گزارش تمرین شماره ۴ بخش ۲ (تشخیص احساسات)

امیرحسین باباجانی ۹۷۲۲۲۰۰۹

درسا الهویی ۹۷۲۲۲۰۰۵

در این پروژه هدف تشخیص احساسات شخص گوینده در یک فایل صوتی می باشد. از ۵ کلاس احساسات در این پروژه استفاده شده است. برای این کار از ۱۹۹۴ داده train استفاده می کنیم تا یادگیری مدل را انجام دهیم.

از آنجایی که کلاس هر داده در نام آن داده مشخص شده است پس باید اطلاعات مربوط به کلاس در هر داده را استخراج کنیم و در متغیری به نام train_label قرار می دهیم.

```
train_label = []
for name in train_name_list:
  train_label.append(name[11:12])
```

```
train_label = pd.get_dummies(train_label)
train_label
```

	A	H	N	S	W
0	1	0	0	0	0
1	1	0	0	0	0
2	0	0	0	0	1
3	0	0	1	0	0
4	1	0	0	0	0
...	...	...	...	...	...
1989	1	0	0	0	0
1990	0	0	1	0	0
1991	1	0	0	0	0
1992	0	0	1	0	0
1993	1	0	0	0	0

1994 rows × 5 columns

تعداد داده ها در هر کلاس را با کمک np.sum مشخص می کنیم.

```
train_class_size = np.sum(train_label)
train_class_size
```

```
A    723
H    130
N    690
S    302
W    149
```

از chroma_cens و mfcc به عنوان feature extractor با feature size=36 برای هر کدام استفاده شده.

```
def extract_mfcc(file_path):
  y,sr=librosa.load(file_path)
  mfccs=np.mean(librosa.feature.mfcc(y=y,sr=sr,n_mfcc=36).T,axis=0)
  return mfccs
```

```
def extract_cens(file_path):
  y,sr=librosa.load(file_path)
  cens =np.mean(librosa.feature.chroma_cens(y=y,sr=sr,n_chroma=36).T,axis=0)
  return cens
```

MFCC :

MFCC در اصل برای شناسایی کلمات تک اصطلاحی در جملات مداوم گفته می شود اما برای شناسایی گوینده پیشنهاد نمی شود.

محاسبه MFCC همان تکرار سیستم شنوایی انسان است که قصد دارد اصل کارکرد گوش را به طور مصنوعی اجرا کند با این فرض که گوش انسان یک تشخیص دهنده فرد گوینده است

ریشه ویژگی های MFCC در مغایرت شناخته شده پهنای باند بحرانی گوش انسان با فیلترهای فرکانس است که به طور خطی در فرکانس های پایین فاصله دارند و به صورت لگاریتمی در فرکانس های بالا برای حفظ خواص حیاتی آوایی سیگنال گفتار استفاده شده اند.
MFCC برای شناسایی رزرو هواپیمایی ، اعداد گفته شده در یک سیستم تشخیص صدا و تلفن برای اهداف امنیتی استفاده می شود. برخی از اصلاحات برای استحکام بهتر به الگوریتم اساسی MFCC پیشنهاد شده است ، مانند بالا بردن دامنه های ورود به سیستم به توان مناسب قبل از استفاده از DCT و کاهش تأثیر قطعات کم انرژی.

https://www.intechopen.com/books/from-natural-to-artificial-intelligence-algorithms-and-applications/some-commonly-used-speech-feature-extraction-algorithms

chroma :

ویژگی های صوتی مبتنی بر کروم به عنوان ابزاری قدرتمند برای انجام کارهای مختلف تجزیه و تحلیل در بازیابی اطلاعات موسیقی از جمله کارهایی مانند chord labeling ، خلاصه سازی موسیقی ، تجزیه و تحلیل ساختار ، همگام سازی موسیقی و ترازبندی صوتی تبدیل شده است.
با شناسایی اجزای طیفی که با یک اکتاو موسیقی متفاوت هستند ، ویژگی های کروم دارای درجه قابل توجهی از استحکام در تغییر میزان صدا و ابزار هستند.
جعبه ابزار Chroma شامل پیاده سازی های MATLAB برای استخراج ویژگی های مختلف معنادار موسیقی از سیگنال های صوتی مبتنی بر شکل موج است. به طور خاص ، این شامل استخراج کننده های ویژگی برای ویژگی های گام و همچنین خانواده های پارامتر شده از انواع ویژگی های کروم است.
ویژگی های CENS :
با در نظر گرفتن آمار کوتاه مدت از توزیع انرژی در باند های کروم ، درجه دیگری از انتزاع را به دست می آوریم ، ویژگی های CENS (Chroma Energy Normalized Statistics) به دست می آید ، که خانواده ای از ویژگی های صوتی مقیاس پذیر و قوی را تشکیل می دهد.
ویژگی های CENS ، به شدت با محتوای هارمونیک کوتاه مدت سیگنال صوتی زمینه ای ارتباط برقرار می کند و تغییراتی از خصوصیات مانند پویایی ، میزان صدا ، بیان ، اجرای گروه های یادداشت و انحرافات ریز زمانی را جذب می کند.
علاوه بر این ، به دلیل temporal resolution پایین ، ویژگی های CENS می توانند به طور کارآمد پردازش شوند.

http://resources.mpi-inf.mpg.de/MIR/chromatoolbox/index.html#MuellerKC05_ChromaFeatures_ISMIR

LPC :

ضرایب پیش بینی خطی LPC از دستگاه صوتی انسان تقلید می کند و ویژگی گفتاری محکم می دهد. این سیگنال گفتاری را با تقریب شکل دهنده ها ارزیابی می کند ، از شر اثرات آن از سیگنال گفتار خلاص می شود و غلظت و فرکانس باقی مانده مانده را تخمین می زند. در نتیجه هر نمونه از سیگنال به عنوان یک ترکیب مستقیم از نمونه های قبلی بیان می شود. ضرایب معادله اختلاف ، فرم ها را مشخص می کند ، بنابراین ، LPC نیاز به تقریبی این ضرایب دارد . LPC یک روش تجزیه و تحلیل گفتار قدرتمند است و به عنوان یک روش تخمین مهم به شهرت رسیده است.

LPC به طور کلی برای بازسازی گفتار استفاده می شود. روش LPC معمولاً در شرکتهای موسیقی و برق برای ایجاد رباتهای متحرک ، در شرکتهای تلفنی ، تجزیه و تحلیل تونال ویولنها و دیگر ابزارهای رشته ای موسیقی اعمال می شود.

- البته در این پروژه از LPC استفاده نشده.

توابع extract_cens و extract_mfcc را روی داده های train و test اجرا می کنیم و برای اینکه برای برای دادن به عنوان ورودی به مدل آماده شود، آن را reshape می کنیم و همچنین به array تبدیل می کنیم.

```
for i in range(len(test_name_list)):
  cens = test_sound_feature_cens[i]
  feature = np.array([cens]).reshape(36,1)
  test_cens_extracted_features.append(feature)

test_cens_extracted_features = np.asarray(test_cens_extracted_features)
```

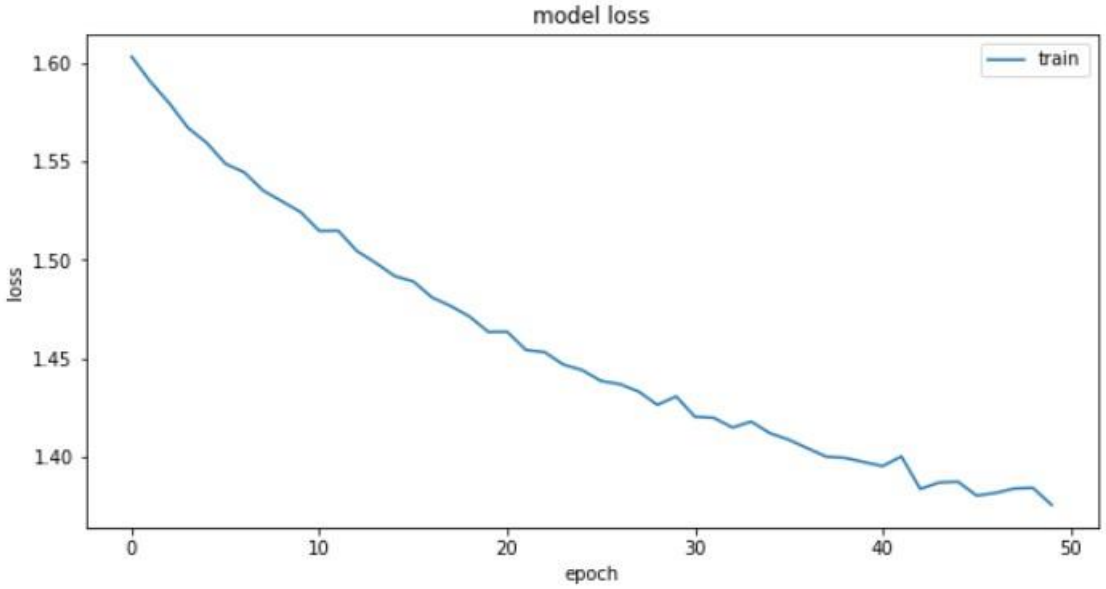
برای تعریف مدل از معماری LSTM با ۱۲۸ unit استفاده می کنیم. بجز لایه خروجی که از softmax برای activation function استفاده شده، در سایر لایه ها از relu استفاده کردیم.

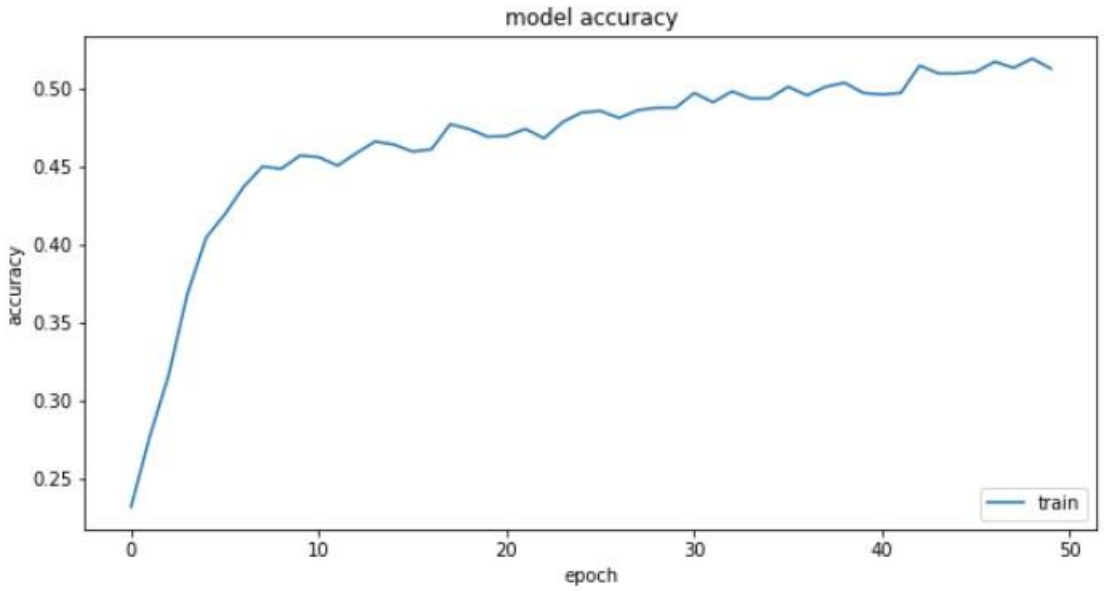
```
model = Model( inputs=input_layer, outputs=output_layer ,name='model')
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input (InputLayer)           [(None, 36, 1)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 128)               66560
_________________________________________________________________
layer1 (Dense)               (None, 32)                4128
_________________________________________________________________
layer2 (Dense)               (None, 16)                528
_________________________________________________________________
dropout (Dropout)            (None, 16)                0
_________________________________________________________________
layer3 (Dense)               (None, 8)                 136
_________________________________________________________________
output (Dense)               (None, 5)                 45
=================================================================
Total params: 71,397
Trainable params: 71,397
Non-trainable params: 0
```

ازSGD به عنوان optimizer با learning rate=0/0001 و momentum=0/9 استفاده شده و همچنین از categorical cross entropy به عنوان loss function استفاده شده.

ابتدا مدل را فقط با کمک feature های استخراج شده از mfcc با ۵۰ epoch اجرا می کنیم که نمودار accuracy و loss function برحسب تعداد epoch ها در زیر مشاهده می شود.





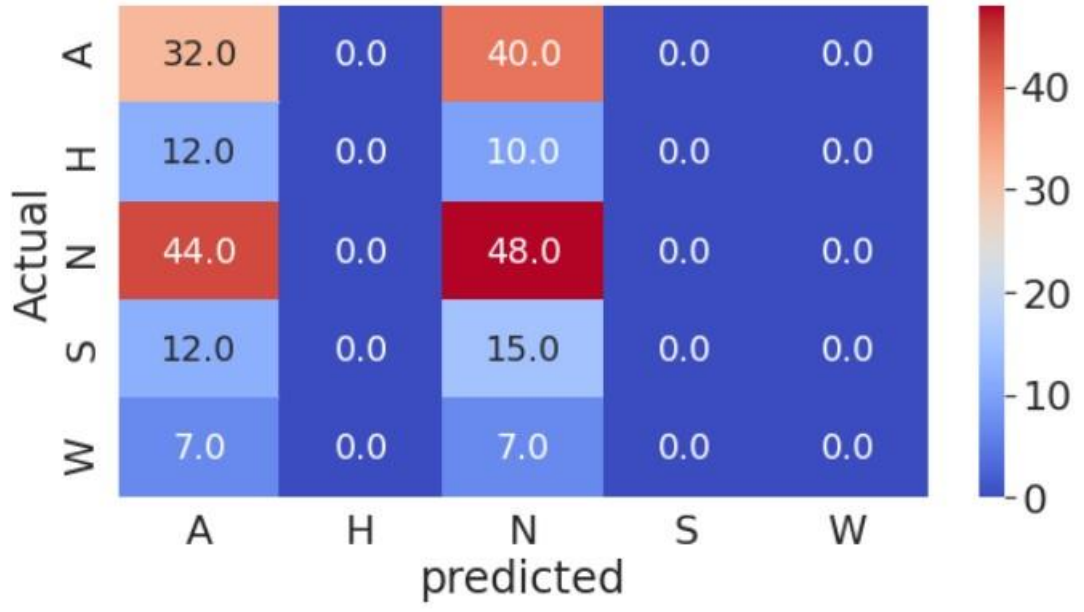
حال داده های test را به مدل می دهیم تا پیش بینی را انجام دهد و نتیجه را در زیر مشاهده میکنیم.

```
predicted = np.argmax(predicted, axis=1)
predicted
```

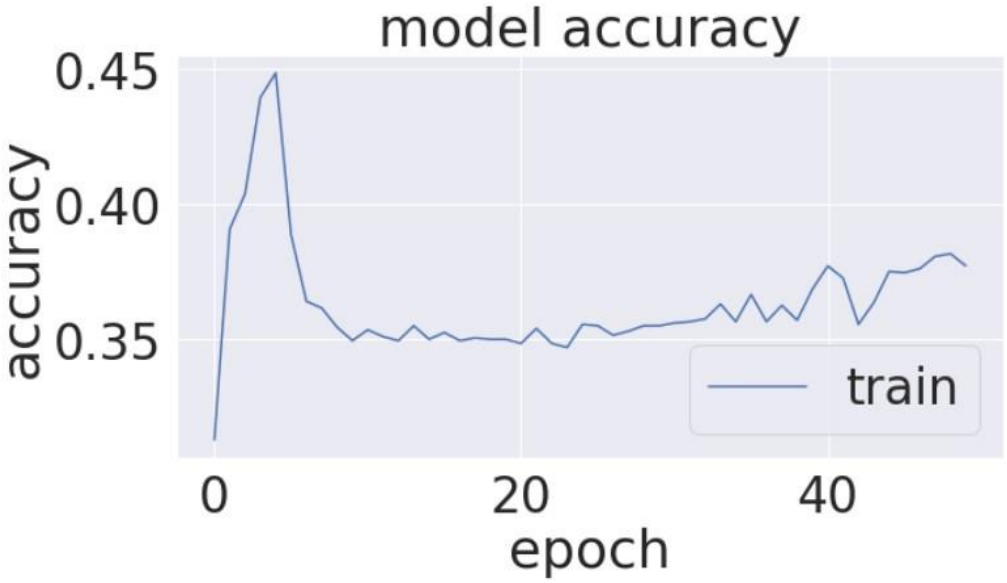
```
array([0, 2, 0, 2, 2, 2, 2, 0, 2, 0, 2, 2, 0, 0, 0, 2, 2, 2, 0, 0, 2, 0,
       0, 2, 0, 0, 2, 2, 0, 0, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 0, 2,
       0, 2, 0, 0, 0, 2, 0, 2, 2, 0, 2, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 2,
       0, 2, 2, 0, 2, 0, 2, 2, 2, 0, 2, 0, 0, 2, 2, 2, 0, 0, 2, 0, 0, 2,
       2, 0, 2, 2, 0, 2, 0, 2, 2, 2, 0, 2, 0, 0, 0, 2, 2, 0, 0, 2, 2, 2,
       2, 2, 2, 0, 2, 0, 0, 2, 0, 2, 0, 0, 2, 2, 0, 0, 0, 0, 2, 0, 0, 2,
       2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 0,
       2, 2, 0, 2, 0, 0, 2, 2, 0, 2, 2, 0, 2, 2, 2, 0, 0, 2, 0, 2, 0, 2,
       2, 0, 2, 2, 2, 2, 0, 2, 0, 0, 0, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 0,
       0, 0, 2, 0, 2, 0, 2, 2, 0, 2, 2, 0, 2, 0, 0, 0, 2, 2, 0, 0, 2, 2,
       2, 0, 2, 2, 0, 0, 2])
```

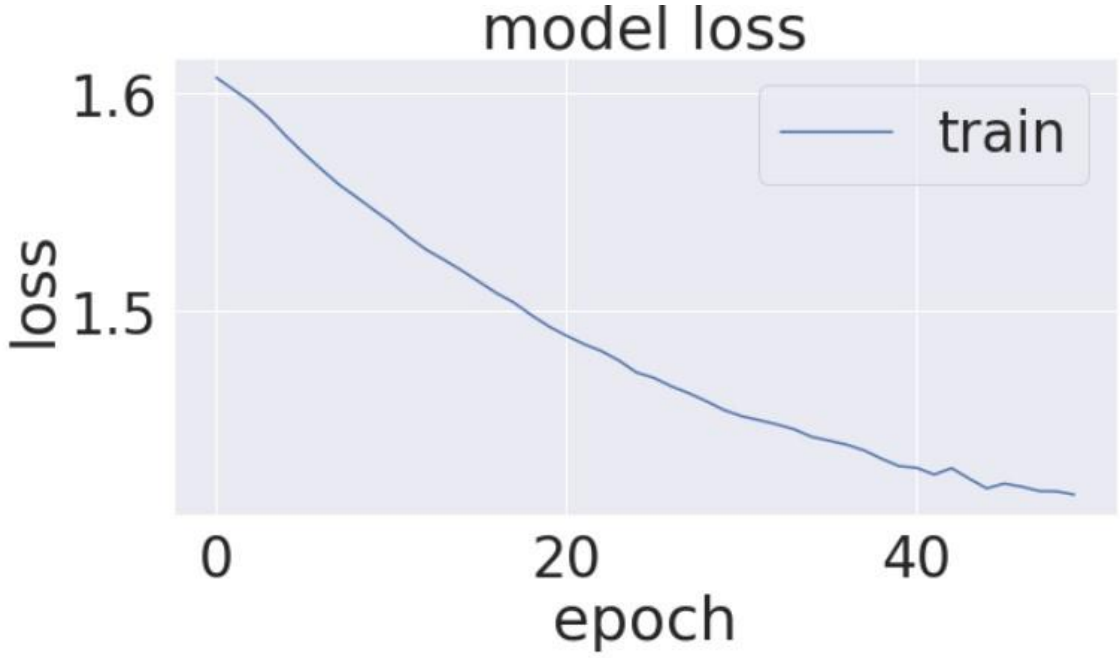
همه داده ها برای کلاس شماره ۰ و ۲ پیش بینی شده اند که به ترتیب همان کلاس های A و N می باشند. با بررسی تعداد اعضای هر کلاس در داده های train مشاهده می شود که تعداد داده ها در کلاس های A و N بسیار بیشتر از تعداد داده ها در کلاس های دیگر (H,S,W) می باشد و همین امر موجب می شود که مدل ما در زمان یادگیری از این کلاس دو کلاس بیشتر تاثیر می پذیرد لذا در برای prediction نیز داده های test را برای این دو کلاس پیش بینی کرده است.

نمودار confusion matrix آن در زیر نمایش داده شده است:

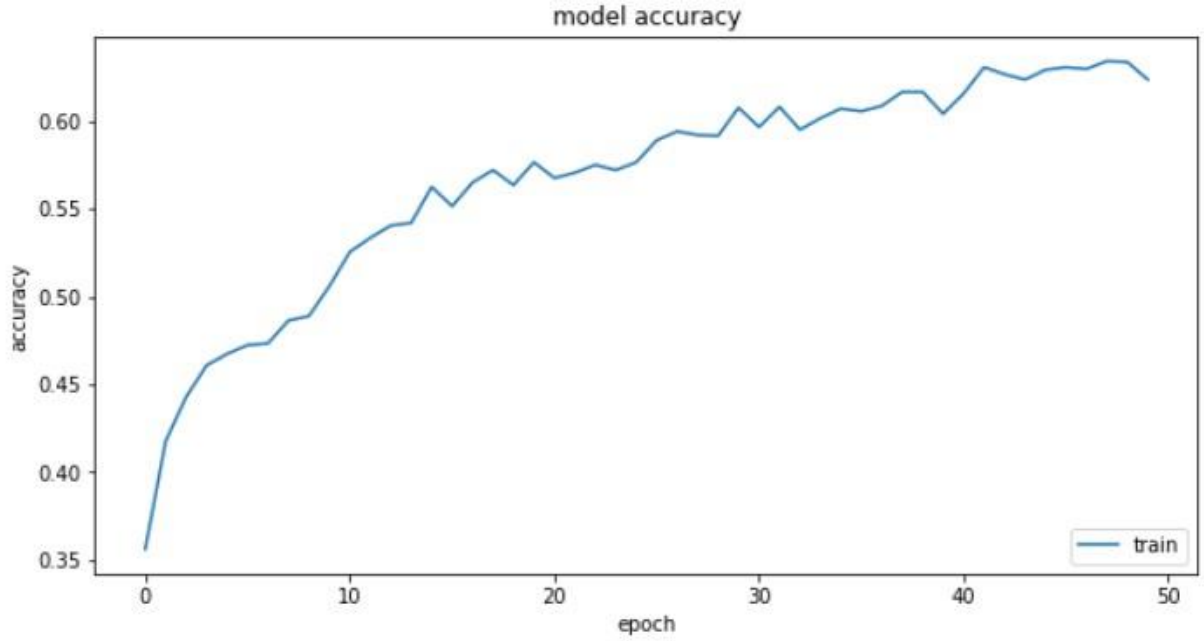


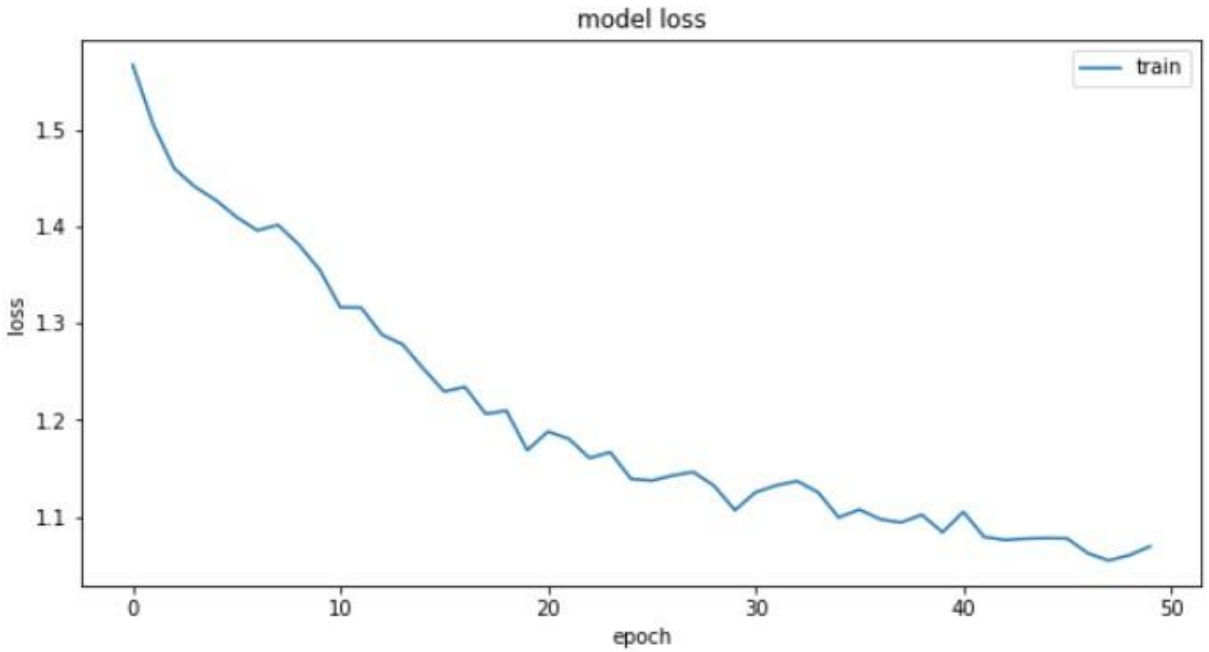
Feature های MFCC وCENS را با یکدیگر ادغام می کنیم و به همان مدل می دهیم و نتیجه بسیار بد تر از حالتی شد که فقط با feature های بدست آمده از MFCC کار کردیم.



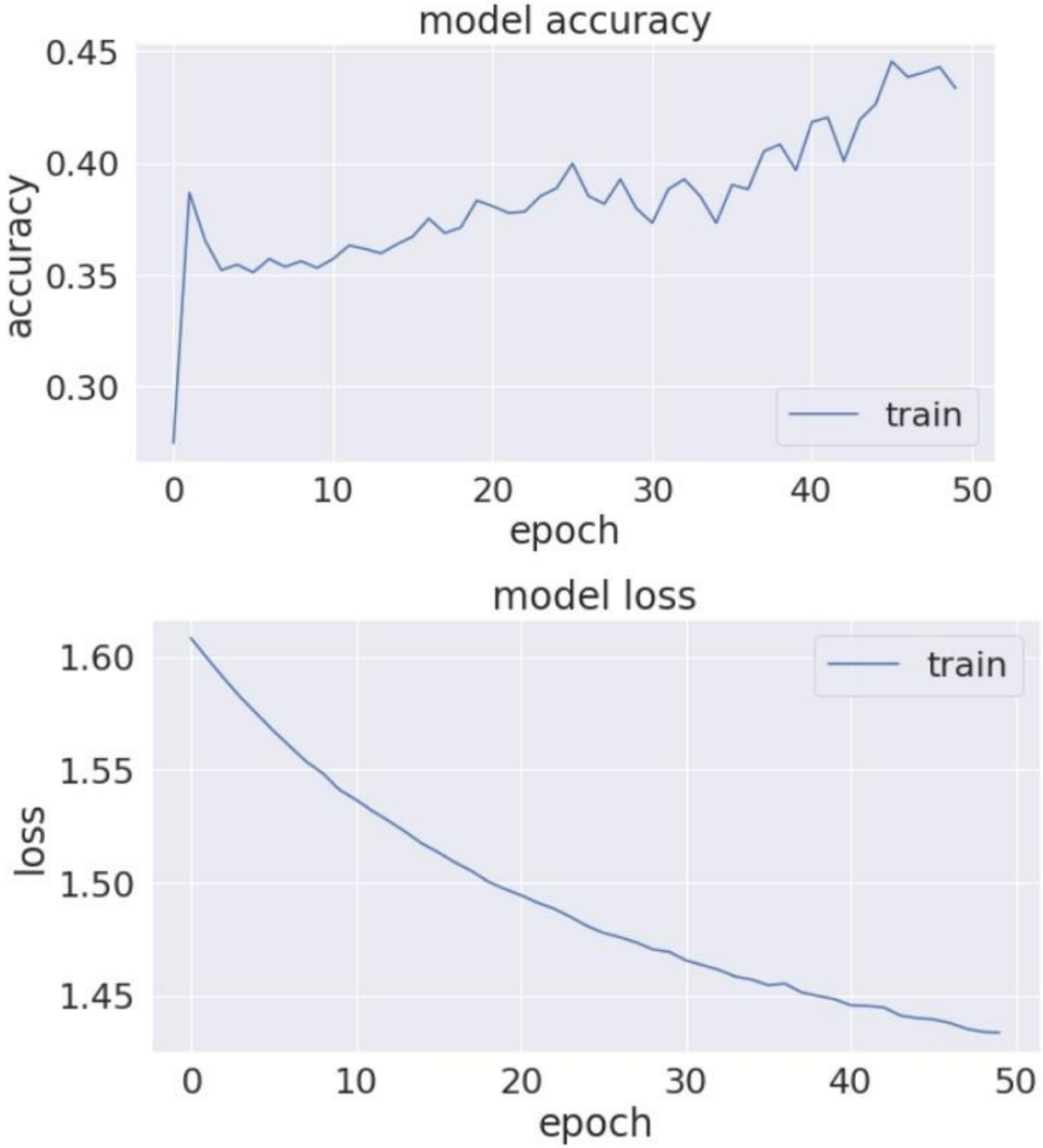


حال optimizer را از SGD به Adam تغییر می دهیم و مشاهده می شود که نتایج به طور چشم گیری بهبود می یابد. در زیر نمودار accuracy و loss function را در حالتی که فقط از feature های MFCC استفاده شده مشاهده می شود.





Accuracy و loss function را در حالتی که از ادغام feature های MFCC و CENS استفاده کردیم در زیر مشاهده می شود.

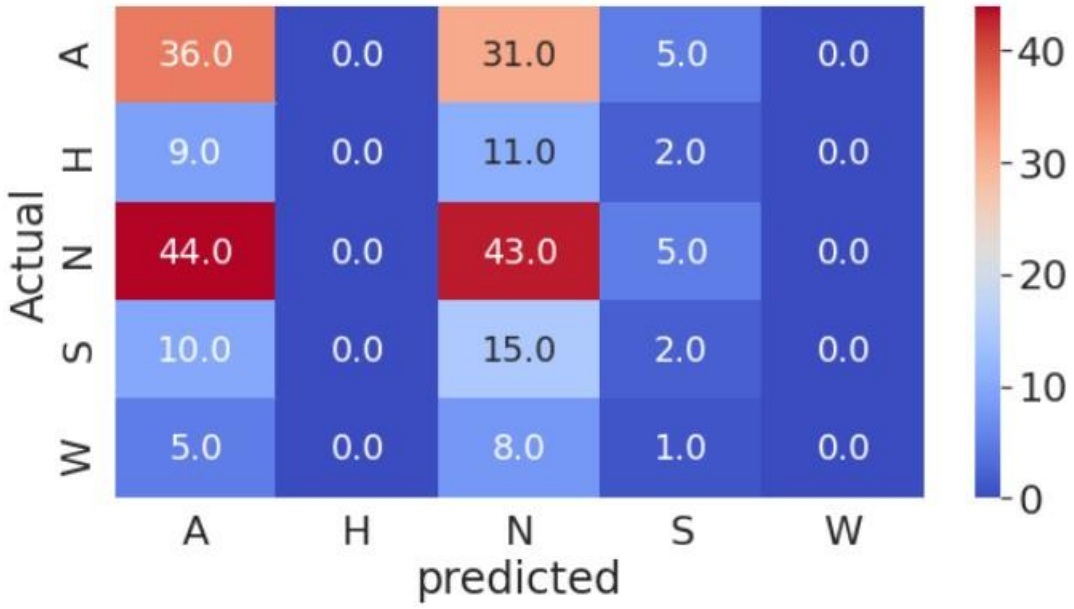


پیش بینی بر روی داده های تست را با مدلی که فقط با کمک feature های mfcc اجرا شده، انجام می دهیم و نتیجه را در زیر مشاهده می کنیم.

```
predicted = np.argmax(predicted, axis=1)
predicted
```

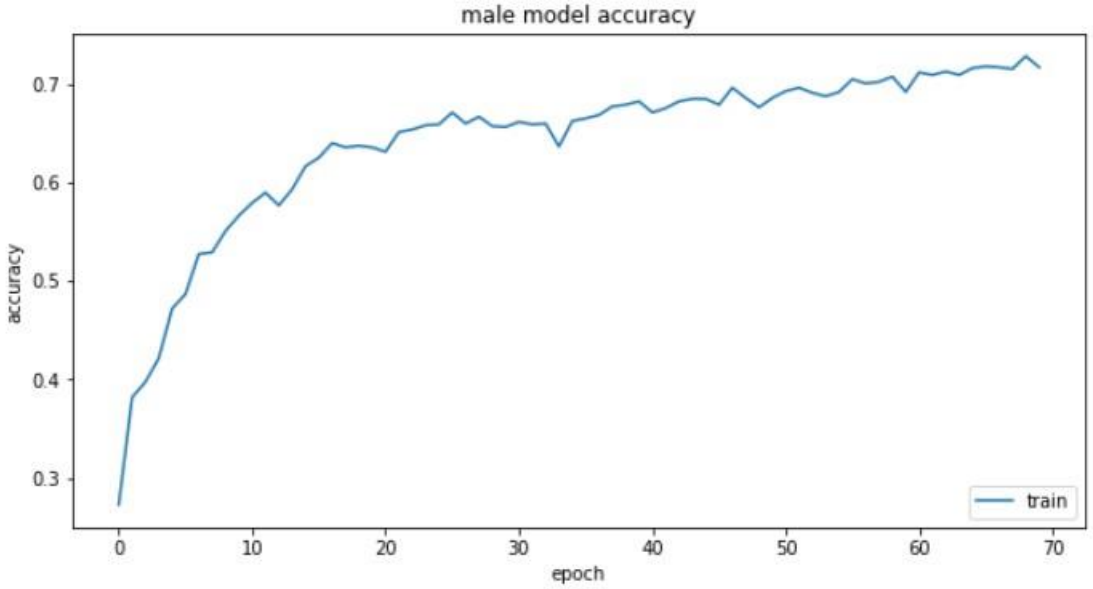
```
array([2, 0, 0, 2, 2, 2, 2, 0, 2, 3, 3, 0, 0, 2, 2, 3, 2, 2, 2, 2, 0, 2,
       2, 2, 2, 0, 2, 0, 0, 0, 0, 2, 2, 0, 0, 2, 2, 2, 0, 2, 0, 2, 0, 0,
       0, 2, 2, 2, 2, 0, 2, 2, 3, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 0,
       2, 2, 2, 0, 2, 2, 2, 0, 0, 2, 2, 2, 0, 0, 0, 3, 3, 0, 3, 0, 0, 2,
       2, 0, 2, 0, 2, 2, 2, 0, 3, 2, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 2, 0,
       3, 0, 0, 2, 2, 3, 0, 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 2, 0,
       0, 0, 0, 0, 2, 0, 2, 2, 0, 3, 2, 2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 0,
       3, 0, 0, 2, 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 2, 0, 0, 0, 2, 0,
       0, 2, 2, 2, 0, 2, 0, 2, 0, 0, 3, 2, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0,
       0, 0, 0, 0, 2, 0, 3, 2, 0, 2, 0, 0, 2, 0, 0, 0, 2, 2, 2, 0, 0, 0,
       0, 2, 2, 2, 2, 2, 3])
```

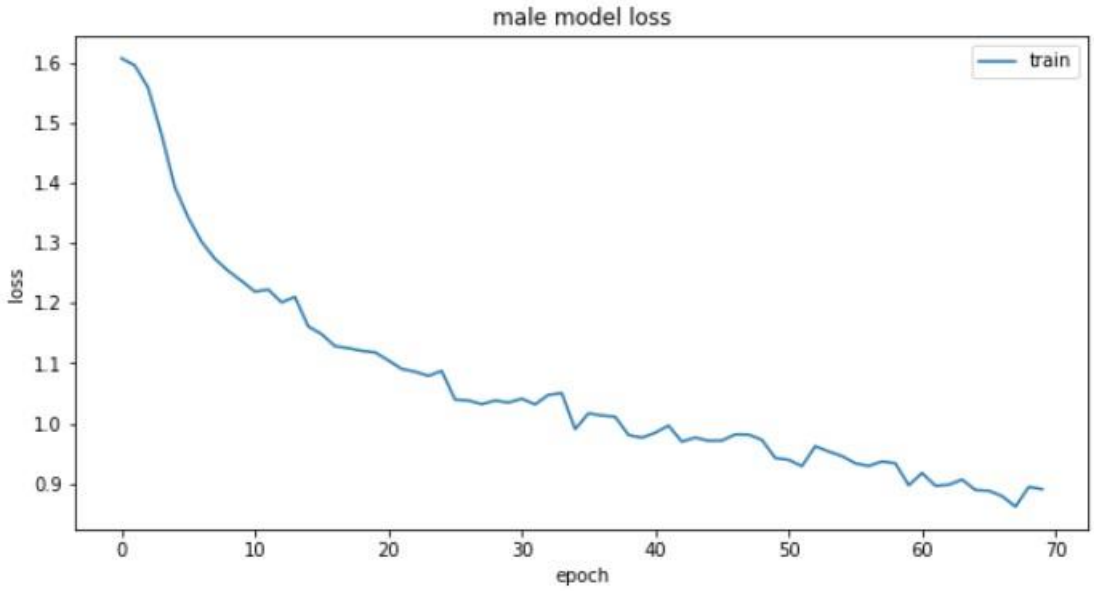
مشاهده می شود که تعدادی از داده ها را متعلق به کلاس شماره ۳ پیش بینی کرده و نسبت به حالتی که باSGD انجام داده بودیم، نتیجه بهتر شده است. تعداد داده های کلاس شماره ۰ که درست پیش بینی شده اند نیز نسبت به حالت قبل افزایش یافته ولی در مورد کلاس شماره ۲ ، خلاف این اتفاق رخ داده و تعداد کم شده است.



حال برای بررسی تاثیر جنسیت، داده های train و test را بر اسا جنسیت تفکیک می کنیم و هر کدام را با مدل مخصوص به خود اجرا می کنیم. تنظیمات مدل بجز اینکه در آن از Adam استفاده شده است در بقیه موارد مانند مدل اولیه می باشد.

دقت مدلی که با داده های train با جنسیت مرد learn شده به مقدار ۷۰٪ رسیده.



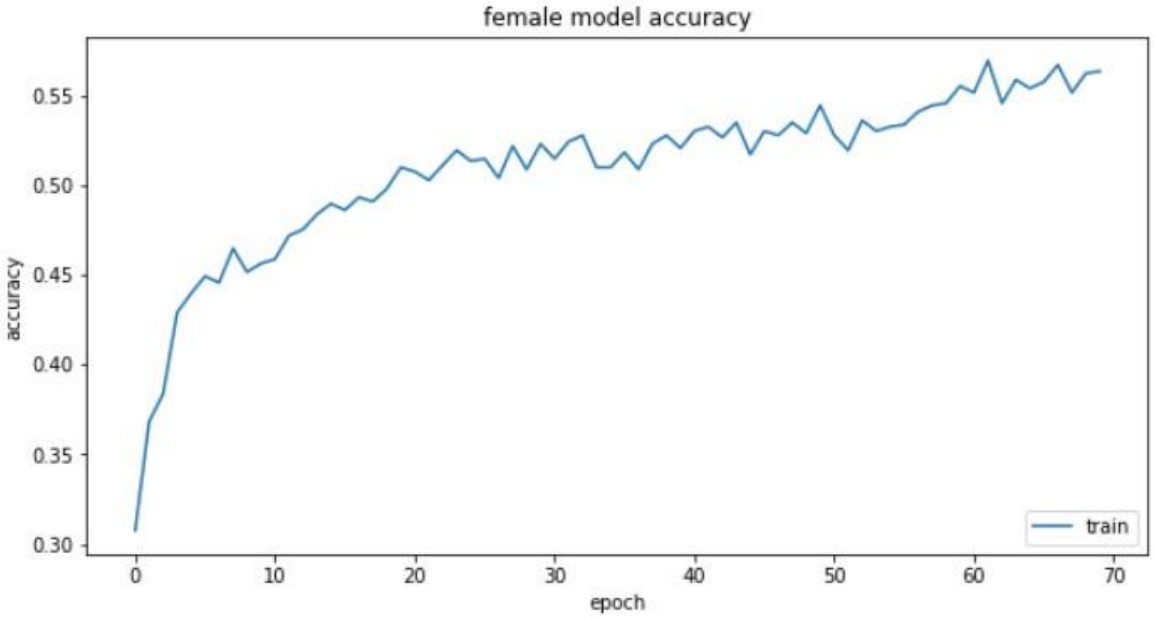


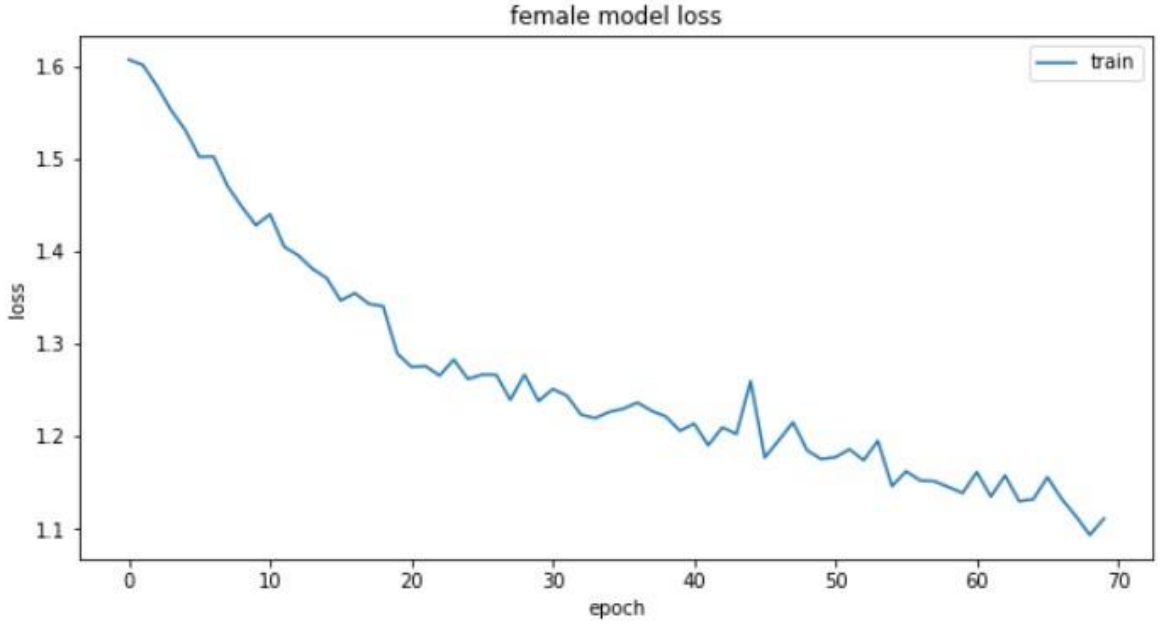
نتیجه پیش بینی داده های test با جنسیت مرد بر روی این مدل به صورت زیر می باشد:

```
male_predicted
```

```
array([2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 0, 2, 2, 0, 2, 2, 0,
       2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 0, 0, 0, 3, 2, 2, 2, 0, 2, 2, 2, 0, 2, 0, 2, 0, 2, 3, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 0, 2, 2, 2, 2, 2, 3, 2, 0, 2,
       0, 0, 0, 2, 2, 0, 2, 2, 2, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 0, 0,
       2, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 2, 2, 0, 2, 2, 2, 0, 0,
       0, 2, 3, 0, 2, 0, 2, 2, 2, 2, 2, 2])
```

دقت مدلی که برای خانم ها استفاده شد نسبت به مدل آقایان کمتر است.





نتیجه پیش بینی مدل بر روی داده هایtest با جنسیت خانم نیز به صورت زیر است:

```
female_predicted = np.argmax(female_predicted, axis=1)
female_predicted
```

```
array([2, 0, 0, 3, 3, 2, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 0, 3, 2, 2, 2,
       0, 2, 0, 2, 2, 0, 2, 0, 0, 2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0,
       0, 2, 3, 3, 3, 2, 2, 0, 2, 3, 2, 2, 2, 0, 0, 0, 2, 0, 3, 0, 0, 3,
       0, 0, 0, 2, 2, 0, 2, 2, 0, 0, 3, 2, 0, 0, 0, 2, 0])
```