**بسم الله الرحمن الرحیم**

**نام دانشجو:** سید محمد علی رضایی

**شماره دانشجویی:** 400131020

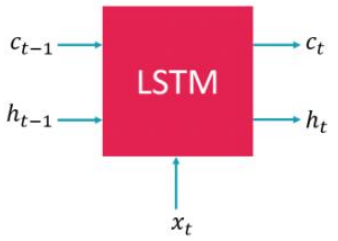
**استاد درس:** دکتر صفابخش

گزارش تمرین شماره۶

کدهای گزارش در فایل زیپ موجود می‌باشد.

**سوال اول** :

شبکه عصبی LSTM همانند شبکه RNN به صورت زنجیره‌ای پشت سر هم قرار می‌گیرد. مزیت این معماری نسبت به RNN می‌توان به استفاده از یک حافظه کوتاه مدت به مدت طولانی تری اشاره کرد. شکل زیر یک بلاک از معماری LSTM را نمایش می‌دهد که در این ساختار $c_{t-1}$ را cell state می‌نامیم که حافظه مورد استفاده در شبکه می‌باشد. $h_{t-1}$ خروجی از بلاک قبلی شبکه در زمان t-1 می‌باشد. $x_t$ نمونه ورودی در زمان t، می‌باشد.




در گذر زمان، در cell state، اطلاعاتی ذخیره یا از آن حذف می‌شود. این حافظه بلندمدت دو خاصیت مهم دارد : ۱- می‌توانیم اطلاعات ان را پاک کنیم(فراموشی)، ۲-می‌توانیم به ان اطلاعاتی اضافه کنیم(به خاطر سپردن)

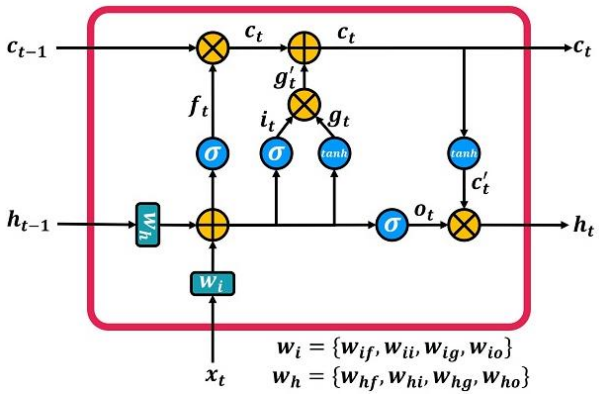
فراموشی و به خاطر سپردن در این ساختار بدین گونه انجام می‌شود که ورودی شبکه ابتدا با استفاده از یک تابع سیگموئید در رنج صفر و یک برده می‌شود و سپس به صورت متناظر با داده‌های ذخیره شده در cell state ضرب می‌شود، ورودی هایی که مقدار صفر بعد از اعمال تابع به خود نسبت گرفته اند منجر به حذف اطلاعات داده‌های متناظر در واحد حافظه می‌شوند و همچنین داده‌هایی که مقدار یک بعد از اعمال تابع به خود می‌گیرند منجر به حفظ داده‌های ذخیره شده در حافظه می‌گردد، این ساختار را forget gate می‌نامیم.

حال بعد از اینکه به ازاء ورودی و خروجی بلاک قبل اطلاعات موجود در حافظه را ثبت یا حذف کردیم حال به دنبال اضافه کردن اطلاعات جدیدتر به واحد حافظه به ازاء همین دو ورودی می‌باشیم(Input Gate ).

در input gate به ازاء دو ورودی که داشتیم ابتدا ان هارو به یک تابع tanh اعمال می‌کنیم و ورودی را بین یک و منفی یک می‌بریم، سپس برای اینکه اطلاعات نا مفید را به حافظه اضافه نکنیم مجدد در یک شبکه MLP دیگر با همان دو ورودی قبلی یک تابع سیگموید اعمال می‌کنیم و با خروجی تابع tanh ضرب می‌کنیم و خروجی را سپس با مقدار موجود در حافظه جمع می‌کنیم به این ترتیب مقدار ذخیره شده بر روی cell state در زمان t بدست می‌آید.

Output gate : گیت خروجی تعیین می‌کند چقدر از حافظه بلندمدت باید به خروجی منتقل شود. در این فاز مشابه قبل دو ورودی $x_t$ و $h_{t-1}$ به یک تابع سیگموید وارد شده و بین رنج صفر و یک می‌رود و مقدار ذخیره شده در cell state نیز با استفاده از یک تابع tanh که مقادیر ذخیره شده در حافظه را بین یک و منفی یک می‌برد با یکدیگر ضرب انجام داده و خروجی در زمان t را می‌سازند.

شکل زیر نمایشی از یک سلول از بلاک حافظه شبکه LSTM :

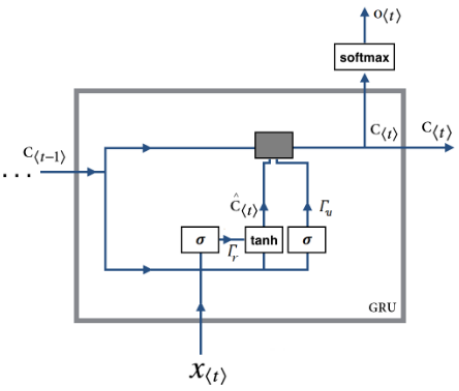


معماری GRU یا همان Gated Recurrent Unit در سال ۲۰۱۴ توسط Cho و همکاران معرفی شد این معماری به منظور برطرف سازی کاستی‌های شبکه عصبی بازگشتی نظیر مشکل محو شدگی گرادیان و همچنین کاهش سربار موجود در معماری LSTM ارائه شده است .GRU عموما به عنوان نسخه ای تغییر یافته از LSTM در نظر گرفته می‌شود چرا که هر دو این معماری‌ها از طراحی مشابه‌ای بهره می‌برند و در بعضی از موارد بصورت یکسان نتایج عالی بدست می‌دهند. در این معماری با استفاده از update gate and reset gate که هر دو، دو بردار هستند و با استفاده از ان‌ها تصمیم گرفته میشود چه اطلاعاتی به خروجی منتقل شده و چه اطلاعاتی منتقل نشود. نکته خاص در باره این دروازه ها این است که این دروازه ها را میتوان آموزش داد تا اینطور اطلاعات مربوط به گام های زمانی بسیار قبل را بدون آنکه در حین گذر زمان (طی گام های زمانی مختلف) دستخوش تغییر شوند حفظ کند.

**Update Gate** : اصطلاحا سویچی است که مشخص می‌کند در یک گام زمانی حالت قبلی مورد استفاده قرار گیرد یا ورودی (و یا ترکیبی از هر دو). با استفاده از این قابلیت جدید شبکه قادر است در دنباله های طولانی براحتی یک حالت از چندین گام زمانی قبل را در چند گام زمانی بعدی اثر دهد به عبارت دیگر شبکه قادر خواهد بود تا المانهایی را از گذشته دور در حافظه خود نگه‌داشته و از آن بهره‌برداری کند. نکته مهم درباره این گیت این است که در عمل مسیرهای میانبری ایجاد می‌کند که چندین گام زمانی را ندید گرفته و پشت سر میگذارد (از روی چندین گام زمانی می‌پرند) این میانبرها به همین صورت به خطای تولیدی اجازه می‌دهد تا بدون آنکه خیلی سریع محو شود براحتی در فاز پس انتشار منتقل گردد و اینطور معضلات مرتبط با گرادیان های محو شونده کاهش می‌یابد.

**Reset gate** : Reset Gate در اصل همانند سویچی عمل می‌کند که شبکه با کمک آن می‌تواند مشخص کند چه میزان از اطلاعات گذشته در گام فعلی مورد نیاز نیست (فراموش شود) و در گام فعلی از چه میزان از اطلاعات گام قبل استفاده شود مشابه گیت فراموشی در معماری lstm.

شکل زیر نمایشی از یک سلول GRU می‌باشد:




با توجه به دو معماری گفته شده از لحاظ ساختاری همانطور که مشاهده کردیم lstm دارای سه گیت در سلول حافظه خود می‌باشد ( forget gate and input gate output gate) در حالی که معماری GRU دارای دو گیت در سلول حافظه خود می‌باشند(update gate and reset gate) از لحاظ عملکرد reset gate و forget gate و همچنین input gate و update gate مشابه هم رفتار می‌کنند. در معماری lstm برای خروجی از گیت output استفاده شد در حالی که در معماری GRU صرفا از یک لایه SoftMax استفاده شده است.

معماری GRU از لحاظ پیچیدگی و ساختار و پارامتر های قابل یادگیری نسبت به معماری LSTM ساده تر می‌باشد (با توجه به کمتر بودن گیت‌ها) . با توجه به بیشتر بودن پارامترهای معماری LSTM این نتیجه گرفته می‌شود که برای اموزش نیاز به داده‌های بیشتری می‌باشد پس در هنگام مواجه شدن با دیتاست‌های بزرگ تر معماری LSTM پیشنهاد می‌گردد و همین‌طور در مقابل ان با توجه به ساختار ساده‌تر و پارامترهای کمتر معماری GRU در هنگام مواجه شدن با دیتاست‌های کوچک‌تر این ساختار نسبت به LSTM ترجیح داده می‌شود.

**سوال دوم** :

برای پیاده سازی شبکه عصبی LSTM و GRU ابتدا به اماده سازی دیتاست مورد نظر می‌پردازیم. در این مسئله ما ۱۶ دیتاست از شاخص‌های مختلف بازار سرمایه داریم که باید مسئله مورد نظر خود را به صورت exogenous در نظر بگیریم و از شاخص‌های دیگر برای پیش بینی شاخص کل بورس استفاده کنیم.

با استفاده از بلاک کد زیر ۱۶ دیتاست موجود را ترکیب کرده و براساس تاریخ یک ساله گذشته از همه شاخص ها ان را merge کرده‌ایم:

```
dfs = [df1,df2,df3,df4,df5,df6,df7,df8,df9,df10,df11,df12,df13,df14,df15,df16]
df_final = ft.reduce(lambda left, right: pd.merge(left, right, on='<DTYYYYMMDD>'), dfs)
```

```
dataset = df_final .loc[(df_final['<DTYYYYMMDD>'] >= 20210501)]
dataset.reset_index(inplace=True)
dataset.drop(['index'], axis=1,inplace=True)
```

نمایشی از دیتاست بدست امده :

| | <DTYYYYMMDD> | <OPEN>_x | <HIGH>_x | <LOW>_x | org_close | <VOL>_x | <OPENINT>_x | <OPENINT>.1_x | <OPENINT>.2_x | <LAST>_x | ... | <LAST>_x | <( |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 20210501 | 1208560.0 | 1208560.0 | 1194303.5 | 1194303.5 | 3701769749 | 1.630000e+13 | 215895 | 4.778443e+10 | 1194303.5 | ... | 159.0 | |
| 1 | 20210502 | 1191400.0 | 1191400.0 | 1180709.1 | 1180709.1 | 2041457656 | 1.060000e+13 | 134221 | 4.724093e+10 | 1180709.1 | ... | 158.8 | |
| 2 | 20210503 | 1176770.0 | 1176770.0 | 1167072.7 | 1167072.7 | 1589299529 | 9.140000e+12 | 144518 | 4.669533e+10 | 1167072.7 | ... | 158.8 | |
| 3 | 20210505 | 1167870.0 | 1172610.0 | 1172249.3 | 1172249.3 | 3573775657 | 1.890000e+13 | 260411 | 4.689892e+10 | 1172249.3 | ... | 158.1 | |
| 4 | 20210508 | 1176190.0 | 1176190.0 | 1167498.3 | 1167498.3 | 1895501673 | 1.250000e+13 | 187024 | 4.670543e+10 | 1167498.3 | ... | 157.4 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 223 | 20220409 | 1460550.0 | 1460550.0 | 1458684.4 | 1458684.4 | 8423628503 | 3.770000e+13 | 462958 | 5.810100e+10 | 1458684.4 | ... | 132.2 | |
| 224 | 20220410 | 1461710.0 | 1474240.0 | 1474099.6 | 1474099.6 | 11270947735 | 4.370000e+13 | 483996 | 5.871501e+10 | 1474099.6 | ... | 130.6 | |
| 225 | 20220411 | 1475490.0 | 1479890.0 | 1479590.9 | 1479590.9 | 10070884564 | 3.960000e+13 | 504631 | 5.893313e+10 | 1479590.9 | ... | 127.5 | |
| 226 | 20220412 | 1474000.0 | 1474000.0 | 1445970.5 | 1445970.5 | 9289109331 | 4.010000e+13 | 430607 | 5.759041e+10 | 1445970.5 | ... | 126.8 | |
| 227 | 20220413 | 1443170.0 | 1454120.0 | 1454126.8 | 1454126.8 | 8473206930 | 3.540000e+13 | 430487 | 5.790589e+10 | 1454126.8 | ... | 128.0 | |

228 rows × 145 columns

۲۲۸ تعداد نمونه‌های ما در یک سال اخیر و ۱۴۵ تعداد ستون‌ها و ویژگی‌های ما می‌باشند.

سپس با استفاده از بلاک کد زیر با توجه به توضیح تدریس یار محترم برای برچسب دهی به داده‌ها ، ابتدا ستون مربوطه به شاخص کل را در این دیتاست مشخص کرده و سپس نسبت اختلاف close هر روز با روز قبلش را حساب کرده اگر این اختلاف دارای عددی مثبت باشد به این معنا است که فردا بازار مثبت خواهد بود و اگر منفی باشد به ان معنا است که فردا بازار منفی خواهد بود به همین ترتیب اعداد مثبت (عدد یک) و اعداد منفی(عدد صفر) نسبت داده‌ام :

```
label_list =[]
for i in range(len(dataset.org_close)-1) :
    m = dataset.org_close[i+1] - dataset.org_close[i]
    if m < 0:
        m = 0
        label_list.append(m)
    else:
        m = 1
        label_list.append(m)
label_list.append(1)
```

با توجه به اینکه رنج داده‌های موجود در دیتا ست طیف وسیعی از اعداد را شامل می‌شود با استفاده از بلاک کد زیر و تابع minmax داده‌ها را بین اعداد صفر و یک می‌بریم:

```
min_max_scaler = MinMaxScaler()
dataset = min_max_scaler.fit_transform(dataset)
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 135 | 136 | 137 | 138 | 139 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.000000 | 0.231993 | 0.215077 | 0.205353 | 0.205353 | 0.164191 | 0.071700 | 0.060317 | 0.206235 | 0.205353 | ... | 0.498047 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| 1 | 0.000101 | 0.196688 | 0.179362 | 0.177034 | 0.177034 | 0.035144 | 0.014620 | 0.000000 | 0.178154 | 0.177034 | ... | 0.496094 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| 2 | 0.000202 | 0.166588 | 0.148914 | 0.148627 | 0.148627 | 0.000000 | 0.000000 | 0.007604 | 0.149965 | 0.148627 | ... | 0.496094 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| 3 | 0.000404 | 0.148277 | 0.140256 | 0.159411 | 0.159411 | 0.154242 | 0.097737 | 0.093192 | 0.160484 | 0.159411 | ... | 0.489258 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| 4 | 0.000706 | 0.165395 | 0.147706 | 0.149514 | 0.149514 | 0.023799 | 0.033647 | 0.038995 | 0.150487 | 0.149514 | ... | 0.482422 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 223 | 0.999596 | 0.750437 | 0.739531 | 0.756104 | 0.756104 | 0.531194 | 0.286000 | 0.242775 | 0.739259 | 0.756104 | ... | 0.236328 | 0.570229 | 0.193463 | 0.198420 | 0.198420 |
| 224 | 0.999697 | 0.752824 | 0.768024 | 0.788217 | 0.788217 | 0.752501 | 0.346085 | 0.258312 | 0.770983 | 0.788217 | ... | 0.220703 | 0.584694 | 0.213781 | 0.210711 | 0.210711 |
| 225 | 0.999798 | 0.781175 | 0.779783 | 0.799656 | 0.799656 | 0.659227 | 0.305027 | 0.273551 | 0.782252 | 0.799656 | ... | 0.190430 | 0.573962 | 0.239399 | 0.244074 | 0.244074 |
| 226 | 0.999899 | 0.778109 | 0.767524 | 0.729619 | 0.729619 | 0.598464 | 0.310034 | 0.218884 | 0.712878 | 0.729619 | ... | 0.183594 | 0.595427 | 0.234099 | 0.198420 | 0.198420 |
| 227 | 1.000000 | 0.714680 | 0.726149 | 0.746610 | 0.746610 | 0.535048 | 0.262968 | 0.218795 | 0.729178 | 0.746610 | ... | 0.195312 | 0.566962 | 0.191696 | 0.194908 | 0.194908 |

228 rows × 145 columns

بعد از این مرحله با توجه به اینکه دیتاست مورد استفاده در مسائل سری زمانی به صورت پنجره ای(اصطلاحا) می‌باشد پس با تابع زیر این بانک اطلاعاتی را به صورت دسته‌های ۴ تایی که دارای overlap می‌باشند در می‌اوریم :

```
def make_window_dataset(X, Y, batch=None):
    x = []
    y = []
    for i in range(batch, len(X)):
        x.append(X[i - batch : i])
        y.append(Y[i])
    return np.array(x), np.array(y)
```

شکل نهایی داده‌ها به صورت زیر می‌باشد:

```
x.shape,y.shape
```

```
((224, 4, 145), (224,))
```

که در اینجا ۲۲۴ تعداد دسته های ما می‌باشد که در هر دسته ۴ نمونه با ابعاد ۱۴۵ داریم و برای برچسب این ۲۲۴ دسته نمونه یک لیست حاوی ۲۲۴ عدد( صفر و یا یک) دارا می‌باشیم.

در نهایت نیز با استفاده از بلاک کد زیر داده‌ها را به مجموعه داده اموزش(۷۰درصد) و تست(۲۰درصد) و اعتبار سنجی (۱۰درصد) در می‌اوریم.

```
xtrain, xtest, ytrain, ytest = train_test_split(x , y, test_size=0.2, random_state=42)
xtrain, xval, ytrain, yval = train_test_split(xtrain, ytrain, test_size=0.125, random_state=42)
```

```
xtrain.shape , ytrain.shape, xtest.shape, ytest.shape
```

```
((156, 4, 145), (156,), (45, 4, 145), (45,))
```

پیش پردازش و اماده سازی اولیه در این مرحله به پایان رسیده و به سراغ اموزش شبکه در مرحله بعد می‌رویم.

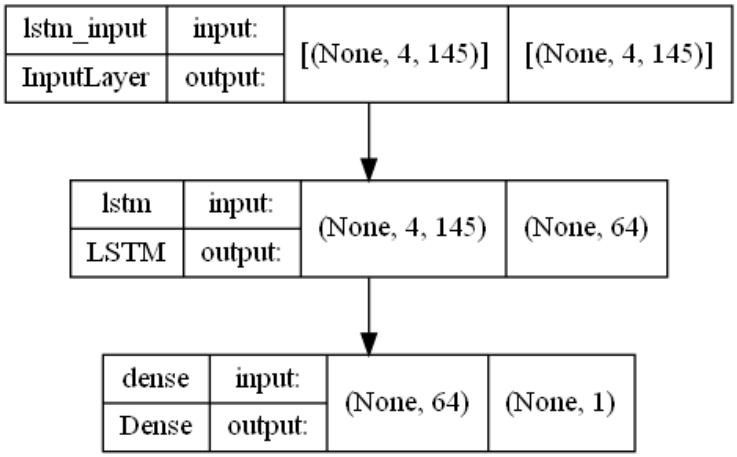
در ابتدا با استفاده از بلاک کد زیر به ساخت مدل LSTM می‌پردازیم:

```
keras.backend.clear_session()
model = Sequential()
model.add(layers.LSTM(64, input_shape=(4,145),kernel_regularizer=keras.regularizers.l1(l1=0.001)))
model.add(layers.Dense(units=1,activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer=keras.optimizers.Adam(learning_rate=0.001),metrics=['accuracy'])
```

در اینجا یک ساختار lstm با تعداد ۶۴ نرون در لایه مخفی و یک نرون در لایه خروجی و ورودی متناسب با اندازه batch داده‌ها و ابعاد نمونه‌ها با تابع فعال سازیsigmoid داریم برای جلوگیری از مشکل overfit شدن مدل در هنگام اموزش از رگولایزر L1 استفاده کرده‌ایم.

شکل زیر تعداد پارامتر های مسئله و نمایشی گرافیکی از نحوه ورودی و خروجی لایه‌های شبکه را نمایش می‌دهد:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                53760

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 53,825
Trainable params: 53,825
Non-trainable params: 0
```




با استفاده از بلاک کد زیر شبکه را با ۲۰۰ دور بر روی مجموعه داده‌های اموزش ، اموزش می‌دهیم و از داده‌های اعتبار سنجی برای دقت validation accuracy استفاده کرده ایم و سپس ارزیابی نهایی را برروی داده‌های ازمون انجام داده‌ایم :

```
model.fit(xtrain ,ytrain,validation_data=(xval,yval) , epochs=200)
model.evaluate(xtest,ytest)
```

```
57
Epoch 194/200
5/5 [==============================] - 0s 22ms/step - loss: 0.6611 - accuracy: 0.6538 - val_loss: 0.6613 - val_accuracy: 0.73
91
Epoch 195/200
5/5 [==============================] - 0s 27ms/step - loss: 0.6524 - accuracy: 0.7051 - val_loss: 0.7026 - val_accuracy: 0.60
87
Epoch 196/200
5/5 [==============================] - 0s 19ms/step - loss: 0.6245 - accuracy: 0.7436 - val_loss: 0.6474 - val_accuracy: 0.73
91
Epoch 197/200
5/5 [==============================] - 0s 15ms/step - loss: 0.6269 - accuracy: 0.7244 - val_loss: 0.7094 - val_accuracy: 0.65
22
Epoch 198/200
5/5 [==============================] - 0s 18ms/step - loss: 0.6341 - accuracy: 0.6923 - val_loss: 0.6465 - val_accuracy: 0.73
91
Epoch 199/200
5/5 [==============================] - 0s 18ms/step - loss: 0.6404 - accuracy: 0.7244 - val_loss: 0.6901 - val_accuracy: 0.69
57
```

همان طور که مشاهده می‌کنیم در انتهای اموزش دقت بر روی داده‌های اموزش تا ۷۲ درصد و بر روی داده‌های اعتبار سنجی تا ۶۹ درصد رسیده است. اما این دقت برای داده‌های تست به مراتب کمتر می‌باشد و در حدود ۴۸ درصد است:

```
2/2 [==============================] - 0s 6ms/step - loss: 0.8237 - accuracy: 0.4889

[0.8237408399581909, 0.4888888895511627]
```

شکل زیر نمایشی از نحوه ساخت و مقادیر موجود در ماتریس درهم ریختگی برای اموزش شبکه فوق می‌باشد:

```
predictions = model.predict(xtest)
predict = []
for i in predictions:
    if i < 0.5 :
        m = 0
        predict.append(m)
    else:
        m = 1
        predict.append(m)
```

```
tn1, fp1, fn1, tp1 = confusion_matrix(ytest, predict).ravel()
tn1, fp1, fn1, tp1
```

```
(10, 14, 6, 15)
```

با توجه به نتایج بدست امده از ماتریس درهم ریختگی مشاهده می‌شود که تعداد false positive به شدت زیاد بوده است(۱۴).

حال بار دیگر شبکه را با تعداد کمتری واحد lstm (۳۲) در لایه مخفی اموزش می‌دهیم تا نتایج را بررسی کنیم :

```
model.add(layers.LSTM(32, input_shape=(4,145),kernel_regularizer=keras.regularizers.l1(l1=0.001)))
```

```
Epoch 95/100
5/5 [==============================] - 0s 17ms/step - loss: 1.6627 - accuracy: 0.6603 - val_loss: 1.6314 - val_accuracy: 0.73
91
Epoch 96/100
5/5 [==============================] - 0s 17ms/step - loss: 1.6551 - accuracy: 0.6538 - val_loss: 1.6249 - val_accuracy: 0.73
91
Epoch 97/100
5/5 [==============================] - 0s 14ms/step - loss: 1.6482 - accuracy: 0.6795 - val_loss: 1.6188 - val_accuracy: 0.69
57
Epoch 98/100
5/5 [==============================] - 0s 17ms/step - loss: 1.6407 - accuracy: 0.6923 - val_loss: 1.6107 - val_accuracy: 0.69
57
Epoch 99/100
5/5 [==============================] - 0s 16ms/step - loss: 1.6333 - accuracy: 0.6795 - val_loss: 1.6031 - val_accuracy: 0.73
91
Epoch 100/100
5/5 [==============================] - 0s 18ms/step - loss: 1.6266 - accuracy: 0.6538 - val_loss: 1.5949 - val_accuracy: 0.69
```

با تعداد واحد LSTM کمتر، مشاهده می‌کنیم صحت برای مجموعه اموزش کاهش پیدا کرده است ، اما همچنان صحت برای داده‌های اعتبارسنجی تقریبا برابر حالت قبل است. در زیر نیز مقدار صحت برای داده‌های ازمون رو مشاهده می‌کنیم که برخلاف تصور نسبت به حالت قبل رشد داشته است.

```
2/2 [==============================] - 0s 0s/step - loss: 1.6648 - accuracy: 0.5556
[1.6648390293121338, 0.5555555820465088]
```

این بار ازمایش را با ۱۲۸ واحدlstm اجرا می‌کنیم :

```
Epoch 195/200
5/5 [==============================] - 0s 25ms/step - loss: 0.6406 - accuracy: 0.6923 - val_loss: 0.6815 - val_accuracy: 0.69
57
Epoch 196/200
5/5 [==============================] - 0s 25ms/step - loss: 0.6346 - accuracy: 0.7244 - val_loss: 0.6523 - val_accuracy: 0.73
91
Epoch 197/200
5/5 [==============================] - 0s 23ms/step - loss: 0.6130 - accuracy: 0.6859 - val_loss: 0.6769 - val_accuracy: 0.69
57
Epoch 198/200
5/5 [==============================] - 0s 23ms/step - loss: 0.6107 - accuracy: 0.7372 - val_loss: 0.6675 - val_accuracy: 0.69
57
Epoch 199/200
5/5 [==============================] - 0s 24ms/step - loss: 0.6049 - accuracy: 0.7115 - val_loss: 0.6620 - val_accuracy: 0.69
57
Epoch 200/200
5/5 [==============================] - 0s 24ms/step - loss: 0.6040 - accuracy: 0.7308 - val_loss: 0.6799 - val_accuracy: 0.69
```

مقدار صحت تا ۷۳ درصد برای اموزش و ۷۰ درصد برای اعتبار سنجی رسیده است و برای داده‌های ازمون:

```
2/2 [==============================] - 0s 17ms/step - loss: 0.8064 - accuracy: 0.5778
[0.8063610792160034, 0.5777778029441833]
```

این بار با ۸۰ واحدlstm ازمایش را اجرا می‌کنیم :

```
Epoch 195/200
5/5 [==============================] - 0s 22ms/step - loss: 0.6306 - accuracy: 0.7051 - val_loss: 0.6577 - val_accuracy: 0.69
57
Epoch 196/200
5/5 [==============================] - 0s 18ms/step - loss: 0.6284 - accuracy: 0.7051 - val_loss: 0.6684 - val_accuracy: 0.69
57
Epoch 197/200
5/5 [==============================] - 0s 25ms/step - loss: 0.6140 - accuracy: 0.7179 - val_loss: 0.6527 - val_accuracy: 0.73
91
Epoch 198/200
5/5 [==============================] - 0s 25ms/step - loss: 0.6205 - accuracy: 0.6923 - val_loss: 0.6790 - val_accuracy: 0.65
22
Epoch 199/200
5/5 [==============================] - 0s 25ms/step - loss: 0.6120 - accuracy: 0.7179 - val_loss: 0.6575 - val_accuracy: 0.69
57
Epoch 200/200
5/5 [==============================] - 0s 25ms/step - loss: 0.6148 - accuracy: 0.7179 - val_loss: 0.6894 - val_accuracy: 0.65
```

در این مرحله به نظر می‌رسد کاهش تعداد واحد های lstm منجر به کاهش در مقدار صحت در اموزش و اعتبار سنجی و ازمون شده است(نسبت به حالت ۱۲۸ واحد)

```
2/2 [==============================] - 0s 0s/step - loss: 0.8022 - accuracy: 0.5333
[0.802226185798645, 0.5333333611488342]
```

بار دیگر ازمایش را ۱۸۰ واحد اجرا می‌کنیم :

```
Epoch 196/200
5/5 [==============================] - 0s 32ms/step - loss: 0.6751 - accuracy: 0.6538 - val_loss: 0.7140 - val_accuracy: 0.65
22
Epoch 197/200
5/5 [==============================] - 0s 32ms/step - loss: 0.6249 - accuracy: 0.7051 - val_loss: 0.6504 - val_accuracy: 0.69
57
Epoch 198/200
5/5 [==============================] - 0s 35ms/step - loss: 0.6046 - accuracy: 0.7051 - val_loss: 0.7204 - val_accuracy: 0.65
22
Epoch 199/200
5/5 [==============================] - 0s 37ms/step - loss: 0.6392 - accuracy: 0.6987 - val_loss: 0.6447 - val_accuracy: 0.69
57
Epoch 200/200
5/5 [==============================] - 0s 34ms/step - loss: 0.5910 - accuracy: 0.6859 - val_loss: 0.7199 - val_accuracy: 0.65
22
2/2 [==============================] - 0s 12ms/step - loss: 0.8019 - accuracy: 0.5111
[0.8019179701805115, 0.5111111402511597]
```

به نظر می‌رسد با افزایش تعداد واحد های lstm لزوما بهبود نخواهیم داشت پس حالت = ۱۲۸ واحد lstm رو به عنوان بالاترین دقت در نظر می‌گیریم.

حال در مرحله بعد به سراغ شبکه با استفاده از واحدGRU می‌رویم:

با بلاک کد زیر مدل شبکه مورد نظر خود را میسازیم:

GRU mdoel

```
keras.backend.clear_session()
gru_model = Sequential()
gru_model.add(layers.GRU(64, input_shape=(4,145),kernel_regularizer=keras.regularizers.l1(l1=0.001)))
gru_model.add(layers.Dense(units=1,activation='sigmoid'))
gru_model.compile(loss='binary_crossentropy', optimizer=keras.optimizers.Adam(learning_rate=0.0001),metrics=['accuracy'])
```

در شرایط مشابه قبل این ازمایش را برای تعداد واحد های ۳۲ و ۶۴ و۸۰و ۱۲۸ و ۱۸۰ انجام می‌دهیم :

Units = 32 :

```
Epoch 196/200
5/5 [==============================] - 0s 17ms/step - loss: 1.0388 - accuracy: 0.6667 - val_loss: 1.0154 - val_accuracy: 0.69
57
Epoch 197/200
5/5 [==============================] - 0s 24ms/step - loss: 1.0363 - accuracy: 0.6667 - val_loss: 1.0132 - val_accuracy: 0.69
57
Epoch 198/200
5/5 [==============================] - 0s 22ms/step - loss: 1.0343 - accuracy: 0.6603 - val_loss: 1.0109 - val_accuracy: 0.69
57
Epoch 199/200
5/5 [==============================] - 0s 22ms/step - loss: 1.0317 - accuracy: 0.6731 - val_loss: 1.0093 - val_accuracy: 0.69
57
Epoch 200/200
5/5 [==============================] - 0s 21ms/step - loss: 1.0296 - accuracy: 0.6731 - val_loss: 1.0065 - val_accuracy: 0.69
57
2/2 [==============================] - 0s 5ms/step - loss: 1.0651 - accuracy: 0.6000

[1.065058946609497, 0.6000000238418579]
```

مقدار دقت بر روی داده‌های تست به ۶۰ درصد رسیده است که با تعداد واحد gru کمتر نسبت به lstm توانسته ایم دقت بالاتری را نسبت به تمام ازمایش های قبلی بدست بیاوریم.

Units = 64 :

```
Epoch 196/200
5/5 [==============================] - 0s 18ms/step - loss: 1.1332 - accuracy: 0.6923 - val_loss: 1.1094 - val_accuracy: 0.73
91
Epoch 197/200
5/5 [==============================] - 0s 16ms/step - loss: 1.1301 - accuracy: 0.6859 - val_loss: 1.1043 - val_accuracy: 0.73
91
Epoch 198/200
5/5 [==============================] - 0s 18ms/step - loss: 1.1276 - accuracy: 0.6987 - val_loss: 1.1042 - val_accuracy: 0.73
91
Epoch 199/200
5/5 [==============================] - 0s 14ms/step - loss: 1.1261 - accuracy: 0.6859 - val_loss: 1.0968 - val_accuracy: 0.73
91
Epoch 200/200
5/5 [==============================] - 0s 14ms/step - loss: 1.1210 - accuracy: 0.6923 - val_loss: 1.0971 - val_accuracy: 0.73
91
2/2 [==============================] - 0s 7ms/step - loss: 1.1682 - accuracy: 0.5556

[1.1682217121124268, 0.5555555820465088]
```

با توجه به اینکه مقدار صحت برای داده‌های ازمون نسبت به ازمایش قبل کمتر بوده است اما مقدار صحت برای داده‌های اعتبار سنجی اندکی بهبود داشته است .

Units = 80 :

```
Epoch 196/200
5/5 [==============================] - 0s 19ms/step - loss: 1.1472 - accuracy: 0.6987 - val_loss: 1.1151 - val_accuracy: 0.73
91
Epoch 197/200
5/5 [==============================] - 0s 19ms/step - loss: 1.1436 - accuracy: 0.6795 - val_loss: 1.1095 - val_accuracy: 0.73
91
Epoch 198/200
5/5 [==============================] - 0s 18ms/step - loss: 1.1383 - accuracy: 0.6795 - val_loss: 1.1070 - val_accuracy: 0.73
91
Epoch 199/200
5/5 [==============================] - 0s 19ms/step - loss: 1.1349 - accuracy: 0.6923 - val_loss: 1.1054 - val_accuracy: 0.73
91
Epoch 200/200
5/5 [==============================] - 0s 21ms/step - loss: 1.1314 - accuracy: 0.7051 - val_loss: 1.1042 - val_accuracy: 0.73
91
2/2 [==============================] - 0s 6ms/step - loss: 1.1776 - accuracy: 0.5333

[1.1775588989257812, 0.5333333611488342]
```

Units = 128 :

```
Epoch 196/200
5/5 [==============================] - 0s 26ms/step - loss: 1.0907 - accuracy: 0.6795 - val_loss: 1.0602 - val_accuracy: 0.78
26
Epoch 197/200
5/5 [==============================] - 0s 27ms/step - loss: 1.0863 - accuracy: 0.6795 - val_loss: 1.0636 - val_accuracy: 0.73
91
Epoch 198/200
5/5 [==============================] - 0s 25ms/step - loss: 1.0827 - accuracy: 0.6987 - val_loss: 1.0608 - val_accuracy: 0.73
91
Epoch 199/200
5/5 [==============================] - 0s 26ms/step - loss: 1.0820 - accuracy: 0.6859 - val_loss: 1.0504 - val_accuracy: 0.78
26
Epoch 200/200
5/5 [==============================] - 0s 26ms/step - loss: 1.0767 - accuracy: 0.6538 - val_loss: 1.0470 - val_accuracy: 0.78
26
2/2 [==============================] - 0s 7ms/step - loss: 1.1404 - accuracy: 0.5556

[1.140446424484253, 0.5555555820465088]
```

Units = 180 :

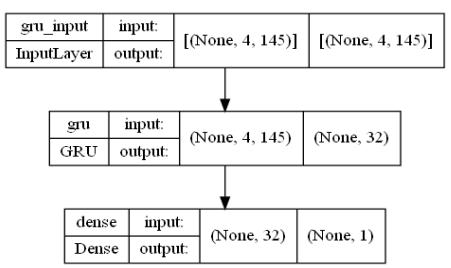
```
Epoch 196/200
5/5 [==============================] - 0s 28ms/step - loss: 1.0777 - accuracy: 0.6923 - val_loss: 1.0535 - val_accuracy: 0.78
26
Epoch 197/200
5/5 [==============================] - 0s 27ms/step - loss: 1.0759 - accuracy: 0.6923 - val_loss: 1.0518 - val_accuracy: 0.78
26
Epoch 198/200
5/5 [==============================] - 0s 27ms/step - loss: 1.0717 - accuracy: 0.6859 - val_loss: 1.0474 - val_accuracy: 0.78
26
Epoch 199/200
5/5 [==============================] - 0s 27ms/step - loss: 1.0716 - accuracy: 0.6731 - val_loss: 1.0473 - val_accuracy: 0.78
26
Epoch 200/200
5/5 [==============================] - 0s 27ms/step - loss: 1.0695 - accuracy: 0.7051 - val_loss: 1.0460 - val_accuracy: 0.78
26
2/2 [==============================] - 0s 9ms/step - loss: 1.1502 - accuracy: 0.4889

[1.15022873878479, 0.4888888895511627]
```

مشاهده می‌کنیم در طی این ازمایشات بهترین مقدار در صحت مربوط به units= 32 می‌باشد و با افزایش تعداد واحد GRU در بین محدوده ۵۵ درصد نوسان داشتیم و مشابه حالتlstm با افزایش به ۱۸۰ واحد مقدار صحت به زیر ۵۰ درصد کاهش پیدا کرد.

بهترین مقدار برای واحدGRU برابر با units= 32 می‌باشد که پارامترهای قابل یادگیری و ساختار شبکه و ماتریس درهم ریختگی به صورت زیر می‌باشد :

```
tn2, fp2, fn2, tp2

(8, 16, 3, 18)
```

```
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 32)                17184

dense (Dense)                (None, 1)                 33

=================================================================
Total params: 17,217
Trainable params: 17,217
Non-trainable params: 0
_________________________________________________________________
```

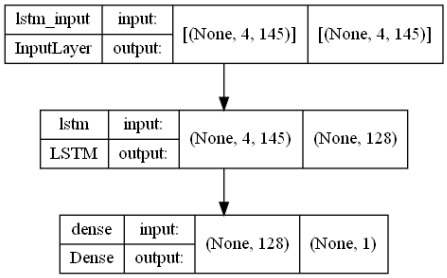


بهترین حالت برای LSTM نیز برابر است با units = 128 :

```
tn1, fp1, fn1, tp1

(10, 14, 7, 14)
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| lstm (LSTM) | (None, 128) | 140288 |
| dense (Dense) | (None, 1) | 129 |

Total params: 140,417
Trainable params: 140,417
Non-trainable params: 0




با توجه با ساختار هر دو مدل و دقت های بدست امده تعداد پارامترهای قابل یادگیری برای ساختار GRU در حدود ۱۷ هزار و برای ساختار LSTM در حدود ۱۴۰ هزار می‌باشد پس GRU با ساختاری کوچک تر توانسته است پیش بینی بهتری را ارائه دهد و با توجه به نکته‌ای که در سوال یک به ان اشاره کردیم GRU درحال حاضر با یک دیتا ست کوچک در ارتباط بوده است و دقت بهتری را نیز از خود نشان داده است. در حالی که lstm با دیتاست های بزرگ تر به دقت‌های بهتری خواهد رسید. همچنین با مراجعه به ماتریس درهم ریختگی مشاهده می‌کنیم مجموع tp+tn در مدل gru نسبت به lstm بیشتر بوده است.

---

**سوال ۳ :**

با توجه به ساختار قبل در این قسمت به پشته کردن واحد های بازگشتی می‌پردازیم

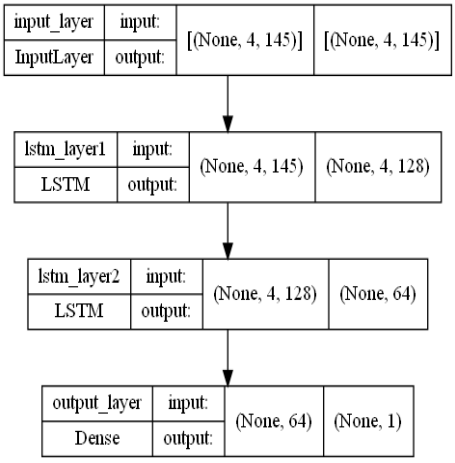
LSTM :

```
keras.backend.clear_session()
inp = keras.Input(shape=(4,145),name='input_layer')
Lstm_hidden1 = keras.layers.LSTM(128, name='lstm_layer1',return_sequences=True)
Lstm_hidden2 = keras.layers.LSTM(64,name='lstm_layer2')
output_layer = keras.layers.Dense(1,name = 'output_layer')

lstm_hideen1_output = Lstm_hidden1(inp)
lstm_hideen2_output = Lstm_hidden2(lstm_hideen1_output)
output_value = output_layer(lstm_hideen2_output)

Deep_lstm_model = keras.Model(inputs= inp , outputs =output_value)
Deep_lstm_model.compile(loss='binary_crossentropy', optimizer=keras.optimizers.Adam(learning_rate=0.0001),metrics=['accuracy'])
```

عمق واحد بازگشتی را یک مرحله افزایش داده‌ایم ساختار این شبکه و پارامترهای ان به صورت زیر می‌باشند :

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer (InputLayer) | [(None, 4, 145)] | 0 |
| lstm_layer1 (LSTM) | (None, 4, 128) | 140288 |
| lstm_layer2 (LSTM) | (None, 64) | 49408 |
| output_layer (Dense) | (None, 1) | 65 |

Total params: 189,761
Trainable params: 189,761
Non-trainable params: 0

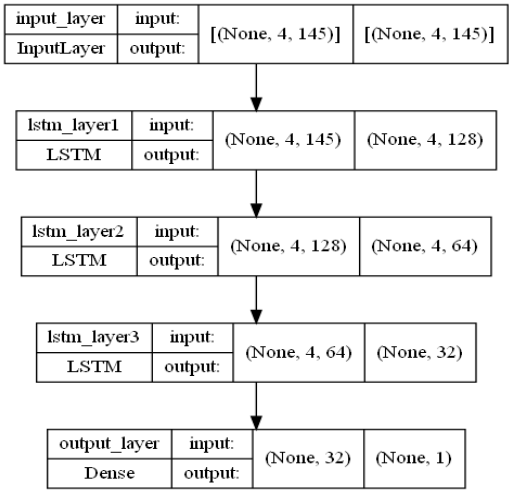


سپس شبکه را با ساختار فوق با ۱۰۰ دور اموزش اجرا می‌کنیم :

```
Epoch 96/100
5/5 [==============================] - 0s 30ms/step - loss: 0.5878 - accuracy: 0.6923 - val_loss: 0.5817 - val_accuracy: 0.65
22
Epoch 97/100
5/5 [==============================] - 0s 36ms/step - loss: 0.5865 - accuracy: 0.6795 - val_loss: 0.5746 - val_accuracy: 0.73
91
Epoch 98/100
5/5 [==============================] - 0s 31ms/step - loss: 0.5832 - accuracy: 0.7179 - val_loss: 0.5877 - val_accuracy: 0.65
22
Epoch 99/100
5/5 [==============================] - 0s 31ms/step - loss: 0.5828 - accuracy: 0.7115 - val_loss: 0.5807 - val_accuracy: 0.65
22
Epoch 100/100
5/5 [==============================] - 0s 31ms/step - loss: 0.5938 - accuracy: 0.6603 - val_loss: 0.5739 - val_accuracy: 0.73
91
2/2 [==============================] - 0s 23ms/step - loss: 0.7285 - accuracy: 0.5111

[0.7285200357437134, 0.5111111402511597]
```

دقت بر روی داده‌های اموزش تا ۶۶ درصد و اعتبار سنجی تا ۷۳ درصد رسیده است در حالی که بر روی داده‌های تست به حدود۵۱ درصد می‌باشد.

حال دوباره عمق شبکه را یک واحد دیگر زیادتر کرده ایم :




| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer (InputLayer) | [(None, 4, 145)] | 0 |
| lstm_layer1 (LSTM) | (None, 4, 128) | 140288 |
| lstm_layer2 (LSTM) | (None, 4, 64) | 49408 |
| lstm_layer3 (LSTM) | (None, 32) | 12416 |
| output_layer (Dense) | (None, 1) | 33 |

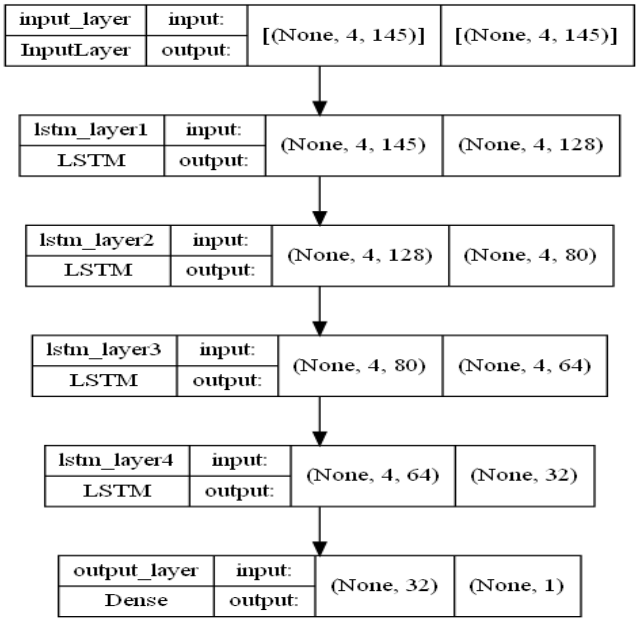
Total params: 202,145
Trainable params: 202,145
Non-trainable params: 0

سپس شبکه را با ۲۰۰ دور اموزش می‌دهیم :

```
22
Epoch 196/200
5/5 [==============================] - 0s 40ms/step - loss: 0.6212 - accuracy: 0.6667 - val_loss: 0.6392 - val_accuracy: 0.65
22
Epoch 197/200
5/5 [==============================] - 0s 42ms/step - loss: 0.6169 - accuracy: 0.6859 - val_loss: 0.6419 - val_accuracy: 0.60
87
Epoch 198/200
5/5 [==============================] - 0s 40ms/step - loss: 0.6183 - accuracy: 0.7051 - val_loss: 0.6412 - val_accuracy: 0.60
87
Epoch 199/200
5/5 [==============================] - 0s 40ms/step - loss: 0.6161 - accuracy: 0.6987 - val_loss: 0.6393 - val_accuracy: 0.65
22
Epoch 200/200
5/5 [==============================] - 0s 39ms/step - loss: 0.6201 - accuracy: 0.6731 - val_loss: 0.6419 - val_accuracy: 0.69
57
2/2 [==============================] - 0s 12ms/step - loss: 0.7179 - accuracy: 0.5333

[0.7178578972816467, 0.5333333611488342]
```

شاهد افزایش دو درصدی بر روی داده‌های ازمون شده ایم .

عمق شبکه را یک واحد دیگر افزایش می‌دهیم :



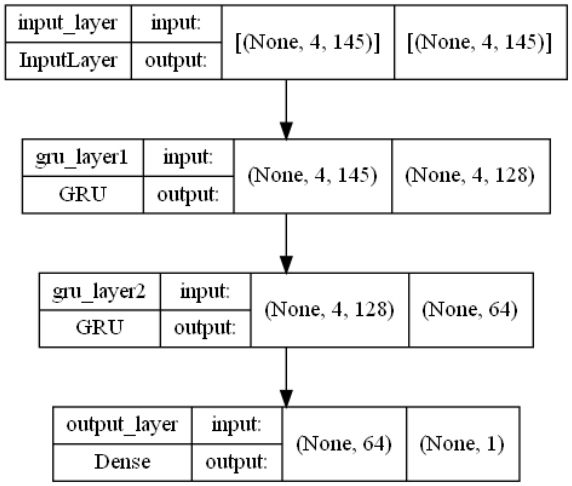

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     [(None, 4, 145)]          0

lstm_layer1 (LSTM)           (None, 4, 128)            140288

lstm_layer2 (LSTM)           (None, 4, 80)             66880

lstm_layer3 (LSTM)           (None, 4, 64)             37120

lstm_layer4 (LSTM)           (None, 32)                12416

output_layer (Dense)         (None, 1)                 33

=================================================================
Total params: 256,737
Trainable params: 256,737
Non-trainable params: 0
```

```
22
Epoch 196/200
5/5 [==============================] - 0s 60ms/step - loss: 0.6543 - accuracy: 0.6603 - val_loss: 0.6350 - val_accuracy: 0.69
57
Epoch 197/200
5/5 [==============================] - 0s 53ms/step - loss: 0.6543 - accuracy: 0.6603 - val_loss: 0.6341 - val_accuracy: 0.65
22
Epoch 198/200
5/5 [==============================] - 0s 56ms/step - loss: 0.6552 - accuracy: 0.6603 - val_loss: 0.6357 - val_accuracy: 0.65
22
Epoch 199/200
5/5 [==============================] - 0s 63ms/step - loss: 0.6535 - accuracy: 0.6603 - val_loss: 0.6338 - val_accuracy: 0.65
22
Epoch 200/200
5/5 [==============================] - 0s 52ms/step - loss: 0.6547 - accuracy: 0.6731 - val_loss: 0.6343 - val_accuracy: 0.69
57
2/2 [==============================] - 0s 14ms/step - loss: 0.7120 - accuracy: 0.5778

[0.7120218276977539, 0.5777778029441833]
```

مشاهده کردیم با افزایش عمق شاهد افزایش دقت بر روی داده‌های ازمون بودیم.

حال این عمیق تر کردن لایه ها را برروی معماری GRU اجرا می‌کنیم :

```
keras.backend.clear_session()
inp_gru = keras.Input(shape=(4,145),name='input_layer')
gru_hidden1 = keras.layers.GRU(128, name='gru_layer1',return_sequences=True)
gru_hidden2 = keras.layers.GRU(64,name='gru_layer2')
output_layer_gru = keras.layers.Dense(1,name = 'output_layer')

gru_hideen1_output = gru_hidden1(inp)
gru_hideen2_output = gru_hidden2(gru_hideen1_output)
output_value_gru = output_layer_gru(gru_hideen2_output)

Deep_gru_model = keras.Model(inputs= inp , outputs =output_value_gru)
Deep_gru_model.compile(loss='binary_crossentropy', optimizer=keras.optimizers.Adam(learning_rate=0.0001),metrics=['accuracy'])
```



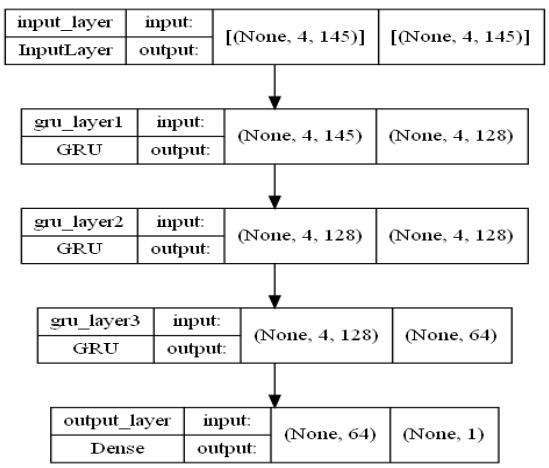

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 4, 145)]          0

 gru_layer1 (GRU)            (None, 4, 128)            105600

 gru_layer2 (GRU)            (None, 64)                37248

 output_layer (Dense)        (None, 1)                 65

=================================================================
Total params: 142,913
Trainable params: 142,913
Non-trainable params: 0
```

سپس مدل ساخته شده را با ۲۰۰ دور اموزش می‌دهیم :

```
Epoch 196/200
5/5 [==============================] - 0s 33ms/step - loss: 0.5619 - accuracy: 0.6923 - val_loss: 0.5886 - val_accuracy: 0.69
57
Epoch 197/200
5/5 [==============================] - 0s 33ms/step - loss: 0.5640 - accuracy: 0.7051 - val_loss: 0.5758 - val_accuracy: 0.73
91
Epoch 198/200
5/5 [==============================] - 0s 34ms/step - loss: 0.5600 - accuracy: 0.6923 - val_loss: 0.5891 - val_accuracy: 0.69
57
Epoch 199/200
5/5 [==============================] - 0s 33ms/step - loss: 0.5578 - accuracy: 0.7115 - val_loss: 0.5818 - val_accuracy: 0.69
57
Epoch 200/200
5/5 [==============================] - 0s 32ms/step - loss: 0.5606 - accuracy: 0.7115 - val_loss: 0.5811 - val_accuracy: 0.73
91
2/2 [==============================] - 0s 11ms/step - loss: 0.7068 - accuracy: 0.4667

[0.706840455532074, 0.46666666865348816]
```

در دقت بر روی داده‌های اموزش و اعتبار سنجی نسبت به حالت مشابهlstm بهبود داشته ایم در حالی که پارامتر های قابل یادگیری به مراتب کمتری وجود دارد.

حال عمق شبکه را یک واحد عمیق تر مشابه حالت قبل می‌کنیم :

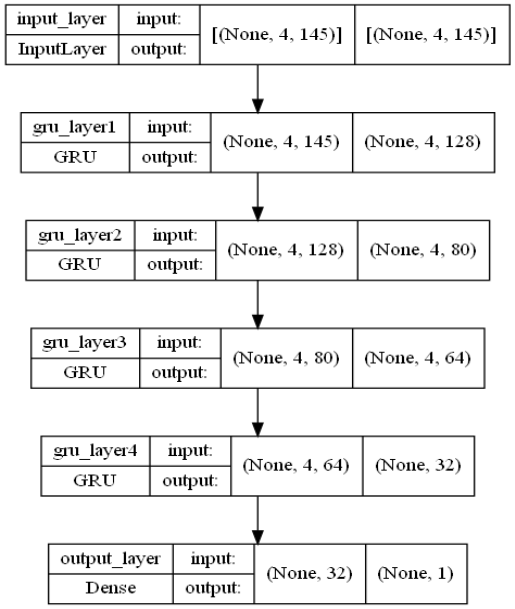



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 4, 145)]          0

 gru_layer1 (GRU)            (None, 4, 128)            105600

 gru_layer2 (GRU)            (None, 4, 64)             37248

 gru_layer3 (GRU)            (None, 32)                9408

 output_layer (Dense)        (None, 1)                 33

=================================================================
Total params: 152,289
Trainable params: 152,289
Non-trainable params: 0
```

```
Epoch 196/200
5/5 [==============================] - 0s 48ms/step - loss: 0.5841 - accuracy: 0.6923 - val_loss: 0.5795 - val_accuracy: 0.73
91
Epoch 197/200
5/5 [==============================] - 0s 56ms/step - loss: 0.5828 - accuracy: 0.6923 - val_loss: 0.5817 - val_accuracy: 0.73
91
Epoch 198/200
5/5 [==============================] - 0s 58ms/step - loss: 0.5837 - accuracy: 0.7051 - val_loss: 0.5949 - val_accuracy: 0.69
57
Epoch 199/200
5/5 [==============================] - 0s 56ms/step - loss: 0.5787 - accuracy: 0.7179 - val_loss: 0.5763 - val_accuracy: 0.73
91
Epoch 200/200
5/5 [==============================] - 0s 43ms/step - loss: 0.5874 - accuracy: 0.6987 - val_loss: 0.5751 - val_accuracy: 0.73
91
2/2 [==============================] - 0s 22ms/step - loss: 0.7115 - accuracy: 0.5111

[0.711453914642334, 0.5111111402511597]
```

شاهد افزایش در مقدار صحت برای داده‌های ازمون بوده ایم که در حدود۵۱ درصد می‌باشد.

عمق شبکه را مجدد یک واحد دیگر افزایش می‌دهیم:




```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 4, 145)]          0

 gru_layer1 (GRU)            (None, 4, 128)            105600

 gru_layer2 (GRU)            (None, 4, 80)             50400

 gru_layer3 (GRU)            (None, 4, 64)             28032

 gru_layer4 (GRU)            (None, 32)                9408

 output_layer (Dense)        (None, 1)                 33

=================================================================
Total params: 193,473
Trainable params: 193,473
Non-trainable params: 0
```

```
Epoch 196/200
5/5 [==============================] - 0s 55ms/step - loss: 0.5353 - accuracy: 0.7179 - val_loss: 0.5846 - val_accuracy: 0.69
57
Epoch 197/200
5/5 [==============================] - 0s 59ms/step - loss: 0.5304 - accuracy: 0.7308 - val_loss: 0.5375 - val_accuracy: 0.78
26
Epoch 198/200
5/5 [==============================] - 0s 68ms/step - loss: 0.5359 - accuracy: 0.7051 - val_loss: 0.6024 - val_accuracy: 0.69
57
Epoch 199/200
5/5 [==============================] - 0s 59ms/step - loss: 0.5574 - accuracy: 0.7372 - val_loss: 0.5376 - val_accuracy: 0.78
26
Epoch 200/200
5/5 [==============================] - 0s 62ms/step - loss: 0.5421 - accuracy: 0.7051 - val_loss: 0.5673 - val_accuracy: 0.69
57
2/2 [==============================] - 0s 15ms/step - loss: 1.3181 - accuracy: 0.5111

[1.3181369304656982, 0.5111111402511597]
```

مقدار صحت بر روی داده‌های اموزش تا۷۰ درصد و بر روی داده‌های اعتبارسنجی تا۶۹ درصد رسیده است .

به نظر می‌رسد که با افزایش عمق، شبکه عملکرد بهتری را از خود نشان می‌دهد اما متناسب با این افزایش عمق پارامترهای قابل یادگیری مسئله نیز افزایش پیدا می‌کنند و اموزش سخت تر خواهد شد برای پیدا کردن تعداد عمق بهینه برای شبکه مذکور با سعی و خطا می‌توان این ساختار را پیدا کرد .

برای پاسخ به این پرسش که کدام یک از این ساختارها را می‌توان عمیق تر کرد می‌توان به این مسئله اشاره کرد که gru نیاز به حافظه کمتری دارد و سریع تر نسبت به lstm می‌باشد پس با افزایش عمق این شبکه بار محاسباتی و پارامترهای قابل یادگیری به مراتب کمتر خواهد بود بدین ترتیب gru را می‌توان عمیق تر مورد استفاده قرار داد در صورت نیاز.