convolutional Deep Belief Network, CDBN)가 개발되었는

데, 기존 CNN과 구조적으로 매우 비슷해서, 그림의 2차원 구조를 잘 이용할 수 있으며 그와 동시에 심층 신뢰 신경망(Deep Belief Network, DBN)에서의 선훈련에 의한 장점도 취할 수 있다. CDBN은 다양한 영상과 신호 처리 기법에 사용될 수 있는 일반적인 구조를 제공하며 CIFAR와 같은 표준 이미지 데이터에 대한 여러 벤치마크 결과에 사용되고 있다.

2.4. 얼굴 검출

얼굴 검출은 컴퓨터 비전의 한 분야로 영상(Image)에서 얼굴이 존재하는 위치를 알려주는 기술이다. 얼굴 검출의 알고리즘적인 기본 구조는 Rowley, Baluja, and Kandae(1995)의 논문에 의해 정의되었다. 다양한 크기의 얼굴을 검출하기 위해 피라미드 영상을 생성한 후, 한 픽셀씩 이동하며 특정 크기(예, 20x20 픽셀)의 해당 영역이 얼굴인지 아닌지를 신경망(Neural Network), 아다부스트(Adaboost), 서포트 백터 머신(Support Vector Machine) 등 분류기로 얼굴인지 아닌지를 결정한다.

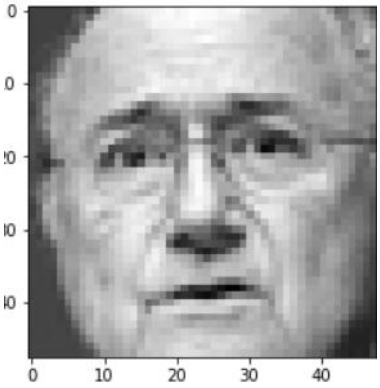
2.5. Haar

Harr는 영상에서 영역 간의 밝기 차를 이용한 특징으로 여러 형태의 기본 특징들이 존재하며 이 특징들을 다양한 크기와 위치로 조합하여 물체에 대한 특징을 추출하는 방법이다. 기본 특징에 대한 특징 값은 특징의 흰색 부분에 해당하는 영상 픽셀들의 밝기 합에서 검은색 부분의 밝기 합을 뺀 차로 계산된다. 그리고 특징을 이용한 대상의 식별은 계산된 영역의 밝기 차가 특징에 부여된 임계값 보다 큰지 작은지 여부를 이용한다. 물론 하나의 특징을 사용하는 것이 아니라 다수의 특징을 조합하여 사용하게 되며 같은 종류의 특징이라 할지라도 물체 내에서의 위치 및 크기에 따라 서로 다른 특징으로 간주하기 때문에 거의 무한대에 가까운 특징 조합이 가능하다. 사람의 얼굴 같은 경우에는 머리카락, 눈썹, 눈동자, 입술 등이 특징적인 밝기 차를 가지기 때문에 Harr 특징을 적용하기에 비교적 적합한 대상으로 볼 수 있다. Harr 특징은 기본적으로 물체의 기하학적 정보를 유지하며 물체의 형태 변화 및 약간의 위치 변화를 어느 정도 커버 할 수 있는 특성을 가진다. 하지만 영상의 contrast 변화, 광원의 방향 변화에 따른 영상 밝기 변화에 영향을 받으며 물체가 회전하는 경우에는 검출이 힘들다는 단점이 있다. Harr 특징은 영상에서 얼굴과 비슷한 특징이 있으면 얼굴로 검출하는 단점이 있기 때문에 다양한 전처리 방법과 함께 사용하여 검출 정확도를 높인다.

3. 본 론

<그림 2> 셀프샷을 이용한 이모티콘 추출

본 논문에서 제시하는 모델은 안드로이드 스마트 디바이스에 장착되어 있는 카메라를 이용하여 사용자의 표정에 따라 이모티콘을 추출하는 모델로 카메라 기능 또는 메신저 이모티콘 보내기 기능 등 다양한 분야에서 활용 할 수 있다.



<그림 3> 모델 학습용 데이터

[그림 3]에서와 같이 캐글 사이트에서 제공되는 사진 데이터로 총 35,887 이미지와 7가지의 표정(분노, 혐오, 공포, 기쁨, 슬픔, 놀라움, 중립)를 이용하여 텐서플로우로 CNN 알고리즘을 적용시켜 표정 인식 모델을 만들었다. 표정 인식 모델을 학습시키는데 사용되는 도구로 노트북 성능 16GB Intel I7-7700HQ CPU 2.80GHz와 고급 프로그래밍 언어인 파이썬을 사용했다.

<그림 4> CNN 모델 구조

[그림 4]에서 CNN 모델의 구조를 보여준다. 본 논문의 연구를 진행하기 위하여 Python 코드로 표정 인식 모델을 작성했다. 본 논문에서의 CNN의 구조는 Deep Learning using Linear Support Vector Machines 논문을 기반으로 만들었다. 텐서플로우 라이브러리를 이용하여 CNN 알고리즘 코드를 작성하여 모델을 학습 시켰으며 사람의 7가지의 표정(분노, 혐오, 공포, 기쁨, 슬픔, 놀라움, 중립)중에서 이미지를 입력하면 학습된 CNN 모델을 통하여 얼마나 정확히 표정을 예측하였는지 정확도를 출력하였다.

[표 1]에 Python 코드는 [그림 4]의 CNN 모델을 Python 코드로 작성한 것이며 fer2013 데이터로 작성한 모델을 학습 및 정확도를 확인한다.

<표 1> CNN 모델 소스 코드

```
# weight initialization
def weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev=1e-4)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)
# convolution
def conv2d(x, W, padd):
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding=padd)
```

```
# pooling
def max_pool_2x2(x):
    return tf.nn.max_pool(x, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1], padding='SAME')
# input & output of NN

# images
x = tf.placeholder('float', shape=[None, image_pixels])
# labels
y_ = tf.placeholder('float', shape=[None, labels_count])
# first convolutional layer 64
W_conv1 = weight_variable([5, 5, 1, 64])
b_conv1 = bias_variable([64])

# (27000, 2304) => (27000,48,48,1)
image = tf.reshape(x, [-1,image_width , image_height,1])
#print (image.get_shape()) # =>(27000,48,48,1)

h_conv1 = tf.nn.relu(conv2d(image, W_conv1, "SAME") + b_conv1)
#print (h_conv1.get_shape()) # => (27000,48,48,64)
h_pool1 = max_pool_2x2(h_conv1)
#print (h_pool1.get_shape()) # => (27000,24,24,1)
h_norm1 = tf.nn.lrn(h_pool1, 4, bias=1.0, alpha=0.001/9.0, beta=0.75)

# second convolutional layer
W_conv2 = weight_variable([5, 5, 64, 128])
b_conv2 = bias_variable([128])

h_conv2 = tf.nn.relu(conv2d(h_norm1, W_conv2, "SAME") + b_conv2)
#print (h_conv2.get_shape()) # => (27000,24,24,128)

h_norm2 = tf.nn.lrn(h_conv2, 4, bias=1.0, alpha=0.001/9.0, beta=0.75)

h_pool2 = max_pool_2x2(h_norm2)
# local layer weight initialization
def local_weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev=0.04)
    return tf.Variable(initial)
```

```
def local_bias_variable(shape):
    initial = tf.constant(0.0, shape=shape)
    return tf.Variable(initial)
# densely connected layer local 3
W_fc1 = local_weight_variable([12 * 12 * 128, 3072])
b_fc1 = local_bias_variable([3072])

# (27000, 12, 12, 128) => (27000, 12 * 12 * 128)
h_pool2_flat = tf.reshape(h_pool2, [-1, 12 * 12 * 128])

h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
#print (h_fc1.get_shape()) # => (27000, 1024)
# densely connected layer local 4
W_fc2 = local_weight_variable([3072, 1536])
b_fc2 = local_bias_variable([1536])

# (40000, 7, 7, 64) => (40000, 3136)
h_fc2_flat = tf.reshape(h_fc1, [-1, 3072])

h_fc2 = tf.nn.relu(tf.matmul(h_fc2_flat, W_fc2) + b_fc2)
#print (h_fc1.get_shape()) # => (40000, 1024)

# dropout
keep_prob = tf.placeholder('float')
h_fc2_drop = tf.nn.dropout(h_fc2, keep_prob)
# readout layer for deep net
W_fc3 = weight_variable([1536, labels_count])
b_fc3 = bias_variable([labels_count])

y = tf.nn.softmax(tf.matmul(h_fc2_drop, W_fc3) + b_fc3)

#print (y.get_shape()) # => (40000, 10)
# settings
LEARNING_RATE = 1e-4
# cost function
cross_entropy = -tf.reduce_sum(y_*tf.log(y))
```

```
# optimisation function
train_step = tf.train.AdamOptimizer(LEARNING_RATE).minimize(cross_entropy)

# evaluation
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))

accuracy = tf.reduce_mean(tf.cast(correct_prediction, 'float'))
# prediction function
#[0.1, 0.9, 0.2, 0.1, 0.1 0.3, 0.5, 0.1, 0.2, 0.3] => 1
predict = tf.argmax(y,1)
# set to 100000 iterations
TRAINING_ITERATIONS = 100000

DROPOUT = 0.5
BATCH_SIZE = 50

epochs_completed = 0
index_in_epoch = 0
num_examples = train_images.shape[0]

# serve data by batches
def next_batch(batch_size):

    global train_images
    global train_labels
    global index_in_epoch
    global epochs_completed

    start = index_in_epoch
    index_in_epoch += batch_size
```

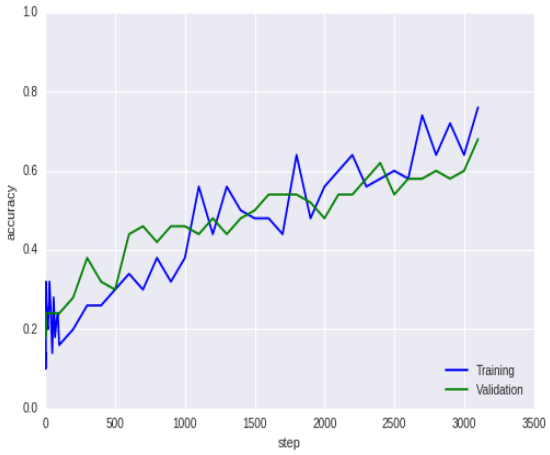
4. 결과

본 논문에서 실행한 연구는 캐글 사이트에서 제공되는 fer2013 데이터세트이며 35887장의 사진을 CNN 알고리즘을 적용시켜서 표정 인식 모델을 학습시켰다.

```
training_accuracy / validation_accuracy => 0.12 / 0.18 for step 98600
training_accuracy / validation_accuracy => 0.12 / 0.18 for step 98700
training_accuracy / validation_accuracy => 0.14 / 0.18 for step 98800
training_accuracy / validation_accuracy => 0.14 / 0.18 for step 98900
training_accuracy / validation_accuracy => 0.20 / 0.18 for step 99000
training_accuracy / validation_accuracy => 0.12 / 0.18 for step 99100
training_accuracy / validation_accuracy => 0.14 / 0.18 for step 99200
training_accuracy / validation_accuracy => 0.14 / 0.18 for step 99300
training_accuracy / validation_accuracy => 0.14 / 0.18 for step 99400
training_accuracy / validation_accuracy => 0.12 / 0.18 for step 99500
training_accuracy / validation_accuracy => 0.16 / 0.18 for step 99600
training_accuracy / validation_accuracy => 0.16 / 0.18 for step 99700
training_accuracy / validation_accuracy => 0.14 / 0.18 for step 99800
training_accuracy / validation_accuracy => 0.14 / 0.18 for step 99900
training_accuracy / validation_accuracy => 0.10 / 0.18 for step 99999
```

<그림 5> 학습 정확도

[그림 5]는 100000번 반복 학습한 데이터의 정확도를 나타낸다. 모델 학습의 가장 높은 정확도는 66%이다.



<그림 6> 정확도 그래프

<그림 6>는 100000번 반복 학습한 데이터의 정확도를 그래프로 보여주는 것이며 66%의 높은 정확도를 보이고 있다.

5. 결론

본 논문에서는 사용자의 표정 인식을 통해 표정과 동일한 이모티콘을 추출하는 서비스를 제시하였다. 여러 사진을 CNN 알고리즘으로 학습시킨 결과 66%의 정확도로 표정을 인식하는 모델을 만들었다. 이 모

델을 기반으로 안드로이드 스마트 디바이스에 적용시켜 앱을 만들거나 얼굴 표정을 인식해야 되는 상황이 발생하는 등 다양한 분야에서 유용하게 쓰일 것이다. 또한 이 모델을 토대로 더 높은 정확도를 연구하면 표정 인식 모델의 성능이 좋아질 것이다.

원하는 이모티콘을 일일이 검색 할 필요 없이 카메라로 표정을 인식하면 이모티콘이 바로 나타나기 때문에 이 서비스는 사람들이 메시지를 주고받을 때 이모티콘 사용에 많은 편리함을 느낄 수 있을 것이다. 수많은 이모티콘 중에서 원하는 이모티콘을 일일이 찾을 필요 없으며 딥 러닝 분야의 연구가 늘어나는 추세라 표정 인식에 더 적합한 알고리즘을 이용해 추후 정확도를 더 높일 필요가 있다.

Reference

Boyle, G. J. (1998). Review of Arousal Seeking Tendency Scale. In J. C. Impara & B. S. Plake (Eds.), *The thirteenth mental measurements yearbook* (pp. 49-50). Lincoln, NE: Buros Institute of Mental Measurements.

Darwin, C. (1871). *The descent of man, and selection in relation to sex.* London: John Murray.

Ekman, P., & Friesen, W. V. (1977). *The Facial Action Coding System (FACS): A Technique for the Measurement of Facial Action.* Palo Alto: Consulting Psychologists Press.

Mehrabian, A. (1971). *Silent Messages* (1st ed.). Belmont, CA: Wadsworth.

NAMU (2017). *Tenser Flow*. Retrieved May 22, 2017. from https://namu.wiki/w/텐서플로우

Pervaiz, A. Z. (2010). Real Time Face Recognition System Based on EBGM Framework. In Computer Modelling and Simulation (UKSim),12th International Conference on 2010, pp.262-266.

Rowley, H. A. Baluja, S., & Kanade, T. (1995). Human face detection in visual scenes. CMUCS-95-158R, Carnegie Mellon University, Retrieved November 22, 1995. from http://www.cs.cmu.edu/˜har/faces.html.

Wikipedia (2017a*). Convolutional Neural Network*. Retrieved May 22, 2017. from https://ko.wikipedia.org/wiki/%EB%94%A5_%EB%9F%AC%EB%8B%9D#.ED.95.A9.EC.84.B1.EA.B3.B1_.EC.8B.A0.EA.B2.BD.EB.A7.9D.28Convolutional_Neural_Network.2C_CNN.29

Wikipedia (2017b*). Deep Lelearning*. Retrieved May 22, 2017. from https://ko.wikipedia.org/wiki/딥_러닝

Wikipedia (2017c*). Deep Lelearning*. Retrieved May 22, from https://ko.wikipedia.org/wiki/%EC%96%BC%EA%B5%B4_%EA%B2%80%EC%B6%9C