



به نام خدا
دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر



درس شبکه‌های عصبی و یادگیری عمیق

تمرین چهارم

نام و نام خانوادگی	فرید سیاهکلی – سعید شکوفا
شماره دانشجویی	810198418 – 810198510
تاریخ ارسال گزارش	1401.10.01

فهرست

پاسخ 1. تخمین آلودگی هوا.....4

پاسخ ۲ - تشخیص اخبار جعلی.....11

شکل‌ها

- شکل 1 اکستراکشن کردن دیتا.....5
- شکل 2 دیتای خوانده شده و مقادیر NaN در دیتاست.....5
- شکل 3 داده های از دست رفته جایگزین و سپس مقادیر NaN را.....6
- شکل 4 ستون wd را طبق خواسته به درجه ها نسبت دادیم.....6
- شکل 5 تمامی داده ها مقدار های valid دارند و دیگر NaN وجود ندارد.....6
- شکل 6 اعمال Scale بر روی دیتا.....7
- شکل 7 نمودار Pearson Correlation بین ویژگی ها.....7
- شکل 8 ذخیره ویژگی های نرمالایز شده.....8
- شکل 9 اعمال Train Test Split.....8
- شکل 10 مدل پیاده سازی شده.....8
- شکل 11 نتایج به دست آمده از train مدل برای Lag 1 در epoch 50.....9
- شکل 12 مقادیر MAE ، RMSE و R^2 برای 1 روز.....9
- شکل 13 نتایج به دست آمده از train مدل برای Lag 7 در epoch 50.....9
- شکل 14 مقادیر MAE ، RMSE و R^2 برای 7 روز.....10
- شکل 15 ساخت مدل های مورد بررسی.....12
- شکل 16 نمودار Accuracy و Loss برای مدل CNN+LSTM.....13
- شکل 17 شکل Confusion Matrix برای مدل CNN+LSTM.....13
- شکل 18 نمودار Classification Score برای مدل CNN+LSTM.....13
- شکل 19 نمودار Accuracy و Loss برای مدل LSTM.....14
- شکل 20 شکل Confusion Matrix برای مدل LSTM.....14
- شکل 21 نمودار Classification Score برای مدل LSTM.....14

جدولها

No table of figures entries found.

پاسخ 1. تخمین آلودگی هوا

1- سوالات تشریحی:

متد Linear Interpolation: برای بازیابی اطلاعات گم شده (به هر دلیلی) از روش Linear Interpolation استفاده می شود. این روش برای داده هایی که نسبت به زمان مستقل هستند به کار گرفته می شود و برای آلاینده های بهتری روش تخمین برای داده های گم شده می باشد. در این روش از فرمول زیر برای به دست آوردن دیتا در یک زمان مشخص می باشد:

$$SL(x) = f(x_{i-1}) \frac{x - x_i}{x_{i-1} - x_i} + f(x_i) \frac{x - x_{i-1}}{x_i - x_{i-1}} \quad x \in [x_{i-1}, x_i], i = 1, 2, 3, \dots, n$$

که برای به دست آوردن هر x به دیتای قبل و بعد از آن یا نزدیک ترین دیتاهای موجود به آن نیاز داریم.

متد Pearson Correlation: این روش همبستگی خطی دو متغیر نسبت به هم را ارائه می دهد و بهترین روش شناخته شده برای یافتن همبستگی خطی می باشد که از فرمول زیر تبعیت می کند:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

\bar{x}, \bar{y} در واقع میانگین همه x ها و y ها می باشند.

متد R^2 : ضریب تعیین تناسب متغیر های وابسته را که می توانند توسط متغیر های وابسته به واسطه رابطه رگرسیون شرح داده شوند را بیان میکند. هر چه این ضریب به یک نزدیک تر باشد بیانگر این است که این متغیر های مستقل بهتر می توانند متغیر های مستقل را توضیح دهند و با فرمول زیر محاسبه می شود:

$$R^2 = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}$$

که n تعداد دیتا ها و y دیتای حقیقی و \hat{y} دیتای پیش بینی شده می باشد.

2- دیتاست:

فراخوانی داده‌ها: ابتدا فایل zip را اکسترکت می‌کنیم:

```
!unzip /content/drive/MyDrive/Datasets/PRSA2017_Data_20130301-20170228.zip -d /content
```

```
Archive: /content/drive/MyDrive/Datasets/PRSA2017_Data_20130301-20170228.zip
  creating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Aotizhongxin_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Changping_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Dingling_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Dongsi_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Guanyuan_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Gucheng_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Huairou_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Nongzhanguan_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Shunyi_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Tiantan_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Wanliu_20130301-20170228.csv
  inflating: /content/PRSA2017_Data_20130301-20170228/PRSA_Data_20130301-20170228/PRSA_Data_Wanshouxigong_20130301-20170228.csv
```

شکل 1 اکسترکت کردن دیتا

تمامی فایل های csv را در فایل zip را باز می‌کنیم (با استفاده از کتابخانه pandas) و همه را در یک

دیکشنری ذخیره می‌کنیم همچنین ستوان های year month day hour را به یک ستون به نام date

تبدیل می‌کنیم:

	No	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	\	No	0
date											PM2.5	925
2013-03-01 00:00:00	1	4.0	4.0	4.0	7.0	300.0	77.0	-0.7	1023.0		PM10	718
2013-03-01 01:00:00	2	8.0	8.0	4.0	7.0	300.0	77.0	-1.1	1023.2		SO2	935
2013-03-01 02:00:00	3	7.0	7.0	5.0	10.0	300.0	73.0	-1.1	1023.5		NO2	1023
2013-03-01 03:00:00	4	6.0	6.0	11.0	11.0	300.0	72.0	-1.4	1024.5		CO	1776
2013-03-01 04:00:00	5	3.0	3.0	12.0	12.0	300.0	72.0	-2.0	1025.2		O3	1719
											TEMP	20
											PRES	20
											DEWP	20
											RAIN	20
											wd	81
											WSPM	14
											station	0
											dtype: int64	

	DEWP	RAIN	wd	WSPM	station
date					
2013-03-01 00:00:00	-18.8	0.0	NNW	4.4	Aotizhongxin
2013-03-01 01:00:00	-18.2	0.0	N	4.7	Aotizhongxin
2013-03-01 02:00:00	-18.2	0.0	NNW	5.6	Aotizhongxin
2013-03-01 03:00:00	-19.4	0.0	NW	3.1	Aotizhongxin
2013-03-01 04:00:00	-19.5	0.0	N	2.0	Aotizhongxin

شکل 2 دیتای خوانده شده و مقادیر NaN در دیتاست

3- پیش پردازش:

معضل Missing Value: با استفاده از روش interpolation داده هایی که مقدار آن ها از دست رفته است را جایگزین میکنیم:

```
for city in dict_of_dfs.keys():
    dict_of_dfs[city] = dict_of_dfs[city].interpolate("time")

dict_of_dfs["Aotizhongxin"].isna().sum()

No          0
PM2.5       0
PM10        0
SO2         0
NO2         0
CO          0
O3          0
TEMP        0
PRES        0
DEWP        0
RAIN        0
wd          81
WSPM        0
station     0
dtype: int64
```

شکل 3 داده های از دست رفته جایگزین و سپس مقادیر NaN را مشاهده میکنیم که فقط در wd موجود است

تبدیل Wind Direction به درجه:

```
wd_encoding = {"N" : 0, "NNE" : 22.5, "NE" : 45, "ENE" : 67.5,
               "E" : 90, "ESE" : 112.5, "SE" : 135, "SSE" : 157.5,
               "S" : 180, "SSW" : 202.5, "SW" : 225, "WSW" : 247.5,
               "W" : 270, "WNW" : 292.5, "NW" : 315, "NNW" : 337.5 }

for city in dict_of_dfs.keys():
    dict_of_dfs[city]["wd"] = dict_of_dfs[city]["wd"].map(wd_encoding)
```

شکل 4 ستون wd را طبق خواسته به درجه ها نسبت دادیم

```
dict_of_dfs["Aotizhongxin"].isna().sum()

No          0
PM2.5       0
PM10        0
SO2         0
NO2         0
CO          0
O3          0
TEMP        0
PRES        0
DEWP        0
RAIN        0
wd          0
WSPM        0
station     0
dtype: int64
```

شکل 5 تمامی داده ها مقدار های valid دارند و دیگر NaN وجود ندارد

پیاده سازی Min-Max Normalization بر روی داده ها: متود MinMax scalar از کتابخانه sklearn می باشد که بر روی دیتا اعمال شده است

```
col_names = Features.columns
dates_train = Features_df_Train_set.index
dates_test = Features_df_Test_set.index
Features_Train_Norm = scaler.fit_transform(Features_df_Train_set)
Features_Test_Norm = scaler.transform(Features_df_Test_set)

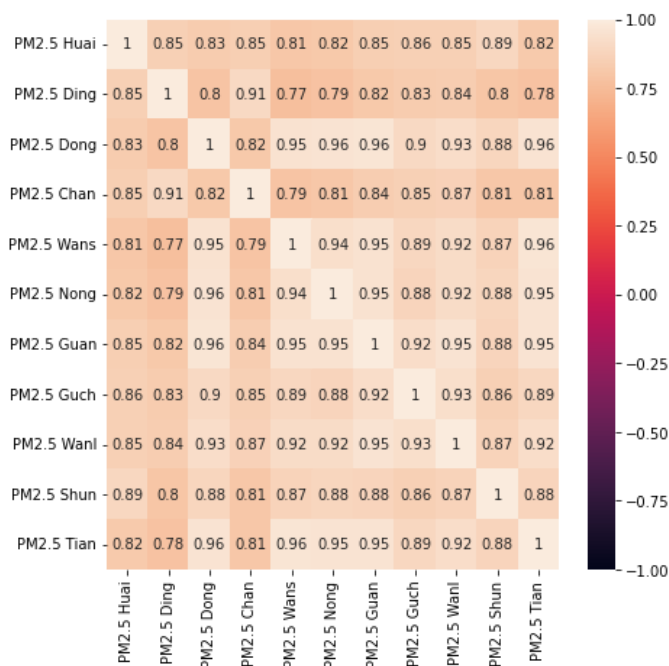
Features_Train_Norm_df = pd.DataFrame(Features_Train_Norm)
Features_Train_Norm_df.columns = col_names
Features_Train_Norm_df.set_index(dates_train,inplace=True)

Features_Test_Norm_df = pd.DataFrame(Features_Test_Norm)
Features_Test_Norm_df.columns = col_names
Features_Test_Norm_df.set_index(dates_test,inplace=True)

dump(scaler, 'MinMaxScaler.joblib')
```

شکل 6 اعمال Scale بر روی دیتا

گزارش Pearson Correlation:



شکل 7 نمودار Pearson Correlation بین ویژگی ها

می توان مشاهده کرد که دیتای PM2.5 برخی از نواحی با نواحی دیگر، از همبستگی بالایی برخوردار هستند.

ایجاد فایل Feature ها مربوط به ایستگاه Aotizhongx: بدین منظور فیچرها در فایل هایی به نام Train.csv و Test.csv ذخیره می گردند که به پیوست قرار گرفته است.

COMBINING TWO DATAFRAMES AND SAVING THEM INTO TRAIN AND TEST CSV FILES

```
[10] Train_df = pd.concat((Features_Train_Norm_df , dict_of_dfs["Aotizhongxin"]["PM2.5"].loc[dates_train]) , axis=1)
      Test_df = pd.concat((Features_Test_Norm_df , dict_of_dfs["Aotizhongxin"]["PM2.5"].loc[dates_test]) , axis=1)

      Train_df.to_csv("train.csv" , date_format = "%Y-%m-%d %H:%M:%S" )
      Test_df.to_csv("test.csv" , date_format = "%Y-%m-%d %H:%M:%S" )
```

شکل 8 ذخیره ویژگی های نرمالایز شده

تبدیل داده ها به فرم Supervised و همچنین Train-Test Split: همانطور که دیده می شود 80 درصد ابتدایی داده ها برای train و 20 درصد نهایی برای Test انتخاب شده اند.

```
index_80_percent = int(rows*0.8)

other_features = dict_of_dfs["Aotizhongxin"].drop(["No","O3" ,"NO2","SO2","PM2.5" , "station"], axis=1)

Features = pd.concat((PMS_df ,other_features ), axis=1)
Features_df_Train_set = Features.iloc[: index_80_percent]
Features_df_Test_set = Features.iloc[index_80_percent :]
```

شکل 9 اعمال Train Test Split

4- آموزش شبکه:

پیاده سازی شبکه و آموزش آن:

```
Model: "sequential"

Layer (type)                 Output Shape                 Param #
=====
conv1d (Conv1D)              (None, 24, 64)              3712

batch_normalization (BatchN  (None, 24, 64)              256
ormalization)

conv1d_1 (Conv1D)             (None, 24, 64)              12352

batch_normalization_1 (Batc  (None, 24, 64)              256
hNormalization)

conv1d_2 (Conv1D)             (None, 24, 32)              6176

max_pooling1d (MaxPooling1D  (None, 8, 32)              0
)

lstm (LSTM)                   (None, 8, 100)              53200

lstm_1 (LSTM)                 (None, 8, 50)              30200

flatten (Flatten)            (None, 400)                 0

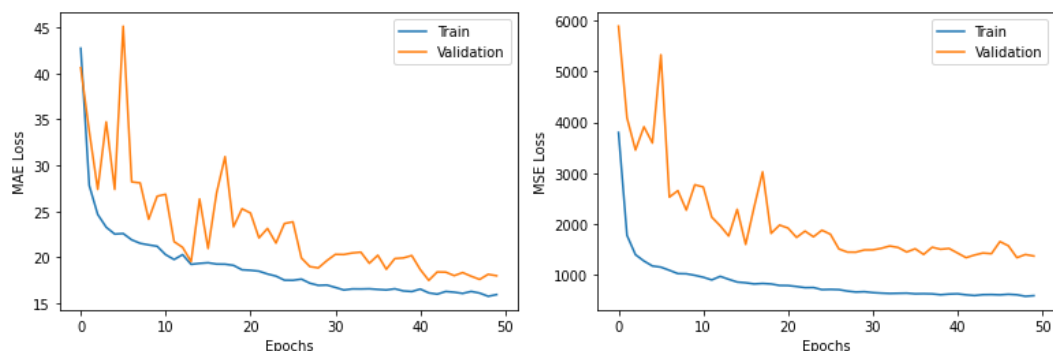
dense (Dense)                 (None, 1)                   401

=====
Total params: 106,553
Trainable params: 106,297
Non-trainable params: 256
```

شکل 10 مدل پیاده سازی شده

مدل را به اندازه epoch 50 برای lag 1 ترین مقادیر epoch آخر:

```
Epoch 50/50
700/701 [=====>.] - ETA: 0s - loss: 454.8447 - mae: 13.3486 - mse: 454.8447
Epoch 50: saving model to checkpoints_lag_1/cp.ckpt
701/701 [=====] - 21s 31ms/step - loss: 454.5803 - mae: 13.3443 - mse: 454.5803 - val_loss: 1285.8156 - val_mae: 15.3436 - val_mse: 1285.8156
```



شکل 11 نتایج به دست آمده از **train** مدل برای **Lag 1** در **epoch50**

گزارش مقادیر خطا به ازای Lag های متفاوت:

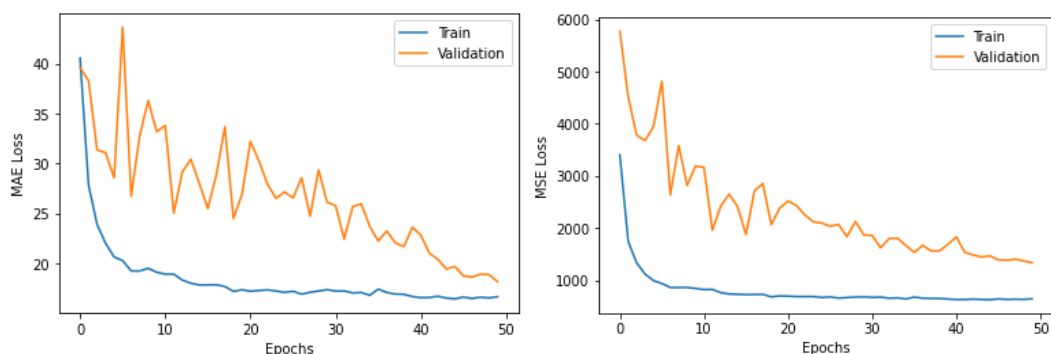
```

[➡] MAE: 14
    R2: 0.9096
    RMSE: 25.1608

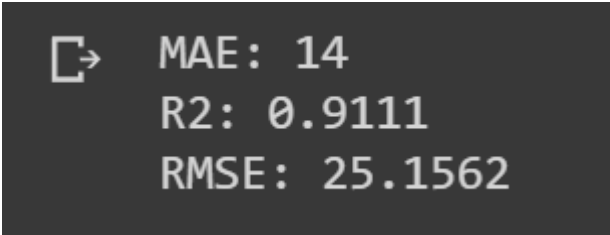
```

شکل 12 مقادیر **MAE**، **RMSE** و **R^2** برای 1 روز

حال مدل را برای Lag 7 آموزش می دهیم. نتیجه آخرین ایپاک به صورت زیر می شود:



شکل 13 نتایج به دست آمده از **train** مدل برای **Lag 7** در **epoch50**



```
➔ MAE: 14  
R2: 0.9111  
RMSE: 25.1562
```

شکل 14 مقادیر MAE ، RMSE و R^2 برای 7 روز

نتایج نشان می دهند که با افزایش Lag به مقدار 7، تغییر و بهبودی در تخمین شبکه رخ نمی دهد در نتیجه می توان دریافت که به جهت تخمین شبکه، نیازی به Lag 7 روز نیست و ویژگی های مفید اضافه ای به شبکه اضافه نمی گردد.

پاسخ ۲ - تشخیص اخبار جعلی

الف) توضیحات مدل‌ها:

تفاوت معماری RNN و LSTM: LSTM در واقع یک RNN پیچیده است و اثر بیشتر و بهتری دارد. در واقع RNN فقط به طور بازگشتی خروجی را حساب میکند ولی LSTM پارامترهای بیشتر دارد و بخشی از داده‌های قبلی را پاک کرده و بخشی دیگر را که بیشتر مورد نیاز است نگه می‌دارد تا بر اساس آن تصمیمی بگیرد (forget gate). همچنین LSTM سبب می‌شود که مشکل vanishing gradient نیز حل شود و در واقع سیستم با حافظه بهتری شود.

علت تاثیر بازگشت در داده‌های متنی: در داده‌های متنی میان هر کلمه با کلمه بعدی ارتباط وجود دارد و نمی‌توان هر کلمه‌ای را بعد از هر کلمه‌ای استفاده کرد این وابستگی هر کلمه به کلمه بعد و قبل از خود سبب می‌شود که این مدل‌ها تاثیرگذار واقع شوند. همچنین علاوه بر کلمات موضوع متن نیز بر موضع بعدی اثر دارد.

تفاوت مدل هیبرید در مقاله با مدل‌های دیگر: در این مدل شبکه CNN و RNN باهم و به طور متوالی کار میکنند به طوری که ابتدا ورودی وارد لایه CNN می‌شود و سپس بعد از آن خروجی CNN به عنوان ورودی RNN (بلوک LSTM) داده می‌شود و سپس خروجی نهایی تولید می‌شود. در واقع یک مرتبه feature map ها توسط CNN تشخیص داده می‌شوند و سپس این feature map وارد لایه LSTM می‌شود که تصمیم گیری نهایی انجام شود.

ب) ورودی مدل:

علت Word Embedding و دلیل استفاده از آن: ورودی شبکه عصبی باید یک بردار و یا یک ماتریس باشد و نمیتواند یک کلمه باشد. برای اینکه کلمه‌ها را به عنوان ورودی به شبکه عصبی بدهیم آن‌ها را به بردار تبدیل میکنیم. در واقع هر کلمه به یک بردار خاص (unique) نظیر می‌شود و هیچ دو کلمه‌ای به یک بردار یکسان نظیر نمی‌شوند (همچنین کلماتی که با یکدیگر ارتباط دارند باید بردارشان نیز مقداری شبیه به هم باشد) و این بردار را به عنوان ورودی شبکه می‌دهیم که بتواند ترین شود و عملیات طبقی بندی را برایمان انجام دهد.

راه های ایجاد Embedding: راه های متفاوتی برای این کار وجود دارد برای مثال در برخی روش ها کلماتی که کلمات دیگر را در بر می گیرند به آن ها اشاره میکنند برای مثال ایران به تهران اشاره میکند و یا در برخی دیگر کلماتی که شبیه به هم هستند به هم لینک می شوند برای مثال پادشاه به مرد لینک می شود و از آن طرف کلمه ملکه به زن لینک می شود.

در این مقاله از word2vec استفاده شده است که توسط گوگل توسعه داده شده است و شامل تجزیه و تحلیل بردارهای آموخته شده و کاوش ریاضیات بردار بر روی نمایش کلمات است ، برای مثال، کم کردن «مرد بودن» از «پادشاه» و افزودن «زن بودن» به کلمه «ملکه» منجر می شود، که تشبیه پادشاه برای ملکه است همانطور که مرد با زن است.

ج) پیاده سازی: پیش پردازش های لازمه: به جهت آموزش شبکه لازم است که ابتدا بر روی متون Embedding انجام شود. در ادامه نیز لازم است که به جملات Padding اضافه شود تا در هنگام عبور از شبکه دچار مشکل نشویم. از طرفی در ادامه با استفاده از پکیج های موجود، Stop words ها از متون پاکسازی شده اند. نهایتاً Train-Test Splitting بر روی دیتا انجام شده است.

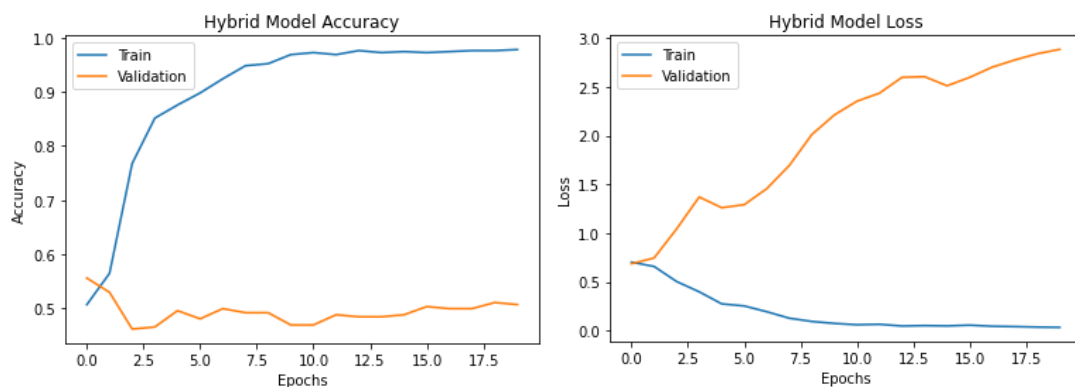
پیاده سازی دو مدل RNN و CNN-LSTM و آموزش و رسم نمودار Loss و Accuracy:

Model: "sequential_1"			Model: "sequential"		
Layer (type)	Output Shape	Param #	Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 40, 100)	500000	embedding (Embedding)	(None, 40, 100)	500000
lstm_1 (LSTM)	(None, 32)	17024	conv1d (Conv1D)	(None, 36, 128)	64128
dense_1 (Dense)	(None, 1)	33	max_pooling1d (MaxPooling1D)	(None, 18, 128)	0
=====			lstm (LSTM)	(None, 32)	20608
Total params: 517,057			dense (Dense)	(None, 1)	33
Trainable params: 517,057			=====		
Non-trainable params: 0			Total params: 584,769		
None			Trainable params: 584,769		
			Non-trainable params: 0		

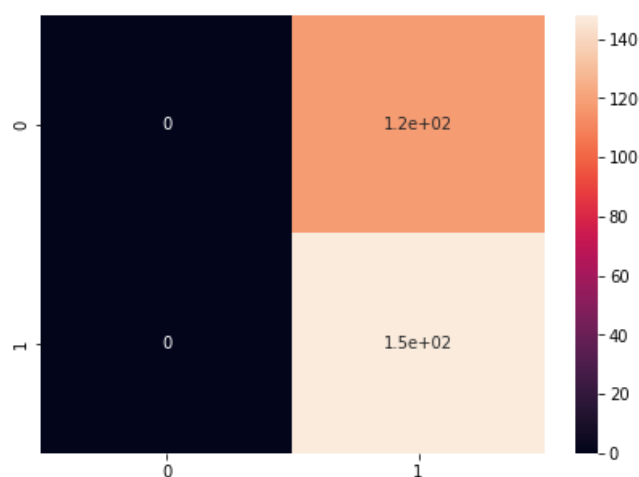
شکل 15 ساخت مدل های مورد بررسی

گزارش معیارهای مختلف برای دو مدل:

برای مدل CNN+RNN:



شکل 16 نمودار Accuracy و Loss برای مدل CNN+LSTM



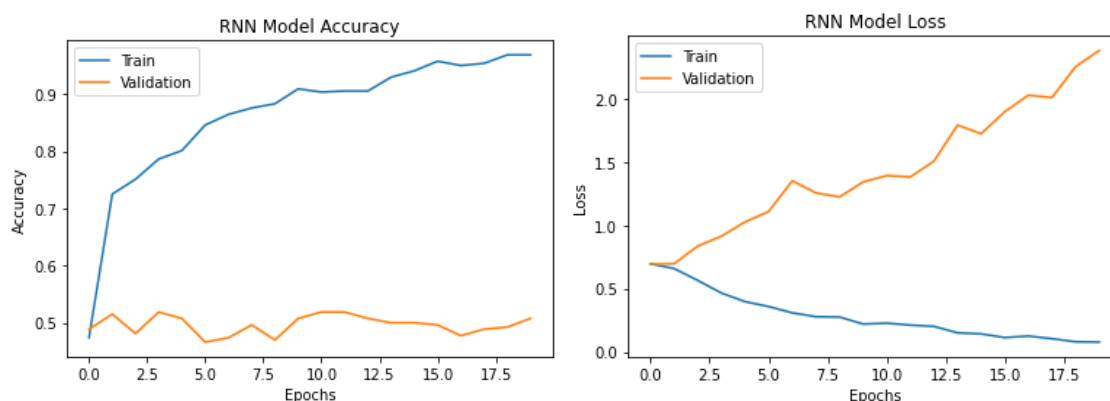
شکل 17 Confusion Matrix برای مدل CNN+LSTM

	precision	recall	f1-score	support
0	0.00	0.00	0.00	118
1	0.56	1.00	0.71	148
accuracy			0.56	266
macro avg	0.28	0.50	0.36	266
weighted avg	0.31	0.56	0.40	266

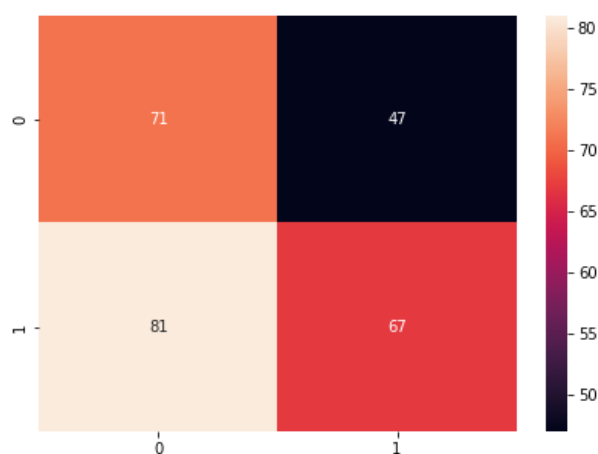
شکل 18 نمودار Classification Score برای مدل CNN+LSTM

که می توان مشاهده کرد که با این مدل به دقت 56 درصد رسیده ایم.

حال برای مدل RNN داریم:



شکل 19 نمودار Accuracy و Loss برای مدل LSTM



شکل 20 Confusion Matrix برای مدل LSTM

	precision	recall	f1-score	support
0	0.47	0.60	0.53	118
1	0.59	0.45	0.51	148
accuracy			0.52	266
macro avg	0.53	0.53	0.52	266
weighted avg	0.53	0.52	0.52	266

شکل 21 نمودار Classification Score برای مدل LSTM

د) تحلیل نتایج: می توان مشاهده کرد که با افزایش لایه کانولوشنال به شبکه، قدرت تصمیم گیری شبکه افزایش یافته و این مدل هیبرید، به نتایج بهتری دست پیدا کرده است.

علت نتایج بدست آمده و نحوه بهبود آنها: علت accuracy پایین این مدل میتواند این دلیل باشد که در واقع تمام data هایی که برای تست میگیرد در واقع دیتاهای جدید و unseen می باشند و تا به حال مشابه آن ها را ندیده است و این قضیه را برای مدل سخت میکند.

نحوه رفع ضعف موجود در مدل ها: از آنجا که مدل سریع overfit می شود میتوان از dropout استفاده کرد تا بهتر ترین شود و دیر تر overfit شود و نتایج بهتری به دست آید.