Deep learning

Homework 1

Parsa eissazadeh - 97412364

سوال 1

الف)

$P(1) = 1/4$

$P(0) = 3/4 = 0.75$

$P(a|0) = 2/3$

$P(b|0) = 1/3$

$P(a|1) = 0$

$P(b|1) = 1$

ب)

برای هر نمونه ، باید احتمال تعلق به هر کلاس را به دست بیاوریم سپس نمونه متعلق به کلاسی است که احتمال بیشتری دارد .

$P (Y = 0 | aabaa) = P(Y = 0) * P (a | 0)^4 * P (b | 0) = 0.75 * 0.66^4 * 0.3333$

$P (Y = 1 | aabaa) = P(Y = 1) * P (a | 1)^4 * P (b | 1) = 0$

قطعا متعلق به کلاس 0 چون تساوی دوم صفر است .

$P (Y = 1 | bb) = P(Y = 1) * P (b | 1)^2 = 0.25 * 1$

$P (Y = 0 | bb) = P(Y = 0) * P (b | 0)^2 = 0.75 * 0.3333^2 = 0.25 * 0.33 < 0.25$

متعلق به کلاس 1 .

$P(Y=1 \mid bba) = P(Y=1) * P(a \mid 1) * P(b \mid 1)^2 = 0$

$P(Y=0 \mid bba) = P(Y=0) * P(a \mid 0) * P(b \mid 0)^2 = 0.75 * 0.66 * 0.3333^2 = 0.055$

متعلق به کلاس 0 .

$P(Y=1 \mid bbbb) = P(Y=1) * P(b \mid 1)^4 = 0.25 * 1 = 0.25$

$P(Y=0 \mid bbbb) = P(Y=0) * P(b \mid 0)^4 = 0.75 * 0.3333^4 = 0.009$

متعلق به کلاس 1 .

سوال 3

الف)

MNIST : این کتابخانه مجموعه بزرگی از عکس های اعداد نوشته شده با دست هستند که ده کلاس ارقام یک تا ده را شامل می شوند . اندازه عکس ها 28 در 28 هستند .

CIFAR-10 : مجموعه ای 60000 از عکس هایی به اندازه 32*32 که ده کلاس دارد که هر کلاس یک شی یا یک حیوان است .

FER-2013 : مجموعه ای از 30000 عکس های صورت در حالت های مختلف . به اندازه 28*28 .

از این dataset برای train کردن اولیه مدل های یادگیری ماشین استفاده می شود .

در صورتی که اختلاف داده ی پیش بینی شده توسط شبکه و برچسب آن داده (loss) از حدی بیشتر باشد ، شبکه جریمه می شود . حال هر چه این اختلاف بیشتر باشد شبکه بیشتر جریمه می شود . در یک شبکه که دسته بندی چند کلاسه می کند ، اگر به جای کلاس 2 ، کلاس 3 انتخاب شود یا کلاس 8 نباید تفاوتی کند و شبکه به مقدار یکسانی جریمه شود .

اگر خروجی به صورت یک عدد باشد خواهیم داشت :

$$|8-2| > |3-2|$$

و این اختلاف شبکه را الکی جریمه می کند ، اما اگر به صورت برداری ذخیره شوند اختلاف 2و 3 برابر خواهد بود با اختلاف 2 و 8 (اختلافشان برابر است با یک مولفه در بردار)

$$2 : [0,0,1,0,0,0,0,0,0,0]$$

3 : [0,0,0,0,0,0,0,0,0,0]

8 : [0,0,0,0,0,0,0,0,1,0]

متد to_categorical خروجی ها را به این شکل برای ما برداری می کند و به ازای هر کلاس یک نورون خروجی اضافه می کند .

در یک مدل یادگیری عمیق ، در ابتدا داده ها با برچسب هایشان به منظور یادگیری شبکه وارد آن می شوند و وزن های شبکه آپدیت می شود ، x_train داده های ورودی هستند و y_train برچسب های آن داده ها هستند ، یعنی خروجی هایی هستند که انتظار داریم شبکه به ما بازگرداند .

در نتیجه در shape آن ها ، مولفه اول تعداد عکس هاست و دو مولفه بعدی اندازه هر عکس (تعداد پیکسل ها در سطر و ستون را نمایش می دهد .)

یک shape با سه مولفه . در cifar-10 اما shape ، چهار مولفه دارد . مولفه چهارم به دلیل سه بعدی بودن عکس ها است . اما بقیه dataset ها که تنها یک کانال دارند در مولفه ۴ ام تنها ۱ هستند برای همین آن مولفه را ندارند .

دلیل استفاده از ImageDataGenerator :

جمع آوری dataset تمیز یکی از مهم ترین و چالش برانگیز ترین مراحل یک پروژه یادگیری ماشین است .
هنگامی که با داده های تصویری کار میکنیم ، می توانیم یا ایجاد تغییراتی در عکس هایی که داریم ، به عکس ها و در نتیجه داده های بیشتری برسیم برای یادگیری مدل .

این تغییر های میتوانند یکی از این ها باشند :

- چرخاندن
- نویز دادن
- تغییر دادن مقیاس عکس
- برش زدن

هر یک از این تغییرات به صورت رندم روی عکس اعمال می شوند . تغییر ها هرگز نباید داده ای به عکس اضافه بکنند .

ب)

کد مدل sequential به این شکل 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, activation='relu'))

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

Summary به شکل زیر است :

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_3 (Flatten)         (None, 2500)              0

 dense_6 (Dense)             (None, 128)               320128

 activation (Activation)     (None, 128)               0

 dense_7 (Dense)             (None, 5)                 645

 activation_1 (Activation)   (None, 5)                 0

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

اطلاعاتی که در summary قابل مشاهده است :

تعداد لایه ها ، متد هر لایه ، نام هر لایه ، تعداد نورون های موجود در هر لایه .

کد قسمت functional :

```
def model_factory(input_shape, num_classes):
    # Input Layer
    # Write your code here
    inputs = keras.Input(shape=input_shape)
    flatten = layers.Flatten(data_format=None)(inputs)

    # Hidden Layer
    # Write your code here
    dense = layers.Dense(128, activation="relu")
    x = dense(flatten)

    # Output Layer
    # Write your code here
    dense = layers.Dense(num_classes, activation="softmax")
    outputs = dense(x)

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

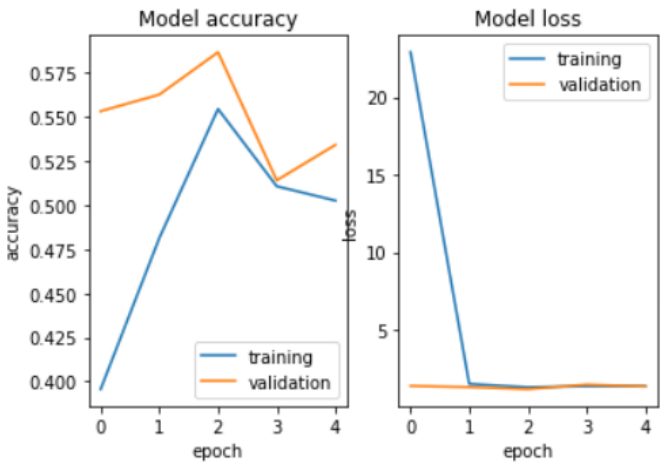
Summary به این شکل هستش :

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_8 (InputLayer)        [(None, 50, 50)]          0

 flatten_7 (Flatten)         (None, 2500)              0

 dense_11 (Dense)            (None, 128)               320128

 activation_5 (Activation)   (None, 128)               0

 dense_12 (Dense)            (None, 5)                 645

 activation_6 (Activation)   (None, 5)                 0

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

Plot هر دو نمودار شبیه عکس سوال بود که در فایل colab قابل مشاهده است .

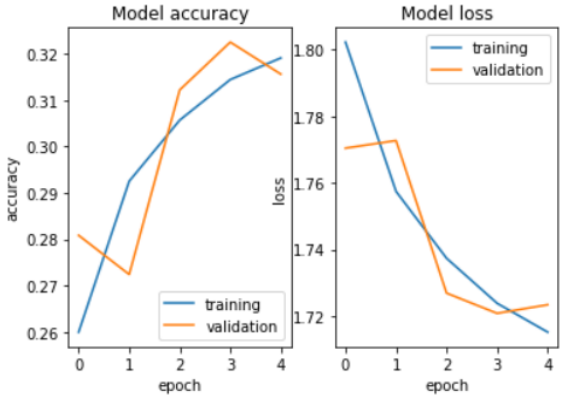
میزان دقت روی داده train و داده val در طی فرایند آموزش در مدل mnist :

طبق نمودار ها :



همانطور که مشاهده می شود مقدار loss کمتر شده و مقدار model accuracy افزایش یافته .

در مدل FER هم به همینگونه است :



امتحان کردن چند نمونه :

کد به شکل زیر بود :

```
prediction = model_mnist.predict(
    x = x_test_1
)
print()
for index in range(13,16):

  print('actual:' , np.argmax(prediction[index] , axis=0) )
  print('label:' , np.argmax(y_test_1[index] , axis=0) )
  print("====================================")
```

اما نتیجه ها چندان جالب نبود :

```
actual: 0
label: 0
====================================
actual: 1
label: 1
====================================
actual: 3
label: 5
====================================
```

حدود یک سوم داده ها اشتباه است . شاید به خاطر کم بودن تعداد epoch هاست که مدل وقت کافی برای یادگیری نداشت .