شبکه عصبی

تشخیص احساسات

پروژه شماره ۵ - علی صالح - ۹۷۲۲۲۰۵۳

رویکرد کلی

رویکرد کلی ما این است که ابتدا از صدا هایی که داریم فیچرهایی استخراج کنیم و سپس با این فیچر ها و یک شبکه RNN بتوانیم آنها را کلاس بندی کنیم.

Feature Extraction

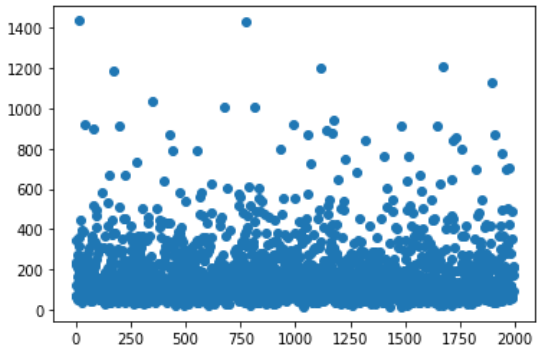
ما برای اینکه بتوانیم یک مدل شبکه عصبی ترین کنیم نیاز داریم دیتا هایی از صدا های ورودی داشته باشیم. برای همین نیاز داریم ویژگی های خاصی از هر صدا را استخراج کنیم. برای این عملیات feature extraction ما از روش mfcc استفاده می‌کنیم. و از هر صدا به اندازه ی ۱۰ ویژگی استخراج می‌کنیم. (مقدار های بیشتر از ۱۰ در دقت های نهایی مدل های ما تاثیر خاصی نداشتند و دقت را زیاد نکردند.)

Padding

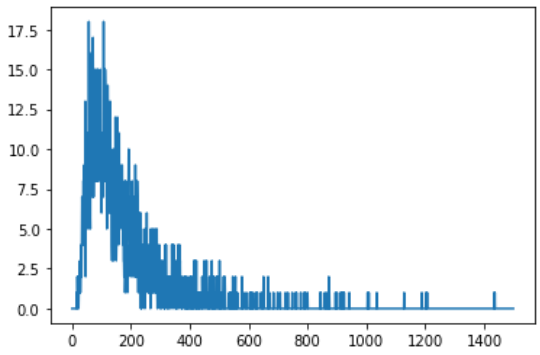
روش mfcc به ازای هر صدا به ما یک تنسور سه بعدی از ویژگی ها میدهد که بعد سوم آن مربوط به مدت زمان صدا استو و از آنجا که زمان صدا ها با هم متفاوت است پس ما تنسور هایی با ابعاد متفاوت داریم و باید آنها را به شکلی در ابعادشان یکسان کنیم.

اندازه ی بعد متفاوت این تنسور ها را بررسی می‌کنیم.

نمودار id-طول زمان



نمودار توزیع طول زمان



از نمودار توزیع مدت زمان هر صدا می‌فهمیم که از ۴۰۰ به بعد مقدار خیلی کمی از داده ها وجود دارند و حذف قسمتی از آن ها مشکل زا نیست.

فرض کنید عدد ۴۰۰ به معنی یک صدا به مدت ۱۵ثانیه است.

پس برای اینکه ابعاد همه‌ی تنسور ها را یکی کنیم از دیتا های با بعد زمانی بیش از ۱۵ ثانیه دارند فقط ۱۵ ثانیه اول را نگه می‌داریم و به دیتا های با بعد زمان کمتر از ۱۵ ثانیه مقداری صفر اضافه می‌کنیم که بعد زمان آنها هم ۴۰۰ شود.

چون ما به وسیله‌ی mfcc مقدار ۱۰ فیچر را استخراج کردیم الان همه‌ی دیتا های ما ابعاد ۱۰ در ۴۰۰ دارند.

Data Preprocess

دیتا های لیبل را چون کتگوریکال اند با one hot encoder انکود می‌کنیم. و دیتا های ورودی را به وسیله‌ی min max scaler اسکیل می‌کنیم. و در پایان دیتا های ولیدیشن را از x و y جدا می‌کنیم.

LSTM Model

می‌خواهیم هر واحد زمانی را به طور پیوسته به شبکه lstm دهیم. پس ما به ازای هر صدا ۴۰۰ دیتا ی پیوسته به شبکه می‌دهیم و بعد از این ۴۰۰ بار از شبکه یک خروجی کلاس بندی می‌گیریم.

اولین مدلی که استفاده می‌کنیم مدل ساده‌ی زیر است.

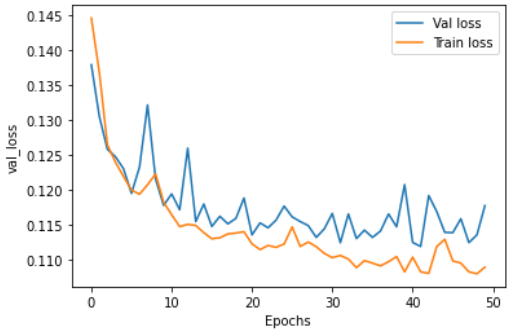
```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_22 (LSTM)               (None, 128)               71168
_________________________________________________________________
dense_42 (Dense)             (None, 5)                 645
=================================================================
```

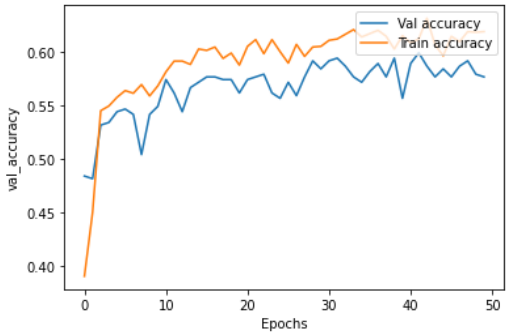
Loss function = mse optimizer = adam

Output activation = softmax batch size = 128

epoch = 50

نتایج:



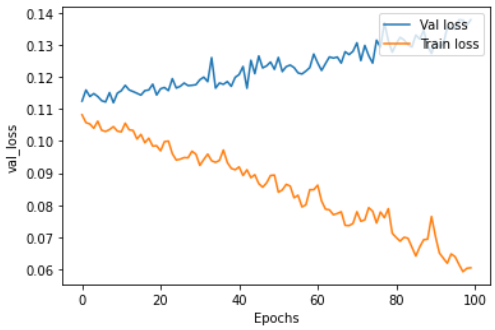


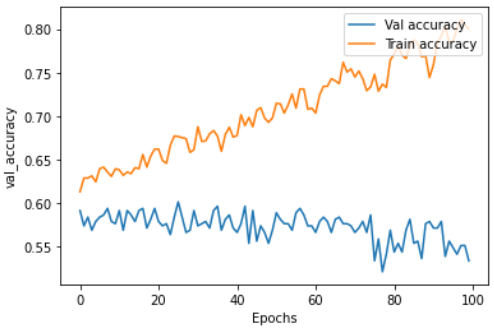
شبکه را بزرگتر می‌کنیم.

این بار مدل را ۱۰۰ epoch ترین می‌کنیم.

Layer (type)	Output Shape	Param #
lstm_25 (LSTM)	(None, 128)	71168
dense_51 (Dense)	(None, 64)	8256
dense_52 (Dense)	(None, 128)	8320
dense_53 (Dense)	(None, 32)	4128
dense_54 (Dense)	(None, 5)	165

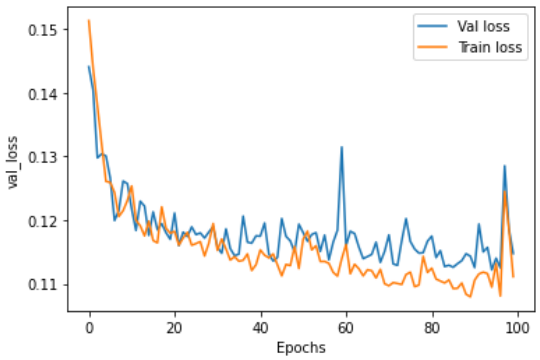
نتایج:

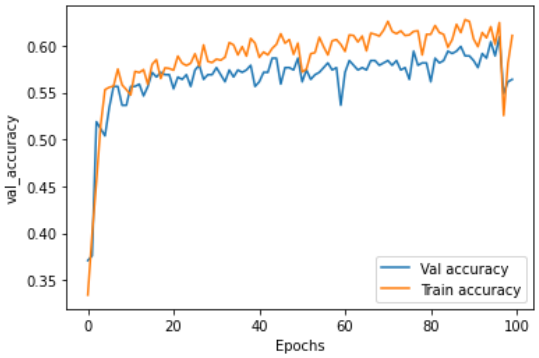




به وضوح مشخص است مدل ما اورفیت شده. پس Dropout به مدل اضافه می‌کنیم و به مدل زیر می‌رسیم.

Layer (type)	Output Shape	Param #
lstm_30 (LSTM)	(None, 128)	71168
dense_71 (Dense)	(None, 64)	8256
dropout_8 (Dropout)	(None, 64)	0
dense_72 (Dense)	(None, 128)	8320
dense_73 (Dense)	(None, 32)	4128
dense_74 (Dense)	(None, 5)	165



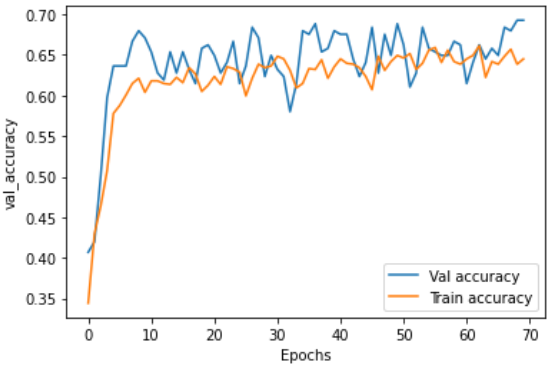


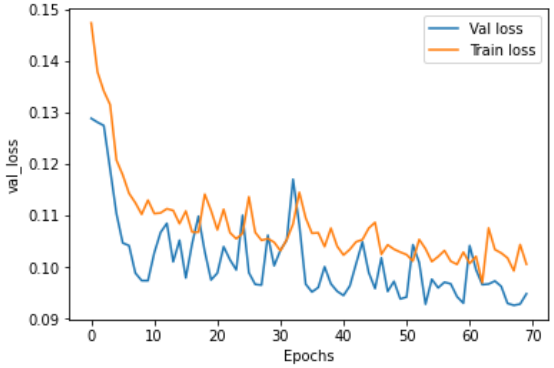
این مدل و دقت ۶۰ درصد بهترین دقتی بود که به آن رسیدیم.

تشخیص احساسات با تفکیک جنسیت

تا الان ما برای آموزش شبکه از همه‌ی دیتا ها استفاده می‌کردیم و حساسیتی روی جنسیت صدا نداشتیم. اما الان دیتا ها را بر اساس جنسیت جدا می‌کنیم و جدا جدا به شبکه می‌دهیم.

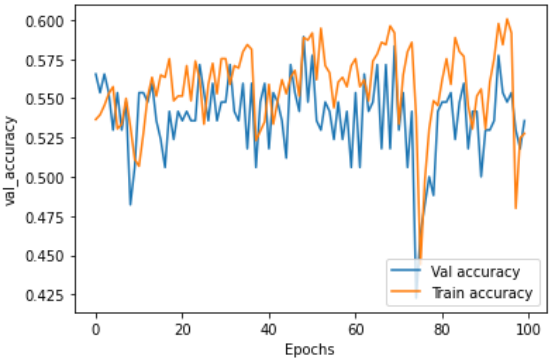
ابتدا صدای مرد ها را به شبکه می‌دهیم.

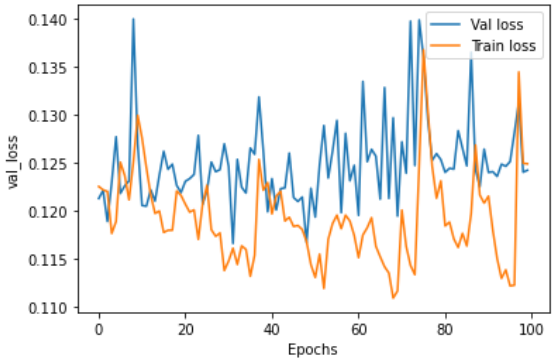




می‌بینیم که وقتی فقط صدای مرد ها را فقط به شبکه می‌دهیم پیشرفت می‌کند و دقتی نزدیک ۷۰ درصد به ما می دهد. (دیتا های تست هم همه صدا های مردها است).

اما برای زن ها متفاوت است و شبکه پیشرفتی نسبت به حالت کلی نمی کند و کمی هم پسرفت دارد.





البته تعداد کمتر دیتا های زن ها می‌تواند عامل این موضوع باشد.

Reference:

https://www.kaggle.com/seriousran/mfcc-feature-extraction-for-sound-classification

https://www.kaggle.com/mychen76/heart-sounds-analysis-and-classification-with-lstm

https://www.kaggle.com/ritzing/speech-emotion-recognition-with-cnn

https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/