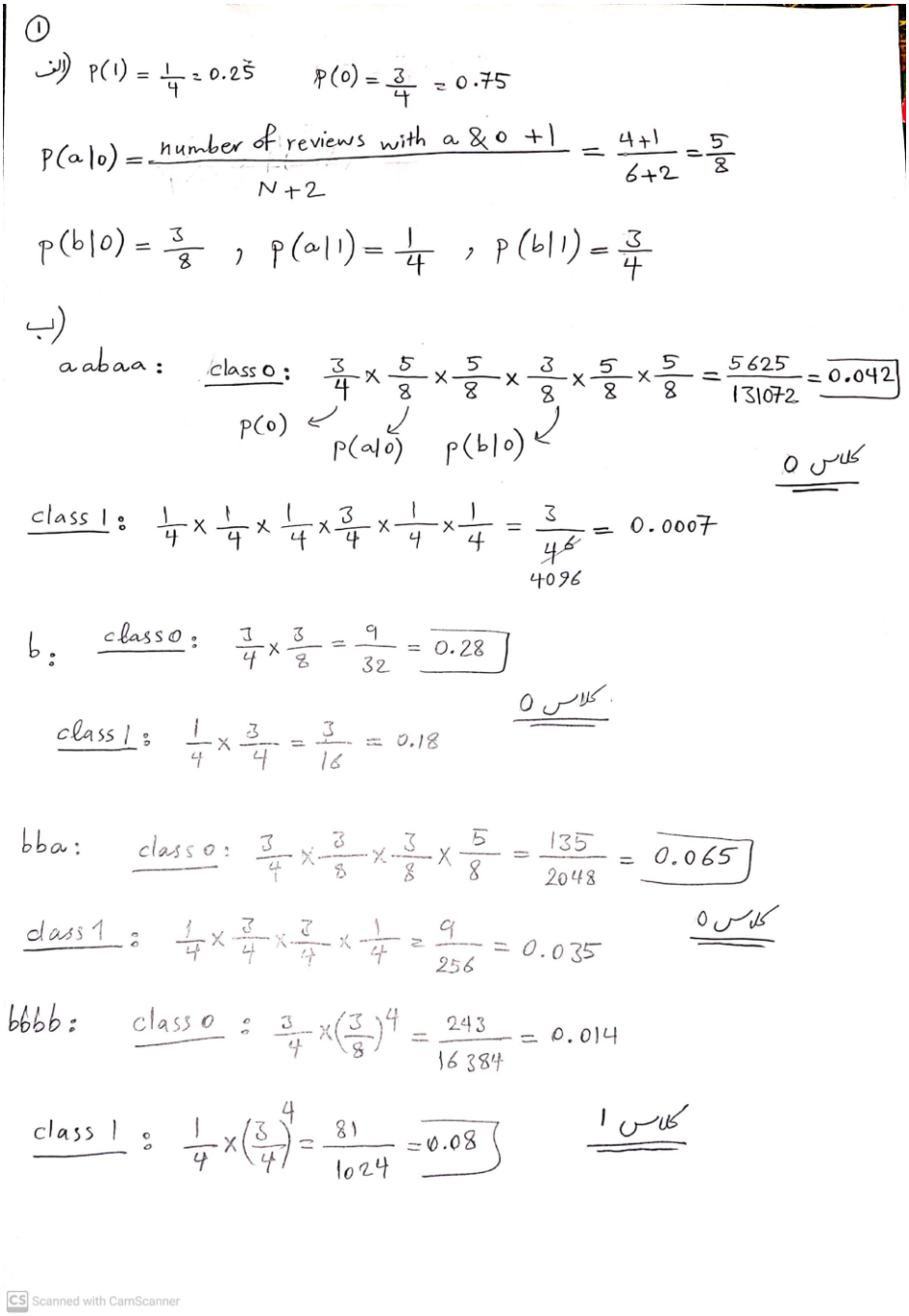
Deep Learning
Dr. DavoodAbadi
Fall 2022
Hoorieh Sabzevari - 98412004
HW1



۱.

①

الف) $P(1) = \frac{1}{4} = 0.25$ $\quad P(0) = \frac{3}{4} = 0.75$

$$P(a|0) = \frac{\text{number of reviews with a \& 0 +1}}{N+2} = \frac{4+1}{6+2} = \frac{5}{8}$$

$P(b|0) = \frac{3}{8}$, $P(a|1) = \frac{1}{4}$, $P(b|1) = \frac{3}{4}$

ب)

aabaa: class 0: $\frac{3}{4} \times \frac{5}{8} \times \frac{5}{8} \times \frac{3}{8} \times \frac{5}{8} \times \frac{5}{8} = \frac{5625}{131072} = 0.042$

$P(0)$ ← $P(a|0)$ $P(b|0)$ ←

کلاس 0

class 1: $\frac{1}{4} \times \frac{1}{4} \times \frac{1}{4} \times \frac{3}{4} \times \frac{1}{4} \times \frac{1}{4} = \frac{3}{4096} = 0.0007$

b: class 0: $\frac{3}{4} \times \frac{3}{8} = \frac{9}{32} = 0.28$

کلاس 0

class 1: $\frac{1}{4} \times \frac{3}{4} = \frac{3}{16} = 0.18$

bba: class 0: $\frac{3}{4} \times \frac{3}{8} \times \frac{3}{8} \times \frac{5}{8} = \frac{135}{2048} = 0.065$

class 1: $\frac{1}{4} \times \frac{3}{4} \times \frac{3}{4} \times \frac{1}{4} = \frac{9}{256} = 0.035$

کلاس 0

bbbb: class 0: $\frac{3}{4} \times \left(\frac{3}{8}\right)^4 = \frac{243}{16384} = 0.014$

class 1: $\frac{1}{4} \times \left(\frac{3}{4}\right)^4 = \frac{81}{1024} = 0.08$

کلاس 1

۳.

۱) دیتاست MNIST مربوط به ارقام دست‌نویس است که ده کلاس ارقام ۰تا۹ دارد. این دیتاست شامل ۶۰۰۰۰ مجموعه‌ی آموزشی و ۱۰۰۰۰ مجموعه‌ی تست است که ورودی هرکدام یک عکس ۲۸*۲۸ پیکسل از رقم دست‌نویس و خروجی آن لیبل کلاس مربوطه است.

دیتاست CIFAR-10 دیتاست مربوط به حیوانات، شامل ۶۰۰۰۰ تصویر رنگی۳۲*۳۲ در ۱۰ کلاس با ۶۰۰۰ تصویر در هر کلاس است. ۵۰۰۰۰ تصویر آموزشی و ۱۰۰۰۰ تصویر تست وجود دارد. مجموعه داده به پنج دسته‌ی آموزشی و یک مجموعه‌ی تست تقسیم شده است که هر کدام دارای ۱۰۰۰۰ تصویر است. مجموعه‌ی تست دقیقاً شامل ۱۰۰۰ تصویر انتخابی تصادفی از هر کلاس است. دسته‌های آموزشی شامل تصاویر باقی‌مانده به ترتیب تصادفی هستند، اما برخی از دسته‌های آموزشی ممکن است حاوی تصاویر بیشتری از یک کلاس نسبت به کلاس دیگر باشند. در بین آن‌ها، دسته‌های آموزشی دقیقاً حاوی ۵۰۰۰ تصویر از هر کلاس هستند.

دیتاست FER_2013 شامل تصاویر ۴۸*۴۸ پیکسل در مقیاس خاکستری از چهره ها است. چهره ها به طور خودکار ثبت شده اند به طوری که صورت کم و بیش در مرکز قرار گرفته و تقریباً همان مقدار فضای هر تصویر را اشغال می کند. وظیفه این است که هر چهره را بر اساس احساسات نشان داده شده در حالت چهره به یکی از هفت دسته (۰ = عصبانی، ۱ = انزجار، ۲ = ترس، ۳ = خوشحال، ۴ = غمگین، ۵ = تعجب، ۶ = خنثی) دسته بندی کنیم. مجموعه آموزشی شامل ۲۸۷۰۹ نمونه و مجموعه‌ی تست شامل ۷۱۷۸ نمونه می باشد.

۲) این تابع لیبل‌های خروجی را به صورت یک بردار با یک سطر و به تعداد کلاس‌ها ستون تبدیل می‌کند. یعنی یک آرایه‌ی numpy یک بردار که دارای اعداد صحیح است که دسته‌های مختلف را نشان می‌دهد، می‌تواند به یک آرایه numpy یا ماتریسی تبدیل شود که دارای مقادیر باینری ستون‌هایی برابر با تعداد دسته‌ها در داده ها است. به این ترتیب کلاس مربوط به هر نمونه برابر است با ایندکسی که مقدار یک داشته باشد.

۳) ابعاد هر دیتاست نشان‌دهنده‌ی سایز نمونه‌ها و ابعاد هر عکس است.

مثلا در دیتاست MNIST ابعاد ورودی نشان‌دهنده‌ی تعداد داده‌های آموزشی (۶۰۰۰۰)، و سایز هر عکس(۲۸*۲۸) است. ابعاد خروجی نشان‌دهنده‌ی تعداد داده‌های آموزشی(۶۰۰۰۰) و تعداد لیبل‌ها یا همان کلاس‌های خروجی(۱۰کلاس) است.

در دیتاست CIFAR-10 نیز ۵۰۰۰۰ دیتای ورودی برای داده‌های آموزشی داریم که ابعاد هر عکس ۳۲*۳۲ پیکسل است و دارای سه کانال می‌باشند. خروجی نیز به همین تعداد لیبل داریم که شامل ۱۰ کلاس هستند.

در دیتاست FER_2013 نیز ۲۸۷۰۹ دیتای آموزشی و ۷ کلاس داریم.

۴) کلاس ImageDataGenerator به ما این امکان را می دهد که تا زمانی که مدل ما هنوز در حال آموزش است، با ارائه یک مقدار صحیح در آرگومان rotation_range، تصاویر را به صورت تصادفی در هر درجه ای بین ۰ تا ۳۶۰ بچرخانیم. همچنین می توانیم هر تبدیل تصادفی را روی هر تصویر آموزشی که به مدل منتقل می‌شود اعمال کنیم. این نه تنها مدل ما را قوی‌تر می‌کند، بلکه در حافظه سربار نیز صرفه‌جویی می‌کند.

ب)

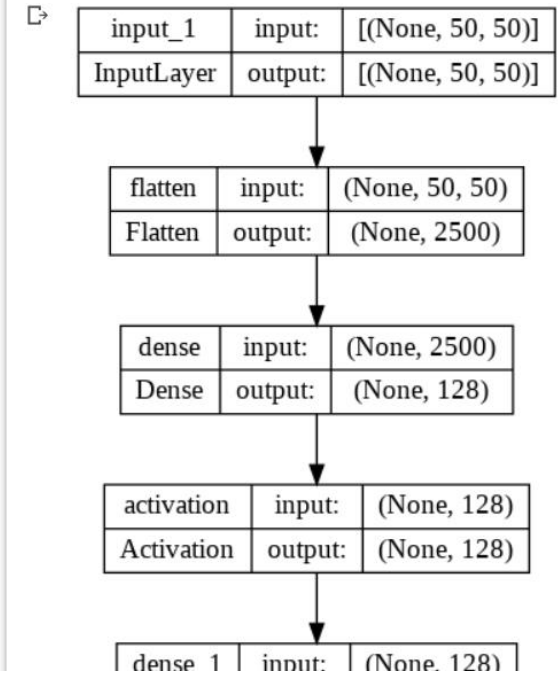
مدل sequential :

```
model_temp_1 = Sequential()
model_temp_1.add(layers.Input(shape=(50,50)))
model_temp_1.add(layers.Flatten())
model_temp_1.add(layers.Dense(units=128))
model_temp_1.add(layers.Activation('relu'))
model_temp_1.add(layers.Dense(units=5))
model_temp_1.add(layers.Activation('softmax'))
```

```
[6]  model_temp_1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 2500)              0

 dense (Dense)               (None, 128)               320128

 activation (Activation)     (None, 128)               0

 dense_1 (Dense)             (None, 5)                 645

 activation_1 (Activation)   (None, 5)                 0

=================================================================
Total params: 320,773
Trainable params: 320,773
Non-trainable params: 0
_________________________________________________________________
```

```
plot_model(
    model_temp_1,
    to_file="structure_1.png",
    show_shapes=True,
    show_layer_names=True,
)
```



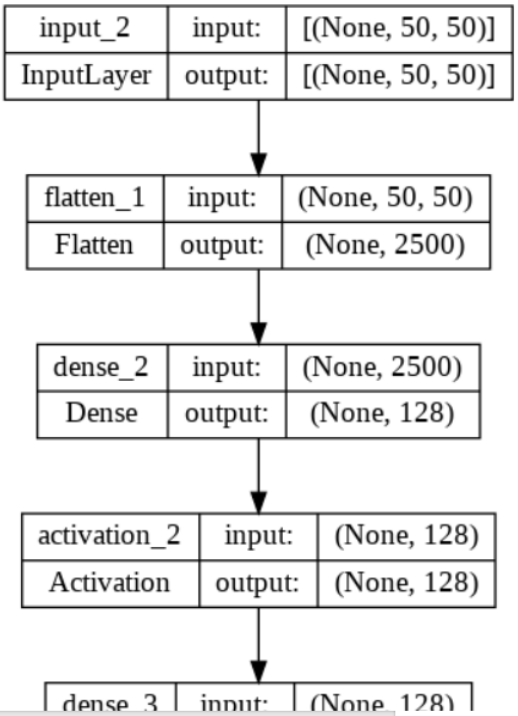
مدل API Function :

```
def model_factory(input_shape, num_classes):
    input_layer = Input(shape=input_shape)
    layer_1 = layers.Flatten()(input_layer)
    layer_2 = Dense(128)(layer_1)
    layer_3 = layers.Activation("relu")(layer_2)
    layer_4 = Dense(num_classes)(layer_3)
    output_layer = layers.Activation("softmax")(layer_4)


    return Model(inputs= input_layer, outputs=output_layer)
```

```
[11] model_template_2.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 50, 50)]          0

 flatten_1 (Flatten)         (None, 2500)              0

 dense_2 (Dense)             (None, 128)               320128

 activation_2 (Activation)   (None, 128)               0

 dense_3 (Dense)             (None, 5)                 645

 activation_3 (Activation)   (None, 5)                 0

=================================================================
Total params: 320,773
Trainable params: 320,773
Non-trainable params: 0
_________________________________________________________________
```

```
[12] plot_model(
         model_template_2,
         to_file="structure_2.png",
         show_shapes=True,
         show_layer_names=True,
     )
```



Summary اطلاعاتی شامل هر لایه، تعداد نورون‌های آن، تعداد وزن‌ها، ابعاد و اسامی لایه‌ها را داراست.

ج)

```
[14] # Write your code here
     sgd_optimizer = SGD(learning_rate=0.01, momentum=0.0, name='SGD')
```

د)

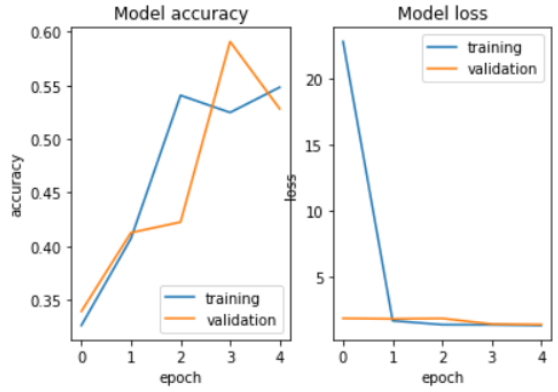
مدل اول:

```
# Write your code here
model_mnist = model_factory(input_shape = (28,28), num_classes = 10)

# Write your code here
model_mnist.compile(loss='categorical_crossentropy', optimizer=sgd_optimizer, metrics=['accuracy'])

# Write your code here
history = model_mnist.fit(
    x_train_1, y_train_1,
    validation_data=(x_test_1, y_test_1),
    batch_size=64,
    epochs=5,
    validation_split=0.2,
    shuffle=True,
)
```

```
Epoch 1/5
938/938 [==============================] - 4s 4ms/step - loss: 22.7738 - accuracy: 0.3262 - val_loss: 1.8369 - val_accuracy: 0.3394
Epoch 2/5
938/938 [==============================] - 3s 3ms/step - loss: 1.6568 - accuracy: 0.4073 - val_loss: 1.7958 - val_accuracy: 0.4127
Epoch 3/5
938/938 [==============================] - 3s 3ms/step - loss: 1.3607 - accuracy: 0.5408 - val_loss: 1.8325 - val_accuracy: 0.4226
Epoch 4/5
938/938 [==============================] - 3s 4ms/step - loss: 1.3482 - accuracy: 0.5248 - val_loss: 1.4040 - val_accuracy: 0.5909
Epoch 5/5
938/938 [==============================] - 3s 4ms/step - loss: 1.3061 - accuracy: 0.5483 - val_loss: 1.3721 - val_accuracy: 0.5283
```

```
ax1.plot(history.history['accuracy'])
ax1.plot(history.history['val_accuracy'])
ax1.set_title('Model accuracy')
ax1.set_ylabel('accuracy')
ax1.set_xlabel('epoch')
ax1.legend(['training', 'validation'], loc='lower right')

ax2.plot(history.history['loss'])
ax2.plot(history.history['val_loss'])
ax2.set_title('Model loss')
ax2.set_ylabel('loss')
ax2.set_xlabel('epoch')
ax2.legend(['training', 'validation'], loc='upper right')
```

```
<matplotlib.legend.Legend at 0x7fe1abd46e90>
```



میزان دقت در داده‌های آموزشی تا ایپاک دوم افزایش، در ایپاک سوم کاهش و در ایپاک سوم به بعد دوباره افزایش یافته است. در داده‌های تست تا ایپاک سوم افزایش دقت و از آن به بعد شاهد کاهش دقت هستیم.

Evaluating Test set

```
# Write your code here
model_mnist.evaluate(x_test_1, y_test_1, batch_size=64)
```

```
157/157 [==============================] - 1s 3ms/step - loss: 1.3721 - accuracy: 0.5283
[1.3721296787261963, 0.5282999873161316]
```

همانطور که قابل ملاحظه است دقت ما روی داده‌ی تست تقریبا ۵۳٪ است.

```
# Write your code here
for i in range(3):
  y_pred = model_mnist.predict(np.array([x_test_1[i]]))
  print("y_predicted: ",y_pred)
  print("y_test: ",y_test_1[i])
  print("---------------------------------")
```

```
1/1 [==============================] - 0s 18ms/step
y_predicted:  [[2.9803159e-15 2.6054654e-14 2.2730036e-07 0.0000000e+00 1.6583941e-26
  8.4005658e-12 0.0000000e+00 9.9999976e-01 0.0000000e+00 1.6570265e-28]]
y_test:  [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
---------------------------------
1/1 [==============================] - 0s 19ms/step
y_predicted:  [[0.04310832 0.02787286 0.07562907 0.1725867  0.04987936 0.06670731
  0.1648519  0.07415597 0.16431308 0.1608955 ]]
y_test:  [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
---------------------------------
1/1 [==============================] - 0s 17ms/step
y_predicted:  [[8.5071105e-26 9.9998546e-01 3.4676497e-16 3.6608537e-14 1.2498444e-08
  2.2267534e-09 1.1050079e-11 1.4579266e-05 7.9547084e-33 1.0461070e-14]]
y_test:  [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
---------------------------------
```

Predicting Some samples from Test set

```
# Write your code here
for i in range(3):
  y_pred = model_mnist.predict(np.array([x_test_1[i]]))
  print("y_predicted: ",np.argmax(y_pred))
  print("y_test: ",np.argmax(y_test_1[i]))
  print("---------------------------------")
```

```
1/1 [==============================] - 0s 18ms/step
y_predicted:  7
y_test:  7
---------------------------------
1/1 [==============================] - 0s 18ms/step
y_predicted:  3
y_test:  2
---------------------------------
1/1 [==============================] - 0s 22ms/step
y_predicted:  1
y_test:  1
---------------------------------
```

از سه نمونه یک نمونه با خطاست.

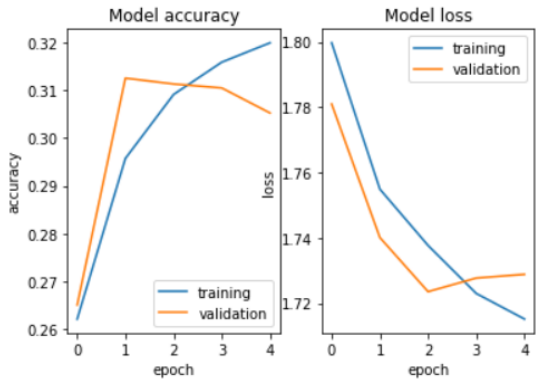
مدل دوم:

```
# Write your code here
model_fer = model_factory(input_shape=(48,48,1), num_classes=7)

# Write your code here
model_fer.compile(loss='categorical_crossentropy', optimizer=sgd_optimizer, metrics=['accuracy'])

# Write your code here
history = model_fer.fit(
    train_set,
    validation_data=(train_set),
    batch_size=64,
    epochs=5
)
```

```
Epoch 1/5
449/449 [==============================] - 45s 99ms/step - loss: 1.7997 - accuracy: 0.2621 - val_loss: 1.7810 - val_accuracy: 0.2651
Epoch 2/5
449/449 [==============================] - 64s 143ms/step - loss: 1.7549 - accuracy: 0.2957 - val_loss: 1.7401 - val_accuracy: 0.3125
Epoch 3/5
449/449 [==============================] - 43s 97ms/step - loss: 1.7377 - accuracy: 0.3091 - val_loss: 1.7235 - val_accuracy: 0.3113
Epoch 4/5
449/449 [==============================] - 43s 96ms/step - loss: 1.7230 - accuracy: 0.3159 - val_loss: 1.7277 - val_accuracy: 0.3105
Epoch 5/5
449/449 [==============================] - 43s 97ms/step - loss: 1.7152 - accuracy: 0.3199 - val_loss: 1.7288 - val_accuracy: 0.3052
```

```
ax1.plot(history.history['accuracy'])
ax1.plot(history.history['val_accuracy'])
ax1.set_title('Model accuracy')
ax1.set_ylabel('accuracy')
ax1.set_xlabel('epoch')
ax1.legend(['training', 'validation'], loc='lower right')

ax2.plot(history.history['loss'])
ax2.plot(history.history['val_loss'])
ax2.set_title('Model loss')
ax2.set_ylabel('loss')
ax2.set_xlabel('epoch')
ax2.legend(['training', 'validation'], loc='upper right')
```

```
<matplotlib.legend.Legend at 0x7fe1a38b9150>
```



دقت روی نمونه‌ی آموزشی در حال افزایش اما روی نمونه‌ی تست ابتدا در حال افزایش و پس از ایپاک اول به صورت کاهشی است.

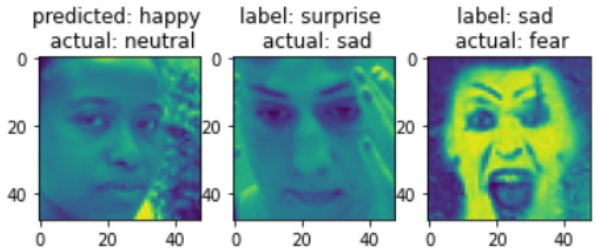
```
model_fer.evaluate(
    x=test_set,
)
```

```
113/113 [==============================] - 3s 26ms/step - loss: 1.7003 - accuracy: 0.3291
[1.7002501487731934, 0.32906103134155273]
```

دقت مدل روی داده‌ی تست تقریبا ۳۳٪ است.

```
# use the created array to output your multiple images.
axarr[0].imshow(np.squeeze(test_set[0][0][5]))
label_1 = label_list[test_set[0][1][5].argmax(axis=-1)]
axarr[1].imshow(np.squeeze(test_set[0][0][29]))
label_2 = label_list[test_set[0][1][29].argmax(axis=-1)]
axarr[2].imshow(np.squeeze(test_set[0][0][10]))
label_3 = label_list[test_set[0][1][10].argmax(axis=-1)]

# Show predicted label for each image
axarr[0].set_title(f'predicted: {label_list[labels[5]]} \n actual: {label_1}')
axarr[1].set_title(f'label: {label_list[labels[29]]} \n actual: {label_2}')
axarr[2].set_title(f'label: {label_list[labels[10]]} \n actual: {label_3}')
```

```
113/113 [==============================] - 3s 23ms/step
Text(0.5, 1.0, 'label: sad \n actual: fear')
<Figure size 432x288 with 0 Axes>
```



همانطور که واضح است لیبل پیش‌بینی‌شده برای هر سه نمونه با لیبل واقعی‌شان متفاوت است. دلیل این موضوع این است که ما از داده‌های آموزشی برای validation استفاده کردیم و مدل ما روی داده‌های آموزشی overfit شده است و برای داده‌های تست جدید دقت خوبی ندارد. در حالی که در مدل اول از داده‌های تست برای validation استفاده کردیم و دقت بالاتری داشت.