

이미지 분류를 활용한 얼굴 감정 인식

오서영 , 최규진 , 최연석 , 박지원



CNN 개념



CNN을 활용한 감정 분류



Face landmark 만들기



시각화

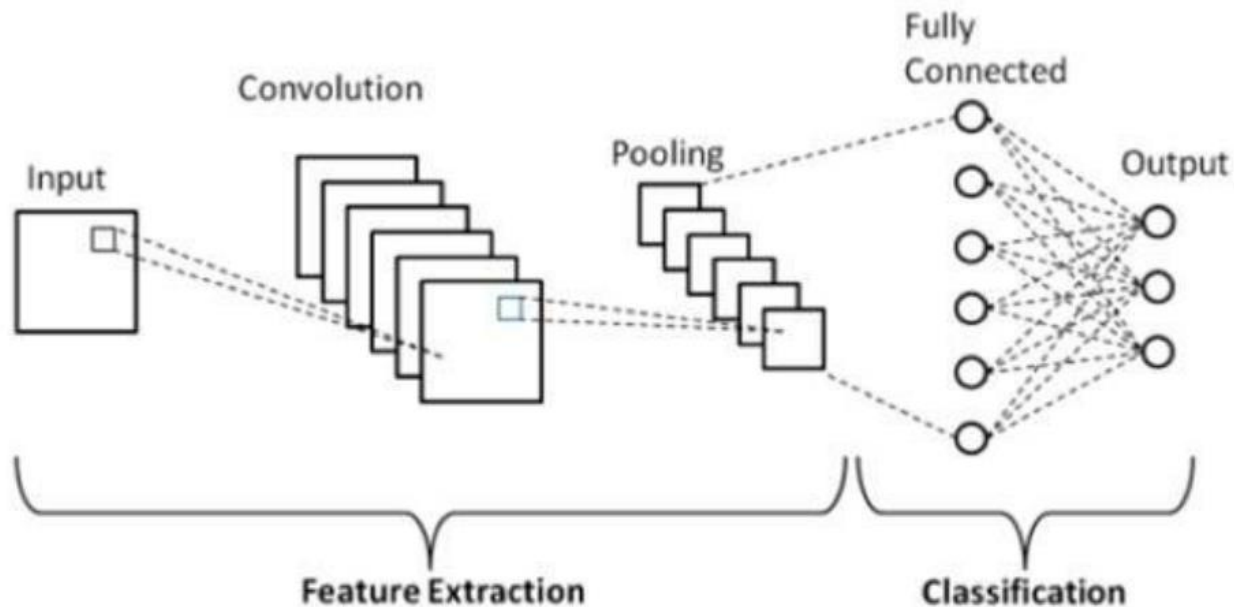


CNN 개념

CNN (Convolutional Neural Network)

: 신경망에 전처리를 추가한 다층 퍼셉트론의 한 종류

: 이미지, 동영상, 음성 학습에 많이 사용



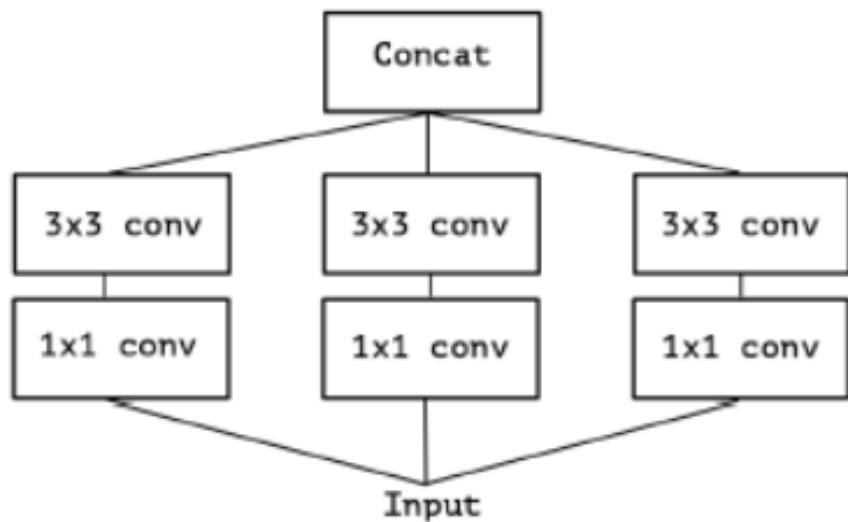
- Conv 층을 통해 입력값에 대한 특징맵들을 여러 개 만들
 - pooling층을 통해 특징맵의 크기를 줄임
- => 이를 반복하면서 마지막에 퍼셉트론 도출!

Xception 개념

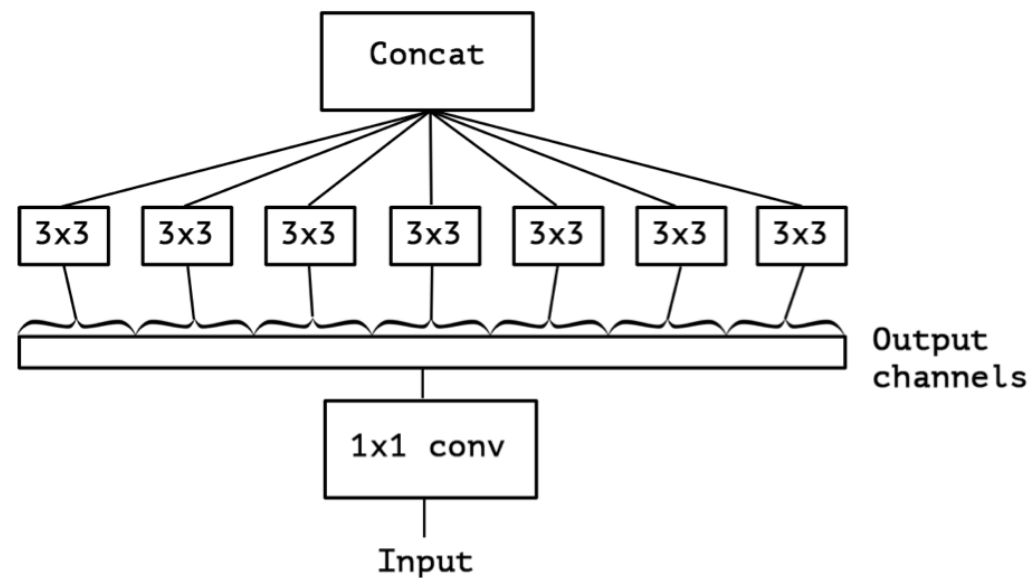
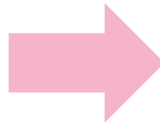
Xception 모델

- : 구글이 2017년에 발표한 모델로 CNN 중 하나
- : extreme inception의 약자

기존 inception 모델이 채널, 공간을 분리한 것을 depthwise separable convolution으로 강화한 모델



Inception 구조



Xception

Inception : 노드간의 연결을 줄임

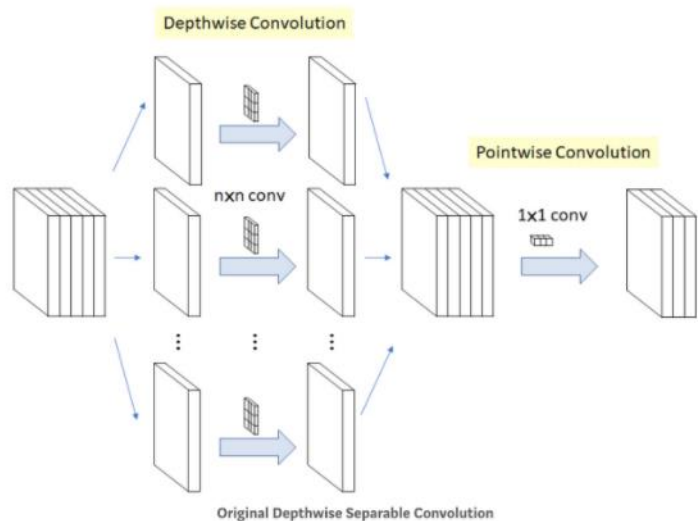
Xception : 채널간의 관계를 찾는 것과 이미지 지역 정보를 찾는 것을 완전히 분리하고자 함!

Xception 개념

Depthwise separable convolution



각 채널별로 conv연산을 시행하고
그 결과에 1x1 연산을 취하는 것



Depthwise ~와 Xception 차이점

- Relu 유무
- 진행 순서



Depthwise separable convolution

Channel-wise $n \times n$ spatial convolution
(k개의 채널에 대해 $n \times n$ conv를 따로 진행해서 합침)
→ pointwise convolution
(채널의 개수를 줄이기 위한 방법)



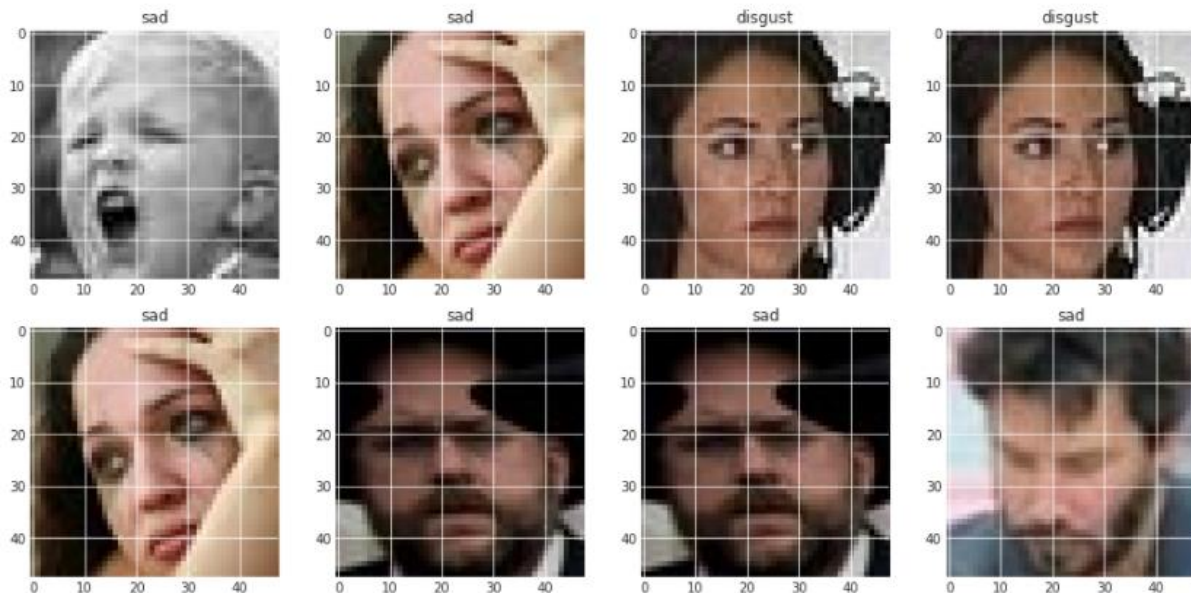
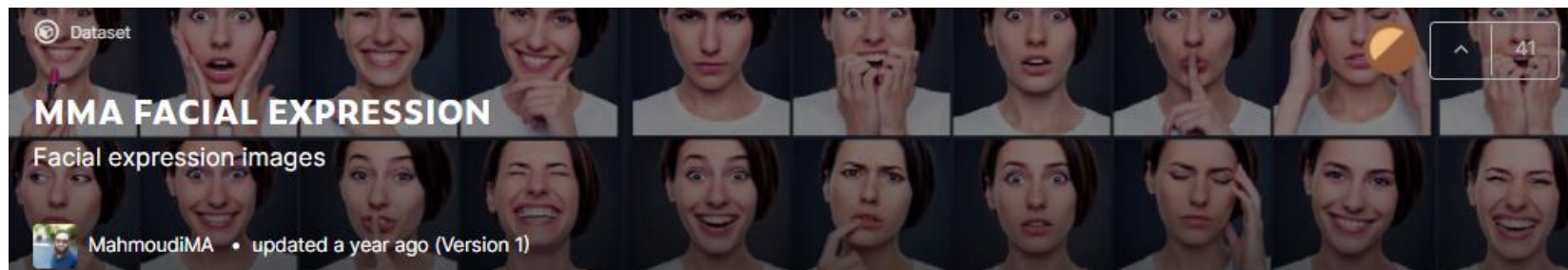
Xception

pointwise convolution
(채널의 개수를 줄이기 위한 방법)
→ Channel-wise $n \times n$ spatial convolution
(k개의 채널에 대해 $n \times n$ conv를 따로 진행해서 합침)

CNN을 활용한 감정분류 1



데이터셋 (캐글) : <https://www.kaggle.com/mahmoudima/mma-facial-expression>



-> 감정별로 분류 되어있음 (7개)
Angry, disgust, fear, happy,
neutral, sad, surprise

CNN을 활용한 감정분류 1

1. Xception 모델

```
train_generator = train_datagen.flow_from_directory("/kaggle/input/emma-facial-expression/MMAFEDB/train",
                                                    target_size=(img_width, img_height),
                                                    batch_size=batch_size,
                                                    class_mode='categorical')
validation_generator = validation_datagen.flow_from_directory("/kaggle/input/emma-facial-expression/MMAFEDB/valid",
                                                             target_size=(img_width, img_height),
                                                             batch_size=batch_size,
                                                             class_mode='categorical')
test_generator = test_datagen.flow_from_directory("/kaggle/input/emma-facial-expression/MMAFEDB/test",
                                                  target_size=(img_width, img_height),
                                                  batch_size=batch_size,
                                                  class_mode='categorical')
```

Found 92968 images belonging to 7 classes.

Found 17356 images belonging to 7 classes.

Found 17356 images belonging to 7 classes.



텐서플로우의 generator를 사용하여 데이터 만들기

CNN을 활용한 감정분류 1

```
# base model : xception -> fine tuning
model = tf.keras.applications.Xception(weights='imagenet', include_top=False, input_shape=(img_width, img_height, 3))
model.summary()
```

```
# Change last layer to fit out needs: 7 classes
x = model.output
x = GlobalAveragePooling2D()(x)
predictions = Dense(7, activation='softmax')(x)
model = Model(model.input, predictions)
model.summary()
```

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
%%time
hist = model.fit([
    train_generator,
    epochs = nb_epoch,
    steps_per_epoch = num_train//batch_size,
    validation_data = validation_generator,
    validation_steps = num_val//batch_size])
```

Imagenet 데이터로 이미 학습된 모델을 미세조정



60.01% accuracy

CNN을 활용한 감정분류 1

2. 기본 CNN 모델

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 64, 64, 32)	896
conv2d_5 (Conv2D)	(None, 64, 64, 64)	18496
batch_normalization_2 (Batch Normalization)	(None, 64, 64, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 64)	0
dropout_3 (Dropout)	(None, 32, 32, 64)	0
conv2d_6 (Conv2D)	(None, 32, 32, 128)	73856
conv2d_7 (Conv2D)	(None, 30, 30, 256)	295168
batch_normalization_3 (Batch Normalization)	(None, 30, 30, 256)	1024
max_pooling2d_3 (MaxPooling2D)	(None, 15, 15, 256)	0
dropout_4 (Dropout)	(None, 15, 15, 256)	0
flatten_1 (Flatten)	(None, 57600)	0
dense_2 (Dense)	(None, 1024)	58983424
dropout_5 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 7)	7175
Total params: 59,380,295		
Trainable params: 59,379,655		
Non-trainable params: 640		

```
model.compile(optimizer='sgd',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

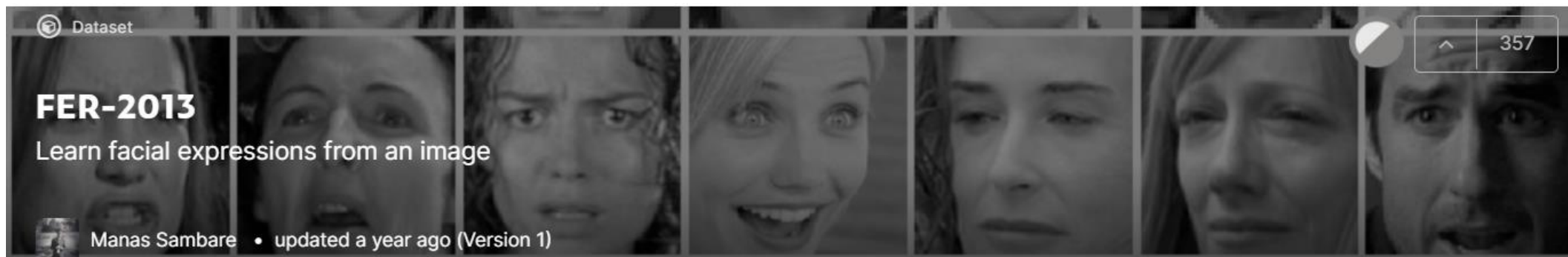


52.67% accuracy

CNN을 활용한 감정분류 2



데이터셋 (캐글) : <https://www.kaggle.com/msambare/fer2013>



-> 감정별로 분류되어있음 (7개)

Angry, disgust, fear, happy,
neutral, sad, surprise

CNN을 활용한 감정분류 2

1. FER2013 데이터 살펴보기

```
train_dir = 'C:/Users/janyq/OneDrive/바탕 화면/템플/train/'
test_dir = 'C:/Users/janyq/OneDrive/바탕 화면/템플/test/'
```

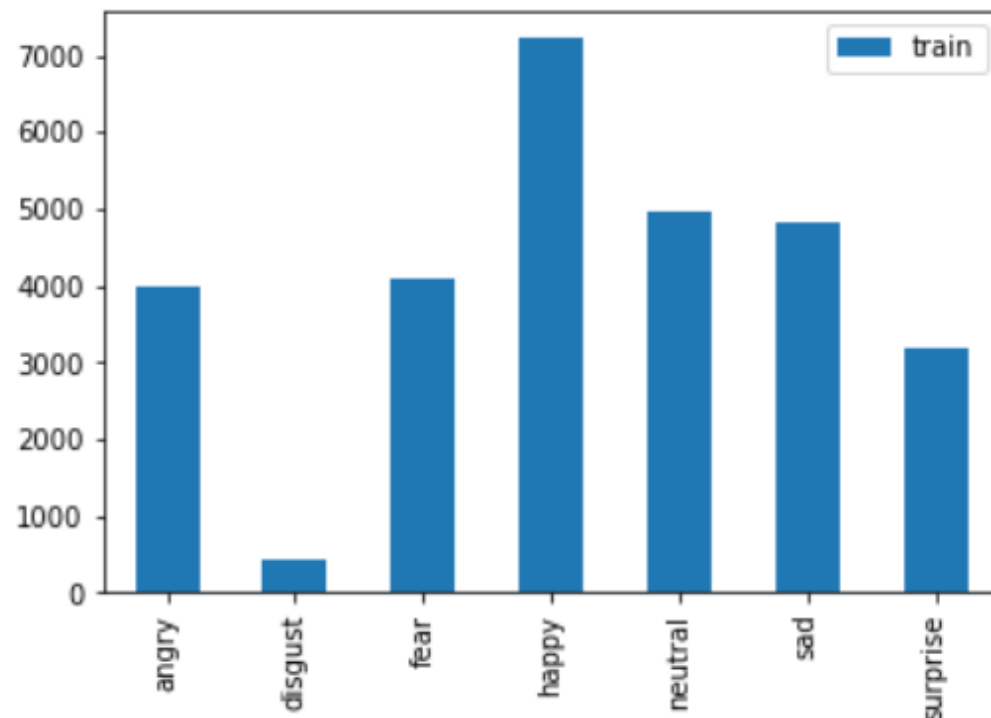
```
row, col = 48, 48
classes = 7
```

```
def count_exp(path, set_):
    dict_ = {}
    for expression in os.listdir(path):
        dir_ = path + expression
        dict_[expression] = len(os.listdir(dir_))
    df = pd.DataFrame(dict_, index=[set_])
    return df
train_count = count_exp(train_dir, 'train')
test_count = count_exp(test_dir, 'test')
print(train_count)
print(test_count)
```

	angry	disgust	fear	happy	neutral	sad	surprise
train	3995	436	4097	7215	4965	4830	3171
test	958	111	1024	1774	1233	1247	831

```
train_count.transpose().plot(kind='bar')
```

<AxesSubplot:>



CNN을 활용한 감정분류 2

2. 각 폴더별 이미지 확인

```
plt.figure(figsize=(14,22))
i = 1
for expression in os.listdir(train_dir):
    img = load_img((train_dir + expression + '/' + os.listdir(train_dir + expression)[1]))
    plt.subplot(1,7,i)
    plt.imshow(img)
    plt.title(expression)
    plt.axis('off')
    i += 1
plt.show()
```

angry



disgust



fear



happy



neutral



sad



surprise



CNN을 활용한 감정분류 2

3. 학습 모델 구축

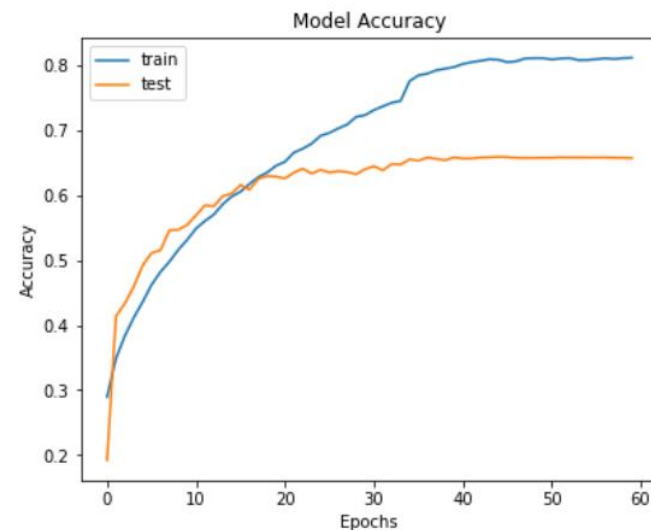
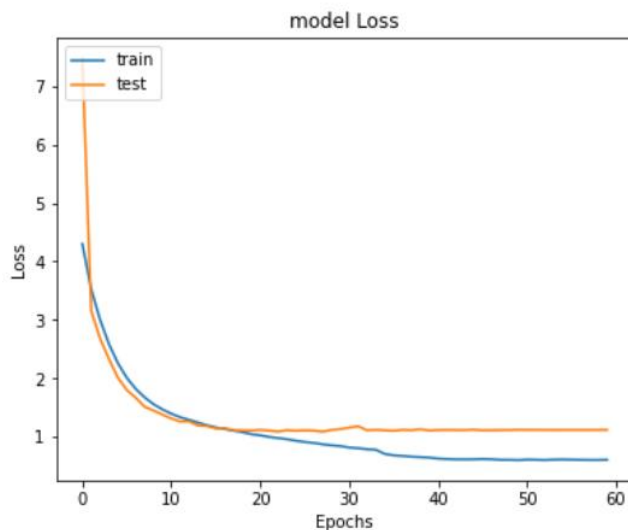
```
def get_model(input_size, classes=7):  
    #Initialising the CNN  
    model = tf.keras.models.Sequential()  
  
    model.add(Conv2D(32, kernel_size=(3, 3), padding='same', activation='relu', input_shape =input_size))  
    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same'))  
    model.add(BatchNormalization())  
    model.add(MaxPooling2D(2, 2))  
    model.add(Dropout(0.25))  
  
    model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same', kernel_regularizer=regularizers.l2(0.01)))  
    model.add(Conv2D(256, kernel_size=(3, 3), activation='relu', kernel_regularizer=regularizers.l2(0.01)))  
    model.add(BatchNormalization())  
    model.add(MaxPooling2D(pool_size=(2, 2)))  
    model.add(Dropout(0.25))  
  
    model.add(Flatten())  
    model.add(Dense(1024, activation='relu'))  
    model.add(Dropout(0.5))  
  
    model.add(Dense(classes, activation='softmax'))  
  
    #Compiling the model  
    model.compile(optimizer=Adam(lr=0.0001, decay=1e-6),  
                  loss='categorical_crossentropy',  
                  metrics=['accuracy'])  
  
    return model
```

CNN을 활용한 감정분류 2

4. 학습

```
steps_per_epoch = training_set.n // training_set.batch_size
validation_steps = test_set.n // test_set.batch_size

hist = fernet.fit(x=training_set,
                  validation_data=test_set,
                  epochs=60,
                  callbacks=callbacks,
                  steps_per_epoch=steps_per_epoch,
                  validation_steps=validation_steps)
```

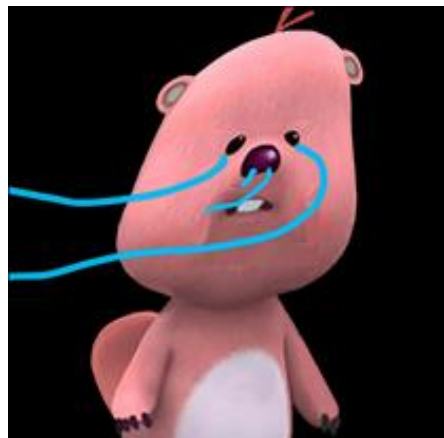


```
train_loss, train_accu = fernet.evaluate(training_set)
test_loss, test_accu = fernet.evaluate(test_set)
print("final train accuracy = {:.2f} , validation accuracy = {:.2f}".format(train_accu*100, test_accu*100))
```

```
449/449 [=====] - 83s 185ms/step - loss: 0.4415 - accuracy: 0.8859
113/113 [=====] - 23s 207ms/step - loss: 1.1210 - accuracy: 0.6571
final train accuracy = 88.59 , validation accuracy = 65.71
```

CNN을 활용한 감정분류 2

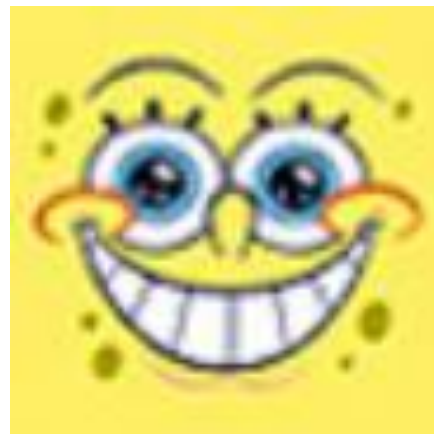
5. 모델 테스트



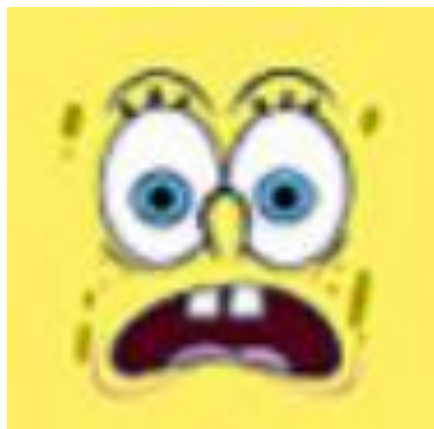
```
1 a=model.predict(x_test)
2 emotions[np.argmax(a)]
sad'
```



'angry'



'happy'



'surprise'

CNN을 활용한 감정분류 2

사람 얼굴 이미지로 테스트

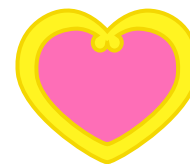
```
1 img_resize=img2.resize((48,48))
2 img_resize = np.array(img_resize)
3 img_resize = np.array(img_resize)
4 img_resize=img_resize/255
5
6 img_resize.shape
7 x_test = np.reshape(img_resize,(1,48,48,1))
8 x_test.shape
```

(1, 48, 48, 1)

```
1 a=model.predict(x_test)
2 emotions[np.argmax(a)]
```



사진을 제공해준 회장님께 감사합니다



'happy'



'happy'

landmark 만들기



데이터셋 (캐글) : <https://www.kaggle.com/drgilermo/face-images-with-marked-landmark-points>



KEY POINTS

	left_eye_center_x	left_eye_center_y	right_eye_center_x	right_eye_center_y	left_eye_inner_corner_x	left_eye_inner_corner_y	left_eye_outer_corner_x
0	66.033564	39.002274	30.227008	36.421678	59.582075	39.647423	73.130346
1	64.332936	34.970077	29.949277	33.448715	58.856170	35.274349	70.722723
2	65.057053	34.909642	30.903789	34.909642	59.412000	36.320968	70.984421
3	65.225739	37.261774	32.023096	37.261774	60.003339	39.127179	72.314713
4	66.725301	39.621261	32.244810	38.042032	58.565890	39.621261	72.515926

5 rows × 30 columns

landmark 만들기

```
# visualization  
ind = 10  
plt.imshow(features[ind,:,:], cmap='gray')  
plt.scatter(key_pts.iloc[ind][0:-1:2], key_pts.iloc[ind][1::2], c='y')  
plt.axis('off')  
plt.show()
```



최대 30개의 key points

landmark 만들기

CNN을 활용한 landmark 학습

```
model = Sequential()

model.add(Input(shape=(img_size, img_size, 1)))

model.add(BatchNormalization())
model.add(Conv2D(32, (3,3), padding="same"))
model.add(LeakyReLU(alpha = 0.1))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(BatchNormalization())
model.add(Conv2D(64, (3,3), padding="same"))
model.add(LeakyReLU(alpha = 0.1))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(BatchNormalization())
model.add(Conv2D(128, (3,3), padding="same"))
model.add(LeakyReLU(alpha = 0.1))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(256))
model.add(LeakyReLU(alpha = 0.1))
model.add(Dropout(0.5))

model.add(Dense(64))
model.add(LeakyReLU(alpha = 0.1))
model.add(Dense(30))
```

```
model.compile(loss = 'mean_squared_error', optimizer = Adam(),
              metrics=['mean_squared_error'])
```



Test MSE : 117.1117

시각화 1

angry face landmark 시각화

```
plt.figure(figsize=(10,10))

plt.subplot(331)
plt.imshow(x_test_a[0,:,:], cmap='gray')
plt.scatter(pred_a[0,:][0:-1:2], pred_a[0,:][1::2], c='b')
plt.axis("off")

plt.subplot(332)
plt.imshow(x_test_a[5,:,:], cmap='gray')
plt.scatter(pred_a[5,:][0:-1:2], pred_a[5,:][1::2], c='b')
plt.axis("off")
```

○
○
○

```
plt.subplot(338)
plt.imshow(x_test_a[14,:,:], cmap='gray')
plt.scatter(pred_a[14,:][0:-1:2], pred_a[14,:][1::2], c='b')
plt.axis("off")

plt.subplot(339)
plt.imshow(x_test_a[149,:,:], cmap='gray')
plt.scatter(pred_a[149,:][0:-1:2], pred_a[149,:][1::2], c='b')
plt.axis("off")

plt.show()
```

앞서 학습시킨 모델을 활용하여 FER2013 데이터의 landmark 생성



시각화 1

각 좌표의 max, min 값 비교

```
angry_d=preds_a.describe()
a_max=angry_d.iloc[7]
a_max=pd.DataFrame(a_max)
a_max=a_max.T
a_max=np.array(a_max)
a_max[0,:][0:-1:2]
```

```
a_min=angry_d.iloc[3]
a_min=pd.DataFrame(a_min)
a_min=a_min.T
a_min=np.array(a_min)
a_min[0,:][0:-1:2]
```

```
array([[50.82007599, 23.9002533, -1.05379391, -1.31817818, -1.02874756,
        -0.97830987, -1.07107711, -1.67900753, -1.38587379, -0.98549318,
        32.68993378, -0.9947753, -1.13942957, -1.00657225, 33.15564728]])
```

```
index=90
plt.imshow(x_test_a[index,:,:], cmap='gray')
plt.scatter(a_max[0,:][0:-1:2], a_max[0,:][1::2], c='r')
plt.scatter(a_min[0,:][0:-1:2], a_min[0,:][1::2], c='b')
plt.axis("off")
plt.show()
```

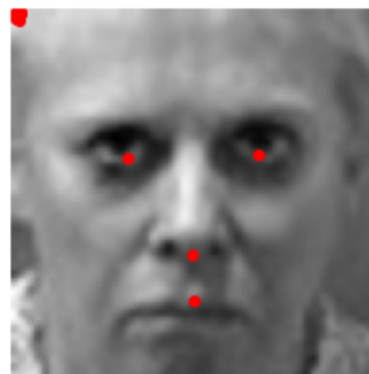


각 좌표의 mean 값 비교

```
a_mean=angry_d.iloc[1]
a_mean=pd.DataFrame(a_mean)
a_mean=a_mean.T
a_mean=np.array(a_mean)
a_mean[0,:][0:-1:2]
```

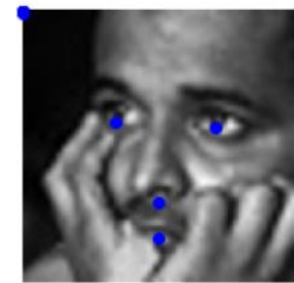
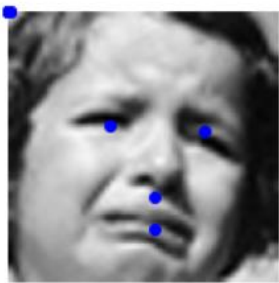
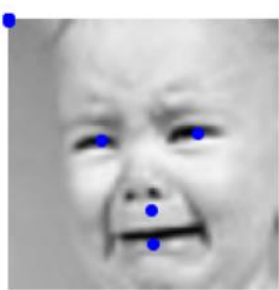
```
array([[63.89535904, 30.04463959, 1.58619165, 1.90227842, 0.77607858,
        0.49255601, 1.4616735, 2.02885294, 0.65742195, 0.2321679,
        46.75931549, 1.67421043, 0.86192524, 1.24880922, 47.21754837]])
```

```
index=90
plt.imshow(x_test_a[index,:,:], cmap='gray')
plt.scatter(a_mean[0,:][0:-1:2], a_mean[0,:][1::2], c='r')
plt.axis("off")
plt.show()
```



시각화 1

sad face landmark 시각화



시각화 1

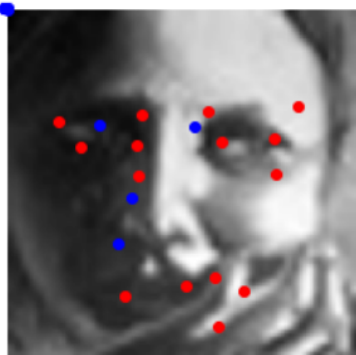
각 좌표의 max, min 값 비교

```
: sad_d=preds_s.describe()
s_max=sad_d.iloc[7]
s_max=pd.DataFrame(s_max)
s_max=s_max.T
s_max=np.array(s_max)
s_max[0,:][0:-1:2]

s_min=sad_d.iloc[3]
s_min=pd.DataFrame(s_min)
s_min=s_min.T
s_min=np.array(s_min)
s_min[0,:][0:-1:2]

: array([50.73066711, 24.69040871, -1.12606144, -1.51337528, -1.27937293,
        -0.66309643, -1.42649245, -1.57622194, -1.56539631, -0.67275333,
        33.5194397 , -1.28190088, -0.67696285, -0.93593693, 30.09754944])
```

```
: index=40
plt.imshow(x_test_s[index,:,:], cmap='gray')
plt.scatter(s_max[0,:][0:-1:2], s_max[0,:][1::2], c='r')
plt.scatter(s_min[0,:][0:-1:2], s_min[0,:][1::2], c='b')
plt.axis("off")
plt.show()
```



각 좌표의 mean 값 비교

```
s_mean=sad_d.iloc[1]
s_mean=pd.DataFrame(s_mean)
s_mean=s_mean.T
s_mean=np.array(s_mean)
s_mean[0,:][0:-1:2]

array([63.32885742, 29.7233429 , 2.17794967, 2.62067676, 1.16092086,
        0.70394498, 1.99097693, 2.79697609, 1.04244983, 0.38606361,
        46.28223038, 2.31201696, 1.17834151, 1.72568595, 46.74061966])
```

```
index=40
plt.imshow(x_test_s[index,:,:], cmap='gray')
plt.scatter(s_mean[0,:][0:-1:2], s_mean[0,:][1::2], c='r')
plt.axis("off")
plt.show()
```



시각화 2

눈, 코, 입의 중심 landmark 시각화

```
center_pts_x = location[:, center_inds]
center_pts_y = location[:, center_inds + 1]

print(center_pts_x, "\n", center_pts_y)
center_inds
```

```
[['left_eye_center_x' 'right_eye_center_x' 'nose_tip_x'
  'mouth_center_bottom_lip_x']]
[['left_eye_center_y' 'right_eye_center_y' 'nose_tip_y'
  'mouth_center_bottom_lip_y']]

array([ 0,  2, -10, -2])
```

angry



disgust



fear



happy



neutral



sad



surprise



시각화 2

중심 점들의 거리 계산

```
def dist_pts(x_pts, y_pts):

    left_eye = (x_pts[0], y_pts[0])
    right_eye = (x_pts[1], y_pts[1])
    nose = (x_pts[2], y_pts[2])
    mouth = (x_pts[3], y_pts[3])

    left_eye_nose = dist(left_eye, nose)
    right_eye_nose = dist(right_eye, nose)
    nose_mouth = dist(mouth, nose)

    return left_eye_nose, right_eye_nose, nose_mouth
```

```
distance = []
for i in range(len(emotions)):

    name = globals()['name_{}'.format(emotions[i])]
    sum1 = 0
    sum2 = 0
    for j in range(len(name)):
        x = np.zeros(4)
        for k in range(4):
            x[k] = globals()['y_test_{}'.format(emotions[i])][j, center_inds[k]]
            y[k] = globals()['y_test_{}'.format(emotions[i])][j, center_inds[k] + 1]
        leye_nose, reye_nose, nose_mouth = dist_pts(x, y)
        sum1 += leye_nose + reye_nose
        sum2 += nose_mouth

    mean1 = sum1 / (len(name) * 2)
    mean2 = sum2 / len(name)

    distance.append([mean1, mean2])

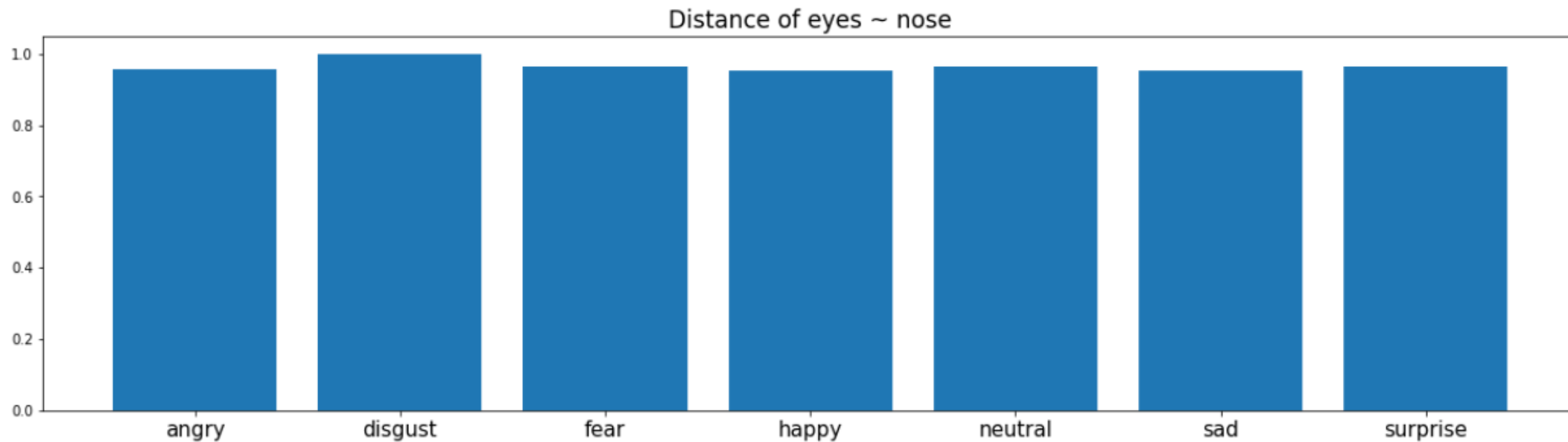
distance = np.array(distance)
```

라벨 별
눈~코 평균거리
코~입 평균거리
계산

시각화 2

1. 두 눈과 코사이의 평균거리

Distance/max(distance) (최대값으로 나눠서 정규화)



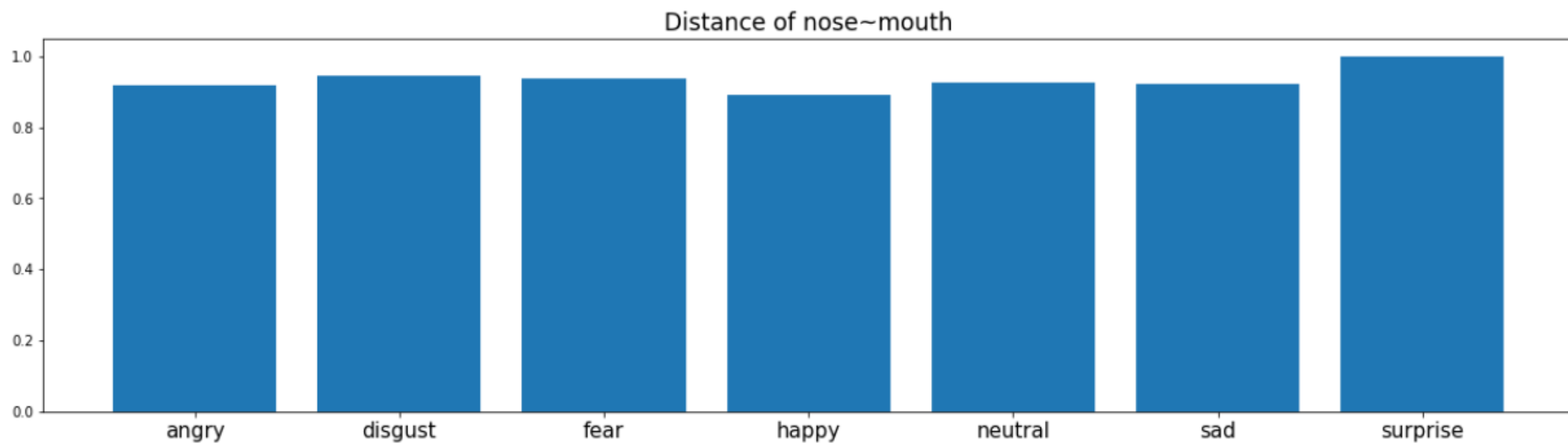
Max : disgust

Min : happy

시각화 2

2. 코와 입사이의 평균거리

Distance/max(distance) (최대값으로 나눠서 정규화)



Max : surprise

Min : happy



데이터셋

- [1] MMA FACIAL EXPRESSION, <https://www.kaggle.com/mahmoudima/mma-facial-expression>
- [2] FER-2013, <https://www.kaggle.com/msambare/fer2013>
- [3] Face Images with Marked Landmark Points,
<https://www.kaggle.com/drgilermo/face-images-with-marked-landmark-points>



참고자료

- [1] Xception, <https://hongl.tistory.com/45?category=922582>
- [2] Inception & Xception, <https://dalsacoo-log.tistory.com/entry/Inception-and-Xception>



발표를 마치겠습니다