

تمرین سری اول

سینا اسکندری 97521054

-1

(الف)

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

$$P(a|0) = \frac{4+1}{6+2} = 0.625 \quad P(b|0) = \frac{2+1}{6+2} = 0.375$$

$$P(a|1) = \frac{0+1}{2+2} = 0.25 \quad P(b|1) = \frac{2+1}{2+2} = 0.75$$

(ب)

چون مخرج کسرها ثابت هستند از نوشتن آنها صرف نظر می‌کنم و فقط صورت‌ها را مقایسه می‌کنم.

$$P(0|aaba) = P(0) * P(a|0)^4 * P(b|0) = 0.75 * (0.625)^4 * 0.375 = 0.04$$

$$P(1|aaba) = P(1) * P(a|1)^4 * P(b|1) = 0.25 * (0.25)^4 * 0.75 = 0.0007$$

$$P(0|b) = P(0) * P(b|0) = 0.75 * 0.375 = 0.28$$

$$P(1|b) = P(1) * P(b|1) = 0.25 * 0.75 = 0.18$$

$$P(0|bba) = P(0) * P(b|0)^2 * P(a|0) = 0.75 * (0.375)^2 * 0.625 = 0.0065$$

$$P(1|bba) = P(1) * P(b|1)^2 * P(a|1) = 0.25 * (0.75)^2 * 0.25 = 0.035$$

$$P(0|bbbb) = P(0) * P(b|0)^4 = 0.75 * (0.375)^4 = 0.015$$

$$P(1|bbbb) = P(1) * P(b|1)^4 = 0.25 * (0.75)^4 = 0.08$$

Data	Class
aaba	0
b	0
bba	1
bbbb	1

-3

(الف)

دیتاست MNIST شامل تصاویری 28 در 28 پیکسل از اعداد 0 تا 9 است که به حالات مختلف نوشته شده‌اند. این دیتاست شامل 60000 داده آموزش و 10000 داده تست است.

دیتاست CIFAR-10 شامل تصاویری 32 در 32 پیکسل با 10 لیبل شامل هواپیما، اتومبیل، پرنده، گربه، آهو، سگ، قوریانگه، اسب، کشتی و کامیون است. این دیتاست شامل 50000 داده آموزش و 10000 داده تست است.

دیتاست FER-2013 شامل تصاویری 48 در 48 پیکسل از صورت افراد است که دارای 7 کلاس برای حالات مختلف صورت است. این دیتاست شامل 28709 داده آموزش و 3589 داده تست است. لینک دانلود این دیتاست در نوتبوک اشتباه بود و آن را تغییر دادم.

تابع `to_categorical` لیبل داده‌ها را به بردارهایی که مپ می‌کند که یکی از المان‌های آن 1 و بقیه 0 هستند (one-hot). این کار برای این است که در تابع `loss` فاصله لیبل‌ها زیاد نشوند و مدل بهتر `train` شود.

ابعاد `x_train` به عنوان مثال در MNIST برابر (10000, 28, 28) است که نشان دهنده این است که 10000 داده داریم که تعداد `28*28` فیچر دارند. در CIFAR-10 که (50000, 32, 32, 3) هست یعنی 50000 داده داریم که هر تصویر 32 در 32 است و چون `rgb` است دارای 3 کانال رنگی می‌باشد.

ابعاد `y_train` نشان دهنده تعداد داده و ابعاد هر لیبل است یعنی (10000, 10) یعنی 10000 داده لیبل‌ی با ابعاد (10,) دارند که به خاطر استفاده از تابع `to_categorical` اینجوری شدند.

از `ImageDataGenerator` وقتی استفاده می‌کنیم که بخواهیم تصاویر را به تنسور تبدیل کنیم همچنین می‌توان تغییراتی نیز اعمال کرد روی تصاویر.

(ب)

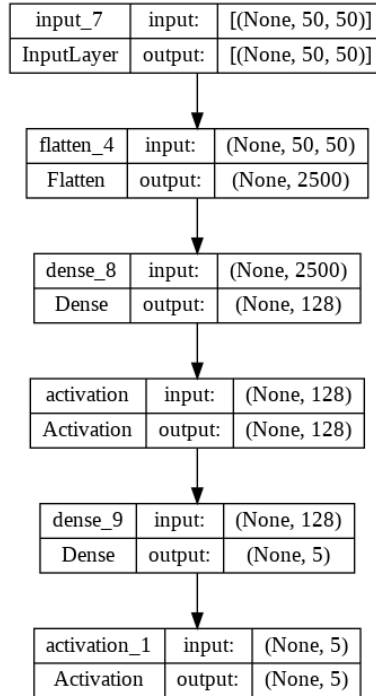
مدل `sequential` به صورت زیر است:

```
model_temp_1 = Sequential()

# Input Layer
# Write your code here
model_temp_1.add(layers.Input(shape=(50, 50)))
model_temp_1.add(layers.Flatten())

# Hidden Layer
# Write your code here
model_temp_1.add(layers.Dense(units=128))
model_temp_1.add(layers.Activation('relu'))

# Output Layer
# Write your code here
model_temp_1.add(layers.Dense(units=5))
model_temp_1.add(layers.Activation('softmax'))
```



:functional مدل

```

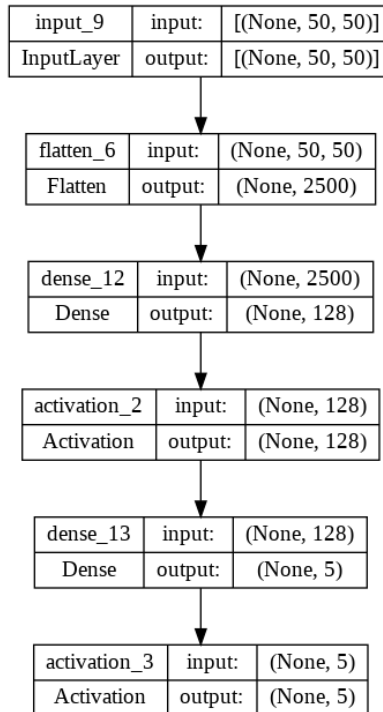
def model_factory(input_shape, num_classes):
    # Input Layer
    # Write your code here
    i = layers.Input(shape=input_shape)
    x = layers.Flatten()(i)

    # Hidden Layer
    # Write your code here
    x = layers.Dense(units=128)(x)
    x = layers.Activation('relu')(x)

    # Output Layer
    # Write your code here
    x = layers.Dense(units=num_classes)(x)
    o = layers.Activation('softmax')(x)

    return Model(inputs=i, outputs=o)

```



در summary مدل می توان ابعاد ورودی و خروجی هر لایه شبکه و تعداد پارامترهای w و b را مشاهده کرد.

(ج)

```
# Write your code here
sgd_optimizer = SGD(learning_rate=0.01)
```

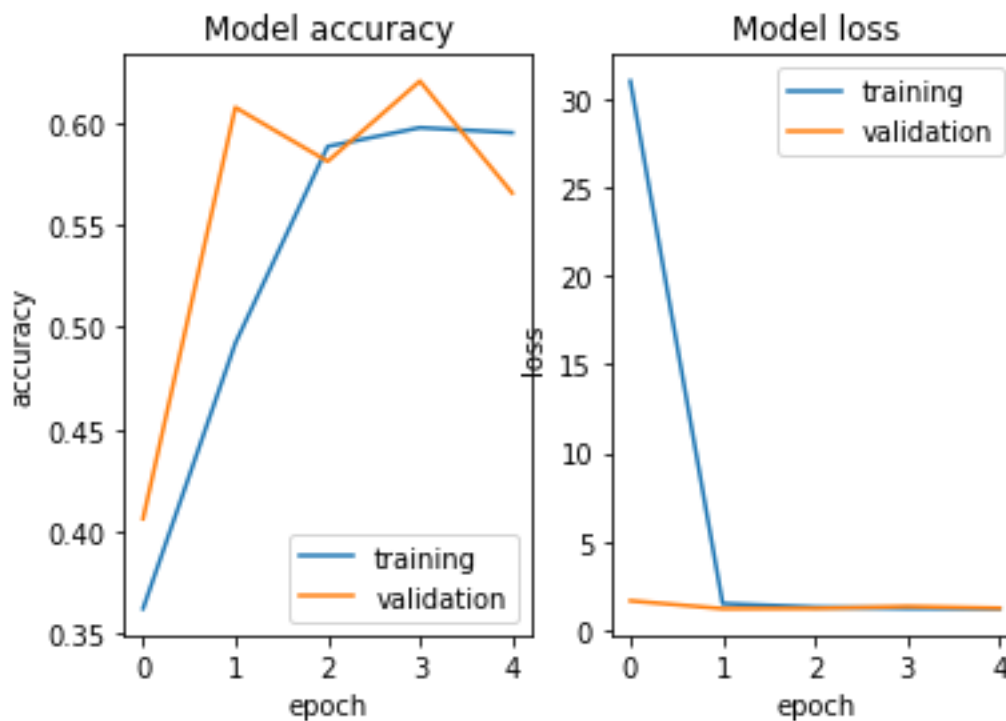
(د)

دیتاست MNIST:

```
# Write your code here
model_mnist = model_factory(
    input_shape=x_train_1[0].shape,
    num_classes=y_test_1.shape[-1]
)

# 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,
    batch_size=64,
    epochs=5,
    validation_split=0.2
)
```



با توجه به نمودارها می توان دید که دقت روی داده train در حال افزایش بوده تا epoch 3 و تا epoch 4 کاهش یافته است. در داده val نیز صعودی و نزولی در چند مرحله بوده است.

```
[54] # Write your code here
model_mnist.evaluate(
    x_test_1, y_test_1
)




313/313 [=====] - 1s 2ms/step - loss: 1.2638 - accuracy: 0.5635
[1.2637537717819214, 0.5634999871253967]
```

مقدار دقت و loss نیز بر روی دیتا تست 0.56 و 1.26 می باشد.

```
# Write your code here
y_pred_1 = model_mnist.predict(x_test_1)
np.set_printoptions(precision=2, linewidth=200)
for i in range(3):
    print(y_test_1[i])
    print(y_pred_1[i])
    plt.figure(figsize=(1, 1))
    plt.imshow(x_test_1[i], cmap='gray')
    plt.xticks([])
    plt.yticks([])
    plt.show()
```

برای پیشبینی مدل روی 3 داده از این کد استفاده می کنیم و نتایج آن به صورت زیر می باشد.

```

313/313 [=====] - 1s 4ms/step
[0. 0. 0. 0. 0. 0. 1. 0. 0.]
[4.97e-33 6.88e-26 0.00e+00 9.89e-26 0.00e+00 1.54e-28 0.00e+00 1.00e+00 0.00e+00 2.67e-10]

[0. 0. 1. 0. 0. 0. 0. 0. 0.]
[7.08e-21 0.00e+00 1.00e+00 1.59e-25 5.22e-17 5.52e-15 7.58e-12 5.60e-10 1.76e-19 6.99e-23]

[0. 1. 0. 0. 0. 0. 0. 0. 0.]
[1.89e-22 1.00e+00 3.25e-10 1.60e-15 5.71e-10 1.77e-11 5.84e-09 1.72e-04 2.31e-06 1.55e-06]


```

مشخص است که شبکه در این 3 مورد به خصوص درست عمل کرده.

دیتاست FER-2013:

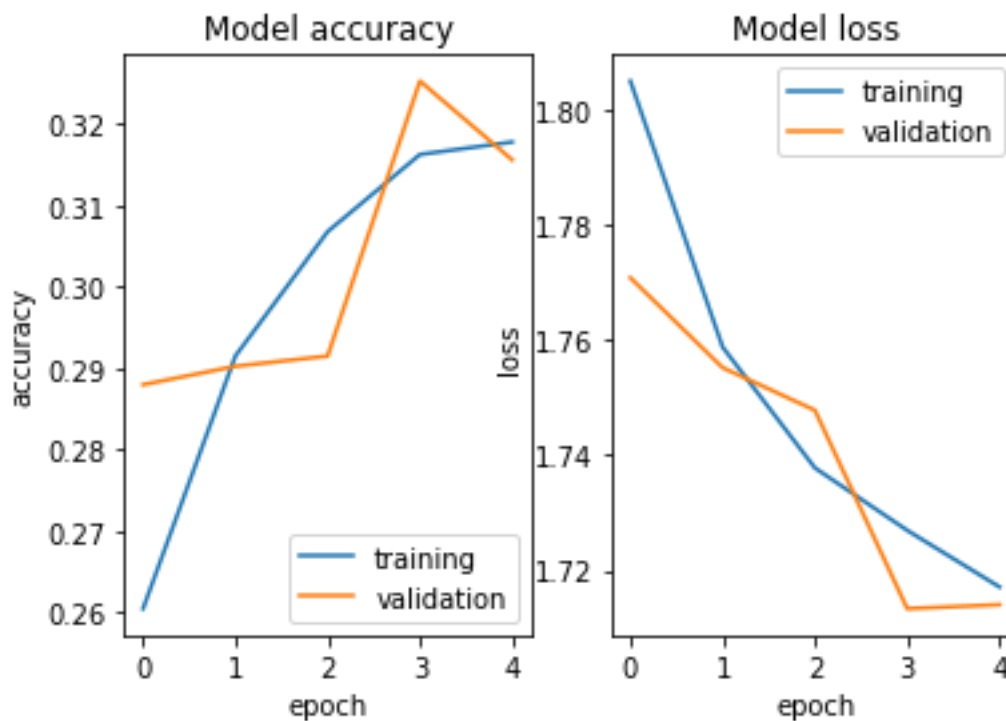
```

# Write your code here
model_fer = model_factory(
    input_shape=(48, 48),
    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(
    x=train_set,
    epochs=5,
    validation_data=train_set
)

```



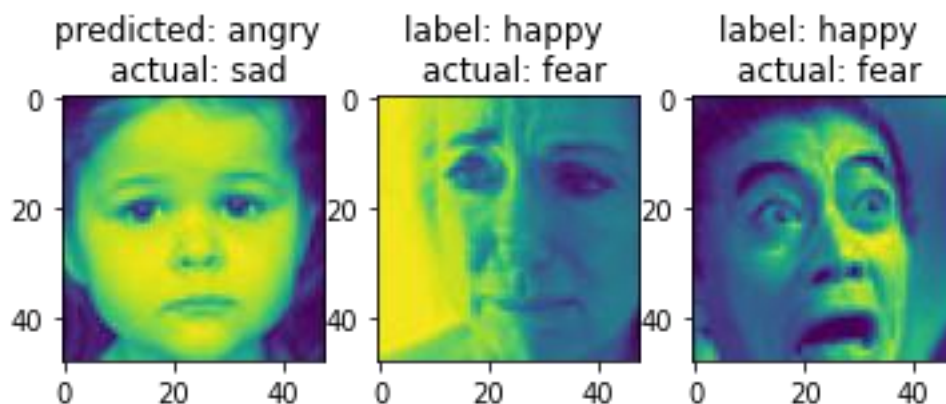
دقت روی داده train همواره زیاد شده ولی هرچه جلوتر رفته با شیب کمتری این اتفاق رخ داده. دقت داده val نیز ابتدا با شیب کم و بعد با شیب زیاد، افزایش یافته ولی در epoch آخر کاهش یافته است.

```
[66] model_fer.evaluate(
      x=test_set,
    )
```

```
113/113 [=====] - 3s 27ms/step - loss: 1.6945 - accuracy: 0.3338
[1.6944997310638428, 0.33379772305488586]
```

دقت مدل بسیار پایین می باشد.

وقتی چند نمونه از داده تست را پیشبینی می کنیم به نتایج زیر می رسیم.



هیچکدام از تصاویر درست پیشبینی نشده اند.

منابع:

• منابع ذکر شده در نوتیوک

• <https://machinelearningmastery.com/keras-functional-api-deep-learning>