گزارش پروژه‌ی پردازش سیگنال دیجیتال

موضوع: دسته‌بندی آریتمی قلب به کمک Spectogram و Recurrence plot حاصل از سیگنال ECG قلب

استاد: آقای دکتر شکفته

ارائه از پدرام یزدی پور

شماره دانشجویی: ۹۹۴۴۳۲۴۱

بهمن ۱۴۰۱

توضیح مختصر موضوع مقاله:

در این مقاله، نوع آریتمی(بی‌نظمی) ضربان قلب به کمک شبکه‌های عصبی کانولوشنی دسته‌بندی شده است. برای این کار، ابتدا نمودار Spectogram نوار قلب افراد به کمک تبدیل فوریه‌ی کوتاه‌مدت رسم شده است و پس از برچسب‌زنی، تصاویر این نمودارها برای آموزش شبکه عصبی کانولوشنی به کار گرفته شده‌است.

توضیح مختصر کارهای پیشین:

اهمیت تشخیص زودهنگام بیماری‌های قلبی به منظور جلوگیری از بروز آسیب‌های شدیدتر در آینده بر هیچ‌کس پوشیده نیست. به همین منظور طی سالیان متمادی در مقالات زیادی به موضوع تشخیص آریتمی قلبی توسط هوش مصنوعی پرداخته شده است. در برخی کارها، سیگنال زمان گسسته‌ی قلب مستقیما توسط شبکه عصبی کانولوشنی یک بعدی پردازش شده است مانند آنچه در پردازش زبان طبیعی می‌بینیم. در بعضی کارها نیز به دلیل وجود وابستگی‌های زمانی، از شبکه‌های بازگشتی برای استخراج ویژگی‌های زمانی موجود استفاده شده است. در برخی دیگر از کارها ترکیبی از دو شبکه‌ی مذکور برای رسیدن به بهترین دقت مورد استفاده قرار گرفته است. در کنار روشهای مستقیم، برخی روشهای غیرمستقیم نیز وجود دارند که ابتدا بازنمایی داده را تغییر داده و سپس داده‌های جدید را پردازش می‌کنند. در بسیاری از مقالات، سیگنال زمان گسسته ابتدا به تصویر تبدیل شده و سپس توسط شبکه‌های عصبی کانولوشنی پردازش می‌شود. در فرآیند تبدیل سیگنال به تصویر، رویکردهای مختلفی وجود دارد از جمله تصویر نمودار تبدیل فوریه‌ی سیگنال یا تصویر نمودار سیگنال در زمان و ... . در این مقاله هدف ما دسته‌بندی نوع آریتمی قلبی با استفاده از تصاویر Spectogram سیگنال است که خود حاصل تبدیل فوریه‌ی زمان کوتاه روی سیگنال اصلی است. تبدیل فوریه‌ی زمان کوتاه این گونه است که سیگنال اولیه در قطعات جداگانه‌ای که با هم هم‌پوشانی دارند، وارد فضای فوریه می‌شوند و پس از ضرب در یک تابع پنجره، مجذور اندازه‌ی ماتریس حاصل پلات گرفته می‌شود. ادعای نویسندگان مقاله این است که با پیاده‌سازی این روش می‌توان به بهبود نتایج امیدوار بود.

توضیح دادگان:

در این مقاله از دادگان مشهور دانشگاه MIT استفاده شده است. این دادگان در دهه‌ی ۷۰ میلادی از ۴۸ فرد بالغ جمع‌آوری شده است. هر نوار قلب شامل ۳۰ دقیقه سیگنال زمان گسسته با نرخ نمونه‌برداری ۳۶۰ هرتز است. برچسب‌زنی سیگنالها بر اساس نوع خاص ضربان قلب انجام شده است و به طور کلی حدود ۱۱۰ هزار ضربان قلب و برچسب متناظرشان را در اختیار داریم. حجم دیتاست حدود ۳۰۰ مگابایت است و به راحتی قابل دانلود است. بیشتر ضربانهای قلب در کلاس نرمال قرار می‌گیرند و به این ترتیب، دیتاست از عدم تعادل رنج

می‌برد. در حالت کلی ۱۸ کلاس مختلف آریتمی وجود دارد که در عموم مقالات ۵ نوع پرتکرار مورد بررسی قرار می‌گیرند. در این مقاله قصد بررسی ۷ نوع مختلف را داریم.

نحوه‌ی پیاده‌سازی:

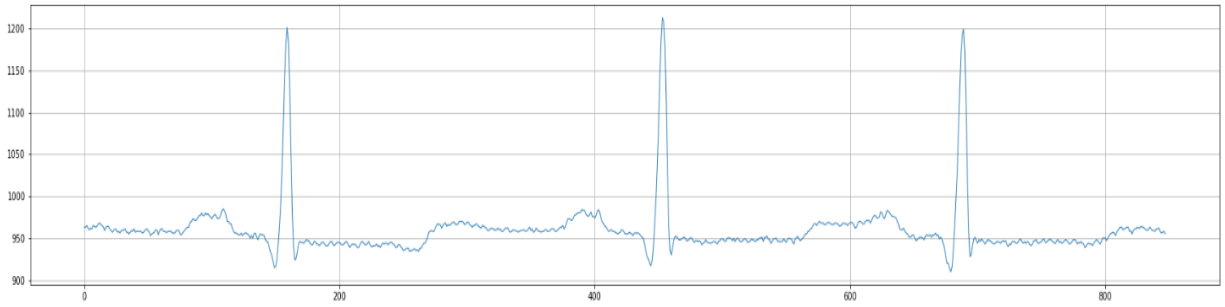
ابتدا باید ذکر کنیم که روشهای متفاوتی برای رسم اسپکتوگرام وجود دارد. در این پروژه، به ازای هر ضربان، ضربان قبل و بعد را می‌گیریم. اگر کلاس هر سه ضربان در لیست برچسبهای ما وجود داشت که ادامه می‌دهیم، در غیر این صورت آن بخش را در نظر نمی‌گیریم. بعد از بررسی کلاس سه ضربان، اگر هر سه از یک کلاس بودند که برچسب این قطعه سیگنال برابر برچسب سه ضربان است. اگر بیش از دو نوع ضربان داشته باشیم، مجددا این قطعه در نظر گرفته نمی‌شود. اگر این حداکثر دو نوع ضربان، فاقد کلاس نرمال بودند، باز این قطعه ارزشی ندارد. در واقع تلاش می‌کنیم ضربانهای خارج نرمال روی برچسب زنی قطعات مختلف از یک سیگنال نقش اصلی را داشته باشند و البته اگر یک فرد سه ضربان متوالی‌اش کاملا متفاوت است، این قطعه از سیگنال را دور می‌ریزیم. پس از تبدیل سیگنالها به اسپکتوگرام با اندازه‌ی پنجره‌ی ۲۵۶، هم‌پوشانی ۱۲۸ و سه تابع پنجره‌ی parzen، hann و triangle، از معماری شبکه عصبی کانولوشنی گفته شده در مقاله برای ارزیابی کار بهره می‌گیریم. برای پیاده‌سازی از فریم‌ورک Keras استفاده می‌کنیم. برای ارزیابی هم از دقت، صحت، فراخوانی و امتیاز F1 استفاده می‌کنیم. همچنین از پلات بازگشتی نیز به عنوان تصاویر مورد نیاز شبکه بهره خواهیم گرفت. در واقع در ۴ حالت مختلف، مقایسه انجام خواهیم داد که سه حالت برای یافتن بهترین پنجره و حالت چهارم برای مقایسه‌ی اسپکتوگرام با پلات بازگشتی است.

توضیح کد:

```
pos_cur = int(next(splitted)) # Sample ID
pos_last = int(next(last_splitted)) # Sample ID
pos_next = int(next(next_splitted)) # Sample ID
arrhythmia_type_cur = next(splitted) # Type
arrhythmia_type_next = next(next_splitted) # Type
arrhythmia_type_last = next(last_splitted) # Type
if(arrhythmia_type_cur in classes and arrhythmia_type_next in classes and arrhythmia_type_last in classes):
    last_index = classes.index(arrhythmia_type_last)
    next_index = classes.index(arrhythmia_type_next)
    cur_index = classes.index(arrhythmia_type_cur)
    tmp_set = list(set([last_index, cur_index, next_index]))
    if len(tmp_set) > 2 :
      pass
    elif 0 not in tmp_set and len(tmp_set) == 2:
      pass
    else:
      if len(tmp_set) == 1:
        #print(tmp_set[0])
        arrhythmia_index = tmp_set[0]
      else:
        if tmp_set[0] == 0:
          arrhythmia_index = tmp_set[1]
          #print(tmp_set[1])
        else:
          arrhythmia_index = tmp_set[0]
          #print(tmp_set[0])
```

در قسمت فوق با خواندن کلاس ضربان فعلی، قبلی و بعدی ابتدا چک می‌کنیم که کلاسهای هر سه در لیست کلاسهای هفت‌گانه‌ی ما باشند. سپس اگر بیشتر از دو نوع ضربان داشته باشیم، مجددا این قطعه در نظر گرفته نمی‌شود. اگر این حداکثر دو نوع ضربان، فاقد کلاس نرمال بودند، باز این قطعه ارزشی ندارد. اگر حداکثر دو نوع ضربان داریم که یکی نرمال است، کلاس این قطعه برابر برچسب دیگر است.

سپس یک نمونه از دیتای برچسب زده شده با برچسب ۳ را می‌بینیم:



بعد از جمع‌آوری دیتا، توزیع کلاسهای مختلف را بررسی می‌کنیم:

```
mycolors = ["black", "hotpink", "b", "#4CAF50",'red', 'green', 'yellow', 'orange']
plt.pie(count_classes, labels = ['N', 'L', 'R', 'A', 'V', '/','F'], colors = mycolors)
plt.show()
```

همان‌طور که قابل مشاهده است، از هر کلاس به تعداد برابر(۱۰۰۰) نمونه داریم.

قطعه کد روبرو نیز برای نرمالسازی مقادیر سیگنال

به بازه‌ی صفر تا یک استفاده می‌شود:

```
def normalize(arr):
    t_min = np.min(arr)
    t_max = np.max(arr)
    norm_arr = []
    for i in arr:
        temp = (i - t_min)/(t_max - t_min)
        norm_arr.append(temp)
    return np.array(norm_arr)
```

کد زیر برای رسم نمودار بازگشتی استفاده می‌شود:

```
from scipy.spatial.distance import pdist, squareform

def rec_plot(s, eps, steps):
    d = pdist(s)
    d = np.floor(d/eps)
    d[d>steps] = steps
    Z = squareform(d)
    return Z

def generate_recurrence_plot(ecg_sample, path, file_name, label):
  my_dpi=300
  f = plt.figure(figsize=(224/my_dpi, 224/my_dpi), dpi=my_dpi)
  b = plt.imshow(rec_plot(np.array(ecg_sample).reshape(-1,1), eps=0.7, steps=10), cmap='gray')
  plt.savefig(path+label+'/'+file_name)
  f.clf()
  plt.close(f)
```

ابتدا فاصله‌ی اقلیدسی تمام نقاط از هم محاسبه می‌شود. می‌دانیم هر قطعه از سیگنال ۸۹۵ نمونه دارد(سه ضربان قلب) پس به دلیل متقارن بودن فواصل نقاط از یکدیگر، می‌توانیم نصف حالات معادل ۴۰۰ هزار فاصله را بسنجیم. سپس با تغییری روی مقادیر فواصل، کف این مقادیر گرفته شده و به ازای مقادیر بیشتر از مقدار step، همین مقدار جایگزین می‌شود. در پایان، بردار فاصله تبدیل به ماتریس مربعی متقارن فواصل شده و با نام Z بازگردانده می‌شود. تابع generate_recurrence_plot برای تولید تصاویر پلاتهای بازگشتی و ذخیره‌ی آنها در گوگل درایو شخصی نوشته شده است.

تقسیم دادگان به دادگان آموزش و تست:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

تولید اسپکتوگرام:

```
import librosa
import librosa.display
from matplotlib.colors import LinearSegmentedColormap

def generate_stft_img(ecg_sample, n_fft , hop_length , pad_mode, window, path, file_name, label):

    cmap=LinearSegmentedColormap.from_list('gyr',["g", "y", "r"], N=256)
    my_dpi=300
    f = plt.figure(figsize=(224/my_dpi, 224/my_dpi), dpi=my_dpi)
    plt.ioff()
    D = np.abs(librosa.stft(ecg_sample, n_fft=n_fft, win_length=n_fft, hop_length = hop_length, pad_mode = pad_mode, window = window))
    db = librosa.amplitude_to_db(D,ref=np.max)
    plt.axis('off')
    img = librosa.display.specshow(db, sr=360, hop_length = hop_length, y_axis='log', x_axis='time', cmap='rainbow')
    plt.savefig(path+label+'/'+file_name)
    f.clf()
    plt.close(f)
```

در قطعه کد بالا نیز تابعی با دریافت قطعه سیگنال، سایز پنجره، مقدار هم‌پوشانی، حالت افزودن حاشیه، نوع تابع پنجره و اسم و آدرس ذخیره‌ی تصویر، اسپکتوگرام هر قطعه را رسم می‌کند.

ذخیره‌ی اسپکتوگرام و پلاتهای بازگشتی در قالب تصویر به ازای دادگان آموزش و تست:

```
labels = ['N', 'L', 'R', 'A', 'V', '/','F']
for i in range(0,len(X_train)):
  tmp_name = 'image_' + str(i)
  if labels[int(y_train[i])] == '/':
    generate_recurrence_plot(normalize(np.array(X_train[i])), path = path_train, file_name = tmp_name, label = 'P')
    #generate_stft_img(normalize(np.array(X_train[i])), n_fft = 256, hop_length = 128, pad_mode='wrap',window='hann', path = path_train,
  else:
    generate_recurrence_plot(normalize(np.array(X_train[i])), path = path_train, file_name = tmp_name, label = labels[int(y_train[i])])
    #generate_stft_img(normalize(np.array(X_train[i])), n_fft = 256, hop_length = 128, pad_mode='wrap',window='hann', path = path_train,
```

ما به این ترتیب عمل کردیم که ابتدا برای هر کدام از چهار روش شامل سه حالت اسپکتوگرام و یک حالت پلات بازگشتی، تصاویر مربوطه را در فولدرهایی به نام برچسب کلاسشان ذخیره کردیم و بعد به کمک کلاس ImageDataGenerator از Keras، دادگان خارجی را ساختیم:

```
tr = ImageDataGenerator(rescale=1/255)
val = ImageDataGenerator(rescale=1/255)
train_dataset = tr.flow_from_directory(path_train, target_size = (224,224),batch_size = 128, class_mode='categorical', shuffle=True, seed = 42)
test_dataset = val.flow_from_directory(path_val, batch_size = 128, target_size = (224,224), class_mode='categorical', shuffle=False, seed = 42)
```

```
Found 5601 images belonging to 7 classes.
Found 1401 images belonging to 7 classes.
```

```
train_dataset.class_indices
```

```
{'A': 0, 'F': 1, 'L': 2, 'N': 3, 'P': 4, 'R': 5, 'V': 6}
```

همانطور که می‌بینید دیتاست ما شامل ۷ کلاس است و دادگان آموزش شامل ۵۶۰۱ تصویر و دادگان تست شامل ۱۴۰۱ تصویر است. رزولوشن تصاویر ۲۲۴ در ۲۲۴ است. مقدار سایز دسته ۱۲۸ در نظر گرفته شده است زیرا با افزایش این مقدار، رم گوگل کولب تمام می‌شود.

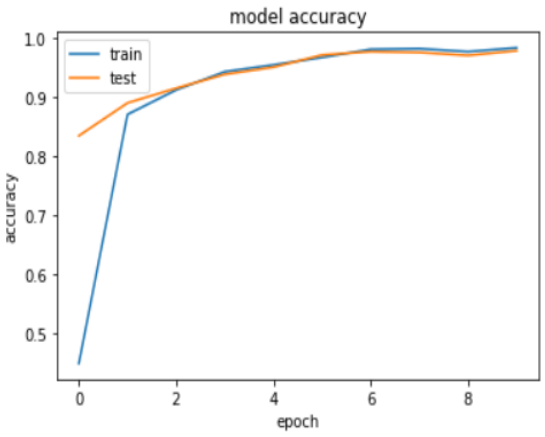
در صفحه‌ی بعد، معماری مدل را خواهیم دید؛ این شبکه دارای چهار لایه‌ی کانولوشنی و پولینگ متوالی است که تعداد فیلترها از ۶۴ تا ۵۱۲ افزایش می‌یابد. سایز فیلترها نیز در تمام لایه‌ها ۳ در ۳ است و تابع فعالسازی relu خواهد بود. پس از صاف کردن تمام فیچرمپها، از یک لایه‌ی تمام متصل با ۱۰۲۴ نورون و در لایه‌ی آخر

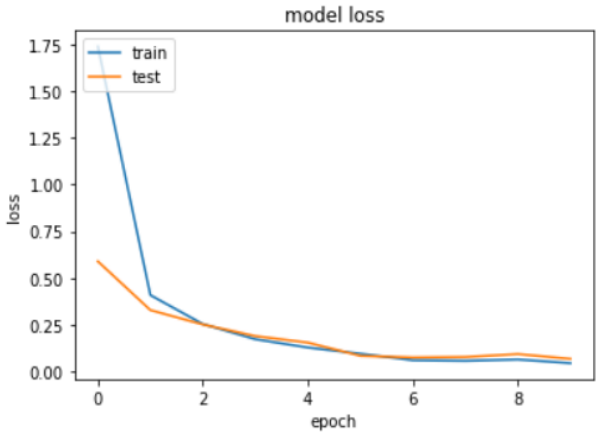
از ۷ نورون با تابع فعالسازی softmax برای دسته‌بندی استفاده شده است. Adam با نرخ یادگیری ۰.۰۰۱ نیز بهینه‌ساز این مدل خواهد بود.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64,(3,3),activation='relu',input_shape = (224,224,3)),
    tf.keras.layers.MaxPool2D(2,2),

    tf.keras.layers.Conv2D(128,(3,3),activation='relu'),
    tf.keras.layers.MaxPool2D(2,2),

    tf.keras.layers.Conv2D(256,(3,3),activation='relu'),
    tf.keras.layers.MaxPool2D(2,2),

    tf.keras.layers.Conv2D(512,(3,3),activation='relu'),
    tf.keras.layers.MaxPool2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(7, activation='softmax')
])


model.compile(loss = 'categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              metrics = ['accuracy'])

model.summary()
```

در ادامه نمودارهای خطا و صحت را روی دادگان آموزش و تست برای هر ۴ مدل خواهیم دید. تفاوت این ۴ مدل نه در معماری(چون معماری همان معماری بهینه‌ی مقاله است تا مقایسه منصفانه باشد) بلکه در تصاویر مورد استفاده برای آموزش و تست است. در سه حالت از تصاویر اسپکتوگرام با پنجره‌های مثلثی، پارزن و هن و یک حالت نیز از تصاویر پلات بازگشتی استفاده شده است.

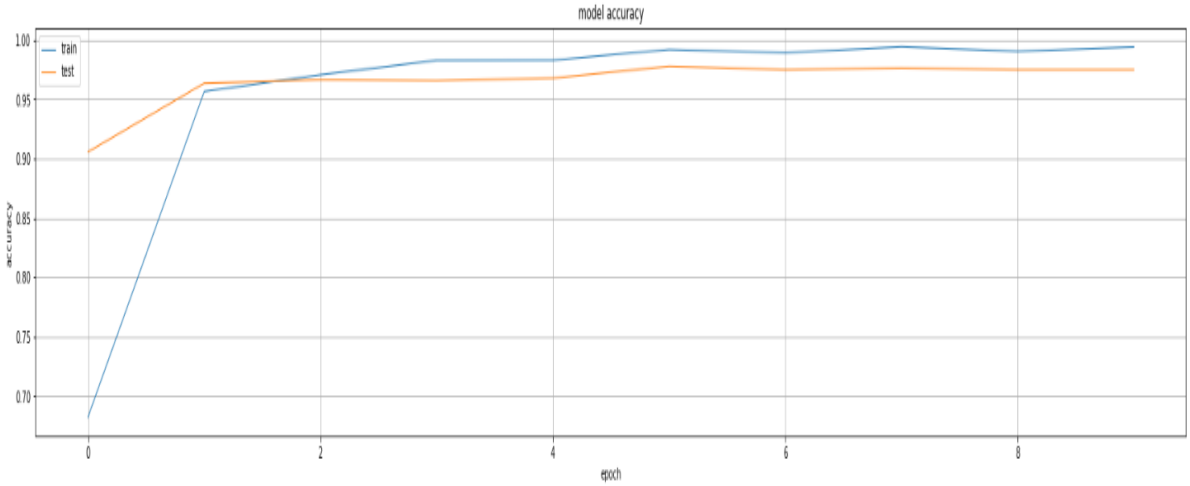
نمودارهای حالت اول، اسپکتوگرام با تابع پنجره‌ی Hann:



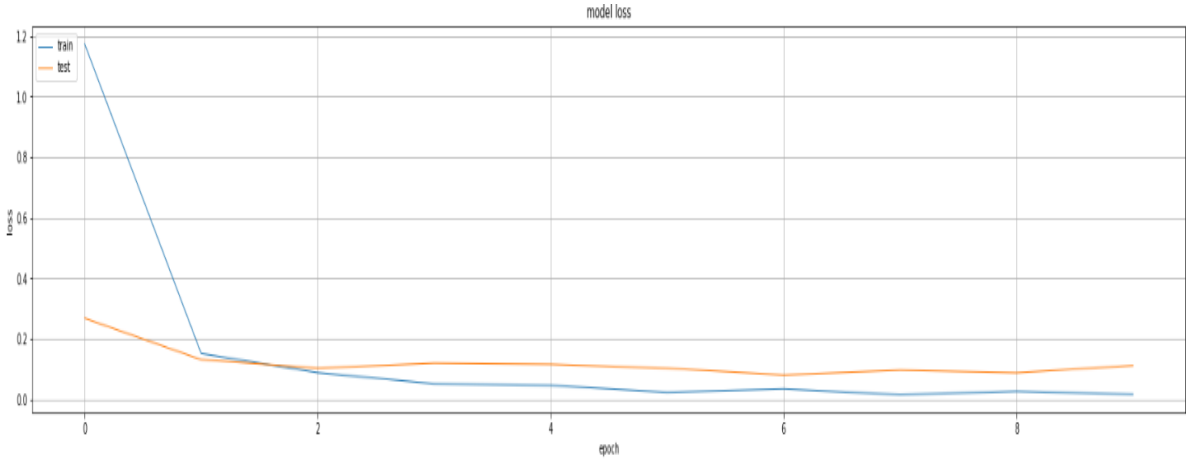


نمودارهای حالت دوم، پلات بازگشتی:

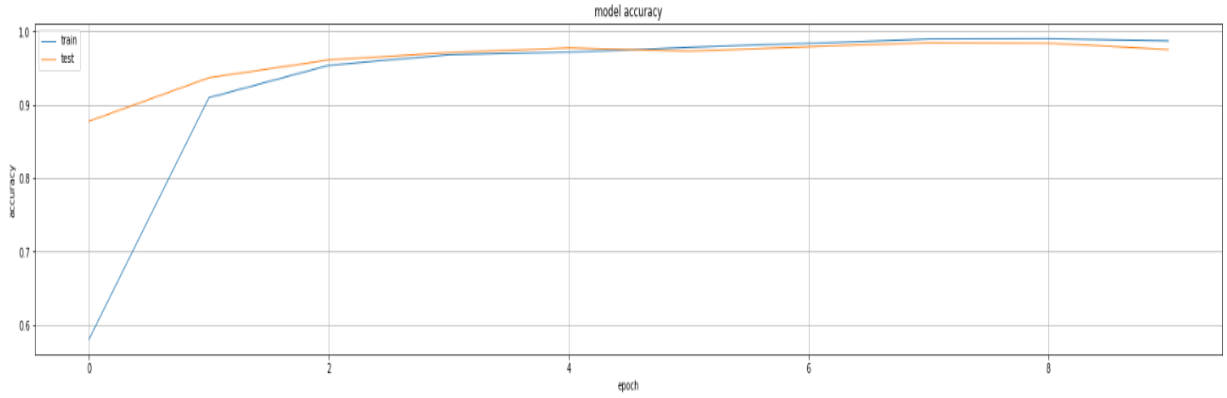
صحت:



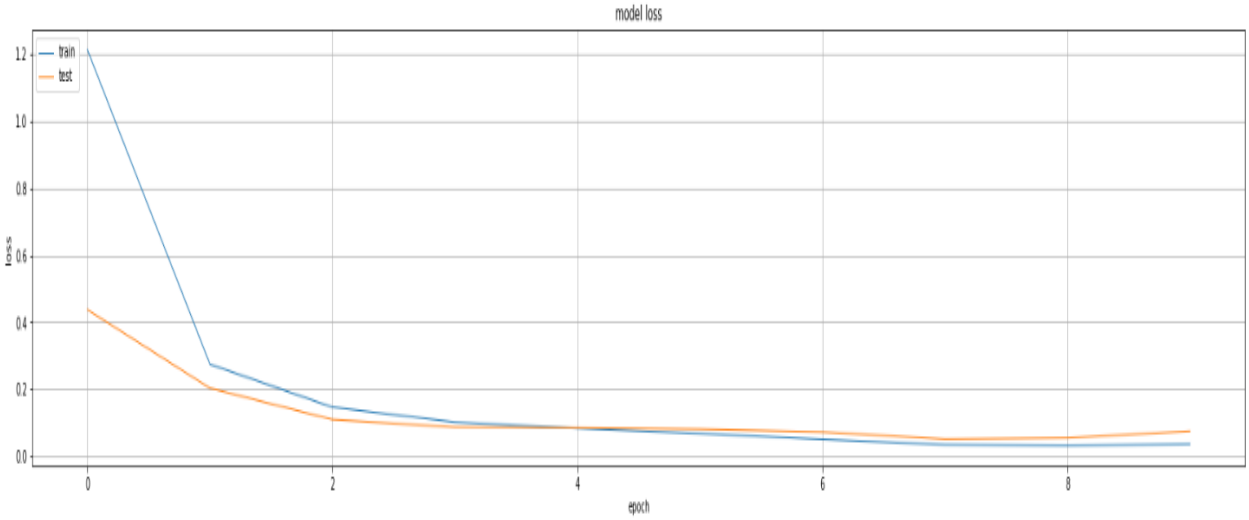
خطا:



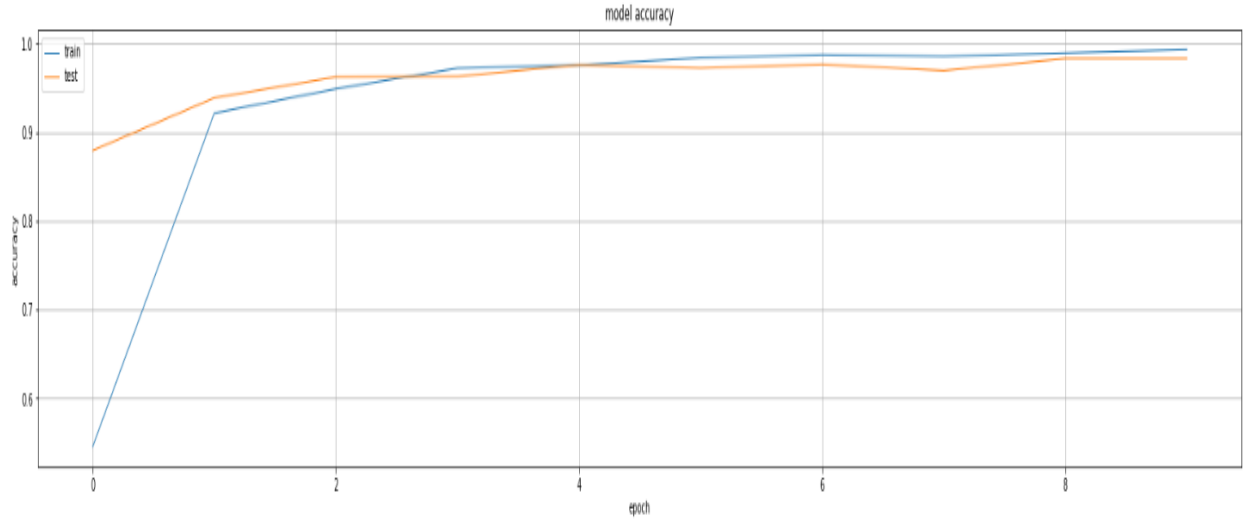
نموارهای حالت سوم، اسپکتوگرام با تابع پنجره‌ی مثلثی: (نارنجی برای تست است) – نمودار صحت



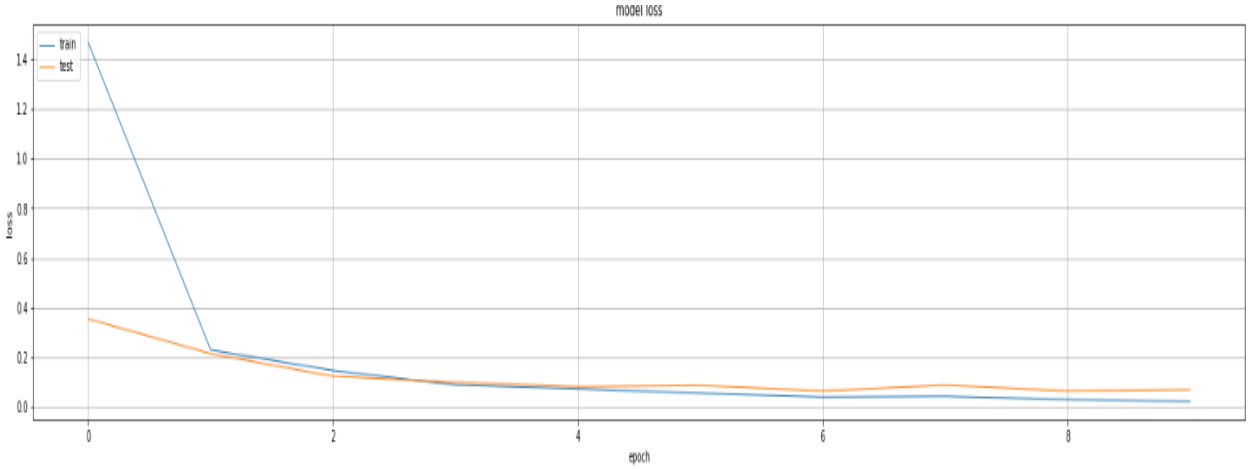
نمودار خطای حالت مثلثی: (نارنجی برای تست است)



نمودارهای حالت چهارم، تصاویر اسپکتوگرام با تابع پنجره‌ی پارزن: (نمودار صحت، نارنجی برای تست است)



نمودار خطا: (نارنجی برای تست است)



با بررسی نمودارهای فوق می‌توان دریافت هیچکدام از حالات فوق دچار بیش‌برازش نشده‌اند و میزان صحت نیز حداقل تا گام هشتم در حال افزایش است.

پس از اتمام آموزش، باید مقادیر معیارهای ارزیابی ذکر شده در ابتدای گزارش شامل دقت، صحت، فراخوانی و امتیاز F1 را نیز گزارش کنیم. برای این کار از کد زیر استفاده می‌کنیم:

```
predictions = model.predict(test_dataset)
final_pred = np.argmax(predictions, axis=-1)
print(final_pred.shape)
```

```
11/11 [==============================] - 5s 447ms/step
(1401,)
```

خروجی مدل به ازای هر نمونه‌ی سیگنال ورودی موجود در دادگان تست، یک بردار ۷ مولفه‌ای است که هر مولفه احتمال تعلق سیگنال ورودی به یکی از کلاسها را بیان می‌کند و مجموع این ۷ مقدار برابر ۱ هستند. کلاس پیش‌بینی شده را برابر عددی بین ۰ تا ۶ که بالاترین احتمال را دارد، در نظر می‌گیریم.

کد مربوط به ذخیره‌ی مدل در گوگل درایو پس از آموزش:

```
model.save('/content/drive/MyDrive/parzen/my_model.h5')
```

کد مربوط به بارگذاری مدل از گوگل درایو:

```
model = load_model('/content/drive/MyDrive/parzen/my_model.h5')
```

کد مربوط به محاسبات معیارهای ارزیابی :

```
model = load_model('/content/drive/MyDrive/parzen/my_model.h5')
predictions = model.predict(test_dataset)
final_pred = np.argmax(predictions, axis=-1)
print('precision_score: ',precision_score(target, final_pred, average='macro'))
print('accuracy_score:  ',accuracy_score(target, final_pred))
print('recall_score:    ',recall_score(target, final_pred, average='macro'))
print('f1_score:        ',f1_score(target, final_pred, average='macro'))
```

```
precision_score:  0.9830114474008692
accuracy_score:   0.9828693790149893
recall_score:     0.9829248585593763
f1_score:         0.9828997234552734
```

جدول معیارهای ارزیابی:

Model	Accuracy	Precision	Recall	F1
Spec-triangle	97.57	97.76	97.56	97.61
Spec-parzen	98.28	98.30	98.29	98.28
Spec-hann	97.85	97.88	97.86	97.87
Recurrence plot	97.50	97.51	97.51	97.46

مدلی که با استفاده از تصاویر اسپکتوگرام با تابع پنجره‌ی پارزن آموزش دیده است، بهترین نتایج را دارد.

لینک دادگان خام:

https://drive.google.com/drive/folders/1NwifQJ8oT3i0k_iyt4gKNtJprk94gcyo?usp=sharing

لینک مدل و دادگان اسپکتوگرام پارزن:

https://drive.google.com/drive/folders/1QpwzDOYgw6ULX7xvvGKuQOeWKv270nz0?usp=share_link

لینک مدل و دادگان اسپکتوگرام مثلثی:

https://drive.google.com/drive/folders/17fc5GUeFCWYI-w2EcBv4Qsd58NtksB3L?usp=share_link

لینک مدل و دادگان اسپکتوگرام هن:

https://drive.google.com/drive/folders/1kRa5-8hnRAPZJNp2jRlhKpohkSu9IADz?usp=share_link

لینک مدل و دادگان پلات بازگشتی:

https://drive.google.com/drive/folders/1iXx-NHaLD_35eHDUc4GAB-jEwBYjNwCz?usp=share_link

لینک نوتبوک(شامل کد و خروجیها):

https://colab.research.google.com/drive/1MxS8PIlmGQhnf_ljv1o4TEnYC-S917Zq?usp=sharing