آرش محمد قلی نژاد
400443183

تمرین سری 3

سوال 1)

ابتدا توضیحات کد و سپس تحلیل نتایج قرار میگیرد

توضیحات)
نکته) راستش برای اینکه بهترین حالت ممکن گزارش کنم چندین بار اجرا کردم و مدل ها را آموزش دادم نکته اینه که گاهی ممکن بود مدل بهتر بدتر عمل کنه آقای یزدی پور گفتن اینجور موقع ها باید میانگین گرفت ولی ما صرفا لازمه بهترین حالت رو گزارش کنیم.

قرار است داده را برای آموزش و تست مدل های lstm آماده کنیم این داده ها را از تاریخ مشخص شده به داده ی تست و ماقبل آن را به داده ی آموزش تقسیم مینماییم

```
data = pd.read_csv ('AABA_2006-01-01_to_2018-01-01.csv', date_parser= True)
data.tail()
```

```
data_training = data[data['Date'] < '2017-01-01'].copy()
data_testing = data[data['Date'] >= '2017-01-01'].copy()

training_data = data_training.drop(['Date','Low','Close','Open','Volume','Name'], axis = 1)

training_data.head()
data_testing.head()
```

فقط به ستونی که در صورت سوال مشخص کردید نیازمندیم لذا آن را جدا خواهیم کرد
سپس کل دیتا را برای این که min max scaler را برای بردن رنج مقادیر به بین 0و1 آماده میکنیم

```
for i in range (60, training_data.shape[0]):
    X_train.append(training_data[i-60: i,0])
    Y_train.append(training_data[i,0])

print(X_train[0])
print(Y_train[0])
```

طبق خواسته ی سوال ما باید با دیتای 60 روز روز 61 را پیشبینی کنیم سپس 60 روز بعدی و 61 امین روز پس از این دیتا و ...
برای اینکار داده ی اموزش را به دو بخش تقسیم کردم 60 تا 60 تا داده را بخش بندی میکنیم یعنی 0 تا 59 سپس 1 تا 60 و ....امین داده ها را با هم یک لیست کرده و درون یک لیست append میکنیم برای y ها 1 روز بعد را در نظر میگیریم مثلا برای داده ی 0 تا 59 داده ی 60 مقدارش در y ذخیره میشود و به همین ترتیب
سپس مدل ها را میسازیم

3 مرحله از ما خواسته شده است
اول خواسته شده است که مدلی را اموزش دهیم سپس ان مدل را با واحد های بیشتر اموزش دهیم و مقایسه کنیم مدل اول من این مدل با 1 لایه میباشد که 15 یونیت دارد

```
regressor3 = Sequential()
regressor3.add(LSTM(units=15, activation= "tanh", return_sequences=False, input_shape = (X_train.shape[1] ,
regressor3.add(Dense(units = 25))
regressor3.add(Dense(units = 1))
regressor3.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 15)                1020

 dense (Dense)               (None, 25)                400

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 1,446
Trainable params: 1,446
Non-trainable params: 0
_________________________________________________________________
```

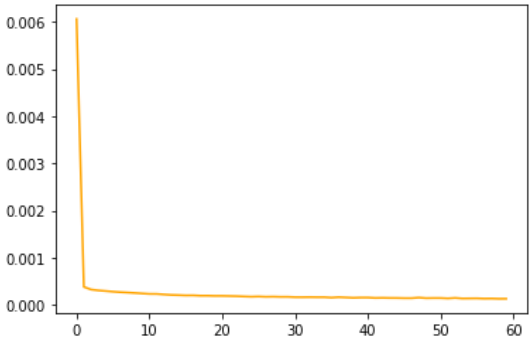
مدل دوم همه چیزش یکسان میباشد اما یونیت های بیشتر دارد

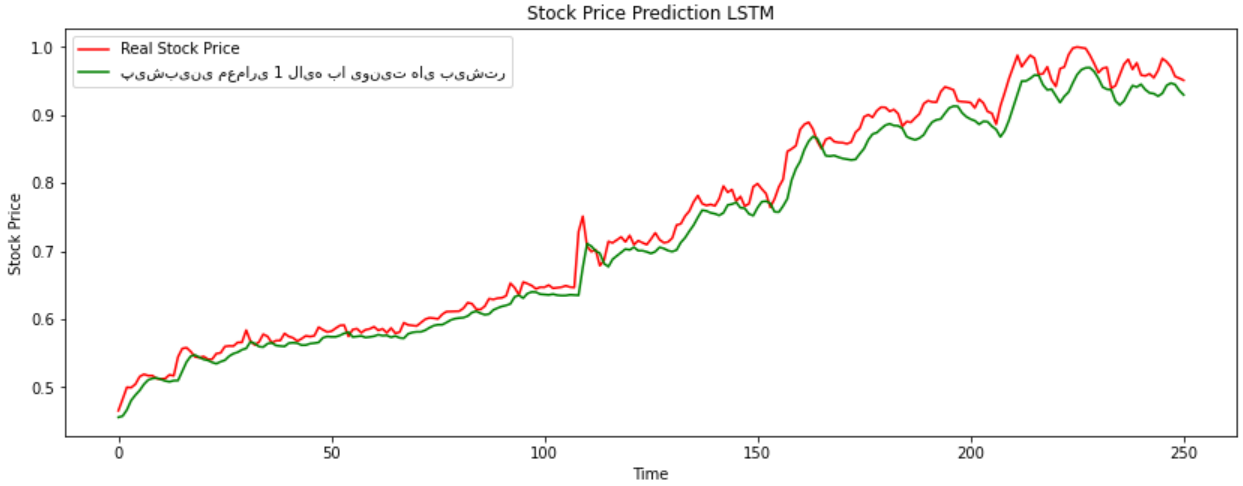
```
regressor = Sequential()
regressor.add(LSTM(units=25, activation= "tanh", return_sequences= True, input_shape = (X_train.shape[1]
regressor.add(LSTM(units=25, activation= "tanh"))
regressor.add(Dense(units = 25))
regressor.add(Dense(units = 1))


regressor.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 60, 25)            2700

 lstm_3 (LSTM)               (None, 25)                5100

 dense_4 (Dense)             (None, 25)                650

 dense_5 (Dense)             (None, 1)                 26

=================================================================
Total params: 8,476
Trainable params: 8,476
Non-trainable params: 0
_________________________________________________________________
```

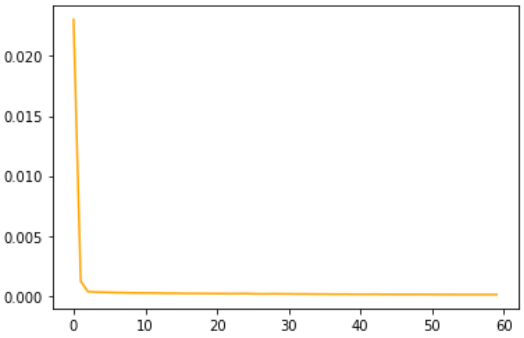
این دو مدل را با هایپر پارامتر های زیر آموزش میدهیم:
هر دو 60 ایپاک تعلیم داده میشوند با بچ سایز 64 و اوپتیمایزر adam نتایج

شکل زیر نمودار مدل اول میباشد که 15 یونیت داشته است.

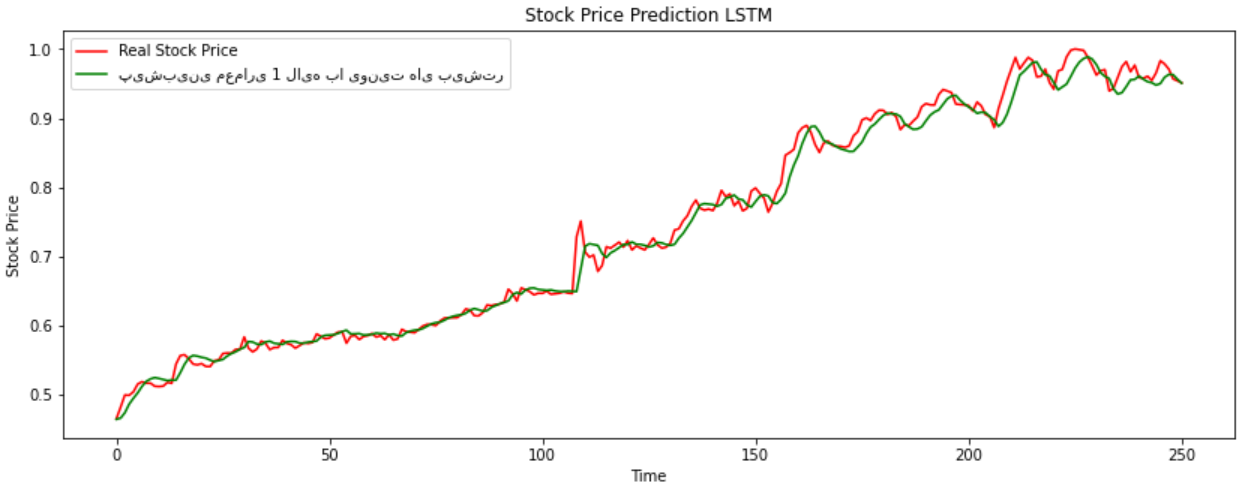




شکل زیر نمودار مدل با یونیت های بیشتر میباشد:



لاس های دو مدل به ترتیب به شکل زیر میباشد:




```
معماری 1 لایه با یونیت های  کمتر LSTM
(251, 1)
0.0006294830645987353
معماری 1 لایه با یونیت های  بیشتر LSTM
(251, 1)
0.0002576432056123457
```

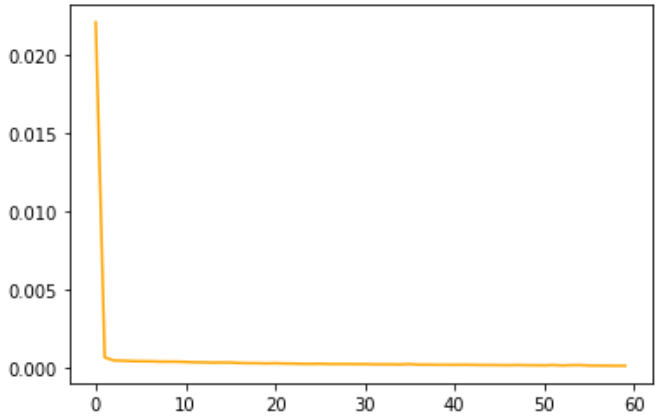
همانطور که مشاهده میشود لاس کمتر شده است به نسبت مدل با یونیت کمتر ، مدل اول گویا underfit میشود.
اما لایه های مدل را افزایش میدهیم و بررسی میکنیم.

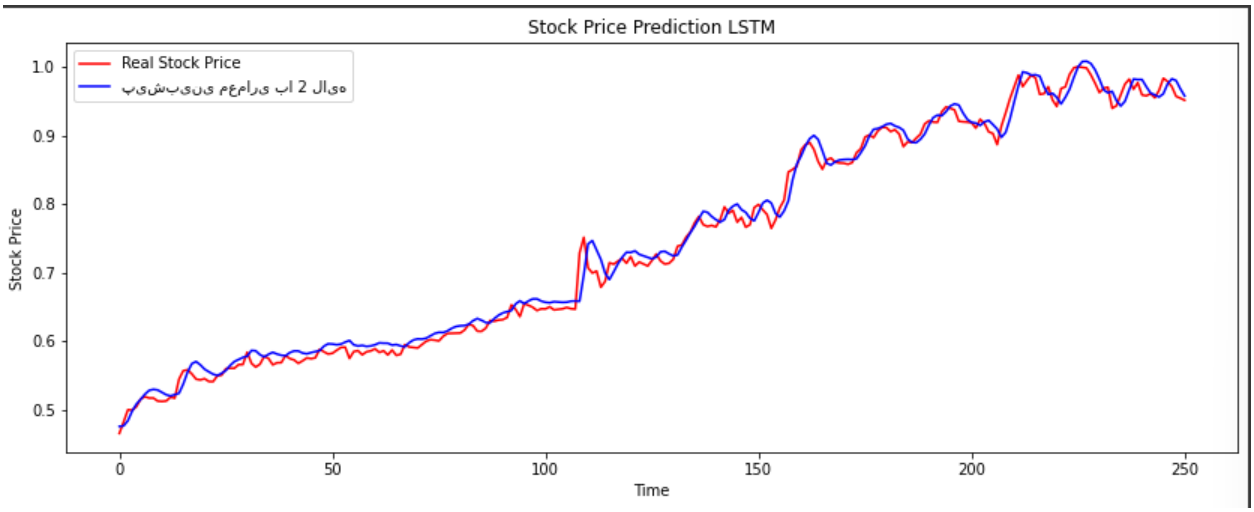
```
regressor = Sequential()
regressor.add(LSTM(units=25, activation= "tanh", return_sequences= True, input_shape = (X_train.shape[1] , 
regressor.add(LSTM(units=25, activation= "tanh"))
regressor.add(Dense(units = 25))
regressor.add(Dense(units = 1))


regressor.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 60, 25)            2700

 lstm_3 (LSTM)               (None, 25)                5100

 dense_4 (Dense)             (None, 25)                650

 dense_5 (Dense)             (None, 1)                 26

=================================================================
Total params: 8,476
Trainable params: 8,476
Non-trainable params: 0
_________________________________________________________________
```

مدل دوم یک لایه بیشتر دارد

نمودار و لاس آن را بررسی میکنیم:





همانطور که مشاهده میشود در پیدا کردن trend رشد و نزول نمودار بهتر عمل کرده است اما تغییرات چشمگیر نبوده است.

```
معماری LSTM 2 لایه
(251, 1)
0.0002241271837333139
```

سپس خواسته شده تا در مرحله ی سوم میزان سایز batch را تغییر دهیم من دو مدل جدید تعریف کردم مدل با بچ سایز کمتر و یکی هم بچ سایز بیشتر:

```
regressor6 = Sequential()
regressor6.add(LSTM(units=50, activation= "tanh", return_sequences= True, input_shape = (X_train.shape[1]
regressor6.add(LSTM(units=50, activation= "tanh"))
regressor6.add(Dense(units = 25))
regressor6.add(Dense(units = 1))

regressor6.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_6 (LSTM)               (None, 60, 50)            10400

 lstm_7 (LSTM)               (None, 50)                20200

 dense_8 (Dense)             (None, 25)                1275

 dense_9 (Dense)             (None, 1)                 26

=================================================================
Total params: 31,901
```

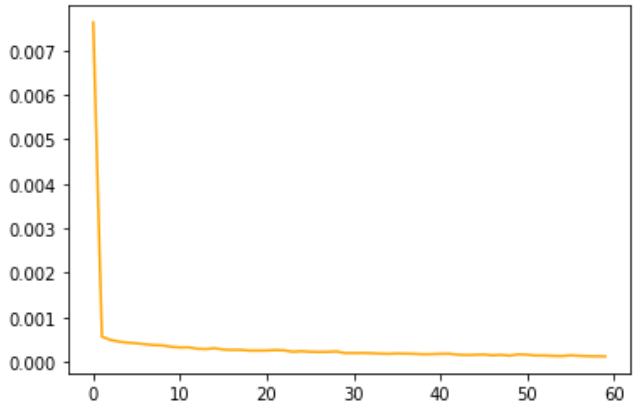
```
regressor5 = Sequential()
regressor5.add(LSTM(units=50, activation= "tanh", return_sequences= True, input_shape = (X_train.shape[1]
regressor5.add(LSTM(units=50, activation= "tanh"))
regressor5.add(Dense(units = 25))
regressor5.add(Dense(units = 1))

regressor5.summary()
```

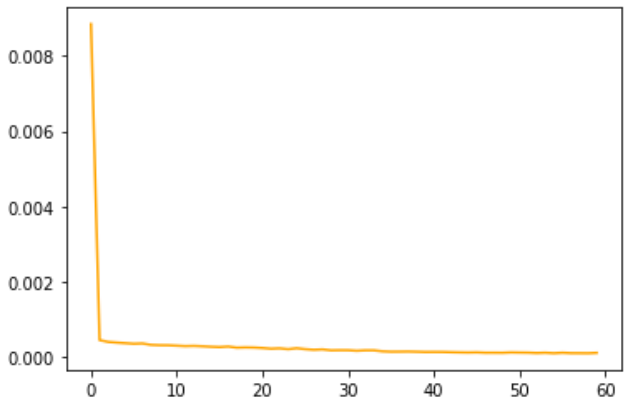
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_4 (LSTM)               (None, 60, 50)            10400

 lstm_5 (LSTM)               (None, 50)                20200

 dense_6 (Dense)             (None, 25)                1275

 dense_7 (Dense)             (None, 1)                 26

=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
```

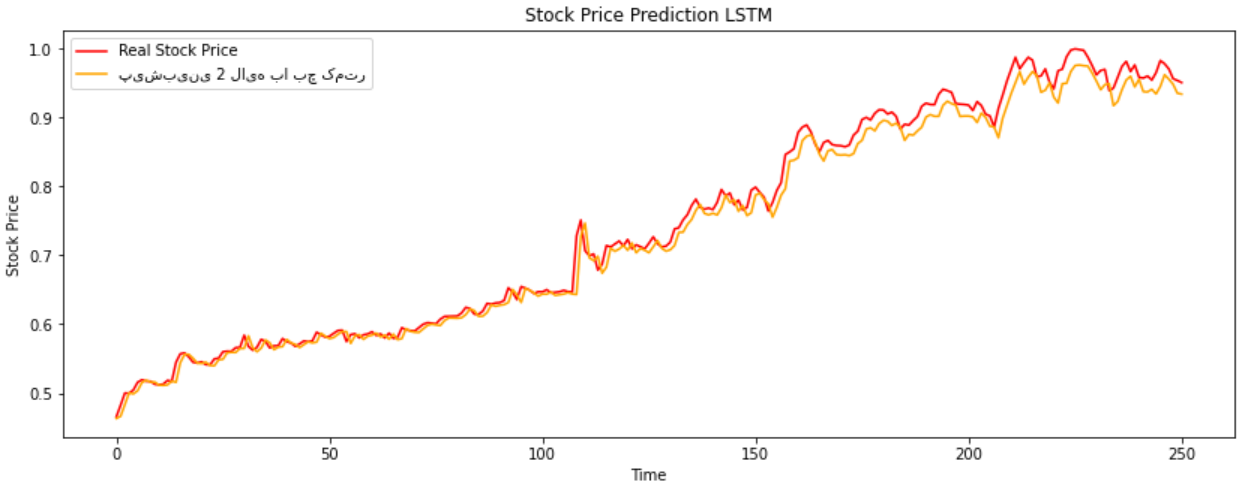
البته یونیت ها را بیشتر کردم ولی نکته اینجاس بچ سایز مدل regressor5 را کمتر کردم و علت اینکه مقدار بسیار کمی نسبت به معماری 2 لایه ی بالا بیشتر لاس دارد این هست نه چیز دیگری سپس معماری 6 را داریم که تقریبا نصف مدل 5 لاس دارد پس با بچ سایز بیشتر هم توانستیم مقداری بهتر عمل کنیم چندین بار مدل ها را آموزش دادم جدای از مدل های قبلی معمولا mse مدل با بچ سایز بیشتر نصف مدل با بچ سایز کمتر میباشد به شرح زیر بهترین حالات رو ثبت کردم و برای سایز بچ بیشتر 64 تا در نظر گرفتم و برای بچ کمتر 32.

```
معماری LSTM 2 لایه با سایز دسته یا بچ سایز بیشتر
(251, 1)
0.0001389397057096183 3
```

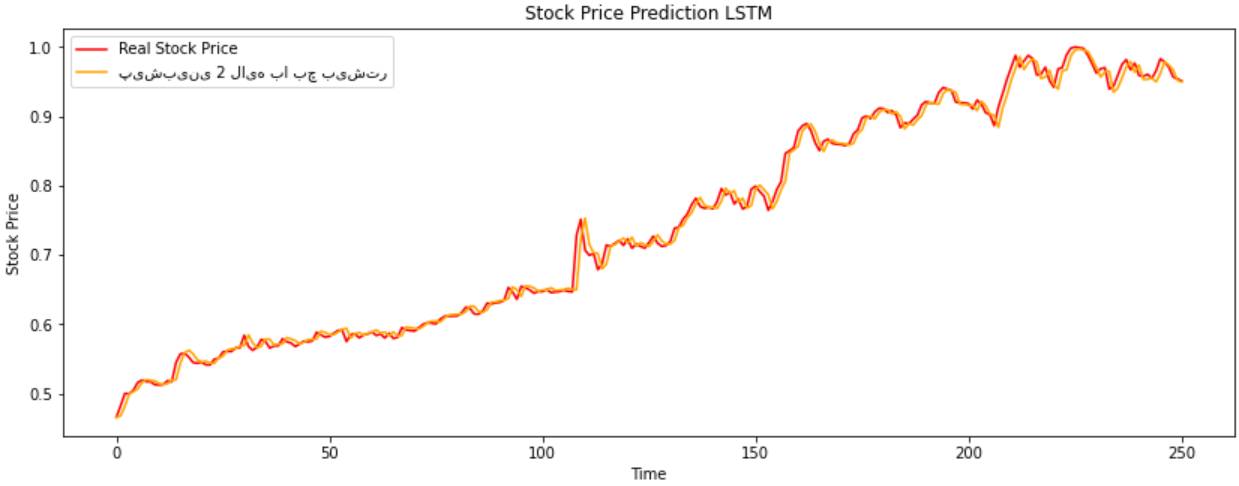
```
معماری LSTM 2 لایه با سایز دسته یا بچ سایز کمتر
(251, 1)
0.0002893398785921067
```







شکل بالا مدل با بچ سایز کمتر میباشد که مقایسه شده با نمودار مقادیر اصلی که در time stamp های اخر کمی دور تر از مقدار اصلی شده است.
اما شکل پایین مدل با بچ سایز بیشتر میباشد که بسیار بهتر در اواخر نمودار عمل میکند.



همانطور که مشاهده میکنید در سه مرحله مدل را بهینه سازی کردیم که بسیار بهتر از مراحل اولیه شده است اینکار را بر اساس سه هایپر پارامتر یعنی تعداد یونیت ها و تعداد لایه ها و اندازه ی بچ را تغییر دادیم.

در ادامه توضیحات کد و راه حل و گزارش سوال 2 را مشاهده میکنیم:

سوال 2)
طبق معمول ابتدا کد ها را توضیح میدهم سپس دقت بدست آمده را گزارش میدهم:

```
ds_train = tfds.load(name="rock_paper_scissors", split="train")
ds_test = tfds.load(name="rock_paper_scissors", split="test")
ds_info = tfds.builder("rock_paper_scissors").info
```

طبق کد بالا از کتابخانه ی tensorflow-datasets دیتاست را دانلود میکنیم

```
train_images, train_labels=np.array([i['image'].numpy()[:,:,0] for i in
([ds_train]),np.array([i['label'].numpy() for i in ds_train

test_images, test_labels=np.array([i['image'].numpy()[:,:,0] for i in
([ds_test]),np.array([i['label'].numpy() for i in ds_test

test_images.shape
```

در این تکه کد که بالاتر گذاشتم ابتدا لیبل را از عکس ها جدا کردم سپس دو ارایه ی numpy یکی برای داده ی آموزش و دیگری برای داده ی تست میسازیم البته فقط یکی از سه کانال تصویر را میگیریم چون نیاز به تصویر RGB نداریم و حجم محاسبات رو کمتر میکنیم.

```
train_images = train_images.reshape(2520,300,300,1)

test_images = test_images.reshape(372,300,300,1)

train_images = train_images/255
test_images = test_images/255

# normalize(train_images)
```

در بلاک اول تصویر بالا چون تصویر را یک کاناله کردیم تو قسمت قبلی (300,300) ابعاد هر تصویر شده بود ولی ما میخواهیم به صورت (300,300,1) تبدیل میکنیم که بعد آخر بیانگر تعداد کانال های تصویر هست، حالا این تصاویر را تقسیم بر 255 که بیشینه ی مقدار عددی هر پیکسل است میباشد.

```
model2 = keras.models.Sequential([
    keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(300, 300, 1)),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.2),
    keras.layers.Conv2D(32, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(3, activation='softmax')
])


model2.compile(optimizer='adam',loss=keras.losses.SparseCategoricalCrossentropy(), metrics=['accuracy'])
history2=model2.fit(train_images, train_labels, epochs=5, batch_size=32)
```

مدل را در سه مرحله بهینه سازی میکنیم اول تعداد لایه ها را افزایش دادم مدل اول دارای دو لایه ی کانولوشنی هست این مدل را مقایسه میکنیم با مدلی که سه لایه ی کانولوشنی دارد.

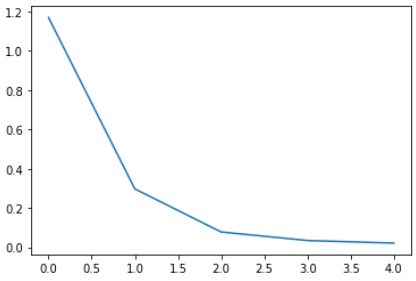
```
model = keras.models.Sequential([
    keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(300, 300, 1)),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.2),
    keras.layers.Conv2D(32, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.2),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(3, activation='softmax')
])


model.compile(optimizer='adam',loss=keras.losses.SparseCategoricalCrossentropy(), metrics=['accuracy'])
history=model.fit(train_images, train_labels, epochs=5, batch_size=32)
```

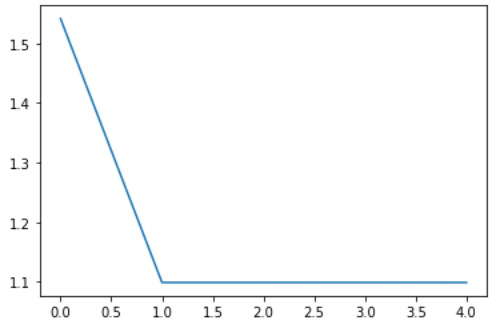
دو مدل را بر اساس لاس و دقت مقایسه میکنیم که نمودار به شرح زیر است:

```
12/12 [==============================] - 1s 32ms/step - loss: 1.0986 - accuracy: 0.3333
[1.0986149311065674, 0.3333333432674408]
```

```
12/12 [==============================] - 0s 25ms/step - loss: 1.2781 - accuracy: 0.6613
[1.2781486511230469, 0.6612903475761414]
```

مشاهده میشود که loss در داده ی آزمون یا تست بهتر شده و دقت نیز دو برابر گشته است





مقایسه ی لاس در دیتای آزمایش نیز نشان دهنده ی این هست که روی دیتای آزمایش نیز بهتر عملکرده است مدل با 1 لایه ی بیشترو در کل گویا مدل اول که 2 لایه داشت underfit شده است.

در آزمایش بعدی سعی کردم تعداد فیلتر ها یا تعداد لایه ی دنس را کم و زیاد کنم مدل اول به شرح زیر است

```
model3 = keras.models.Sequential([
    keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(300, 300, 1)),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.3),
    keras.layers.Conv2D(32, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.3),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.3),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.3),
    keras.layers.Flatten(),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dropout(0.3),
    keras.layers.Dense(3, activation='softmax')
])


model3.compile(optimizer='adam',loss=keras.losses.SparseCategoricalCrossentropy(), metrics=['accuracy'])
history3=model3.fit(train_images, train_labels, epochs=5, batch_size=32)
```

مدل دوم اما دارای تعداد فیلتر بیشتری در لایه ی نهایی و تعداد نورون بیشتر در لایه ی dense میباشد:

```
model4 = keras.models.Sequential([
    keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(300, 300, 1)),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.3),
    keras.layers.Conv2D(32, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.3),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.3),
    keras.layers.Conv2D(128, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.3),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.3),
    keras.layers.Dense(3, activation='softmax')
])


model4.compile(optimizer='adam',loss=keras.losses.SparseCategoricalCrossentropy(), metrics=['accuracy'])
history4=model4.fit(train_images, train_labels, epochs=5, batch_size=32)
```

هر دو مدل را 4 لایه در نظر گرفتم تا فقط و فقط هایپر پارامتر خاصی تغییر را ایجاد نماید دقت های بدست آمده در داده ی آموزش به شرح زیر میباشد:

```
Epoch 1/5
79/79 [==============================] - 5s 60ms/step - loss: 1.1208 - accuracy: 0.3734
Epoch 2/5
79/79 [==============================] - 5s 59ms/step - loss: 0.8797 - accuracy: 0.5607
Epoch 3/5
79/79 [==============================] - 5s 59ms/step - loss: 0.5507 - accuracy: 0.7405
Epoch 4/5
79/79 [==============================] - 5s 59ms/step - loss: 0.2979 - accuracy: 0.8540
Epoch 5/5
79/79 [==============================] - 5s 62ms/step - loss: 0.1361 - accuracy: 0.9329
```

```
Epoch 1/5
79/79 [==============================] - 6s 62ms/step - loss: 1.1570 - accuracy: 0.3722
Epoch 2/5
79/79 [==============================] - 5s 61ms/step - loss: 0.9160 - accuracy: 0.5698
Epoch 3/5
79/79 [==============================] - 5s 61ms/step - loss: 0.2672 - accuracy: 0.8968
Epoch 4/5
79/79 [==============================] - 5s 61ms/step - loss: 0.0846 - accuracy: 0.9734
Epoch 5/5
79/79 [==============================] - 5s 61ms/step - loss: 0.0316 - accuracy: 0.9877
```

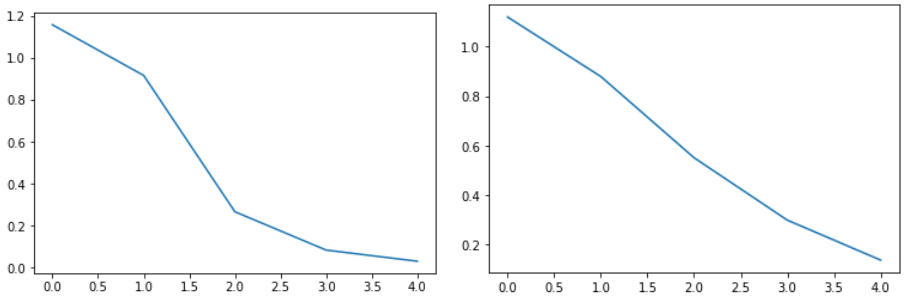
در داده ی آموزش مدل بهتر عمل کرده است.

```
12/12 [==============================] - 0s 24ms/step - loss: 0.5551 - accuracy: 0.7957
[0.555065929889679, 0.7956989407539368]
```

تصویر بالا دقت و Loss در مدل با فیلتر کمتر را نشان میدهد که مدل با فیلتر بیشتر را در پایین میتوانید مشاهده نمایید

```
12/12 [==============================] - 0s 25ms/step - loss: 0.3968 - accuracy: 0.8522
[0.39684566855430603, 0.852150559425354]
```

دقت بیشتر و loss کمتر شده است میتوان این موضوع که مدل بهبود پیدا کرده را در تصویر زیر که به ترتیب از راست به چپ نمودار تغییرات loss برای داده ی train مدل های معرفی شده ی فیلتر کمتر و فیلتر بیشتر را نشان میدهد گذاشته شده است:



که مشاهده میشود که عملکرد بهتری در مدل دوم بدست آمده است.

سپس در مرحله ی آخر همین مدل را با سایز بچ بیشتر سعی میکنیم عمل کرد سایز بچ بیشتر را بررسی کنیم:

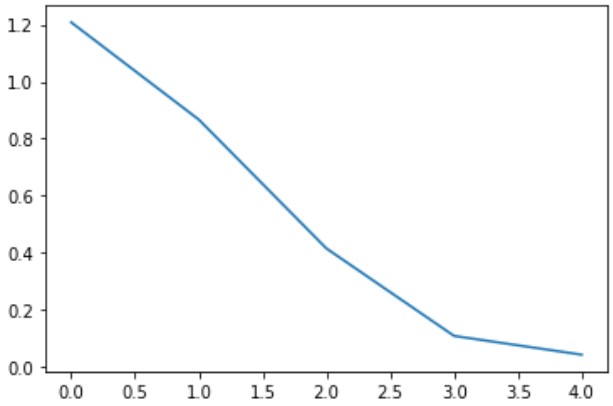
مدل را با بچ سایز 64 تایی آموزش میدهیم:

```
Epoch 1/5
40/40 [==============================] - 7s 110ms/step - loss: 1.2083 - accuracy: 0.3837
Epoch 2/5
40/40 [==============================] - 4s 110ms/step - loss: 0.8665 - accuracy: 0.6044
Epoch 3/5
40/40 [==============================] - 4s 111ms/step - loss: 0.4139 - accuracy: 0.8417
Epoch 4/5
40/40 [==============================] - 4s 112ms/step - loss: 0.1067 - accuracy: 0.9635
Epoch 5/5
40/40 [==============================] - 5s 113ms/step - loss: 0.0408 - accuracy: 0.9873
```

لاس و دقت روی داده ی آموزش تقریبا در یک رده قرار دارد اما

```
model5.evaluate(test_images,test_labels)

12/12 [==============================] - 0s 26ms/step - loss: 0.4493 - accuracy: 0.8683
[0.4492816627025604, 0.8682795763015747]
```

روی داده ی تست loss باز در همان میزانیست که مدل با سایز بچ کمتر است اما دقت کمی بیشتر شده است که آنقدر ها هم پیشرفت چشم گیری نیست
در کل هر دو مدل آخر دقت بالای 70 میدهند و به دفعات دقت بالای 80 هم داده اند ولی متوجه شدم گویا وقتی بچ سایز را بیشتر میکنیم گاهی در مینیمم محلی گیر میکنیم نمودار loss در هنگام آموزش نیز به شرح زیر است:



این نمودار هم در حد و اندازه ی همان نمودار قبلیست.

در کل سایز batch در آزمایشات من گاهی مدل را بهتر میکرد دفعات کمی بود که 5 درصد کمتر دقت داد.