Deep Learning
Dr. DavoodAbadi
Fall 2022
Hoorieh Sabzevari - 98412004
HW3

۱.

الف) Overfittingیک خطای مدل‌سازی در آمار است و زمانی رخ می‌دهد که یک تابع خیلی نزدیک به مجموعه محدودی از نقاط داده fit شده باشد. در نتیجه، این مدل تنها در ارجاع به مجموعه‌ی داده های اولیه خود مفید است و نه به هیچ مجموعه داده دیگری. هنگامی که الگوریتم های یادگیری ماشین ساخته می‌شوند، از مجموعه داده‌های نمونه برای آموزش مدل استفاده می کنند. با این حال، زمانی که مدل برای مدت طولانی روی داده‌های نمونه آموزش می‌یابد یا زمانی که مدل بسیار پیچیده است، می‌تواند شروع به یادگیری اطلاعات نامربوط و جزئیات در مجموعه‌ی داده‌ها کند. وقتی مدل این اطلاعات را به خاطر می‌سپارد و خیلی نزدیک به مجموعه آموزشی منطبق می‌شود، مدل overfit می‌شود و نمی‌تواند به خوبی روی داده‌های جدید تعمیم یابد. اگر یک مدل نتواند به خوبی روی داده‌های جدید تعمیم یابد، آنگاه نمی‌تواند وظایف طبقه‌بندی یا پیش‌بینی را که برای آن در نظر گرفته شده است، انجام دهد. (لینک)

ج) از راه‌های جلوگیری از آن می‌توان به موارد زیر اشاره کرد:

۱) تقسیم و شکستن داده‌های آموزشی به تعدادی fold و اجرای مدل برای هرکدام از آن‌ها. خطای کل در این روش برابر است با میانگین خطاهای هر بخش.

۲) افزودن داده به مجموعه‌ی داده‌های آموزشی به همراه قوی‌تر کردن مدل

۳) داده‌افزایی یا data augmentation: به روش‌های گوناگون انجام می‌شود تا داده‌ها هم زیاد شود و هم متفاوت از هم دیده شوند.

۴) ساده‌سازی داده‌ها

۵) افزودن نویز به دیتای ورودی

۶) Dropout : در هر تکرار، به صورت تصادفی مقدار تعدادی نورون را صفر می‌کند.

۷) افزودن نویز در خروجی مطلوب

۸) جریمه کردن اندازه‌ی پارامترها (L1 , L2)

ب) زیرا مدل روی داده‌های آموزشی overfit شده و تمام جزئیات بی‌اهمیت را نیز یاد گرفته است. لذا اگر داده‌ی جدیدی را ببیند چون احتمالا آن جزئیات را ندارد با خطای زیادی مواجه خواهیم شد. گرچه خط قرمز در شکل۱ به بهترین وجه داده‌های آموزشی را دنبال می‌کند، اما بیش از اندازه به آن داده‌ها وابسته است و احتمالاً در مقایسه با شکل۲ ، میزان خطای بالاتری در داده‌های جدید و دیده‌نشده خواهد داشت.

الف)

Forward:

$$z = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5$$

$$L = (y-z)^2 + \frac{\lambda}{2}\left(\sum_{i=1}^{5} w_i^2\right)$$

Backward:

$$\frac{\partial L}{\partial z} = -2(y-z), \quad \frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_i} = -2x_i(y-z)$$

update: $w_i = (1-\eta\lambda)w_i - \eta\frac{\partial L}{\partial w_i}$

داده‌ی اول: $(1, 20)$

Forward: $z = 3 \Rightarrow L = (20-3)^2 + \frac{0.9}{2}\left(\sum_{i=1}^{5} w_i^2\right)$

$$= (20-3)^2 + \frac{0.9}{2}(1+4+9+4+1) = 297.55$$

Backward:

$$\frac{\partial L}{\partial z} = -2(17) = -34 \qquad \frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_1} = -2\times 17 = -34$$

به‌طور مشابه: $\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial w_5} = -34$

update: $w_1 = (1-(0.1\times 0.9)) - 0.1\times(-34) = 4.31$

$w_2 = (1-0.09)\times 2 + 3.4 = 5.22$

$w_3 = (0.91)\times 3 + 3.4 = 6.13$

$w_4 = (0.91) \times (-2) + 3.4 = 1.58$

$w_5 = (0.91) \times (-1) + 3.4 = 2.49$

ب) براساس روابط بدست آمده در بخش الف، افزایش λ باعث افزایش مقدار loss و در نتیجه آن باعث افزایش تغییر وزن θ می‌شود. لذا با کاهش آن موارد بالا نیز کم می‌شوند. هم چنین با افزایش λ ممکن است واریانس بالا و overfitting برطرف شود و کاهش λ سبب برطرف کردن بایاس بالا و underfitting می‌شود.

۳. ابتدا تصاویر را خوانده و مجموعه‌ی عکس‌ها و لیبل‌ها را می‌سازیم.

```
images = []
labels = []
classnames = ['cat', 'dog']
folder = '/content/data'
for filename in os.listdir(folder):
    img = cv2.imread(os.path.join(folder, filename))
    img = cv2.resize(img, (28, 28))
    images.append(img)
    if filename.split('.')[0]=="cat":
      labels.append(0)
    else:
      labels.append(1)
```

تصاویر را نرمالایز کرده و به آرایه تبدیل می‌کنیم. سپس روی آن‌ها flatten زده و به صد بردار ۲۳۵۲ تایی تبدیل می‌کنیم. لیبل‌ها را نیز به آرایه تبدیل می‌کنیم.

```
images = np.divide(images, 255)
x_train = np.array(images)
x_train = x_train.flatten().reshape(100, 2352)
y_train = np.array(labels)
```

```
print(x_train.shape)
print(y_train.shape)
```

```
(100, 2352)
(100,)
```

در این مرحله مدل خود را بر اساس معیارهای مناسب کلاس‌بندی دوکلاسه انتخاب می‌کنیم. در لایه‌ی آخر از تابع فعالساز sigmoid استفاده می‌کنیم.

```
model1 = Sequential()
model1.add(Dense(512, input_shape=(2352,), activation='relu'))
model1.add(Dense(10, kernel_initializer='normal', activation='relu'))
model1.add(Dense(1, kernel_initializer='normal', activation='sigmoid'))
```

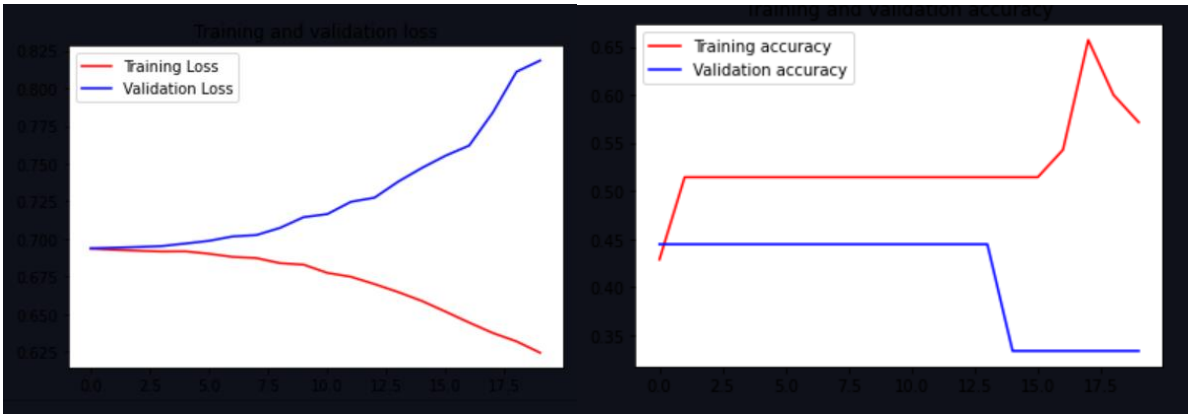
از تابع train_test_split() برای تقسیم دیتاست به دو مجموعه‌ی آموزشی و تست استفاده می‌کنیم.

```
x_train, x_test, y_train, y_test = train_test_split(x_train, y_train, test_size=0.33)
```

مدل خود را کامپایل و فیت می‌کنیم. از تابع ضرر binary_crossentropy و تابع بهینه‌ساز adam و متریک accuracy استفاده می‌کنیم.

```python
model1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# uncomment below and complete the code
history = model1.fit(
      x_train, y_train,
      epochs=20,
      batch_size=5,
      validation_split=0.2,
      shuffle=True,
      verbose=1)
#########################
```

```
7/7 [==============================] - 0s 14ms/step - loss: 0.6749 - accuracy: 0.5143 - val_loss: 0.7247 - val_accuracy: 0.4444
Epoch 13/20
...
Epoch 19/20
7/7 [==============================] - 0s 15ms/step - loss: 0.6320 - accuracy: 0.6000 - val_loss: 0.8110 - val_accuracy: 0.3333
Epoch 20/20
7/7 [==============================] - 0s 16ms/step - loss: 0.6245 - accuracy: 0.5714 - val_loss: 0.8185 - val_accuracy: 0.3333
```




```python
model1.evaluate(x_test, y_test)
```

```
1/1 [==============================] - 0s 27ms/step - loss: 0.7719 - accuracy: 0.3913

[0.7718790173530579, 0.3913043439388275]
```

همانطور که می‌بینیم دقت ما به ۳۹٪ رسیده است.

در بخش دوم، عملیات data augmentation را به صورت دستی اعمال می‌کنیم و نتیجه را با قسمت اول مقایسه می‌کنیم.

```python
aug_images = []
for img in images:
    aug_images.append(img)

    img2 = horizontal_shift(image)
    aug_images.append(img2)

    img3 = vertical_shift(image)
    aug_images.append(img3)

    img4 = brightness(image, 10, 30)
    aug_images.append(img4)

    img5 = zoom(image, 0.3)
    aug_images.append(img5)

    img6 = channel_shift(image, 5)
    aug_images.append(img6)

    img7 = horizontal_flip(image)
    aug_images.append(img7)

    img8 = vertical_flip(image)
    aug_images.append(img8)
```

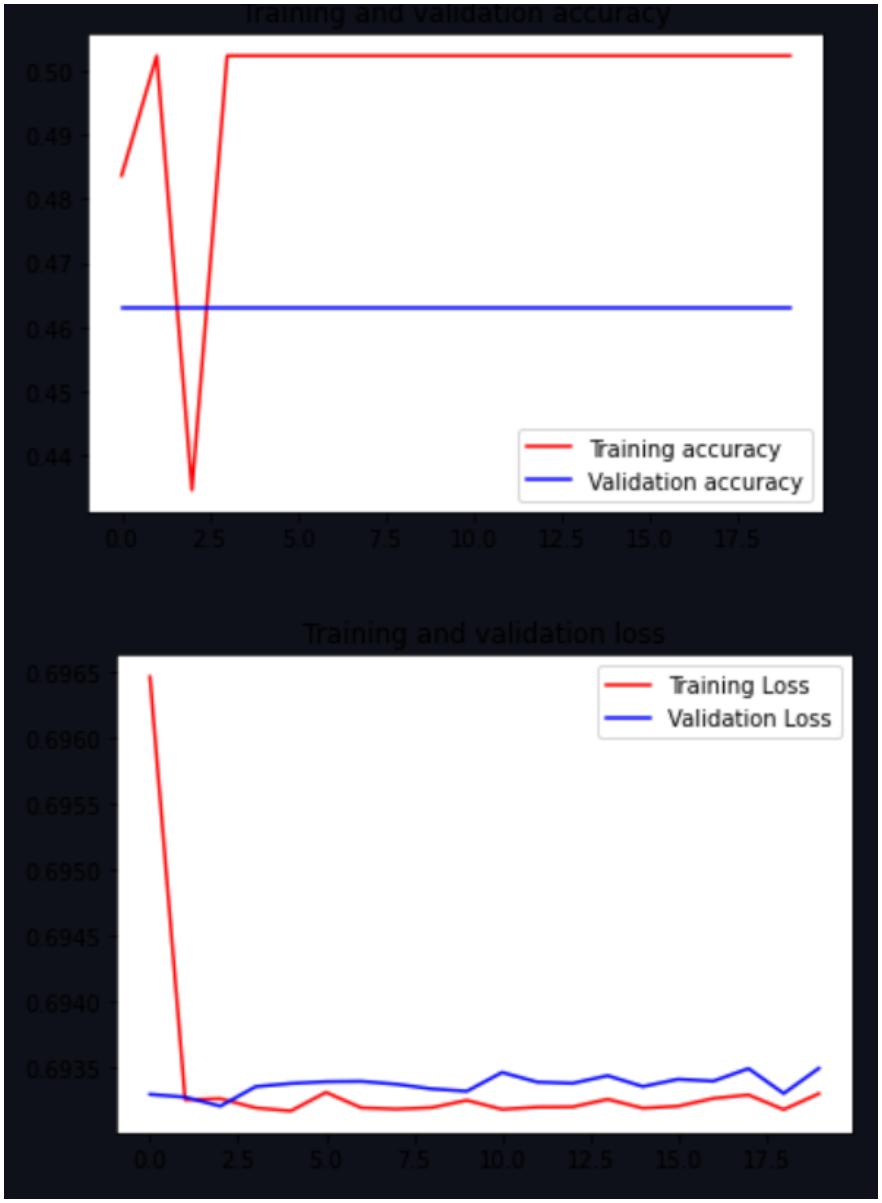
هر لیبل را نیز ۸ بار تکرار کرده و به لیست جدیدی اضافه می‌کنیم.

```python
aug_labels = []
for l in labels:
  for i in range(0, 8):
    aug_labels.append(l)
```

```python
print(x_train.shape)
print(y_train.shape)
```

```
(800, 2352)
(800,)
```

```
Epoch 13/20
...
Epoch 19/20
86/86 [==============================] - 1s 10ms/step - loss: 0.6932 - accuracy: 0.5023 - val_loss: 0.6933 - val_accuracy: 0.4630
Epoch 20/20
86/86 [==============================] - 1s 10ms/step - loss: 0.6933 - accuracy: 0.5023 - val_loss: 0.6935 - val_accuracy: 0.4630
```




```
model2.evaluate(x_test, y_test)
```

```
9/9 [==============================] - 0s 5ms/step - loss: 0.6931 - accuracy: 0.5114

[0.6930540800094604, 0.5113636255264282]
```

همانطور که می‌بینیم دقت در داده‌های تست به ۵۱٪ رسید.

در بخش امتیازی هم همانطور که در فایل نوت‌بوک هست، دیتاست را با استفاده از تابع آماده‌ی keras درست کرده و با استفاده از توابع آماده عملیات data augmentation را انجام می‌دهیم. سپس مدل قبلی را دوباره آموزش داده و می‌بینیم که دقت به ۸۶٪ رسیده است.

```
Epoch 13/20
...
Epoch 19/20
11/11 [==============================] - 0s 16ms/step - loss: 0.4408 - accuracy: 0.8491 - val_loss: 0.7461 - val_accuracy: 0.4286
Epoch 20/20
11/11 [==============================] - 0s 16ms/step - loss: 0.4081 - accuracy: 0.8679 - val_loss: 1.2465 - val_accuracy: 0.4286
```

ها با تولید نقاط داده ها برای افزایش مصنوعی مقدار داده‌ها مجموعه ای از تکنیک‌افزایش داده های ها یا استفاده از مدل‌های موجود است. این شامل ایجاد تغییرات کوچک در داده‌جدید از داده اگر مجموعه داده در یک مدل یادگیری یادگیری عمیق برای تولید نقاط داده جدید است.
برای مدل‌های یادگیری ماشین، ماشینی غنی و کافی باشد، مدل بهتر و دقیق‌تر عمل می‌کند.
جمع‌آوری و برچسب‌گذاری داده‌ها می‌تواند فرآیندهای طاقت‌فرسا و پرهزینه باشد. تغییر در ها اجازه می دهد تا این هزینه مجموعه داده ها با استفاده از تکنیک های افزایش داده به شرکت های عملیاتی را کاهش دهند.

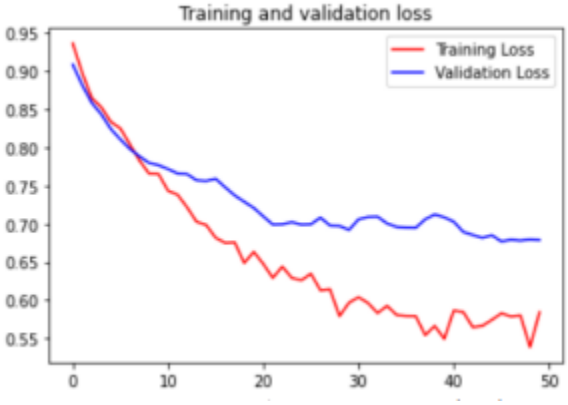
.۴

مدل خود را طبق صورت سوال می‌سازیم و روی داده‌های آموزشی ترین می‌کنیم. نتایج بدست آمده:

طبق نمودارها مدل اورفیت شده است و روی داده‌های validation عملکرد خوبی ندارد.

برای بهبود این مشکل از روش‌های اضافه کردن به دیتا، کم کردن تعداد نورون‌ها، L_1 و L_2 و dropout استفاده می‌کنیم.

سپس مدل خود را کامپایل و فیت کرده و نتایج جدید را پلات می‌کنیم.




```
12/12 [==============================] - 0s 2ms/step - loss: 0.5536 - accuracy: 0.7833
Train evaluation is:  [0.5535984635353088, 0.7833333611488342]
2/2 [==============================] - 0s 6ms/step - loss: 0.4980 - accuracy: 0.8500
Test evaluation is:  [0.49796637892723083, 0.8500000238418579]
```

همانطور که می‌بینیم دقت به خوبی افزایش یافته است.