



به نام خدا



دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر
شبکه های عصبی و یادگیری عمیق

مینی پروژه سری دوم

| | |
|--------------------|----------------------------------|
| نام و نام خانوادگی | محمدحسین بدیعی، سعید محمدی دشتکی |
| شماره دانشجویی | 810199106، 810199266 |
| تاریخ ارسال گزارش | 20 دی 1400 |

فهرست گزارش سوالات

- 3..... سوال اول
- 18..... سوال دوم – تشخیص نُت موسیقی
- 48..... سوال سوم

سوال اول

قسمت اول: Load کردن دیتا برای مجموعه داده گان Chopin

در این قسمت از سوال ما از دیتای Classical Music MIDI استفاده می کنیم و از آهنگ های Chopin استفاده می کنیم و شبکه خود را آموزش می دهیم. لذا برای این منظور کلاسی با نام MusicGeneration تعریف کردیم. برای Load کردن دیتا آهنگ های این هنرمند را از پوشه به نام آن با استفاده از کتابخانه music21 می خوانیم. و سپس note ها و chord های موجود در هر آهنگ را بدست می آوریم.

Note: نُت های موسیقی بلوک های سازنده موسیقی هستند. مربوط به گامی است که با یک لرزش صوتی خاص مرتبط است. موسیقی غربی از دوازده نت موسیقی بهره می برد.

در موسیقی، نُت نمادی است که یک صدای موسیقی را نشان می دهد. در کاربرد انگلیسی، نُت نیز خود صدا است. نت ها می توانند زیر و بم و مدت یک صدا را در نوتیشن موسیقی نشان دهند. یک نُت همچنین می تواند نشان دهنده یک کلاس pitch باشد.

نُت ها بلوک های سازنده بسیاری از موسیقی های نوشته شده هستند: گسسته سازی پدیده های موسیقی که اجرا، درک و تحلیل را تسهیل می کند. نُت های موسیقی بصورت زیر هستند:



شکل 1 نمونه ای از نام برخی نُت ها

(A = La; B = Si; C = Do; D = Re; E = Mi; F = Fa; G = Sol)

Chord: کورد، در موسیقی، به هر مجموعه هارمونیک از زیر و بم ها/فرکانس ها متشکل از نت های متعدد (که به آنها pitch نیز گفته می شود) گفته می شود که به طور هم زمان شنیده می شوند. نت های کورد به جای هم زمان، یکی پس از دیگری به صدا در می آیند، یا توالی tone های کورد نیز ممکن است به عنوان کورد در زمینه موسیقی مناسب در نظر گرفته شوند.

برای مثال در شکل زیر می توان در ردیف بالایی نت ها را مشاهده کرد و در ردیف پایینی کوردهای ساخته شده از نت های قطعه ای است که در قسمت بالایی نشان داده شده است:



شکل 2 ردیف بالایی نت ها و ردیف پایینی کوردهای ساخته شده از آن

قسمت دوم: پیش پردازش دیتا برای مجموعه داده گان Chopin

سپس با توجه به تعاریف گفته شده لیست corpus را بدست می آوریم. سپس تعداد کل نت های موجود در corpus را بدست می آوریم که در زیر می توانیم بینیم که در مجموعه دیتای chopin، برابر با 63429 است. و تعداد نت های منحصر بفرد (غیر تکراری) برابر با 317 است. و در نهایت 10 تا نت ابتدایی موجود در corpus را در زیر می بینیم:

```
Length of All Notes in The Midis : 63429
Length of All Unique Notes in The Corpus: 317
Some Samples of Notes in The Corpus (first 10 samples from 63429):
['F3', 'F2', 'G3', 'G2', 'B-3', 'B-2', 'C#4', 'C#3', 'E3', 'E2']
```

شکل 3 خروجی ترمینال شامل تعداد کل نت ها و نت های واحد و 10 تا نت اول Corpus

و سپس تعداد تکرار های هر نُت را می شماریم که این مقدار تکرار از 1 بار تا 1869 بار تکرار وجود دارد که در زیر با نُت مورد نظر نیز نمایش داده شده است. و میانگین تکرار هر نُت نیز برابر با 200 می باشد. در زیر همه موارد گفته شده را می توان از خروجی ترمینال مشاهده کرد:

Average Frequencies of Each Notes in The Corpus: 200.09148264984228

| | Note | Frequency |
|-----------------------------------|--------|-----------|
| Most Frequent Note in The Corpus | G#3 | 1869 |
| Least Frequent Note in The Corpus | 2.6.10 | 1 |

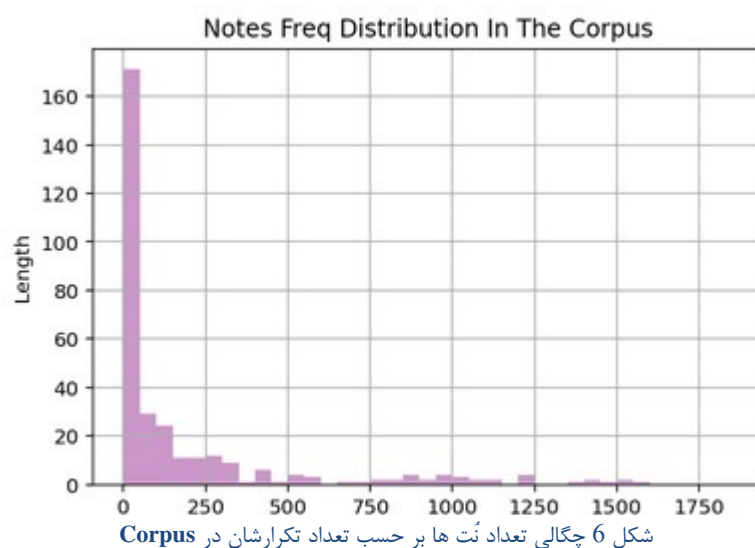
شکل 4 میانگین تعداد تکرار هر نُت و نُت با کمترین و بیشترین تکرار به همراه تعداد تکرار آنها

اگر نُت هایی که تعداد تکرارشان کمتر از 100 است را نُت های کم تکرار (rare) در نظر بگیریم، تعدادشان برابر با 200 است. سپس نُت های کم تکرار (rare) را از corpus حذف کنیم، تعداد کل نُت های corpus ما به 59853 می رسد. همانطور که از خروجی ترمینال میتوان بصورت زیر مشاهده کرد:

Number of Rare Notes: 200
Length of Corpus after Rare Notes Elemination: 59853

شکل 5 تعداد نُت های کم تکرار و طول Corpus پس از حذف کم تکرارها

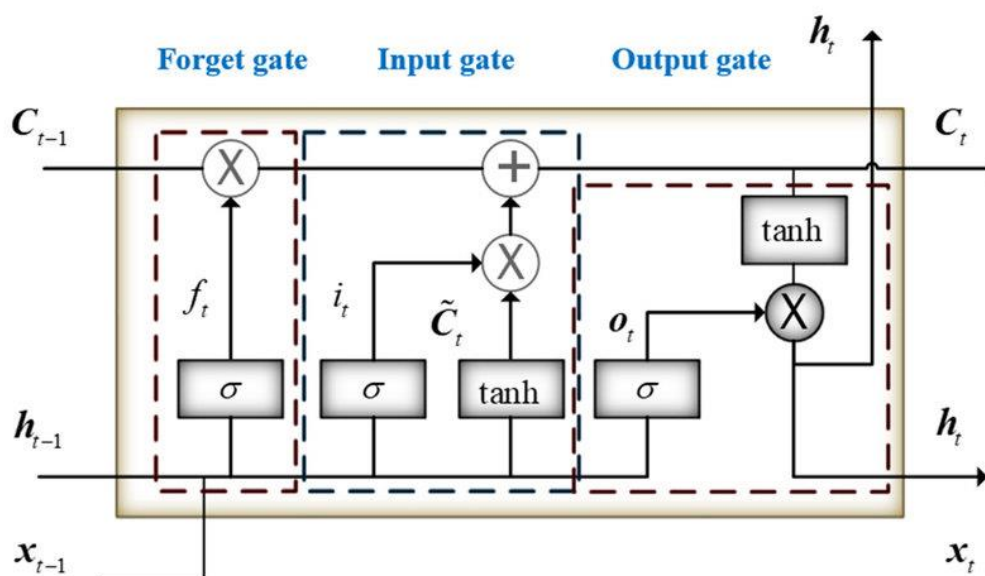
حال نمودار چگالی تعداد نُت ها را بر حسب تعداد تکرارشان بصورت زیر می توان دید:



حال از corpus حاصل شده در مرحله قبل استفاده می کنیم و نت ها را بصورت sequence های 40 تایی در می آوریم و نت 41 امی بع از sequence را به عنوان لیبل و دیتای تست در نظر می گیریم. و ایندکس 1 تا 40 به عنوان sequence بعدی و ایندکس 41 به عنوان لیبل این sequence در نظر می گیریم و سپس دیتای test و train را از هم جدا می کنیم که برای این جداسازی 20% داده ها به test اختصاص داده می شوند و بقیه به train اختصاص داده می شوند. و سپس شبکه عصبی بازگشتی خود را می سازیم.

قسمت سوم: پیاده سازی شبکه عصبی بازگشتی برای مجموعه داده گان Chopin

شبکه عصبی بازگشتی LSTM همان گونه که در کلاس درس نیز مطرح شد با در کنار هم قرار گرفتن چندین بلوک LSTM بصورت زیر حاصل می شود:



شکل 7 یک بلوک LSTM

که با توجه به شکل بالا که یک بلوک LSTM را نشان می دهد دارای ۳ دروازه (گیت) است به نام های دروازه ورودی (i_t) و خروجی (o_t) و دروازه فراموشی (f_t) که دروازه فراموشی تصمیم می گیرد که اطلاعاتی از حالت سلول قبلی را نگه دارد یا دور بریزد. معادلات ما بصورت زیر هستند:

$$\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) & x_t &\in \mathbb{R}^n \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) & W_{f,i,C,o} &\in \mathbb{R}^{h \times (h+n)} \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t & b_{f,i,C,o} &\in \mathbb{R}^{h \times 1} \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) & f_t, i_t, o_t, h_t, c_t &\in \mathbb{R}^h \\
h_t &= o_t * \tanh(C_t) \\
\hat{y}_t &= f(V \cdot h_t + b_v) & V &\in \mathbb{R}^{m \times h} \quad \hat{y}_t, y_t \in \mathbb{R}^m
\end{aligned}$$

Parameters: $\{(W_i, b_i), (W_f, b_f), (W_C, b_C), (W_o, b_o), (V, b_V)\}$

شبکه عصبی بازگشتی ما از دو لایه LSTM ساخته می شود، که مدل آن در زیر آورده شده است:

loss function: categorical_crossentropy

activation function: softmax

optimizer = Adamax(learning_rate = 0.01)

خروجی ترمینال که مدل ساخته شده با دستور summary را به نمایش می گذارد، خدمتتان ارائه می کنیم.

```

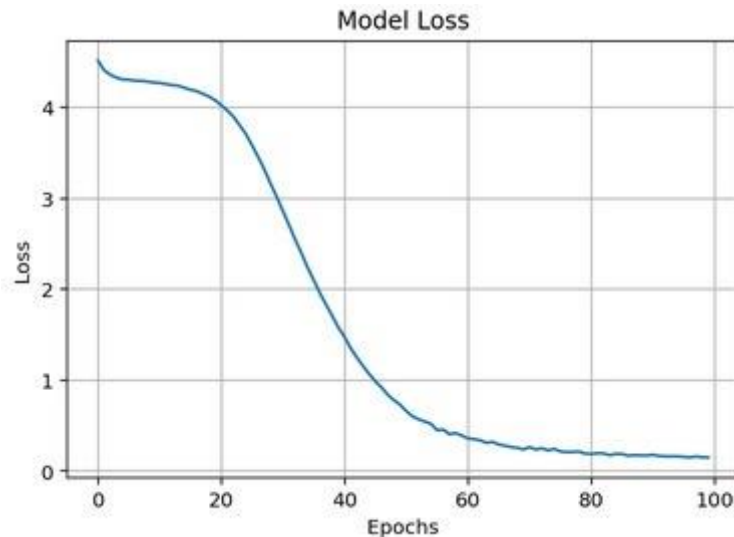
Model: "sequential_1"
_____
Layer (type)                 Output Shape              Param #
-----
lstm_2 (LSTM)                 (None, 40, 512)          1052672
lstm_3 (LSTM)                 (None, 256)              787456
dense_2 (Dense)               (None, 256)              65792
dense_3 (Dense)               (None, 228)              58596
=====
Total params: 1,964,516
Trainable params: 1,964,516
Non-trainable params: 0
_____

```

شکل 8 ساختار انتخابی شبکه عصبی بازگشتی

قسمت چهارم: ارزیابی شبکه برای مجموعه داده‌گان Chopin

مقدار loss را برای فرآیند بصورت زیر داریم:



شکل 9 نمودار Loss برای آهنگ های chopin

سپس با استفاده از تابع داده شده یک آهنگ به نام Chopin.midi تولید کرده که فرمت این فایل midi است و آنرا به فرمت wav. به نام Chopin.wav تبدیل کردیم که خروجی ها در پوشه Results آمده است.

سپس بعد از هر لایه LSTM و لایه های fully connected یک لایه dropout با مقدار 0.15 اضافه می کنیم و قسمت های سوم و چهارم را دوباره تکرار می کنیم:

قسمت سوم: پیاده سازی شبکه عصبی بازگشتی برای مجموعه داده‌گان Chopin با Dropout

برای فعال سازی dropout یک متغیر جدید برای کلاس با نام dropoutFlag تعریف می کنیم، این Flag دو مقدار مجاز را می تواند اختیار کند: مقدار اول True است و مقدار دوم False می باشد، به جهت اینکه لایه های dropout فعال شوند کافی است که مقدار این متغیر را برابر با True قرار دهید، برای غیرفعالسازی نیز مقدارش را False قرار می دهیم.

شبکه عصبی بازگشتی LSTM مان بصورت زیر می شود و نتایج این اضافه کردن dropout بصورت زیر حاصل می شوند:

loss function: categorical_crossentropy

activation function: softmax

optimizer = Adamax(learning_rate = 0.01)

خروجی ترمینال که مدل ساخته شده با دستور summary را به نمایش می گذارد، خدمتتان ارائه می کنیم.

```
Model: "sequential_3"
```

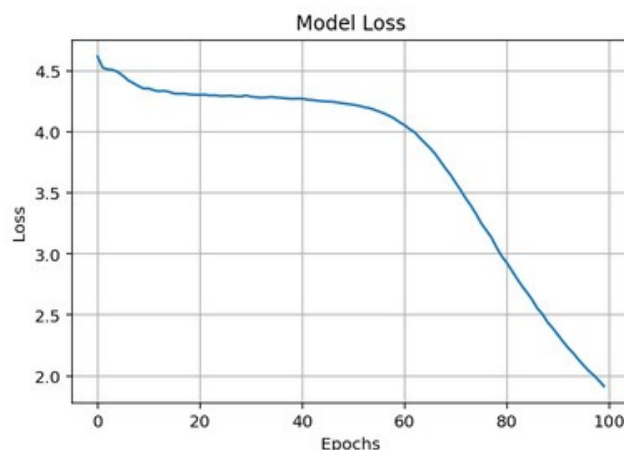
| Layer (type) | Output Shape | Param # |
|---------------------|-----------------|---------|
| lstm_6 (LSTM) | (None, 40, 512) | 1052672 |
| dropout_6 (Dropout) | (None, 40, 512) | 0 |
| lstm_7 (LSTM) | (None, 256) | 787456 |
| dropout_7 (Dropout) | (None, 256) | 0 |
| dense_6 (Dense) | (None, 256) | 65792 |
| dropout_8 (Dropout) | (None, 256) | 0 |
| dense_7 (Dense) | (None, 228) | 58596 |

```
=====  
Total params: 1,964,516  
Trainable params: 1,964,516  
Non-trainable params: 0  
=====
```

شکل 10 ساختار انتخابی شبکه عصبی بازگشتی با اضافه کردن لایه های dropout پس از هر لایه

قسمت چهارم: ارزیابی شبکه برای مجموعه داده‌گان Chopin به همراه Dropout

مقدار loss را برای فرآیند بصورت زیر داریم:



شکل 11 نمودار Loss برای آهنگ های Dropout پس از اضافه کردن لایه های dropout به همه لایه ها

با استفاده از تابع داده شده یک آهنگ به نام Chopin_with_dropout.midi تولید کرده که فرمت این فایل midi است و آنرا به فرمت wav. به نام Chopin_with_dropout.wav تبدیل کردیم که خروجی ها در پوشه Results آمده است.

همین مراحل قبل را برای موسیقی هایی که آهنگ سازی آن Mozart است نیز به همین ترتیب انجام می دهیم:

قسمت اول: Load کردن دیتا برای مجموعه داده‌گان Mozart

در این قسمت از سوال ما از دیتای Classical Music MIDI استفاده می کنیم و از آهنگ های Mozart استفاده می کنیم و شبکه خود را آموزش می دهیم.

برای Load کردن دیتا آهنگ های این هنرمند را از پوشه به نام آن با استفاده از کتابخانه music21 خوانیم. و سپس note ها و chord های موجود در هر آهنگ را بدست می آوریم.

قسمت دوم: پیش پردازش دیتا برای مجموعه داده‌گان Mozart

سپس با توجه به تعاریف گفته شده لیست corpus را بدست می آوریم. سپس تعداد کل نُت های موجود در corpus را بدست می آوریم که در زیر می توانیم بینیم که در مجموعه دیتای mozart، برابر با 59618 است. و تعداد نُت های منحصر بفرد (غیر تکراری) برابر با 197 است.

و در نهایت 10 تا نُت ابتدایی موجود در corpus را در زیر می بینیم:

```
Length of All Notes in The Midis : 59618

Length of All Unique Notes in The Corpus: 197

Some Samples of Notes in The Corpus (first 10 samples from 59618):

['C#5', 'A3', 'E4', 'D5', 'B3', 'C#5', '4.9', 'E5', '1.4', 'E5']
```

شکل 12 خروجی ترمینال شامل تعداد کل نُت ها و نُت های واحد و 10 تا نُت اول Corpus

و سپس تعداد تکرار های هر نُت را می شماریم که این مقدار تکرار از 1 بار تا 3116 بار تکرار وجود دارد که در زیر با نُت مورد نظر نیز نمایش داده شده است. و میانگین تکرار هر نُت نیز برابر با 302.62 می باشد. در زیر همه موارد گفته شده را می توان از خروجی ترمینال مشاهده کرد:

```
Average Frequencies of Each Notes in The Corpus: 302.6294416243655

+-----+-----+-----+
|           | Note | Frequency |
+-----+-----+-----+
| Most Frequent Note in The Corpus | D5 | 3116 |
| Least Frequent Note in The Corpus | 10.1.4 | 1 |
+-----+-----+-----+
```

شکل 13 میانگین تعداد تکرار هر نُت و نُت با کمترین و بیشترین تکرار به همراه تعداد تکرار آنها

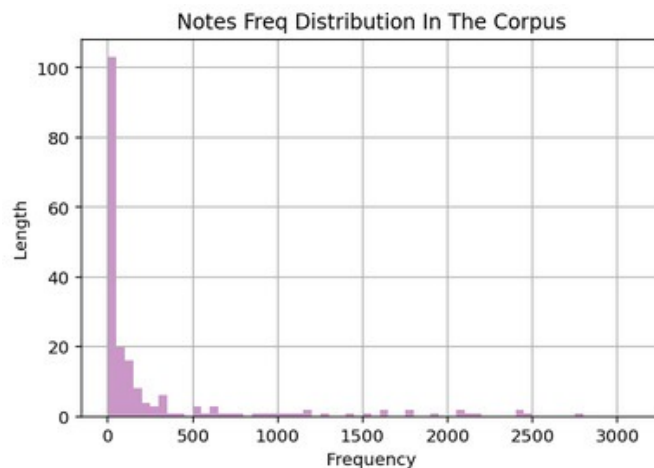
اگر نُت هایی که تعداد تکرارشان کمتر از 100 است را نُت های کم تکرار (rare) در نظر بگیریم، تعدادشان برابر با 123 است. سپس نُت های کم تکرار (rare) را از corpus حذف کنیم، تعداد کل نُت های corpus ما به 57248 می رسد. همانطور که از خروجی ترمینال میتوان بصورت زیر مشاهده کرد:

```
Number of Rare Notes: 123

Length of Corpus after Rare Notes Elimination: 57248
```

شکل 14 تعداد نُت های کم تکرار و طول Corpus پس از حذف کم تکرارها

حال نمودار چگالی تعداد نُت ها را بر حسب تعداد تکرارشان بصورت زیر می توان دید:



شکل 15 چگالی تعداد نُت ها بر حسب تعداد تکرارشان در Corpus

حال از corpus حاصل شده در مرحله قبل استفاده می کنیم و نُت ها را بصورت sequence های 40 تایی در می آوریم و نُت 41 امی بعد از sequence را به عنوان لیبل و دیتای تست در نظر می گیریم. و ایندکس 1 تا 40 به عنوان sequence بعدی و ایندکس 41 به عنوان لیبل این sequence در نظر می گیریم و ادامه می دهیم

سپس دیتای test و train را از هم جدا می کنیم که برای این جداسازی 20% داده ها به test اختصاص داده می شوند و بقیه به train اختصاص داده می شوند. و سپس شبکه عصبی بازگشتی خود را می سازیم.

قسمت سوم: پیاده سازی شبکه عصبی بازگشتی برای مجموعه داده گان Mozart

حال شبکه عصبی بازگشتی ما از دو لایه LSTM ساخته می شود، که مدل آن در زیر آورده شده است:

loss function: categorical_crossentropy

activation function: softmax

optimizer = Adamax(learning_rate = 0.01)

```
Model: "sequential_8"
```

| Layer (type) | Output Shape | Param # |
|------------------|-----------------|---------|
| lstm_16 (LSTM) | (None, 40, 512) | 1052672 |
| lstm_17 (LSTM) | (None, 256) | 787456 |
| dense_16 (Dense) | (None, 256) | 65792 |
| dense_17 (Dense) | (None, 99) | 25443 |

```

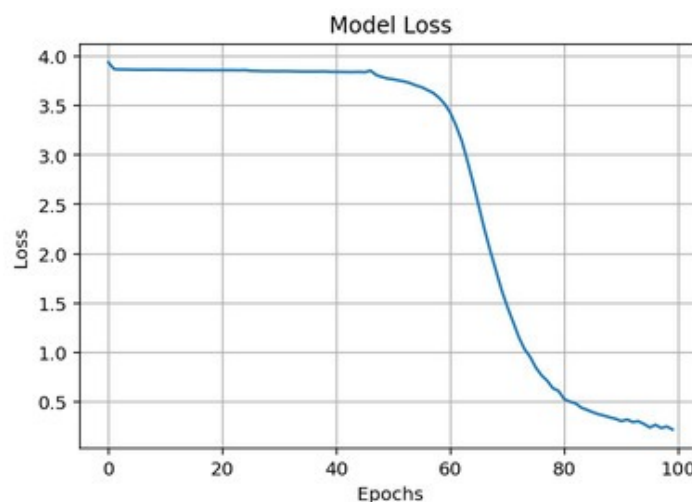
=====
Total params: 1,931,363
Trainable params: 1,931,363
Non-trainable params: 0
=====

```

شکل 16 ساختار انتخابی شبکه عصبی بازگشتی

قسمت چهارم: ارزیابی شبکه برای مجموعه داده‌گان Mozart

مقدار loss را برای فرآیند بصورت زیر داریم:



شکل 17 نمودار Loss برای آهنگ های mozart

با استفاده از تابع داده شده یک آهنگ به نام Mozart.midi تولید کرده که فرمت این فایل midi است و آنرا به فرمت wav. به نام Mozart.wav تبدیل کردیم که خروجی ها در پوشه Results آمده است.

سپس بعد از هر لایه LSTM و لایه های fully connected یک لایه dropout با مقدار 0.15 اضافه می کنیم و قسمت های سوم و چهارم را دوباره تکرار می کنیم:

قسمت سوم: پیاده سازی شبکه عصبی بازگشتی برای مجموعه داده گان Mozart به همراه

Dropout

شبکه عصبی بازگشتی LSTM مان بصورت زیر می شود و نتایج این اضافه کردن dropout بصورت زیر حاصل می شوند:

loss function: categorical_crossentropy

activation function: softmax

optimizer = Adamax(learning_rate = 0.01)

```
Model: "sequential_9"
```

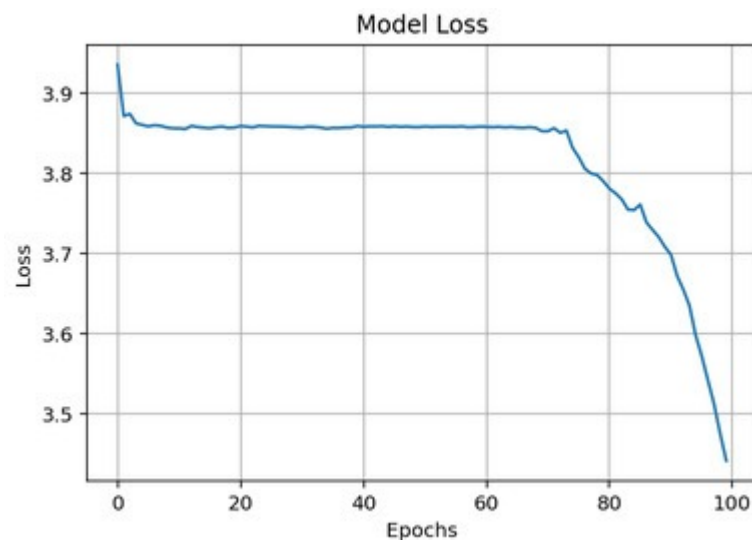
| Layer (type) | Output Shape | Param # |
|----------------------|-----------------|---------|
| lstm_18 (LSTM) | (None, 40, 512) | 1052672 |
| dropout_24 (Dropout) | (None, 40, 512) | 0 |
| lstm_19 (LSTM) | (None, 256) | 787456 |
| dropout_25 (Dropout) | (None, 256) | 0 |
| dense_18 (Dense) | (None, 256) | 65792 |
| dropout_26 (Dropout) | (None, 256) | 0 |
| dense_19 (Dense) | (None, 99) | 25443 |

Total params: 1,931,363
Trainable params: 1,931,363
Non-trainable params: 0

شکل 18 ساختار انتخابی شبکه عصبی بازگشتی با اضافه کردن لایه های dropout پس از هر لایه

قسمت چهارم: ارزیابی شبکه برای مجموعه داده گان Mozart به همراه Dropout

مقدار loss را برای فرآیند بصورت زیر داریم:



شکل 19 نمودار Loss برای آهنگ های mozart پس از اضافه کردن لایخ dropout پس از هر لایه

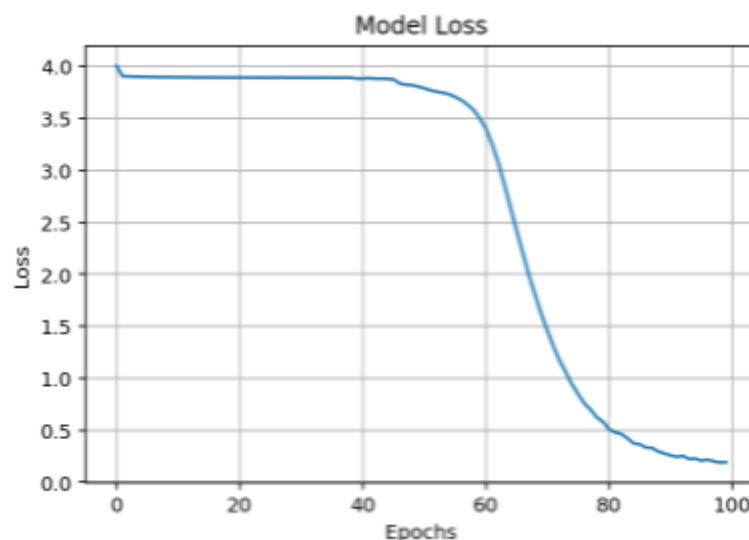
و با استفاده از تابع داده شده یک آهنگ به نام Mozart_with_dropout.midi تولید کرده که فرمت این فایل midi است و آنرا به فرمت wav. به نام Mozart_with_dropout.wav تبدیل کردیم که خروجی ها در پوشه Results آمده است.

سپس یک بار دیگر شبکه را آموزش می دهیم که ما dropout را فقط پس از لایه اول به میزان 0.05 اعمال می کنیم که برای Mozart مدل شبکه ما بصورت زیر می شود:

| Layer (type) | Output Shape | Param # |
|-----------------------------|-----------------|---------|
| lstm_2 (LSTM) | (None, 40, 512) | 1052672 |
| dropout_3 (Dropout) | (None, 40, 512) | 0 |
| lstm_3 (LSTM) | (None, 256) | 787456 |
| dense_2 (Dense) | (None, 256) | 65792 |
| dense_3 (Dense) | (None, 140) | 35980 |
| Total params: 1,941,900 | | |
| Trainable params: 1,941,900 | | |
| Non-trainable params: 0 | | |

شکل 20 ساختار انتخابی شبکه عصبی بازگشتی با اضافه کردن یک لایه **dropout** پس از لایه اول برای آهنگ های **mozart**

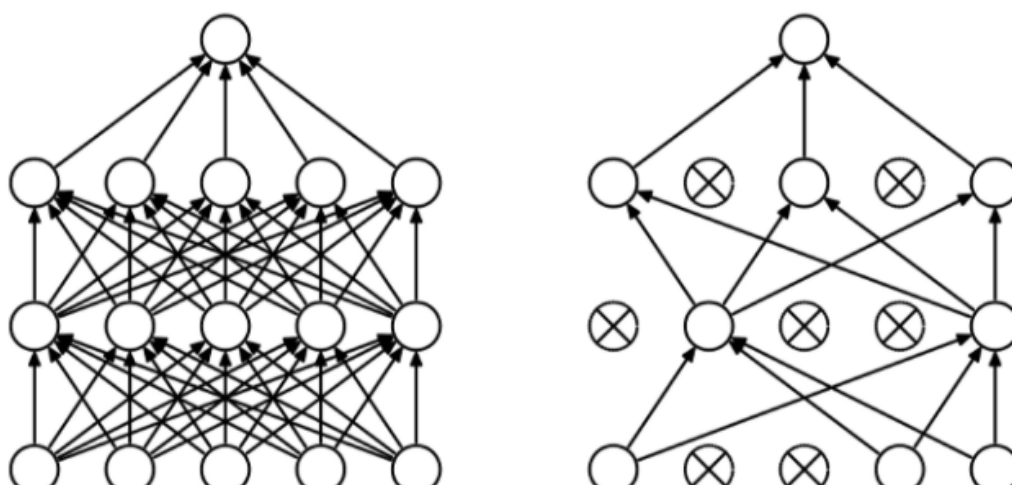
و خروجی Loss برای این شبکه به صورت زیر می شود:



شکل 21 نمودار Loss با اضافه کردن یک لایه **dropout** پس از لایه اول برای آهنگ های **mozart**

دلیل این کار این بود که مشاهده کنیم که قرارگیری dropout پس از تمامی لایه ها می تواند کارایی کمتری نسبت به قرارگیری تنها یک dropout بعد لایه ی مناسب در شبکه باشد که در نتایج فوق مشهود است.

کاربرد های Dropout: یک تکنیک است که جلوی «over-fit» شدن شبکه را می گیرد. یعنی یک راه ساده برای جلوگیری از اتصال بیش از حد در شبکه های عصبی استفاده از dropout است این روش روشی



شکل 22 یک مثال از اضافه کردن **Dropout** به شبکه

برای منظم سازی است و خطای تعمیم پذیری را با کاهش گنجایش مدل کمتر می کند. بدین ترتیب در هر دور آموزشی، به جای استفاده از همه نورون ها، تنها برخی از نورون ها (با احتمال p) فعال می شوند.

منظم سازی (regularization): مقدار خطایی که شبکه های عصبی روی داده های آموزشی بدست می آورند بسیار کمتر از خطای آنها روی داده های آزمایشی است. به همین دلیل در معرض بیش برآزش قرار می گیرند. برای مقابله با این مساله، از تکنیک های منظم سازی استفاده می شود که گنجایش مدل را کاهش دهیم. بدین ترتیب تفاوت بین خطای آموزش و آزمایش کاهش یافته و مدل آموزش و آزمایش، عملکرد مشابهی از خود نشان خواهد داد.

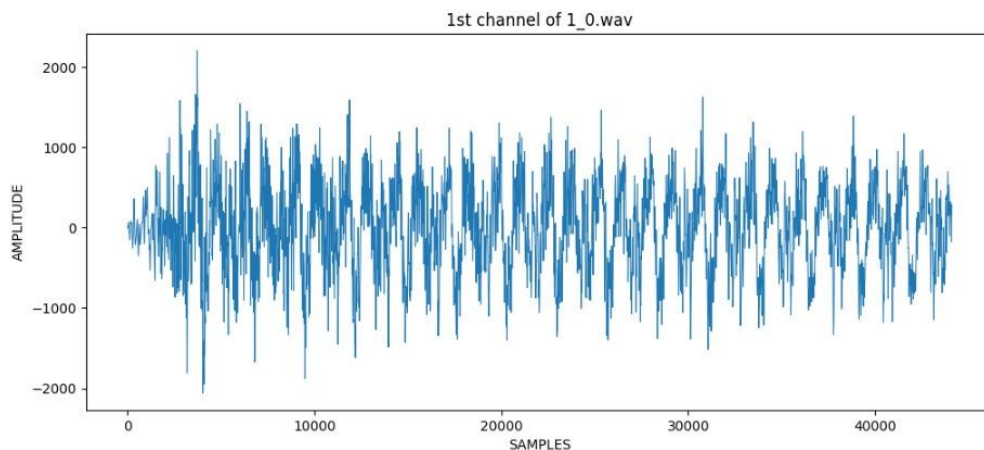
تاثیر لایه dropout بر روی عملکرد شبکه ما: همان طور که در خروجی های شبکه ما مشاهده شده است با اضافه نمودن dropout با مقدار 0.05 و برای یک لایه کاملاً مشهود است که از نظر خطا (Loss) مقدارش کم تر شده و می بینیم که واقعا این dropout تاثیر گذار بوده است و به اندازه قابل توجهی مقدار خطا کاهش می یابد که این یک تاثیر مثبت بر شبکه تلقی می شود. اما همانطور که از نتایج ملاحظه کردیم اگر با توجه به خواسته صورت سوال، ما در این مساله پس از هر لایه یک لایه dropout اضافه کنیم می بینیم که نتایج به این صورت است که به نظر می رسد که کلیت ساختار شبکه را تحت الشعاع قرار می دهد و شبکه از نظر خطا (Loss) مقدارش کم نشده و یا بهبود پیدا نمی کند و به نظر می رسد که بخشی از شبکه از بین می رود که نتیجه مثبت نمی دهد.

پس می توان نتیجه گرفت که مقدار dropout را نباید خیلی بزرگ و نباید خیلی کوچک در نظر گرفت که اگر خیلی بزرگ باشد، بخش عمده ای از شبکه از دست خواهد رفت و همچنین نیازی نیست که بر روی همه ی لایه های شبکه اضافه شود که این امر باعث می شود که باز هم بخش عمده و مهمی از شبکه از دست خواهد رفت و اینکه کلیت ساختار را با زیاد نمودن dropout از بین ببریم طبیعتاً کار درستی نیست لذا برای ساختار شبکه ای که در این مساله داشتیم، به نظر می رسد برای یک لایه اضافه کردن dropout با مقدار برابر با 0.05 به بهبودی تشخیص توسط مدل می پردازد.

سوال دوم – تشخیص نُت موسیقی

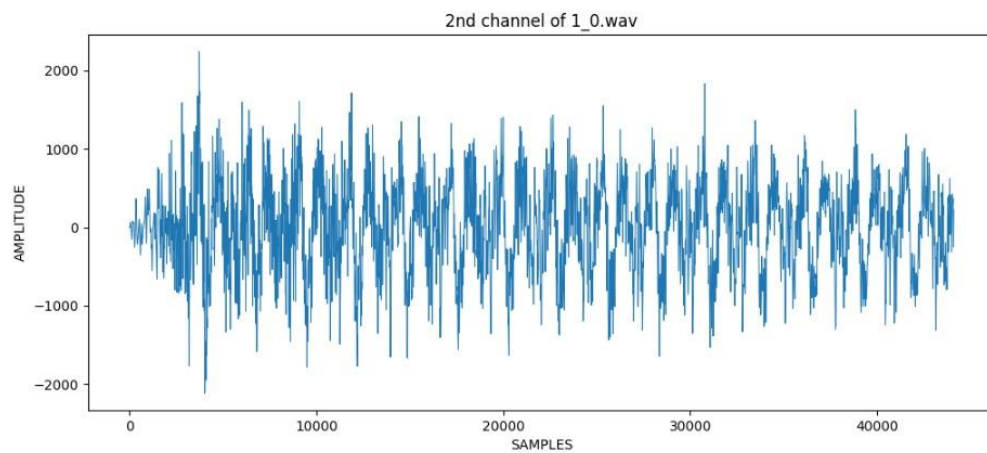
قسمت اول: مجموعه داده

در این قسمت در ابتدا فایل های فرمت mid. را به فرمت wav. تبدیل می کنیم تا بتوانیم با استفاده از کتابخانه های صوتی روی آنها پردازش انجام دهیم. و سپس سیگنال های موجود در فایل های 1_0.wav و 1_12.wav را برای هر دو کانال پلات می کنیم که نتیجه بصورت زیر حاصل می شود:



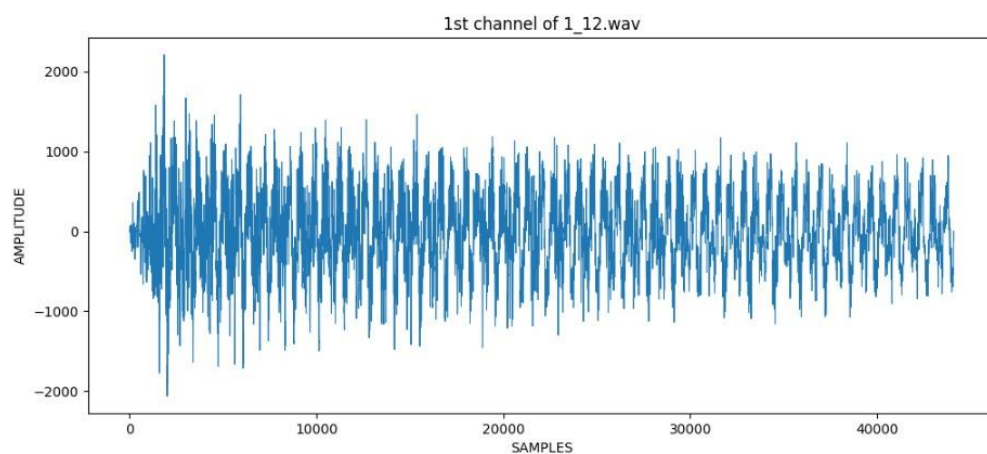
شکل 23 سیگنال 1_0 برای کانال اول

حال نتایج را برای کانال دوم بررسی می کنیم.



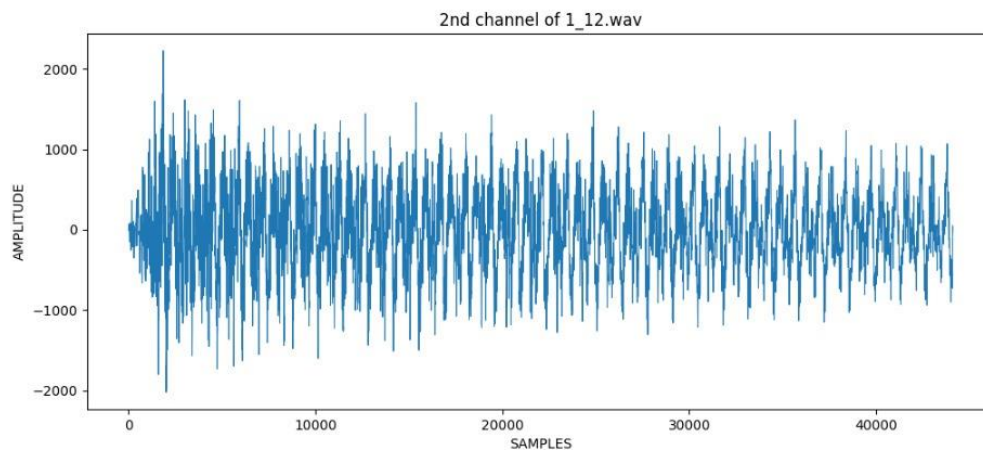
شکل 24 سیگنال 1_0 برای کانال دوم

سپس برای فایل بعدی ذکر شده در صورت سوال نتایج به صورت زیر است.



شکل 25 سیگنال 1_12 برای کانال اول

نهایتاً برای کانال دوم فایل مذکور نتایج را خدمتتان ارائه می‌کنیم.



شکل 26 سیگنال 1_12 برای کانال دوم

و جدولی را که مشخصات نته‌ها را از فایل دریافتی بیان می‌کند رسم نمودیم که به صورت زیر است.

| Characteristics of Note | | | |
|-------------------------|---------------|----------|---------------------|
| Note | Time Duration | Channels | Sample Rate |
| 1_0.wav | 1.0 (s) | 2 | 44100 (samples/sec) |
| 1_12.wav | 1.0 (s) | 2 | 44100 (samples/sec) |

شکل 27 اطلاعات مربوط به سیگنال‌های 1_0 و 1_12

حال به موارد خواسته شده در صورت سوال می‌پردازیم.

❖ شباهت‌ها:

رنج دامنه‌ها در یک instrument تقریباً مشابه هم هستند و می‌بینیم که در این instrument رنج دامنه بین $[-2000, 2000]$ است.

در نمونه‌های اولیه (در زمانهای اولیه) پوش منحنی صعودی و سپس یک روند نزولی به خود می‌گیرد.

❖ تفاوت:

تفاوت این دو سیگنال در فرکانس آنها می باشد و اینها مربوط به دو نُت متفاوت برای یک instrument هستند.

❖ محور افقی و عمودی در تصویر:

در اشکال فوق نشان دادیم که محور افقی همان Samples یعنی نمونه‌ها سیگنال ما در هر ریکورد (یا مقدار هر بُعد در هر ریکورد) است و محور عمودی Amplitude (دامنه) است.

قسمت دوم: تولید مجموعه های training, validation و test

در رابطه با تقسیم بندی داده‌های آموزش، تست و ارزیابی در ابتدا آمادیم و لیستی از نام فایلها تهیه کردیم. در این لحظه نام تمامی فایل‌ها در یک لیست ذخیره شده اند. سپس لیست مذکور را با استفاده از دستور random.shuffle بُر زدیم یا به عبارت علمی‌تر اقدام به مخلوط و درهم‌ریختگی این نام‌ها در لیست ذکر شده کردیم. سپس سائز داده‌های آموزش را 85 درصد آموزش نموده و مابقی را به عنوان تست در نظر گرفتیم و با داشتن نام هر یک از فایل‌های wav اقدام به کپی برداری از فایل‌های متناظرشان و ذخیره در پوشه‌ی مربوط به هر کلاس (کلاس آموزش یا تست) کردیم. حال در زمان fit کردنِ مدل بر داده‌های آموزش، validation-size را برای 15 درصد مجموعه داده‌های آموزش اولیه در نظر گرفتیم و نهایتاً با این تقسیم بندی اقدام به آموزش نمودیم. دقت بفرمایید که به طور معمول 15 تا 30 درصد از مجموعه داده‌گان را به عنوان تست در نظر می‌گیرند که چون تعداد کلاس‌های ما زیاد بود تصمیم گرفتیم مدل به خوبی آموزش دیده باشد و عدد 15 درصد را به عنوان داده‌های تست در نظر گرفتیم. از طرفی در بین داده‌گان یک کلاس همبستگی بسیار بالایی از نظر دامنه مشاهده نمی‌شد و این موضوع ایجاب می‌کرد که داده‌های بیشتری را مورد آموزش قرار دهیم. داده‌های ارزیاب هم نه باید زیاد و نه کم انتخاب شوند. در صورتیکه تعداد داده‌های ارزیاب ما کم باشند، مدل ممکن از تغییرات خوبی در وزن‌های خود و آپدیت شبکه در هر iteration بوجود نیارد و تا حدی موجب overfitting نیز بشود. از طرفی داده‌های ارزیابی بسیار زیاد نیز ممکن است باعث تغییرات و جهش‌های زیاد در آپدیتِ وزن‌ها شود که این هم مطلوب نیست. لذا از نظر بنده این اعداد با استدلال‌هایی که ارائه نمودم مناسب بودند.

قسمت سوم: پیاده سازی شبکه عصبی

توضیحات مقدماتی در مورد شیوهی حل این سوال توسط گروه ما:

در ابتدا برای قسمت پیاده سازی چند نکته را ذکر می کنیم. متأسفانه سخت افزار اینجانب و هم گروهی من بسیار معمولی و اصلاً برای پردازش این حد از داده پاسخگو نبود. از طرفی سایت کلب هم ما را بدلیل استفاده ی زیاد در این تمرین بُن نمود. لذا ما مجبور شدیم تعداد نُت ها را کمتر (به طور معمول 30 تا) و تعداد ابزارها را نیز در همین حدود در نظر بگیریم. البته در قسمت فریم های زمانی این اعداد تغییر کردند چون فضای حافظه را به یکباره زیاد می کرد و کامپیوتر پاسخگو نبود.

توضیحات پیاده سازی:

برای پیاده سازی کلاسی با نام noteRecognizer طراحی نمودیم. در این کلاس قابلیت های زیادی از جمله کاهش ابعاد با نرخ های مجاز، تعریف فریم های زمانی، تعداد نُت، تعداد ابزار، تنظیمات ساختار مدل و ... را پیاده کردیم.

لذا در گام اول کافی است که بجای timeframe عدد 250 که در حقیقت 250 ms است را قرار دهید. در این حالت ورودی ها را به عنوان قطعه های صوتی متشکل از فریم های صوتی 250 ms به عنوان ورودی شبکه در نظر می گیرید. با توجه به سخت افزار خود تعداد نُت ها را 30 و تعداد ابزارهای مربوط به نُت های انتخابی را نیز 30 (یعنی 30 تا ساز برای هر نُت) در نظر گرفتیم. نرخ یادگیر تابع بهینه ساز adam را 0.001 لحاظ کرده و شبکه را برای 50 اپاک آموزش دیدیم.

مهمترین قسمت شبکه: این شبکه بدلیل پیش پردازش های مناسبی که بر روی آن انجام گرفته در بسیاری از موارد نتایجی را (البته برای تعداد نُت کمتر از 30 و تعداد ساز کمتر از 30) ارائه می دهد که از بسیاری از افراد دیگر که نتایجشان را پرسیدم بهتر است. دلیل این امر دو چیز است.

✓ دلیل اول عملکرد نسبتاً مطلوب شبکه در این مسأله بخصوص:

پیش پردازش بر روی داده ها. انجام پیش پردازش بر روی داده ها بدین صورت بود که من هر داده را با توجه به سمپل های خودش اسکیل کردم. این امر باعث افزایش صحت عملکرد شبکه ام شد.

✓ دلیل دوم: یادگیری برای شبکه های بازگشتی با تعداد ابعاد بالا به ازای هر ریکورد در اکثر مواقع امکان پذیر نیست. لذا تصمیم گرفتیم از ابعاد هر ریکورد، نمونه برداری کنیم. این امر نیز به طرز بسیار خوبی توانست قدرت شبکه را در پیش بینی داده های تست و نیز در صحت داده های ارزیابی بالا ببرد.

نهایتاً دلیل عملکرد نسبتاً خوب شبکه را علاوه بر تنظیم مناسب پارامترها و هایپرپارامترها، بلکه ناشی از دو پیاده‌سازی فوق می‌دانم.

دقت فرمایید که نرخ نمونه برداری از داده‌گان دریافتی برای کاهش ابعاد در کد توسط پارامتر طول دیتای فشرده (compressionLength) قابل تنظیم است. تنها توجه کنید که نرخ بسیار بالا زمان پیش‌پردازش‌ها را به شدت افزایش می‌دهد و نرخ دیفالت قرار گرفته در کد تا ده برابر هم می‌تواند سرعت خوبی از خود نشان دهد. حال به پاسخگویی به بخش‌های مسأله می‌پردازیم.

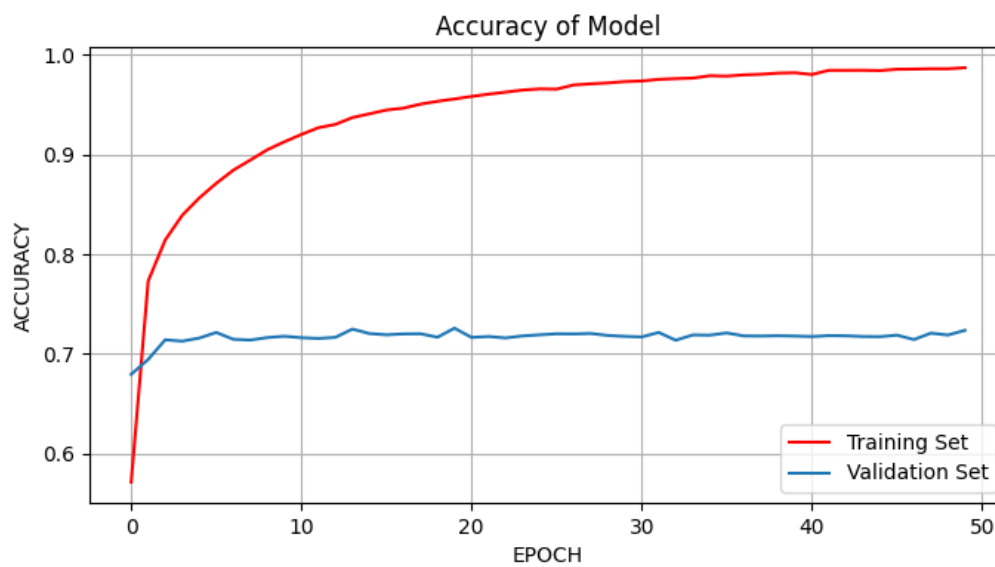
(الف)

ابتدا نتایج را برای شبکه RNN بررسی می‌کنیم. فلذا شبکه را با سلول‌های RNN طراحی کردیم و ساختار شبکه را بصورت زیر می‌توانید مشاهده فرمایید:

| Layer (type) | Output Shape | Param # |
|--------------------------|--------------|---------|
| ===== | ===== | ===== |
| simple_rnn (SimpleRNN) | (None, 128) | 29312 |
| dense (Dense) | (None, 64) | 8256 |
| dense_1 (Dense) | (None, 30) | 1950 |
| ===== | ===== | ===== |
| Total params: 39,518 | | |
| Trainable params: 39,518 | | |
| Non-trainable params: 0 | | |

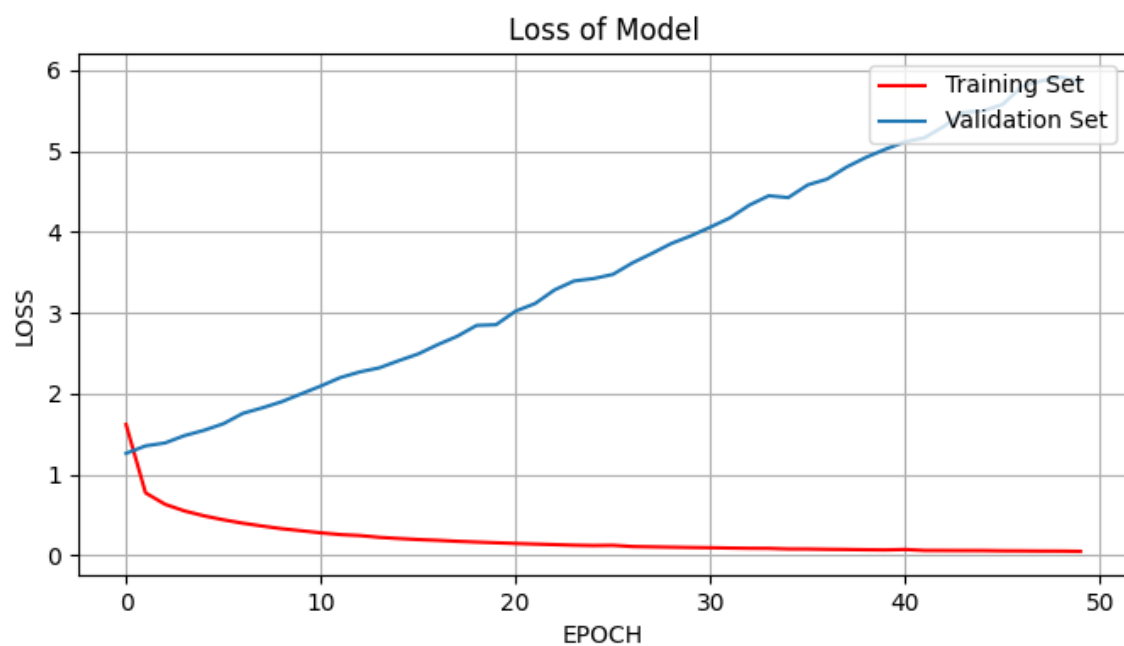
شکل 28 ساختار شبکه انتخابی با سلول‌های RNN

صحت مدل بصورت زیر می باشد:



شکل 29 صحت مدل شبکه RNN

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 30 خطا مدل شبکه RNN

زمان آموزش نیز بصورت زیر شد:

```
Training Time: 290.16404390335083
```

مقادیر Test Loss و Test Accuracy بصورت زیر شد:

```
1266/1266 [=====] - 3s 3ms/step - loss: 7.6772 - accuracy: 0.6378  
Test Loss, Test Accuracy: [7.67722749710083, 0.6377530694007874]
```

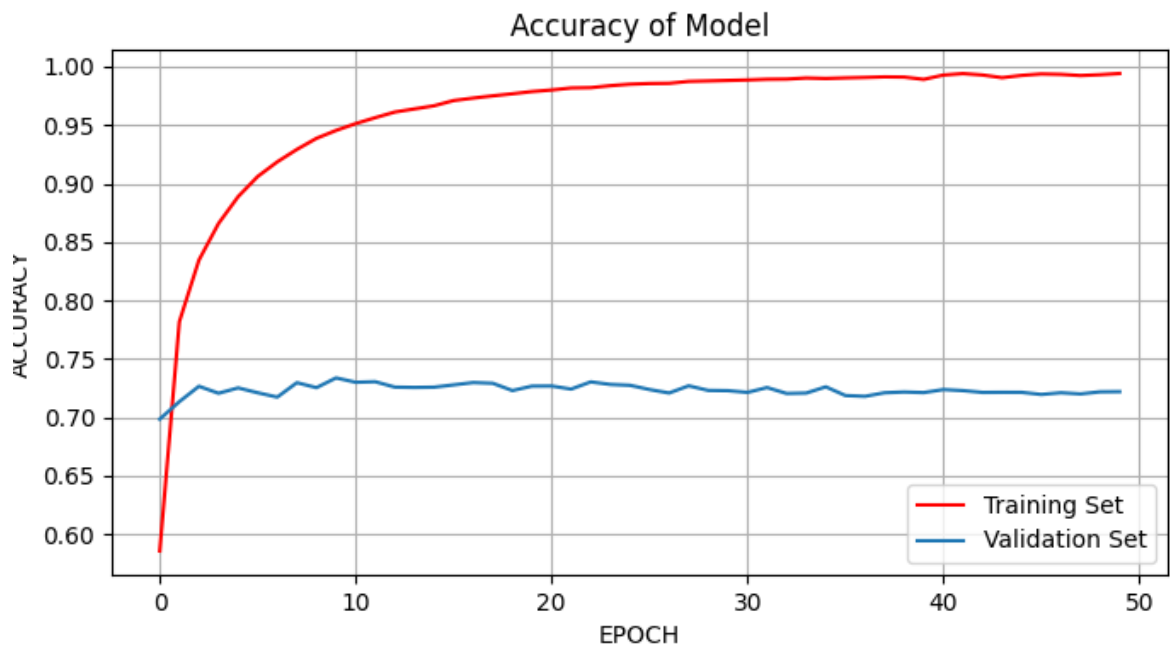
همانطور که مشاهده می‌فرمایید شبکه RNN که ساده‌ترین شبکه در این قسمت است نتایج نسبتاً خوبی برای پیش‌بینی و نیز آموزش شبکه از خود نشان داده. لکن برای مقایسه دقیق سائز شبکه‌ها را نیز بررسی می‌کنیم.

حال شبکه را با سلول‌های LSTM طراحی کردیم و ساختار شبکه بصورت زیر می‌باشد:

| Layer (type) | Output Shape | Param # |
|---------------------------|--------------|---------|
| lstm (LSTM) | (None, 128) | 117248 |
| dense (Dense) | (None, 64) | 8256 |
| dense_1 (Dense) | (None, 30) | 1950 |
| Total params: 127,454 | | |
| Trainable params: 127,454 | | |
| Non-trainable params: 0 | | |

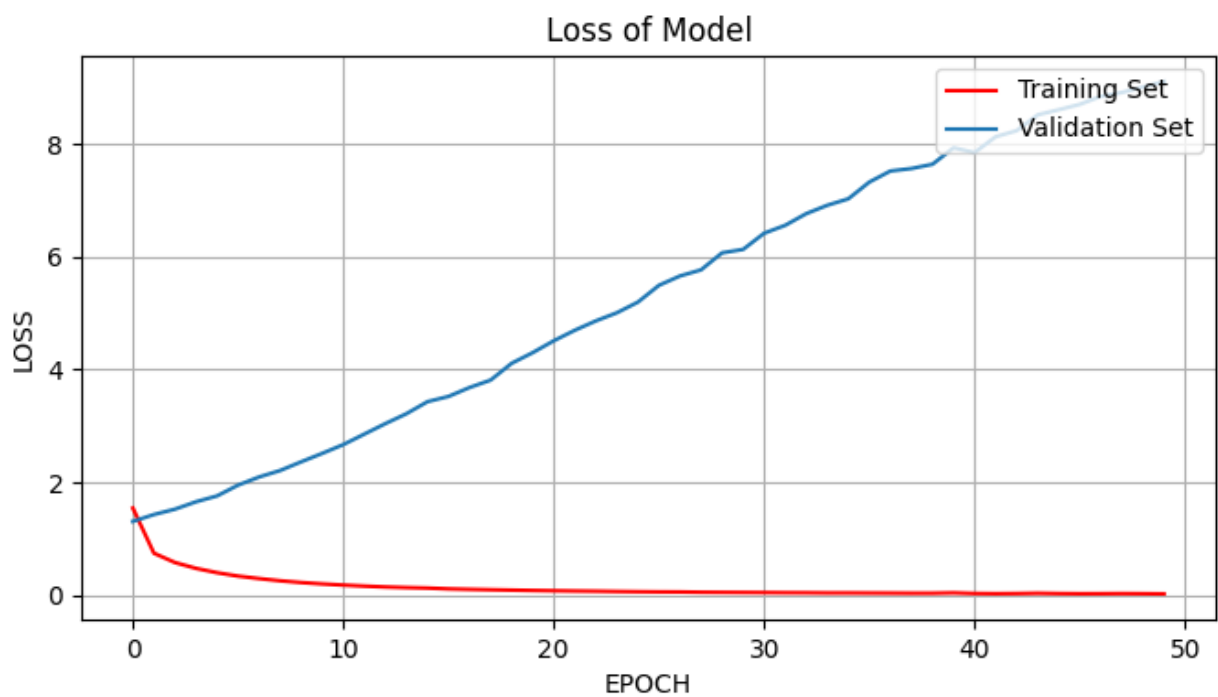
شکل 31 ساختار شبکه انتخابی با سلول‌های LSTM

صحت مدل بصورت زیر می باشد:



شکل 32 صحت مدل شبکه LSTM

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 33 خطا مدل شبکه LSTM

زمان آموزش نیز بصورت زیر شد:

```
Training Time: 578.1516191959381
```

مقادیر Test Accuracy و Test Loss بصورت زیر شد:

```
1266/1266 [=====] - 4s 3ms/step - loss: 10.3972 - accuracy: 0.6334  
Test Loss, Test Accuracy: [10.397180557250977, 0.6333580017089844]
```

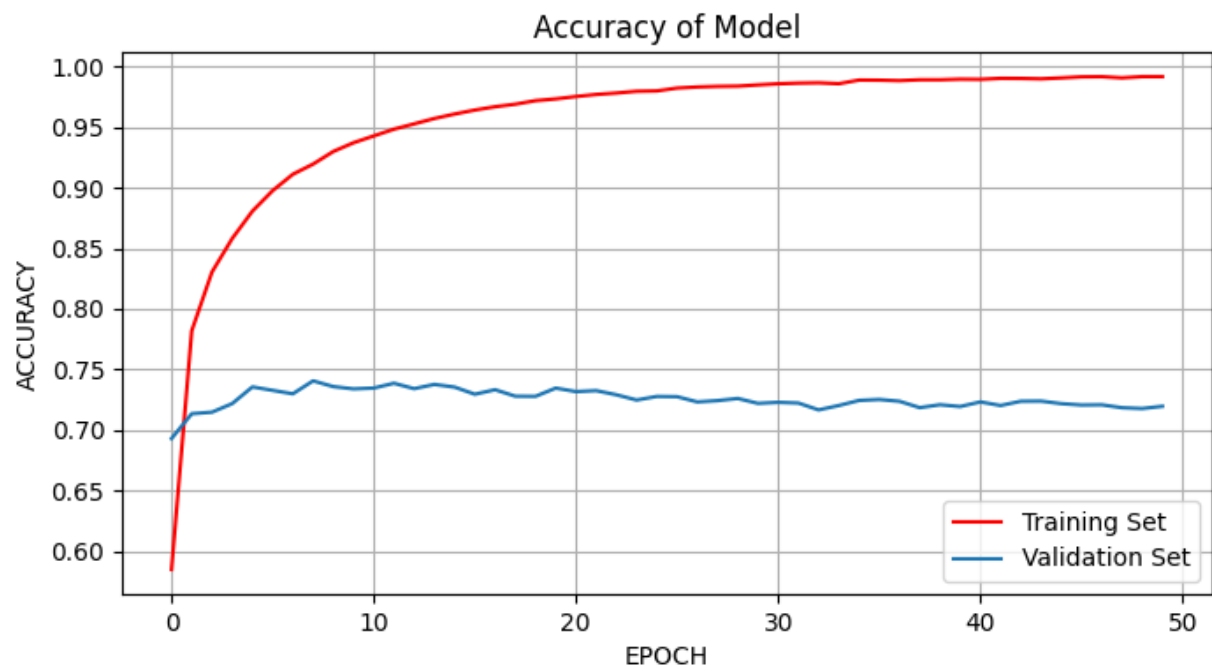
سپس شبکه را با سلول های GRU طراحی کردیم.

ساختار شبکه بصورت زیر می باشد:

| Layer (type) | Output Shape | Param # |
|--------------------------|--------------|---------|
| ===== | ===== | ===== |
| gru (GRU) | (None, 128) | 88320 |
| dense (Dense) | (None, 64) | 8256 |
| dense_1 (Dense) | (None, 30) | 1950 |
| ===== | ===== | ===== |
| Total params: 98,526 | | |
| Trainable params: 98,526 | | |
| Non-trainable params: 0 | | |

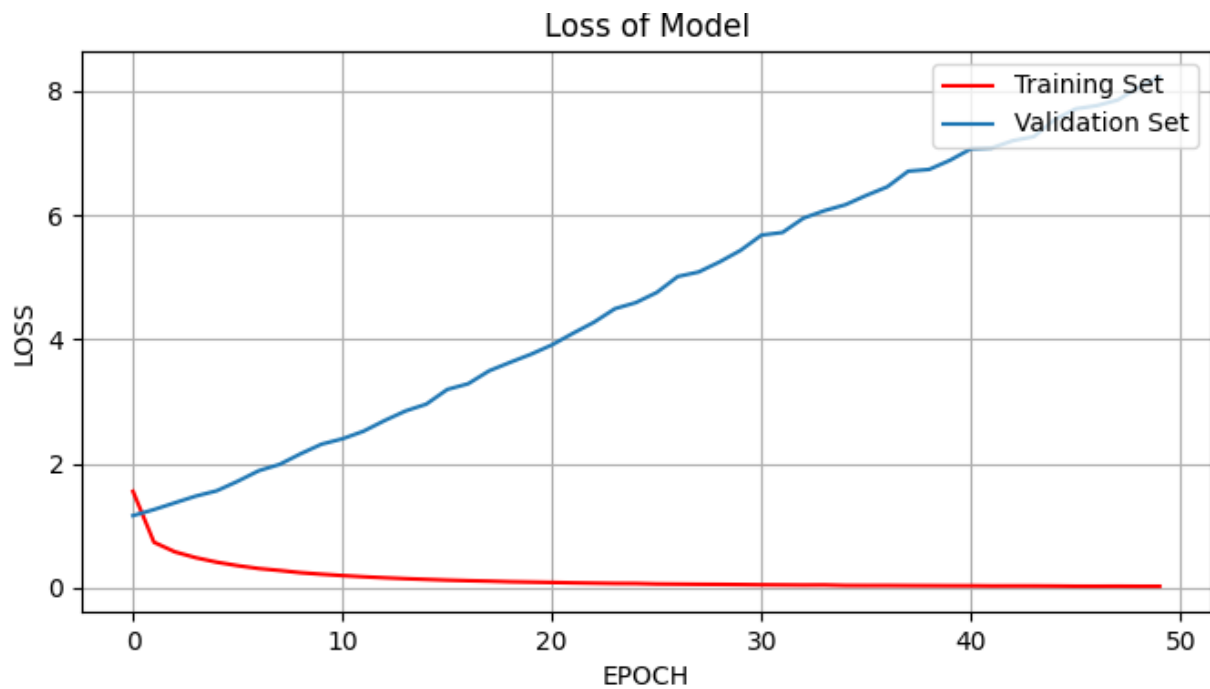
شکل 34 ساختار شبکه انتخابی با سلول های GRU

صحت مدل بصورت زیر می باشد:



شکل 35 صحت مدل شبکه GRU

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 36 خطا مدل شبکه GRU

زمان آموزش نیز بصورت زیر شد:

Training Time: 484.50505685806274

مقادیر Test Accuracy و Test Loss بصورت زیر شد:

1266/1266 [=====] - 6s 5ms/step - loss: 9.3452 - accuracy: 0.6469
Test Loss, Test Accuracy: [9.3451509475708, 0.6468641757965088]

نتایج کلی بصورت زیر می شود:

جدول 1 نتایج کلی پس از طراحی شبکه با RNN و LSTM و GRU

| | Training Time | Test Accuracy | Test Loss |
|------|---------------|---------------|-----------|
| RNN | 290.1640 | 0.63775 | 7.6772 |
| LSTM | 578.1516 | 0.633358 | 10.3971 |
| GRU | 484.5051 | 0.646864 | 9.34515 |

همانطور که از نتایج فوق مشاهده می‌فرمایید GRU موفق شده است که در پیش‌بینی نهایی بر روی مجموعه داده‌گان تست به بالاترین صحت برسد. از طرفی زمان بهتری را نسب به LSTM از خود نشان داده است. در مراجع مختلفی که مطالعه نمودم نوشته شده بود که برای داده‌گان با ابعاد کم مطلوب است. از طرفی توجه کنید که ما ابعاد هر ریکورد را بسیار کاهش دادیم. فلذا یکی از دلایلی که GRU را در این گام انتخاب می‌کنیم همین است. هر چند در قسمت‌های بعدی، GRU جای خود را به LSTM خواهد داد که در بخش‌های بعدی شرح می‌دهیم. توجه نمایید که در نتایج برآمده از این قسمت، gru تقریباً 22 درصد سریعتر از lstm است.

ب) بررسی تاثیر dropout روی هر سه شبکه قبل:

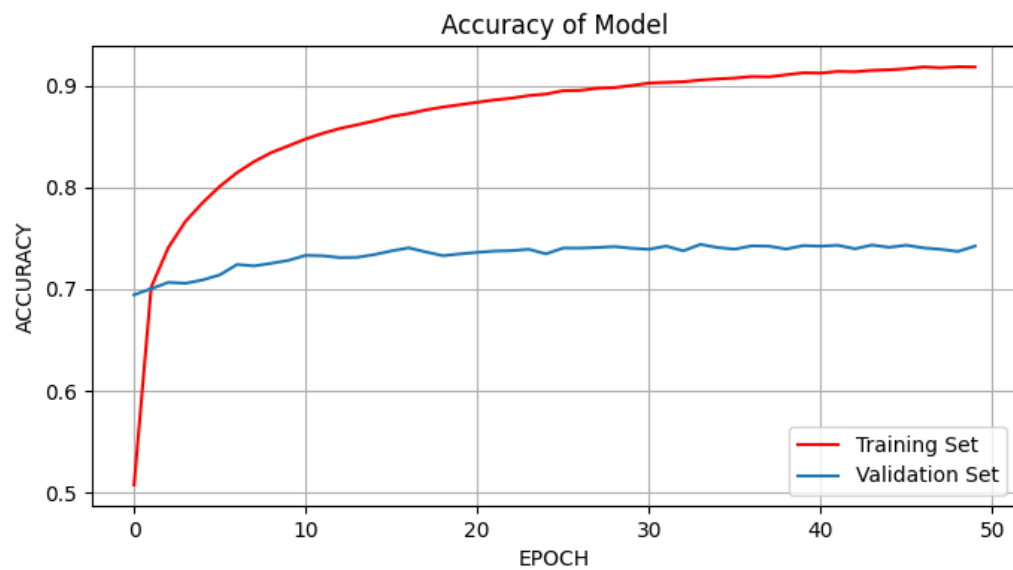
بررسی RNN با Dropout:

بعد از لایه RNN یک dropout با مقدار 10% و نیز یک dropout با مقدار 5% برای fully connect لایه میانی قرار دادیم که ساختار شبکه بصورت زیر می باشد:

| Layer (type) | Output Shape | Param # |
|--------------------------|--------------|---------|
| ===== | | |
| simple_rnn (SimpleRNN) | (None, 128) | 29312 |
| dropout (Dropout) | (None, 128) | 0 |
| dense (Dense) | (None, 64) | 8256 |
| dropout_1 (Dropout) | (None, 64) | 0 |
| dense_1 (Dense) | (None, 30) | 1950 |
| ===== | | |
| Total params: 39,518 | | |
| Trainable params: 39,518 | | |
| Non-trainable params: 0 | | |

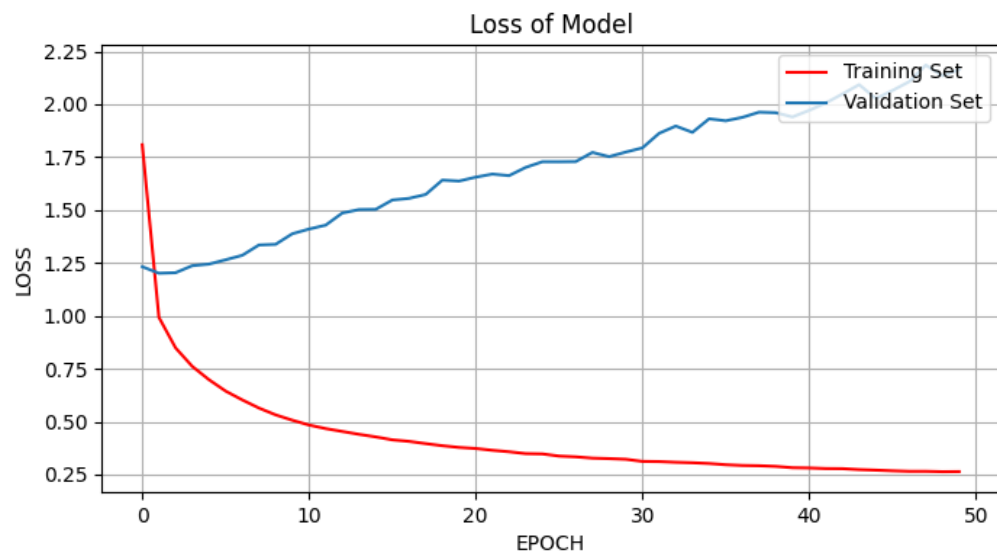
شکل 37 ساختار شبکه انتخابی با سلول های RNN با اعمال dropout

صحت مدل بصورت زیر می باشد:



شکل 38 صحت مدل شبکه RNN با اعمال dropout

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 39 خطا مدل شبکه RNN با اعمال dropout

زمان آموزش نیز بصورت زیر شد:

```
Training Time: 351.5059199333191
```

مقادیر Test Accuracy و Test Loss بصورت زیر شد:

```
1266/1266 [=====] - 4s 3ms/step - loss: 2.2082 - accuracy: 0.6479  
Test Loss, Test Accuracy: [2.2081570625305176, 0.6478765606880188]
```

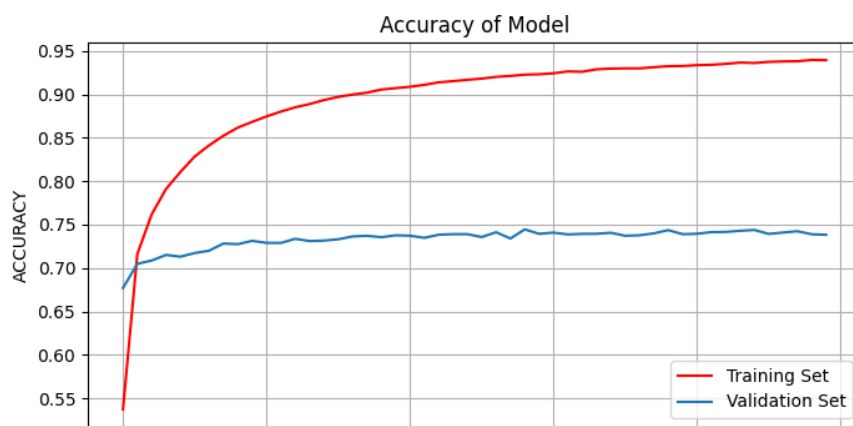
بررسی LSTM با Dropout:

بعد از لایه LSTM یک dropout با مقدار 10% و نیز یک dropout برای fully connect لایه میانی با مقدار 5% قرار دادیم که ساختار شبکه بصورت زیر می باشد:

| Layer (type) | Output Shape | Param # |
|---------------------------|--------------|---------|
| lstm (LSTM) | (None, 128) | 117248 |
| dropout (Dropout) | (None, 128) | 0 |
| dense (Dense) | (None, 64) | 8256 |
| dropout_1 (Dropout) | (None, 64) | 0 |
| dense_1 (Dense) | (None, 30) | 1950 |
| Total params: 127,454 | | |
| Trainable params: 127,454 | | |
| Non-trainable params: 0 | | |

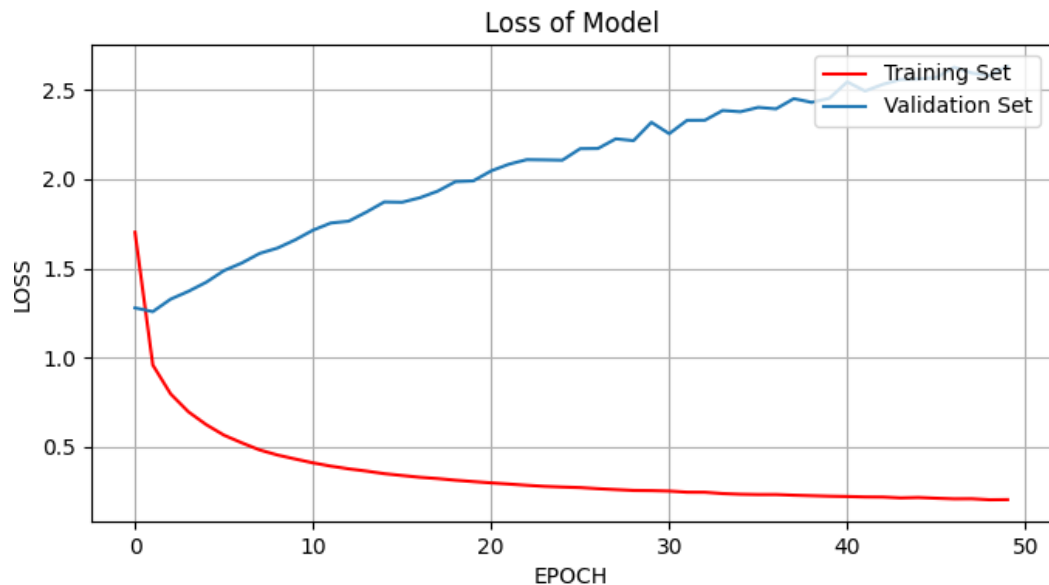
شکل 40 ساختار شبکه انتخابی با سلول های LSTM با اعمال dropout

صحت مدل بصورت زیر می باشد:



شکل 41 صحت مدل شبکه LSTM با اعمال dropout

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 42 خطا مدل شبکه LSTM با اعمال dropout

زمان آموزش نیز بصورت زیر شد:

Training Time: 663.2843239307404

مقادیر Test Accuracy و Test Loss بصورت زیر شد:

```
1266/1266 [=====] - 4s 3ms/step - loss: 2.5930 - accuracy: 0.6632  
Test Loss, Test Accuracy: [2.592963695526123, 0.6632345914840698]
```

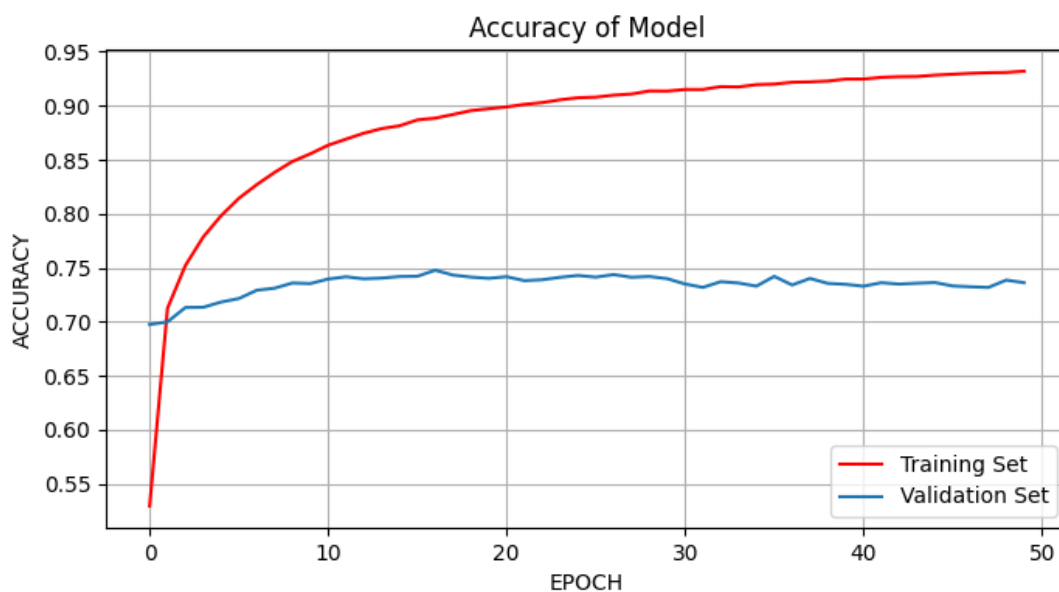
بررسی GRU با Dropout:

بعد از لایه GRU یک dropout با مقدار 10% و نیز یک dropout برای fully connect لایه میانی با مقدار 5% قرار دادیم که ساختار شبکه بصورت زیر می باشد:

| Layer (type) | Output Shape | Param # |
|--------------------------|--------------|---------|
| gru (GRU) | (None, 128) | 88320 |
| dropout (Dropout) | (None, 128) | 0 |
| dense (Dense) | (None, 64) | 8256 |
| dropout_1 (Dropout) | (None, 64) | 0 |
| dense_1 (Dense) | (None, 30) | 1950 |
| Total params: 98,526 | | |
| Trainable params: 98,526 | | |
| Non-trainable params: 0 | | |

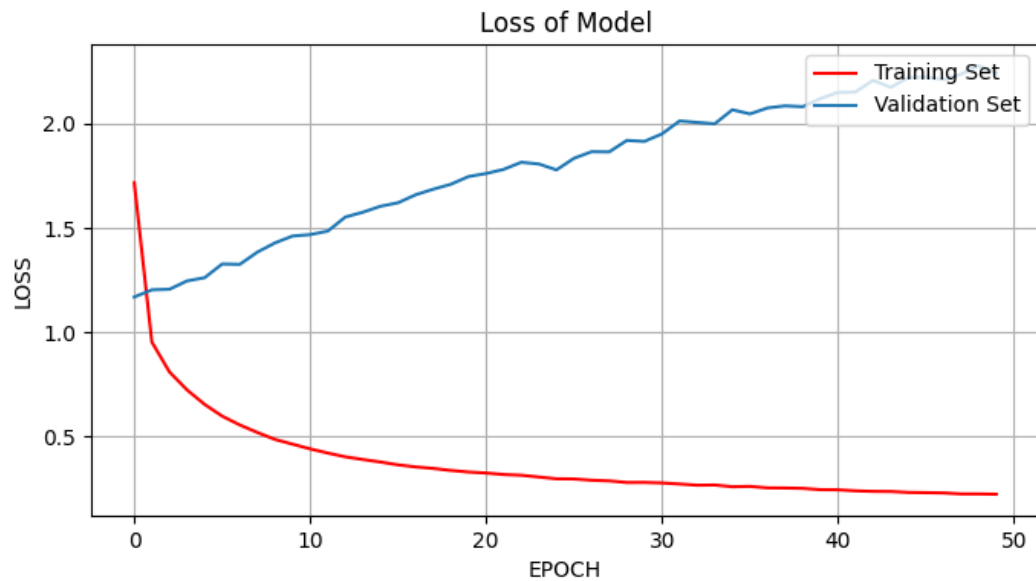
شکل 43 ساختار شبکه انتخابی با سلول های GRU با اعمال dropout

صحت مدل بصورت زیر می باشد:



شکل 44 صحت مدل شبکه GRU با اعمال dropout

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 45 خطا مدل شبکه GRU با اعمال dropout

زمان آموزش نیز بصورت زیر شد:

Training Time: 560.7549819946289

مقادیر Test Accuracy و Test Loss بصورت زیر شد:

```
1266/1266 [=====] - 4s 3ms/step - loss: 2.4874 - accuracy: 0.6564
Test Loss, Test Accuracy: [2.487448215484619, 0.6564197540283203]
```

نتایج کلی بصورت زیر می شود:

جدول 2 نتایج کلی پس از طراحی شبکه با RNN و LSTM و GRU پس از اعمال dropout

| | Training Time | Test Accuracy | Test Loss |
|------|---------------|---------------|-----------|
| RNN | 351.5059 | 0.647876 | 2.2082 |
| LSTM | 663.284 | 0.663235 | 2.5929 |
| GRU | 560.755 | 0.65642 | 2.48745 |

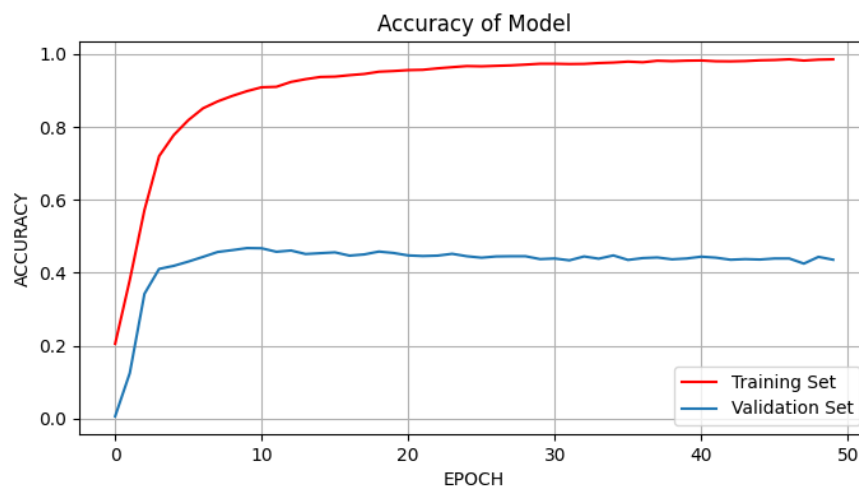
همانطور که از نتایج این قسمت برآمده است، LSTM بهترین عملکرد را بالاخص در پیش‌بینی داده‌های تست انجام داده. لذا زمان زیاد این مدل را به عملکرد خوبش به جان می‌خریم و ادامه‌ی روند در بخش‌های بعدی را با از استفاده از لایه‌ی LSTM پیش می‌گیریم.

پاسخ بخش ج)

در این بخش که فریم‌های زمانی مطرح بود و نیز با لحاظ محدودیت‌های سخت‌افزاری، تعداد نُت‌ها و نیز تعداد سازهای مختص این نُت‌ها را هر یک 10 در نظر می‌گیریم و روند را ادامه می‌دهیم. دیدیم که بهترین شبکه‌ای که در قسمت قبل بدست آمد، شبکه LSTM بود. حال برای ادامه مساله از این شبکه استفاده می‌کنیم. برای فریم‌های زمانی مختلف داریم:

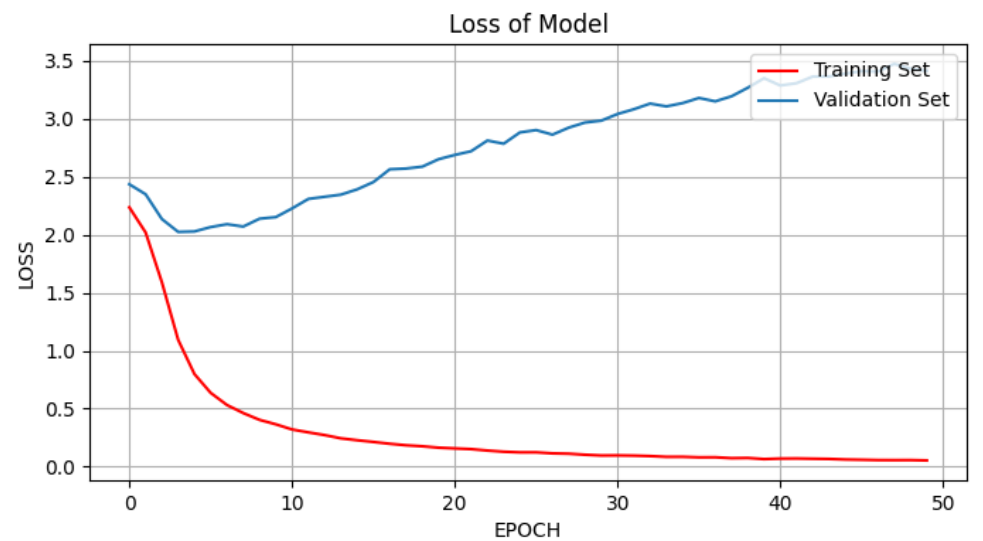
فریم زمانی 300:

صحت مدل بصورت زیر می‌باشد:



شکل 46 صحت مدل شبکه LSTM با اعمال dropout و فریم زمانی 300

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 47 خطا مدل شبکه LSTM با اعمال dropout و فریم زمانی 300

زمان آموزش نیز بصورت زیر شد:

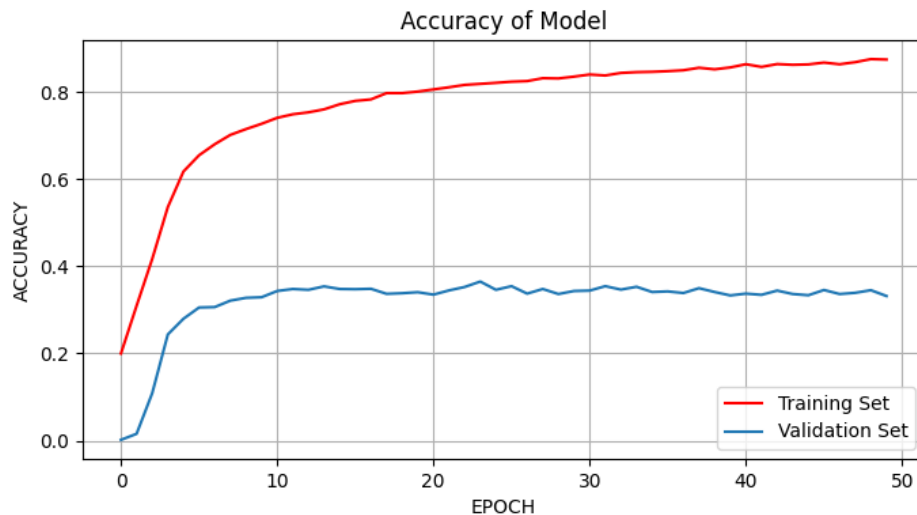
Training Time: 33.82352542877197

مقادیر Test Accuracy و Test Loss بصورت زیر شد:

```
66/66 [=====] - 0s 5ms/step - loss: 2.5242 - accuracy: 0.7381  
Test Loss, Test Accuracy: [2.5242090225219727, 0.738095223903656]
```

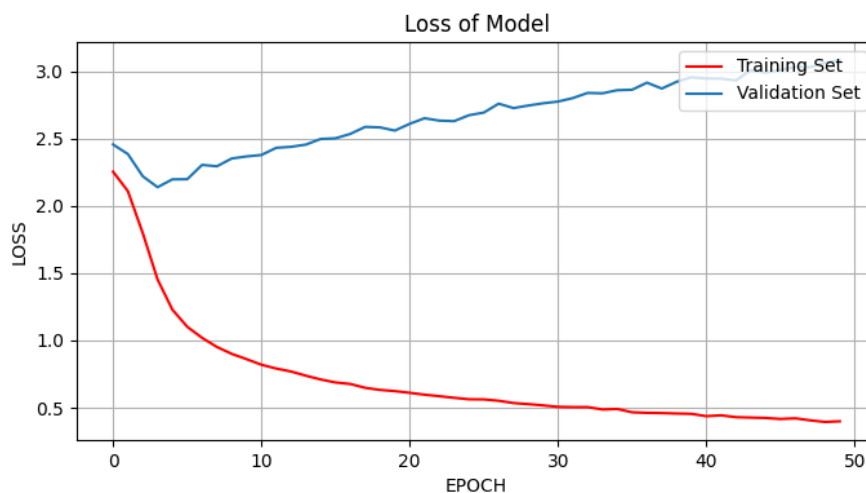
فریم زمانی 150:

صحت مدل بصورت زیر می باشد:



شکل 48 صحت مدل شبکه LSTM با اعمال dropout و فریم زمانی 150

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 49 خطا مدل شبکه LSTM با اعمال dropout و فریم زمانی 150

زمان آموزش نیز بصورت زیر شد:

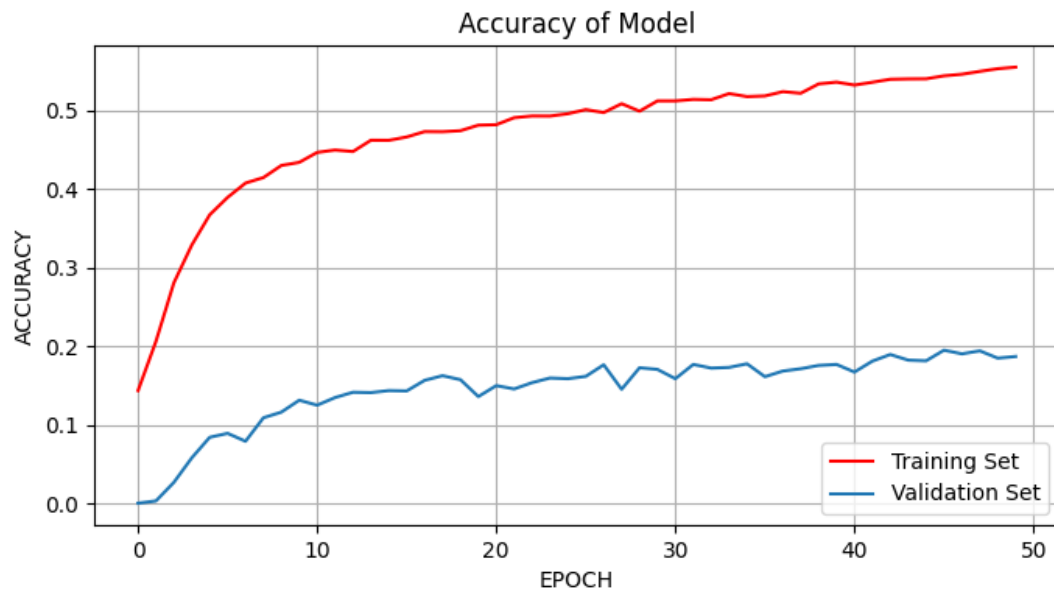
Training Time: 36.53097486495972

مقادیر Test Accuracy و Test Loss بصورت زیر شد:

```
80/80 [=====] - 0s 3ms/step - loss: 1.9544 - accuracy: 0.5769
Test Loss, Test Accuracy: [1.9543766975402832, 0.5768627524375916]
```

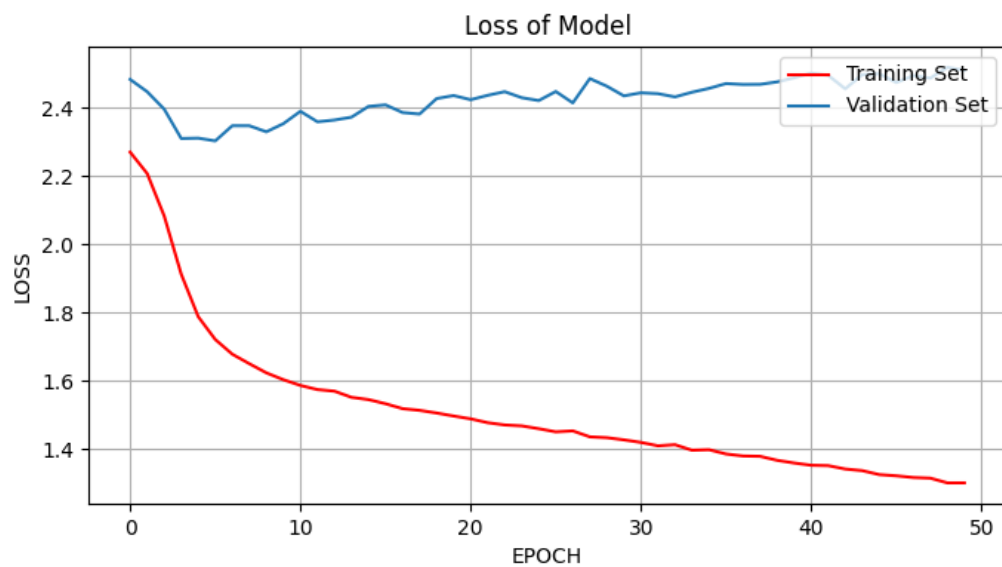
فریم زمانی 70:

صحت مدل بصورت زیر می باشد:



شکل 50 صحت مدل شبکه LSTM با اعمال dropout و فریم زمانی 70

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 51 خطا مدل شبکه LSTM با اعمال dropout و فریم زمانی 70

زمان آموزش نیز بصورت زیر شد:

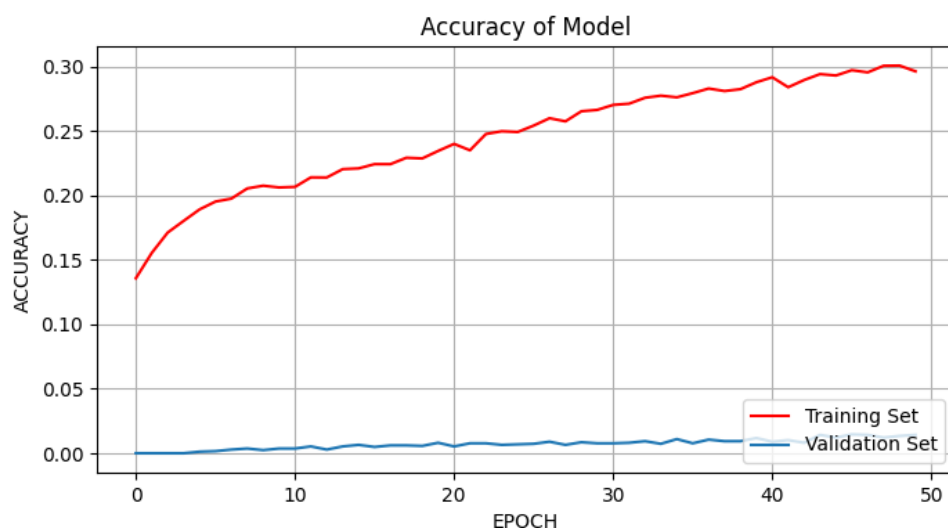
```
Training Time: 42.726794481277466
```

مقادیر Test Loss و Test Accuracy بصورت زیر شد:

```
88/88 [=====] - 0s 4ms/step - loss: 2.1458 - accuracy: 0.2222  
Test Loss, Test Accuracy: [2.1457836627960205, 0.222222238779068]
```

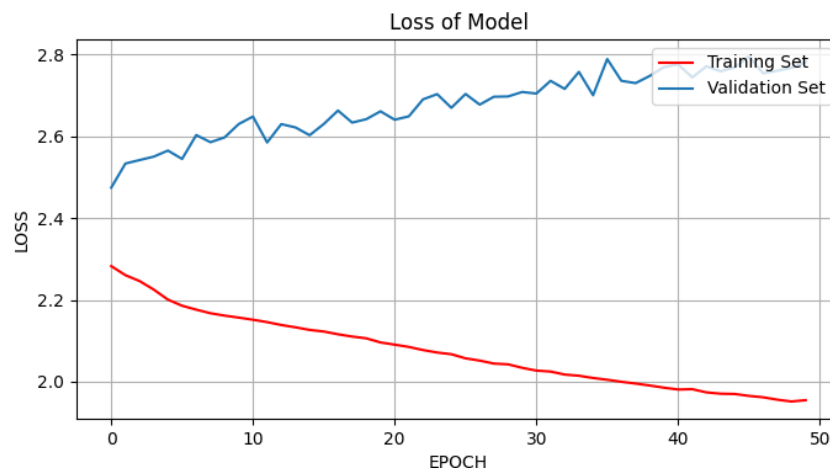
فریم زمانی 35:

صحت مدل بصورت زیر می باشد:



شکل 52 صحت مدل شبکه LSTM با اعمال dropout و فریم زمانی 35

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 53 خطا مدل شبکه LSTM با اعمال dropout و فریم زمانی 35

زمان آموزش نیز بصورت زیر شد:

Training Time: 36.55024242401123

مقادیر Test Accuracy و Test Loss بصورت زیر شد:

```
91/91 [=====] - 0s 3ms/step - loss: 2.5583 - accuracy: 0.0788
Test Loss, Test Accuracy: [2.5582573413848877, 0.07875647395849228]
```

نتایج کلی به شرح زیر است:

جدول 3 نتایج کلی پس از فریم های زمانی مختلف روی مدل برنده قسمت قبل

| Time Frame | Training Time | Test Accuracy | Test Loss |
|------------|---------------|---------------|-----------|
| 300 | 33.8235 | 0.7381 | 2.5242 |
| 150 | 36.531 | 0.57686 | 1.95437 |
| 70 | 42.727 | 0.2222 | 2.1458 |
| 35 | 36.55 | 0.0787 | 2.5583 |

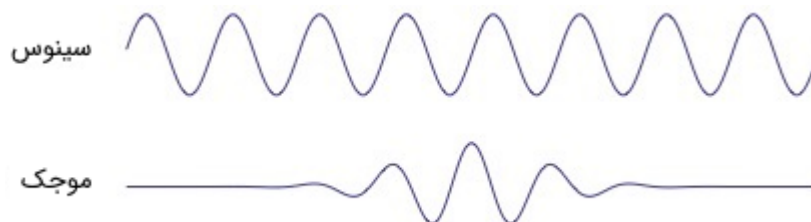
نتایج نشان می دهد که فریم زمانی 300 میلی ثانیه توانسته بهترین نتیجه را در آموزش داده گان و نیز پیش بینی ریکورهای تست و همچنین از نقطه نظر زمانی به ما ارائه کند. مسلماً هر اندازه که فریم زمانی

کاهش می‌یابد، اطلاعات همبستگی بیشتر پیدا کرده و عملاً قسمت غیر مفید افزایش یافته فلذا نتایج از نظر ما به این دلیل به شکل فوق صورت گرفته است. لذا بهترین نتیجه برای فریم زمانی 300ms است.

پاسخ بخش د) الگوریتم‌های استخراج ویژگی متنوعی وجود دارد که در اینجا به چند مورد می‌پردازیم:

تبدیل wavelet و تبدیل فوریه: تبدیل موجک (Wavelet Transform) یکی از تبدیلات مهم ریاضی است که در حوزه‌های مختلف علوم کاربرد دارد. ایده اصلی تبدیل موجک این است که بر ضعف‌ها و محدودیت‌های موجود در تبدیل فوریه غلبه کند. این تبدیل را بر خلاف تبدیل فوریه، می‌توان در مورد سیگنال‌های غیر ایستا و سیستم‌های دینامیک نیز مورد استفاده قرار داد.

تبدیل فوریه برای آنالیز سیگنال از یک سری امواج سینوسی با فرکانس‌های مختلف استفاده می‌کند. در این حالت، سیگنال به صورت ترکیبی خطی از سیگنال‌های سینوسی نمایش داده می‌شود. اما تبدیل موجک از تعدادی توابع به نام موجک استفاده می‌کند که هر کدام مقیاس متفاوتی دارند. همان طور که می‌دانیم معنی واژه موجک، موج کوچک است و توابع موجک نیز دقیقاً به همین صورت کوچک هستند. در تصویر زیر تفاوت بین یک سیگنال سینوسی و یک موجک نشان داده شده است:



شکل 54 مثالی از یک سیگنال سینوسی و موجک

همان طور که در تصویر بالا، کاملاً مشخص است که سیگنال سینوسی در یک لحظه خاص از زمان واقع نشده است. این سیگنال از بی‌نهایت شروع می‌شود و تا بی‌نهایت ادامه می‌یابد، در حالی که یک موجک در لحظه خاصی از زمان واقع شده است. این ویژگی به تبدیل موجک اجازه می‌دهد تا علاوه بر اطلاعات فرکانسی، اطلاعات زمانی را نیز به دست آورد. چون موجک در زمان واقع شده است، در نتیجه می‌توان سیگنال اصلی را در لحظات مختلف از زمان در موجک ضرب کرد. در گام نخست، با نقاط ابتدایی سیگنال شروع می‌کنیم و به تدریج موجک را به سمت انتهای سیگنال حرکت می‌دهیم. این عمل را کانولوشن (Convolution) می‌گویند.

بعد از این که کانولوشن را با سیگنال موجک اصلی (موجک مادر) انجام دادیم، می‌توانیم آن را به نحوی مقیاس‌دهی کنیم که بزرگ‌تر شود و دوباره فرایند را تکرار کنیم. این فرایند را تبدیل موجک گوئیم. تبدیل موجک یک سیگنال تک بعدی، دارای دو بعد است. این خروجی دو بعدی مربوط به تبدیل موجک، نمایش سیگنال اصلی بر حسب مقیاس و زمان است که به طیف اسپکتروگرام (Spectrogram) یا اسکالوگرام (Scaleogram) معروف است.

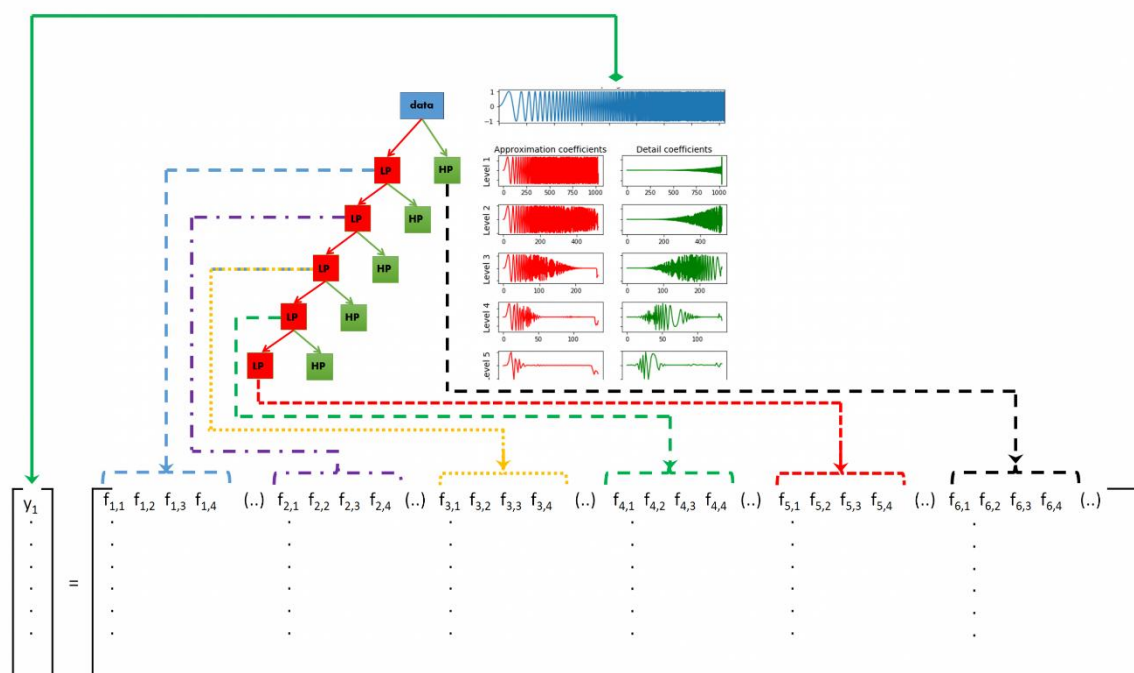
تبدیل فوریه معمولاً برای تبدیل یک سیگنال در طیف زمانی به سیگنالی در طیف فرکانسی مورد استفاده قرار می‌گیرد. در حالی که تبدیل فوریه نمایشی از سیگنال در حوزه فرکانس ایجاد می‌کند، تبدیل موجک نمایشی از سیگنال را در حوزه زمان و فرکانس ایجاد می‌کند و در نتیجه امکان دسترسی کارآمد به اطلاعات محلی در مورد سیگنال را فراهم می‌کند.

اشکال تبدیل فوریه این است که تمام اطلاعات سیگنال در کل محدوده تبدیل است. یعنی یک مشخصه محلی سیگنال به یک مشخصه جهانی (global) تبدیل می‌شود. که در مقاله زیر توضیحات بیشتر موجود است:

Wirsing, Karlton. "Time Frequency Analysis of Wavelet and Fourier Transform." *Wavelet Theory*. IntechOpen, 2020.

ایده‌ای که در پس طبقه‌بندی سیگنال با استفاده از تبدیل موجک گسسته وجود دارد را می‌توان به صورت زیر بیان کرد:

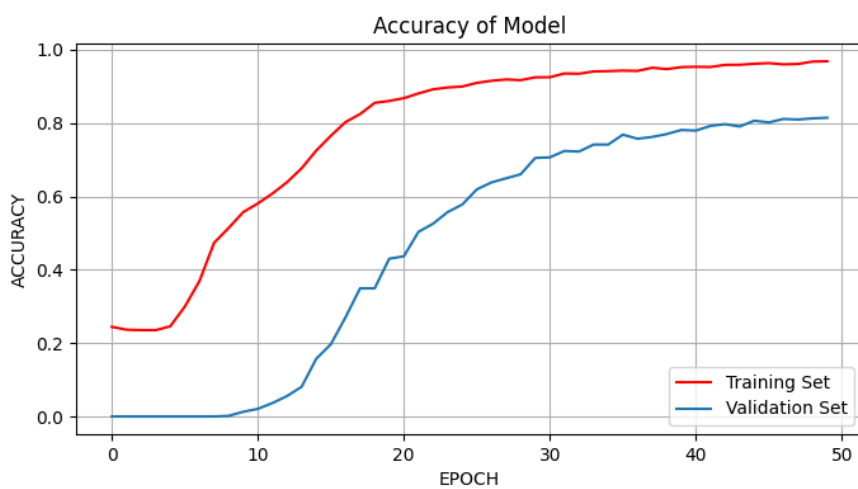
ابتدا از تبدیل موجک گسسته برای تفکیک یک سیگنال به زیرباندهای فرکانسی مختلف استفاده می‌کنیم. این کار را تا حد ممکن یا تا حد لازم تکرار می‌کنیم. چون انواع مختلف سیگنال‌ها، مشخصه‌های فرکانسی مختلفی را از خود نشان می‌دهند، در نتیجه این تمایز در رفتار باید در یکی از زیرباندهای فرکانسی خود را نشان دهد. بنابراین اگر ویژگی‌ها (Features) را از هریک از زیرباندهای مختلف ایجاد کنیم و از مجموعه تمام ویژگی‌ها به عنوان ورودی در یک طبقه‌بند (Classifier) استفاده کنیم و الگوریتم طبقه‌بندی را با استفاده از این ویژگی‌ها آموزش دهیم، آن‌گاه الگوریتم قادر خواهد بود که تمایز بین سیگنال‌های مختلف را تشخیص دهد و طبقه‌بندی را بر اساس آن انجام دهد. این مفهوم در تصویر زیر به خوبی نشان داده شده است:



شکل 55 عملکرد موجک

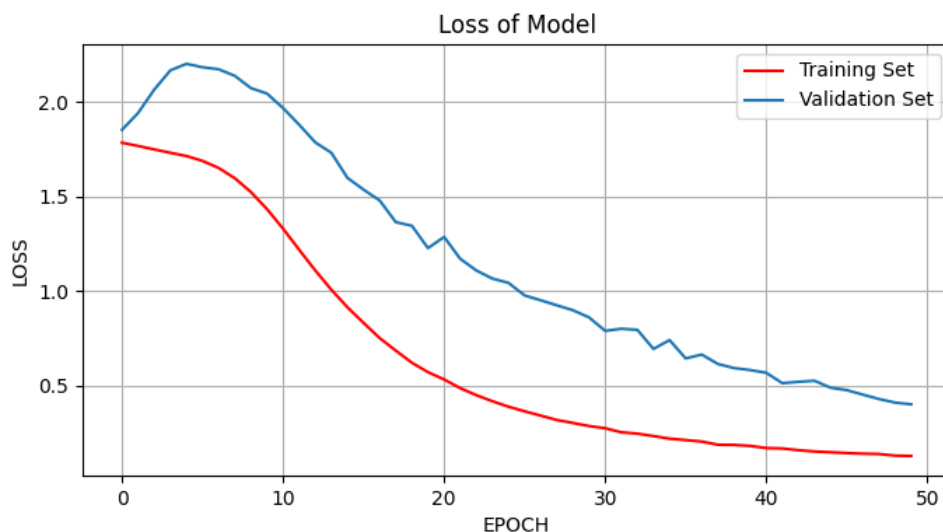
حال برای این مساله از تبدیل فوریه استفاده کردیم و نتایج زیر حاصل شد:

صحت مدل بصورت زیر می باشد:



شکل 56 صحت مدل شبکه LSTM با اعمال dropout و فریم زمانی 300 و اعمال تبدیل فوریه

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 57 خطا مدل شبکه LSTM با اعمال dropout و فریم زمانی 300 و اعمال تبدیل فوریه

زمان آموزش نیز بصورت زیر شد:

```
Training Time: 31.893112659454346
```

مقادیر Test Accuracy و Test Loss بصورت زیر شد:

```
27/27 [=====] - 0s 3ms/step - loss: 0.4547 - accuracy: 0.8238  
Test Loss, Test Accuracy: [0.4547199010848999, 0.8238095045089722]
```

همانطور که مشاهده می فرمایید تبدیل فوریه توانسته نتایج را به میزان زیادی بهبود بخشد. از نظر من این موضوع باید ریشه در شباهت فرکانسی مابین نوت های یکسان از سازهای گوناگون باشد. در واقع نُت ها اگرچه در قالب سیگنال زمانی توانسته اند شباهت نسبتاً خوبی را با پیش پردازش های قبلی از خود نشان دهند ولی به نظر می رسد که داده گان موجود در هر کلاس از نقطه نظر فرکانسی توانسته اند شباهت بیشتری از خود نشان دهند فلذا این جابجایی از مختصات زمانی به فرکانس به بهبودی نتایج و تقسیم بندی نُت ها و تشخیص بسیار بهر برای داده گان تست منجر شده است.

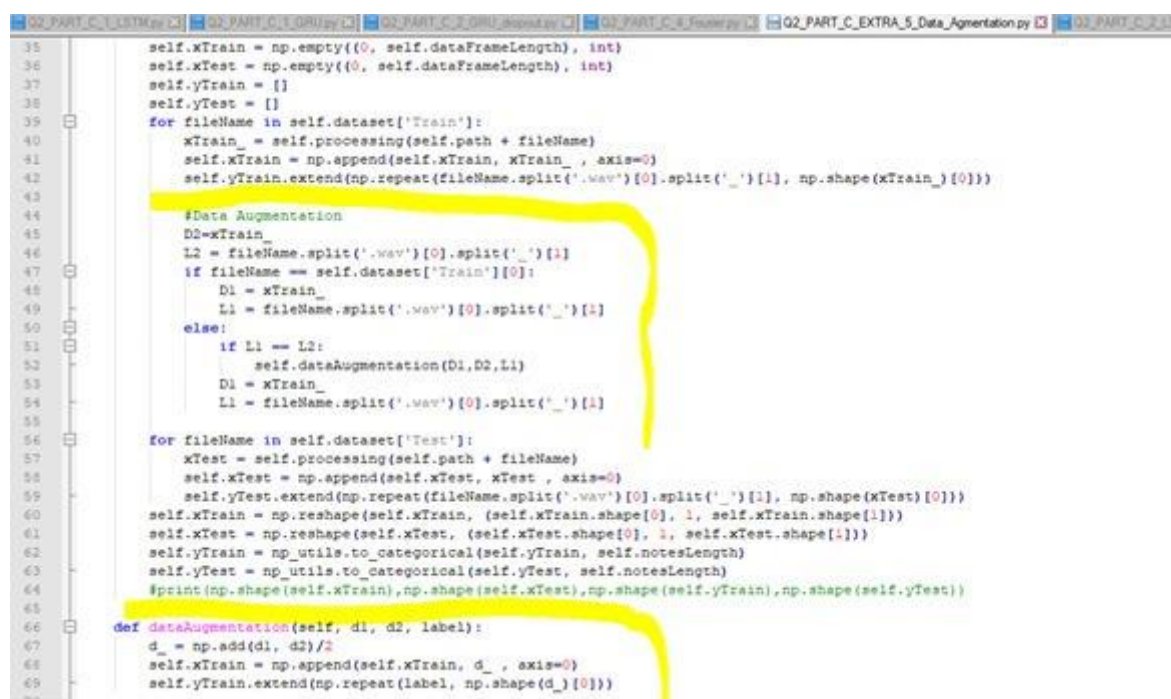
پاسخ بخش ه - امتیازی) برای این بخش روش هایی موجود است که به چند مورد می پردازیم:

❖ **روش اول:** برای data augmentation یک روش اینست که به ازای یک نت خاص سیگنال های دو instrument را با هم میانگین بگیریم و همینطور ادامه می دهیم و دو به دو میانگین می گیریم بدین صورت سمپل های جدید ساخته و ادغام آنها برای دو داده به یک ریکورد جدید می رسم. فلذا سیگنال های جدیدی را می سازیم، که منجر به اضافه نمودن دیتاهای جدید به دیتاست می شود.

❖ **روش دوم:** برای یک instrument خاص مختص یک نت بین دو سمپل اش از ابتدا تا انتها دو به دو میانگین می گیریم و سمپل های جدید بدست می آوریم که تعدادشان یکی کمتر از تعداد سمپل های اولیه مان می شود به همین خاطر اولین سمپل را به آنها اضافه می کنیم و دیتاهای جدیدی را به دیتاست اضافه می کنیم.

در این بخش مشابه با آنچه در روش یک شرح دادیم عمل می کنیم. برای وضوح بیشتر در ضمیمه ی توضیح روش یک، کد زده شده برای این بخش را در گزارش خدمتتان ارائه می کنیم هر چند که تمامی کدها در فایل Codes موجود است:

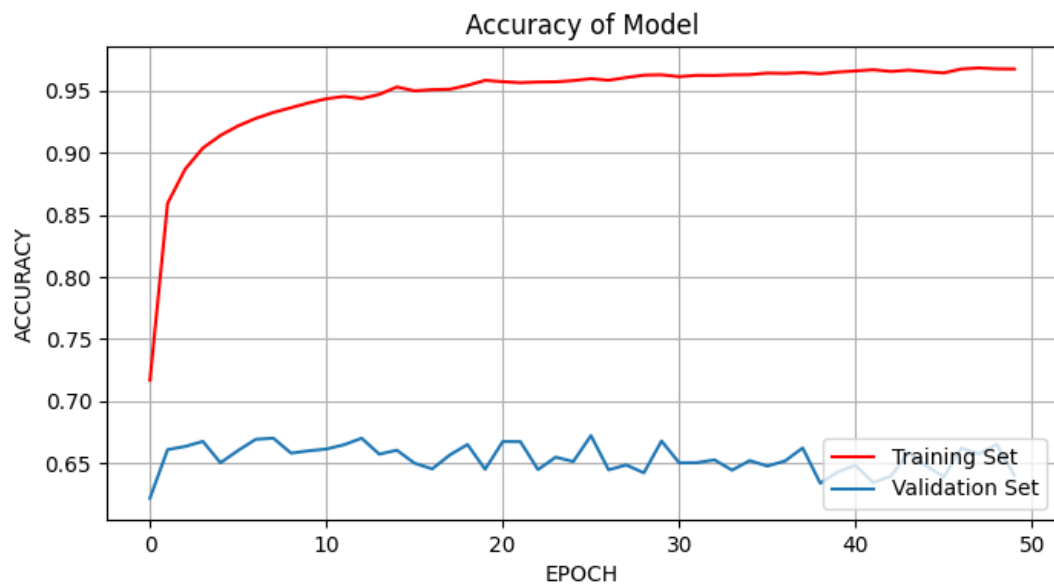
کد های این بخش بصورت زیر می باشد:



```
35 self.xTrain = np.empty((0, self.dataFrameLength), int)
36 self.xTest = np.empty((0, self.dataFrameLength), int)
37 self.yTrain = []
38 self.yTest = []
39 for fileName in self.dataset['Train']:
40     xTrain_ = self.processing(self.path + fileName)
41     self.xTrain = np.append(self.xTrain, xTrain_, axis=0)
42     self.yTrain.extend(np.repeat(fileName.split('.')[0].split('.')[1], np.shape(xTrain_)[0]))
43
44 #Data Augmentation
45 D2=xTrain_
46 L2 = fileName.split('.')[0].split('.')[1]
47 if fileName == self.dataset['Train'][0]:
48     D1 = xTrain_
49     L1 = fileName.split('.')[0].split('.')[1]
50 else:
51     if L1 == L2:
52         self.dataAugmentation(D1,D2,L1)
53     D1 = xTrain_
54     L1 = fileName.split('.')[0].split('.')[1]
55
56 for fileName in self.dataset['Test']:
57     xTest = self.processing(self.path + fileName)
58     self.xTest = np.append(self.xTest, xTest, axis=0)
59     self.yTest.extend(np.repeat(fileName.split('.')[0].split('.')[1], np.shape(xTest)[0]))
60 self.xTrain = np.reshape(self.xTrain, (self.xTrain.shape[0], 1, self.xTrain.shape[1]))
61 self.xTest = np.reshape(self.xTest, (self.xTest.shape[0], 1, self.xTest.shape[1]))
62 self.yTrain = np_utils.to_categorical(self.yTrain, self.notesLength)
63 self.yTest = np_utils.to_categorical(self.yTest, self.notesLength)
64 #print(np.shape(self.xTrain),np.shape(self.xTest),np.shape(self.yTrain),np.shape(self.yTest))
65
66 def dataAugmentation(self, d1, d2, label):
67     d_ = np.add(d1, d2)/2
68     self.xTrain = np.append(self.xTrain, d_, axis=0)
69     self.yTrain.extend(np.repeat(label, np.shape(d_)[0]))
```

شکل 58 کد های بخش Data Augmentation بر روی داده ها

صحت مدل بصورت زیر می باشد:



شکل 59 صحت مدل شبکه LSTM با اعمال dropout و فریم زمانی 300 و اعمال data augmentation

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 60 خطا مدل شبکه LSTM با اعمال dropout و فریم زمانی 300 و اعمال data augmentation

زمان آموزش نیز بصورت زیر شد:

Training Time: 277.37434363365173

مقادیر Test Accuracy و Test Loss بصورت زیر شد:

```
525/525 [=====] - 3s 5ms/step - loss: 5.1408 - accuracy: 0.6415  
Test Loss, Test Accuracy: [5.140750885009766, 0.6415476202964783]
```

نتایج این بخش با بخش دوم قسمت پیاده‌سازی این سوال مقایسه شده است. همانطوری که مشاهده می‌فرمایید بهبود دو الی سه درصدی نتایج به طور کامل مشهود است. این برآمده از دیتا آگمنتیشن می‌باشد. در واقع ما مدل را تعمیم دادیم و خود را برای داده‌های ورودی جدید آماده نمودیم. طبیعتاً مدل توانسته خود را با داده‌های جدیدتر ورزیده و بهتر آموزش دهد فلذا انتظار این بهبودی کاملاً مطابق با انتظار است که در نتایج فوق نیز مشهود است.

سوال سوم

پاسخ بخش الف) در این سوال هدف ما طراحی یک شبکه عصبی بازگشتی است که با استفاده از مجموعه داده های LYRICS_DATASET که شامل اشعار تعدادی از موسیقی های برتر دنیا است، بتواند یک بیت شعر تولید کند. دیتاست ما بصورت زیر است:

| | Artist Name | Song Name | Lyrics |
|---|-----------------|------------------|--|
| 0 | Phoebe Bridgers | Motion Sickness | I hate you for what you did And I miss you li... |
| 1 | Phoebe Bridgers | Killer | Sometimes I think I'm a killer I scared you i... |
| 2 | Phoebe Bridgers | Georgia | Georgia, Georgia, I love your son And when he... |
| 3 | Phoebe Bridgers | Kyoto | Day off in Kyoto Got bored at the temple Look... |
| 4 | Phoebe Bridgers | Would You Rather | Playing "would you rather" When it comes to f... |

شکل 61 سطرهای ابتدایی دیتاست

این دیتاست شامل 3 ستون و 347 سطر است که ستون های آن شامل Artist Name و Song Name و Lyrics می باشد.

که ستون Artist Name شامل نام شاعر مربوط به آن بیت می باشد.

ستون Song Name شامل نام آهنگ می باشد.

ستون Lyrics شامل متن شعر می باشد.

در این دیتاست تعداد کاراکتر های یکتا برابر با 70 می باشد.

| Shape of The Dataset | Length of Unique Characters |
|----------------------|-----------------------------|
| (347, 3) | 70 |

شکل 62 ابعاد دیتاست و تعداد کاراکتر های یکتا

سپس در مرحله بعد Corpus را تشکیل می دهیم که می خواهیم شامل تمام Lyrics ها باشد. در این مرحله که تمام Lyrics ها را فراخوانی می کنیم، پی میبریم که یک سطر miss value داریم. پس به همین خاطر شرطی می گذاریم که در صورت string نبودن Lyrics، آنرا به Corpus اضافه نکنیم. پس از این مرحله تمام Corpus را lower case می کنیم و با آن کار می کنیم.

پس از این کار کاراکتر های واحد را پرینت کردیم که بصورت زیر شدند:

The Unique Characters: [' ', '!', '"', '&', '(', ')', '*', ',', '-', '.', '/', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ':', ';', '?', ']', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '{', '\x93', 'ı', '£', '"', 'á', 'é', 'í', 'ó', '\u2005', '—', '‘', '’', '“', '”', '...', '\u205f']

پس در صدد حذف برخی از این کاراکتر ها شدیم که بی معنی و بدون کاربرد هستند.

حال می بینیم که کاراکتر های واحد نهایی که پس از حذف کاراکتر های غیر ضروری، نگه می داریم بصورت زیر هستند:

The Unique Characters Corpus after remove unimportant symbols: [' ', '"', '"', ',', '.', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ':', ';', '?', ']', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

می بینیم که پس از پیش پردازش های لازم طول کل کاراکتر ها برابر با 419781 و تعداد کاراکترهای واحدمان برابر با 45 عدد است:

| Length of All Characters After Remove Unimportant Symbols | Length of Unique Characters After Remove Unimportant Symbols |
|---|--|
|---|--|

شکل 63 تعداد کل کاراکتر ها در Corpus و تعداد کاراکتر های یکتا پس از پیش پردازش داده ها

در این مساله شبکه عصبی بازگشتی LSTM مان بصورت زیر طراحی می کنیم:

activation function: softmax

optimizer = Adamax(learning_rate = 0.015)

```
Model: "sequential_2"
```

| Layer (type) | Output Shape | Param # |
|-----------------|--------------|---------|
| lstm_2 (LSTM) | (None, 300) | 362400 |
| dense_2 (Dense) | (None, 45) | 13545 |

```
=====  
Total params: 375,945  
Trainable params: 375,945  
Non-trainable params: 0
```

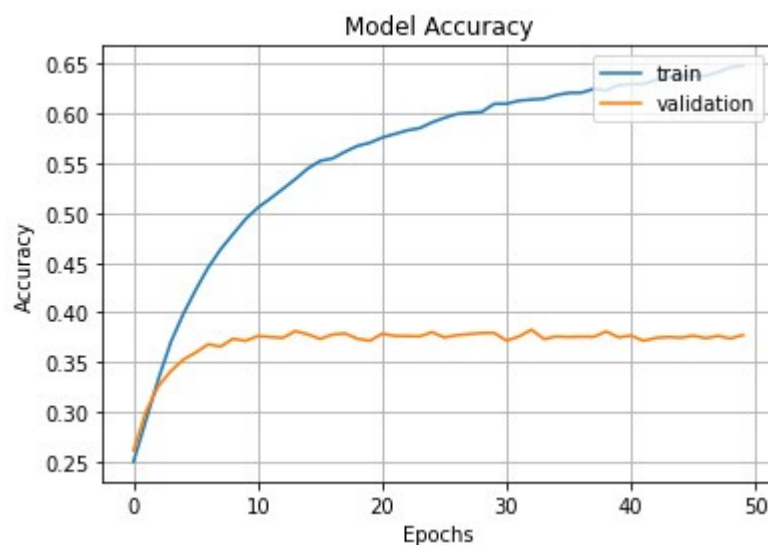
شکل 64 ساختار شبکه انتخابی

Loss و Accuracy را برای تعداد 50 اپاک نیز رسم می کنیم.

loss function: categorical_crossentropy

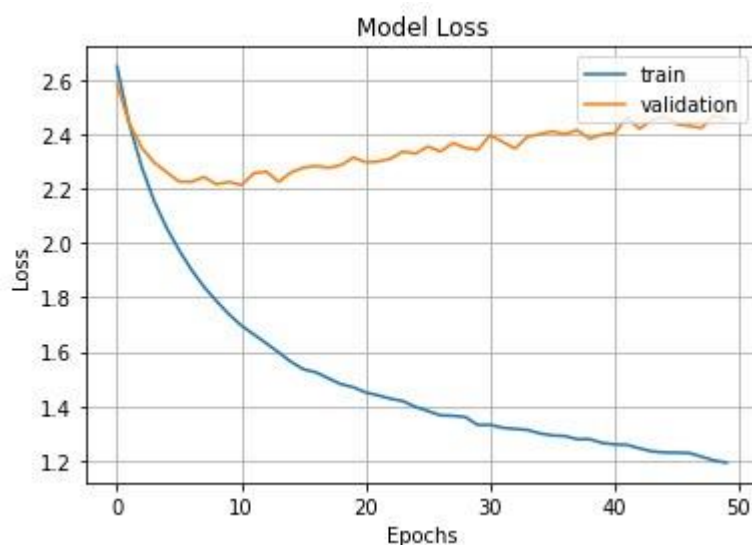
نتایج بصورت زیر حاصل می شود:

نمودار صحت:



شکل 65 صحت مدل شبکه

نمودار خطا:



شکل 66 خطا مدل شبکه

بیت خروجی شبکه به همراه ترجمه در گوگل ترنسلیت:

google translate

×

🔍

×

poem it's only
been three
weeks and a bag
of dartinn that
would be conle
ae in my beb i
sake i seelldre
your heart and
the sime if you
fanl in love with
you and i don't
wanna be the
reased to lo in
the dlolds and i
know these's a
qigeting of t

شعر فقط سه هفته گذشته است و
یک کیسه دارتین که می شود به
خاطر من اگر عاشق تو شوی و
من نمی خواهم برای دیدن در
دلهره ها خوشحال شوم و دلت را
بفروشم من می دانم که این یک
گیجت تی است

شکل 67 بیت خروجی شبکه به همراه ترجمه آن

پیش پردازش های انجام شده:

Lower case کردن همه حروف (تبدیل همه حروف به حروف کوچک): کلمات متناسب با قرارگیری در ابتدا وسط یا انتهای جمله و یا حتی متناسب با قرارگیری در موقعیت های خاصی در اشعار می توانند بصورت Lower case یا Upper case ظاهر شوند، این موضوع در اشعار که متونی بصورت کاملاً grammatical نیست بطور واضح مشهود است. لذا ما کلمات را متون را از این نقطه نظر یکپارچه سازی کرده و با استفاده از دستور lowercase، تمامی این کاراکترها را بصورت حرف کوچک لحاظ کردیم.

حذف کاراکتر های بی معنا و بدون استفاده: ما در ابتدا کاراکتر های منحصر بفرد را استخراج کرده که به تعداد 70 بودند و پس از حذف کاراکتر ها به تعداد 45 می رسد.

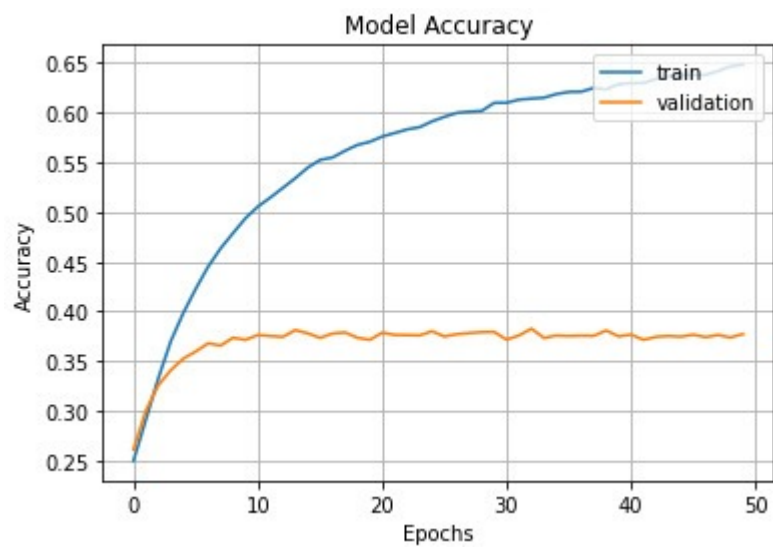
لحاظ کردن missing value ها: در ابتدا missing value ها را حذف می کنیم. پس در صدد حذف برخی از این کاراکتر ها شدیم که بی معنی و بدون کاربرد هستند.

انجام چنین پردازش هایی باعث می شود کرولیشن بین دادگان تا حد خوبی از بین رفته و نیز اطلاعات مفید موجود در هر ریکورد را نگه داشته و اطلاعات غیر مفید و غیر ضروری را تا حد قابل قبولی حذف نماید.

پاسخ بخش ب) در این مساله از سه تابع هزینه مختلف استفاده می کنیم و نتایج به شرح زیر است:

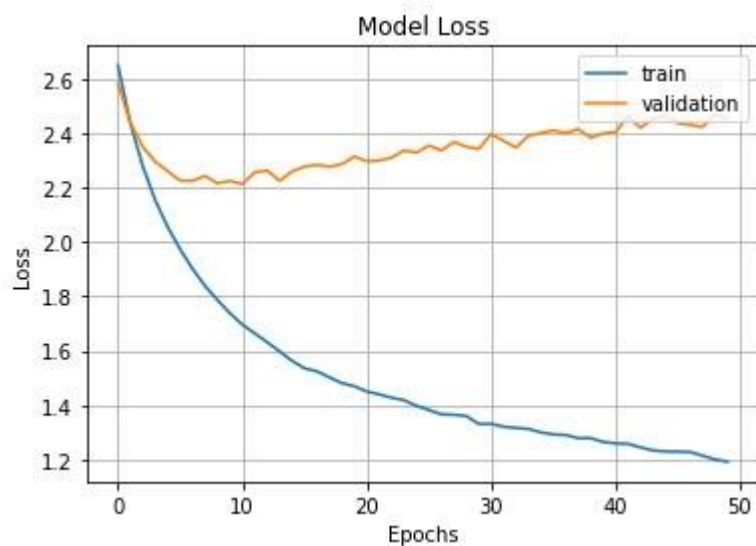
❖ الف) **loss function: categorical_crossentropy**

نمودار صحت:



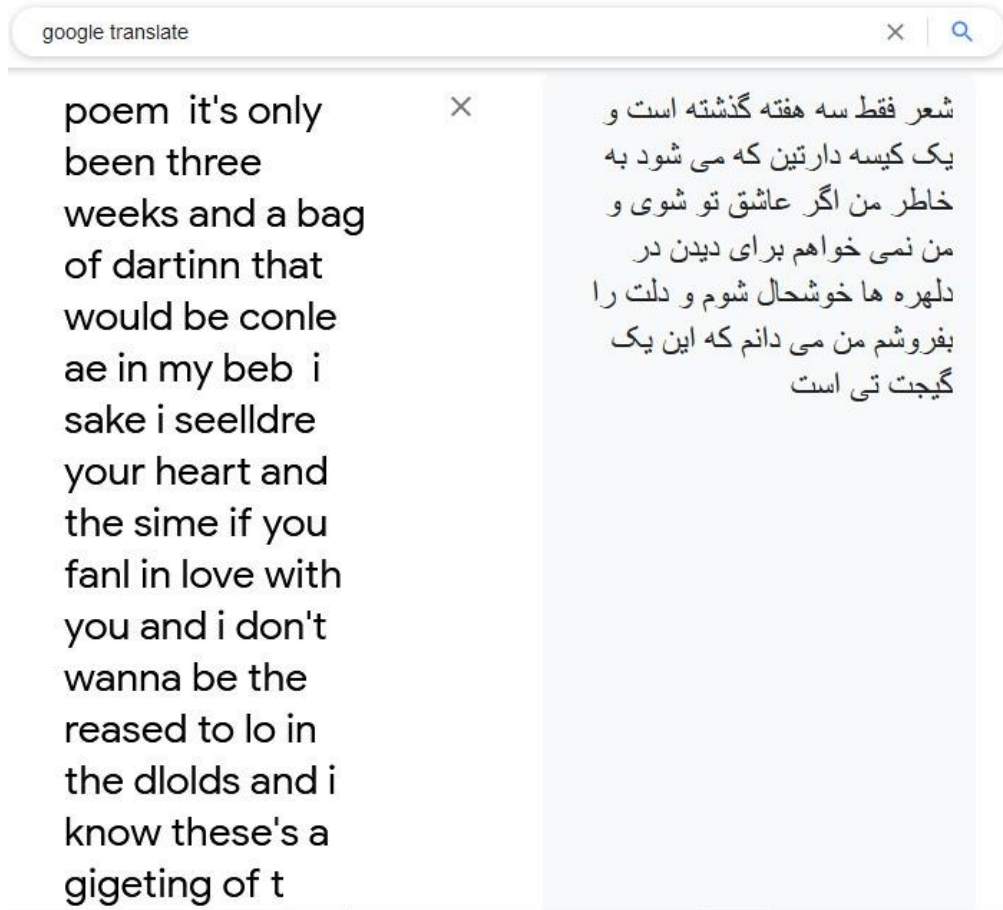
شکل 68 صحت مدل شبکه با `categorical_crossentropy` Loss function:

نمودار خطا:



شکل 69 خطا مدل شبکه با `categorical_crossentropy` Loss function:

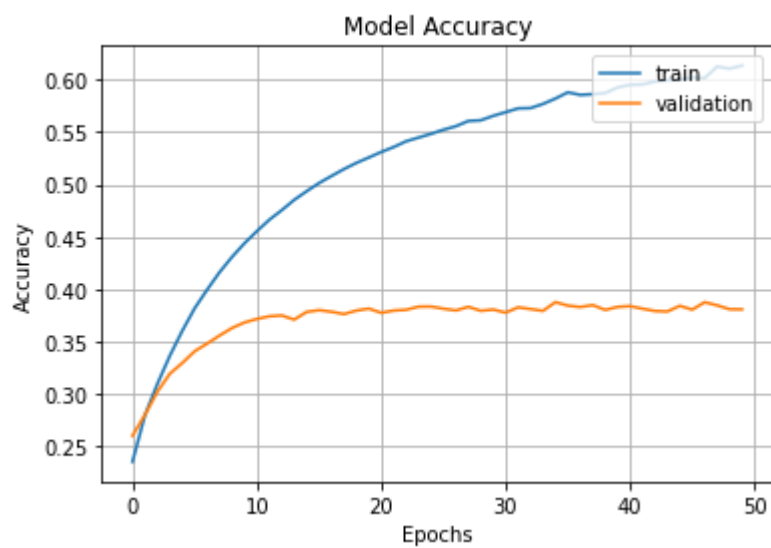
بیت خروجی شبکه به همراه ترجمه در گوگل ترنسلیت:



شکل 70 بیت خروجی شبکه به همراه ترجمه آن

❖ loss function: mse

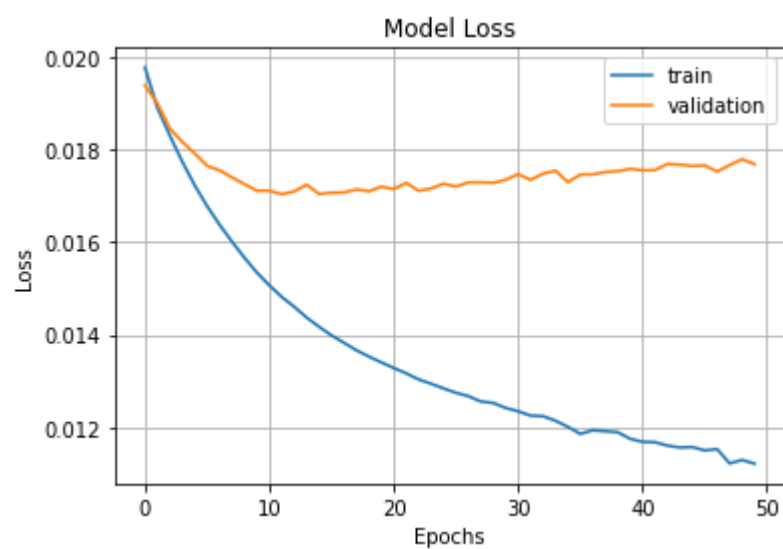
نمودار صحت:



شکل 71 صحت مدل شبکه با **Loss function: mse**

خطا:

نمودار



شکل 72 خطا مدل شبکه با **Loss function: mse**

بیت خروجی شبکه به همراه ترجمه در گوگل ترنسلیت:

English

↔

Persian

×

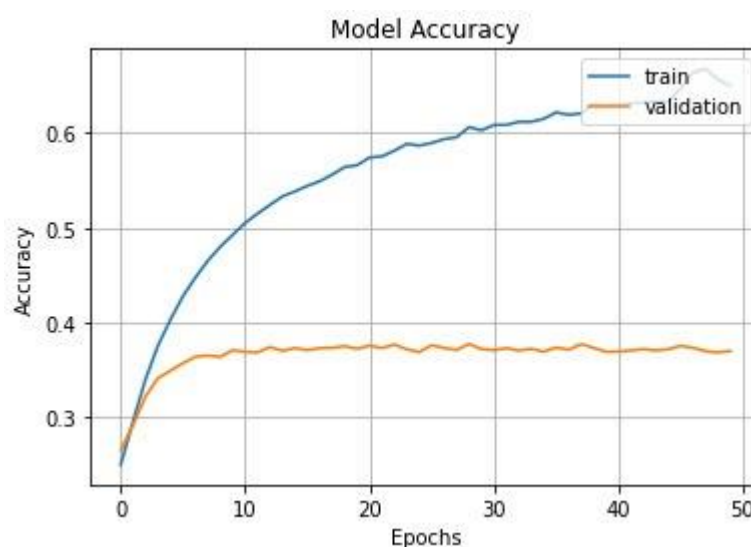
it's only been
three weeks and
a bag of me in
the shaton and
the coie bares all
the waits of to
the same and
when i was just a
whnl bnd the
shated sale mf to
call io the back of
the aay you leate
the world the
waits of the bay
you leateth

فقط سه هفته است که یک کیسه از
من در شاتون و کوی تمام
انتظارات را به هم می زند و
زمانی که من فقط یک لحظه از
فروش مشکوک بودم که به آیو
زنگ بزنم پشت آن ay تو دنیا را
انتظار می کشی از خلیجی که می
زنی

شکل 73 بیت خروجی شبکه به همراه ترجمه آن

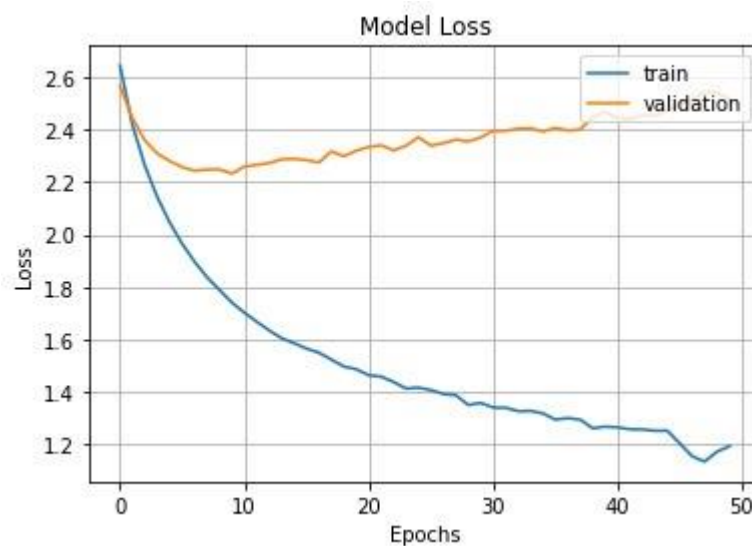
❖ **loss function: KLDivergence**

نمودار صحت:



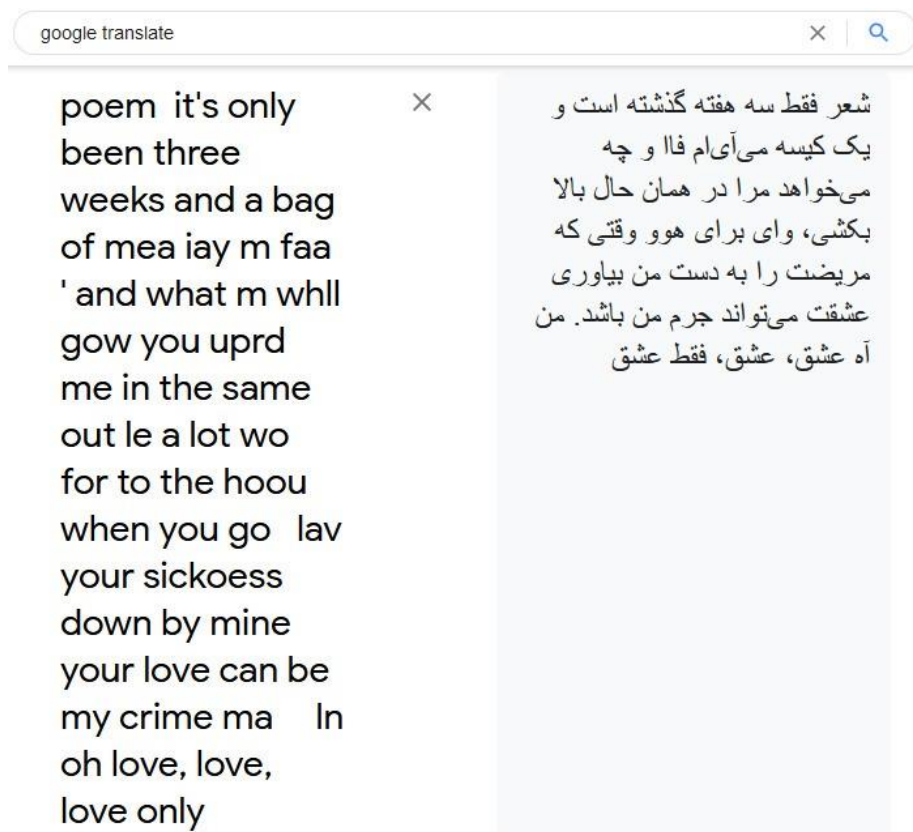
شکل 74 صحت مدل شبکه با Loss function: KLDivergence

نمودار خطا:



شکل 75 خطا مدل شبکه با Loss function: KLDivergence

بیت خروجی شبکه به همراه ترجمه در گوگل ترنسلیت:



شکل 76 بیت خروجی شبکه به همراه ترجمه آن

❖ در این مساله عملکرد مدل را یکبار برای 60 اپیاک و همان

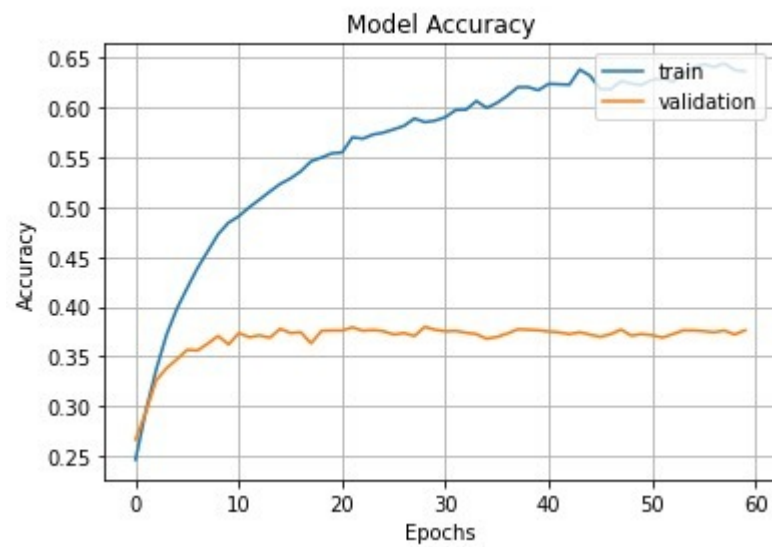
`optimizer = Adamax(learning_rate = 0.015)`

بررسی می‌کنیم و سپس با `optimizer` های متفاوت زیر عملکرد مدل را بررسی می‌کنیم و به نتایج زیر می‌رسیم:

(1) عملکرد مدل یکبار برای 60 اپیاک و همان `optimizer = Adamax(learning_rate = 0.015)`

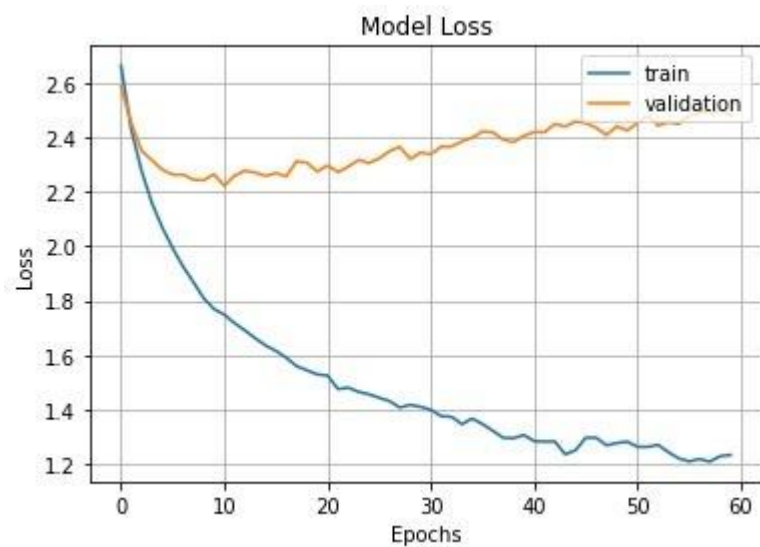
loss function: categorical_crossentropy

نمودار صحت:



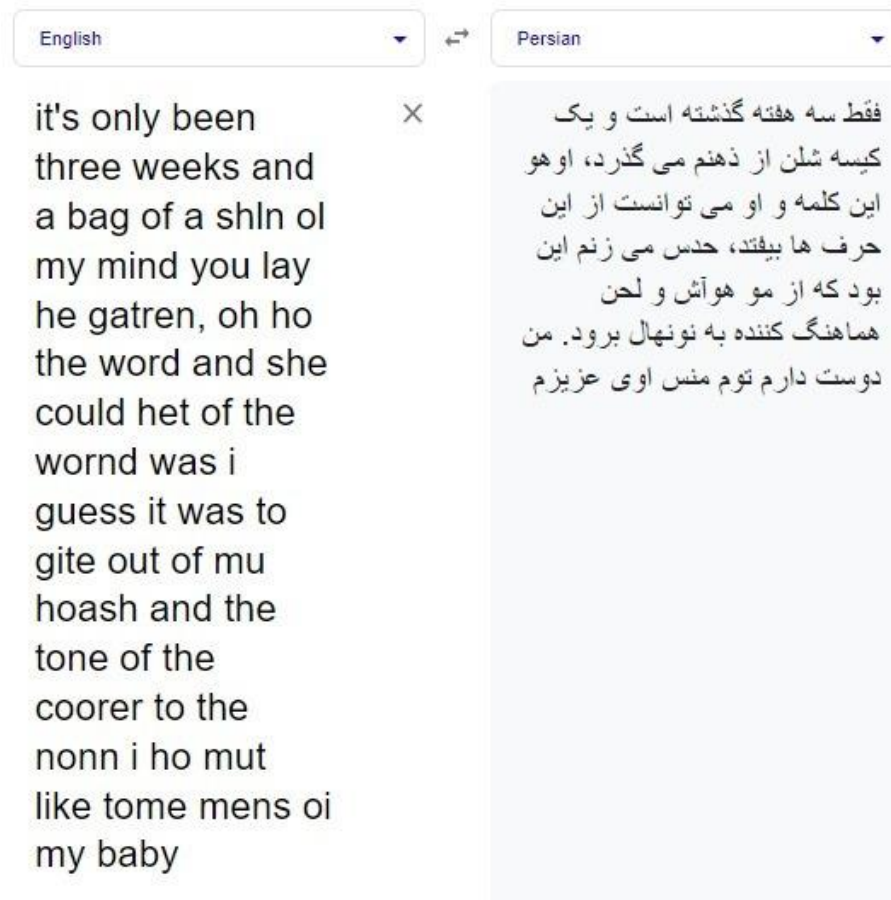
شکل 77 صحت مدل شبکه با **Loss function: categorical_crossentropy** و **optimizer: Adamax(Learning_rate=0.05)** برای 60 اپیاک

نمودار خطا:



شکل 78 خطا مدل شبکه با **Loss function: categorical_crossentropy** و **optimizer: Adamax(Learning_rate=0.05)** برای 60 اپیاک

بیت خروجی شبکه به همراه ترجمه در گوگل ترنسلیت:



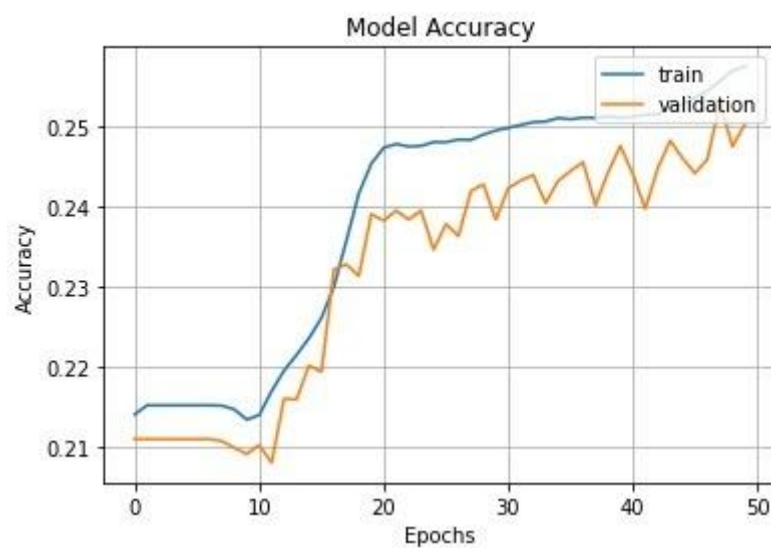
شکل 79 بیت خروجی شبکه به همراه ترجمه آن

می بینیم که با تعداد ایتريشن 60 تا که بیشتر از قبلی که 50 تا بود عملکرد مدل بهتر از همه موارد قبل است. و accuracy تا حدود 0.65 بالا می رود و loss نیز به مقدار قابل توجهی کم می شود.

2) عملکرد مدل برای `optimizer = SGD` و `loss function: categorical_crossentropy`

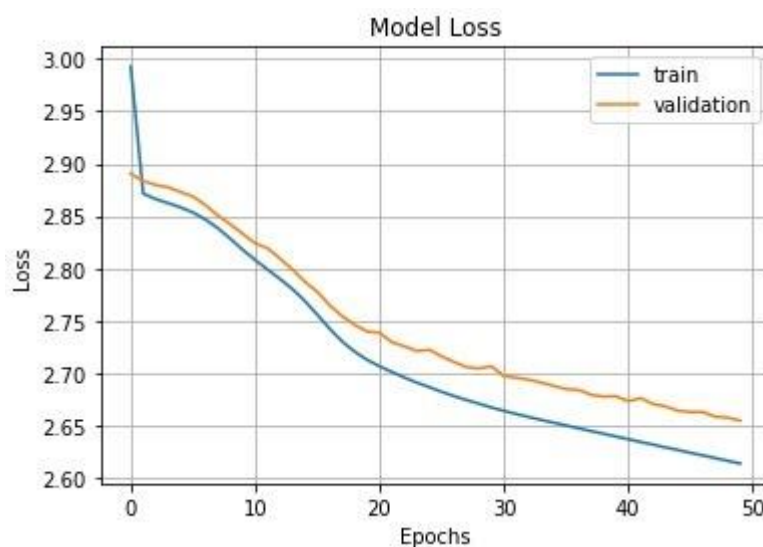
در این مرحله optimizer خود را به SGD تغییر می دهیم تا عملکرد مدل را بررسی کنیم:

نمودار صحت:



شکل 80 صحت مدل شبکه با `optimizer: SGD` و `Loss function: categorical_crossentropy`

نمودار خط:



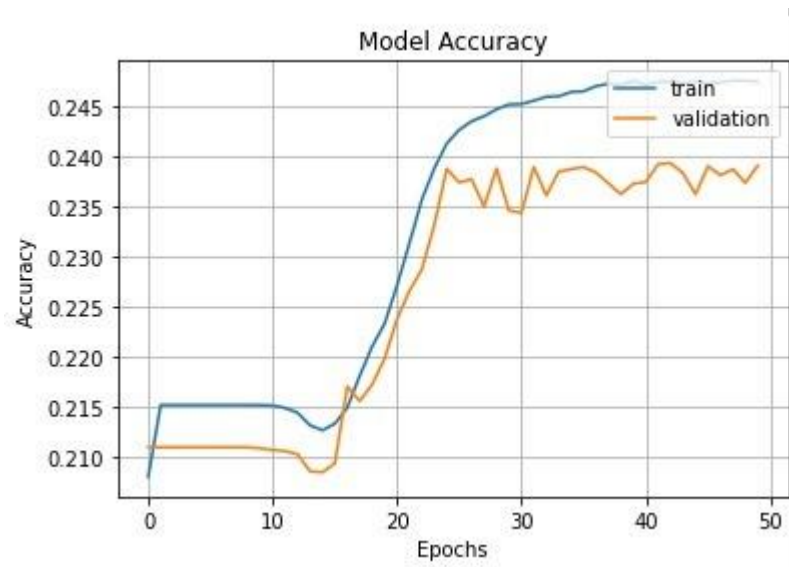
شکل 81 خطا مدل شبکه با `optimizer: SGD` و `Loss function: categorical_crossentropy`

حال می بینیم که نتایج نسبت به مرحله قبل که بهتر نشد.

(3) عملکرد مدل برای `optimizer = adadelta` و `loss function: categorical_crossentropy`

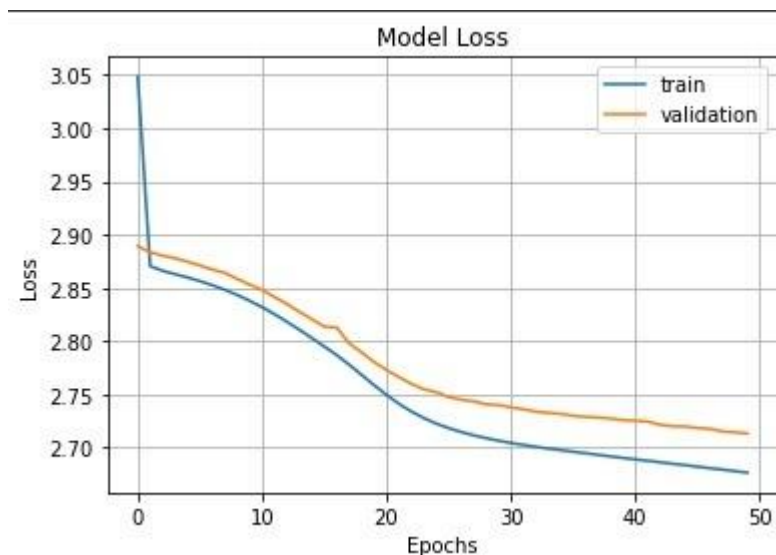
در این مرحله `optimizer` خود را به `adadelta` تغییر می دهیم تا عملکرد مدل را بررسی کنیم:

نمودار صحت:



شکل 82 صحت مدل شبکه با `optimizer: adadelta` و `Loss function: categorical_crossentropy`

نمودار خطا:



شکل 83 خطا مدل شبکه با `optimizer: adadelta` و `Loss function: categorical_crossentropy`

با توجه به نتایج بدست آمده در این مساله می بینیم که بهترین عملکرد این شبکه با

loss function: categorical_crossentropy

optimizer: Adamax(learning_rate = 0.015)

بدست آمده است که مشاهده کردیم که مقدار صحت به مقدار حدود 0.65 می رسد و برای این مساله خروجی بسیار مطلوبی است. شعر بدست آمده از شبکه بصورت معقول است و جمله بندی مرتب و بصورت معنی داری است.

دیدیم که با توابع هزینه متفاوت mse و KLDivergence نیز تقریباً مقادیر صحت و خطا تا حدودی مشابه نتایج گفته شده می شود و خروجی شعر نیز خیلی معقول است که نشان دهنده خوب بودن عملکرد شبکه است. اما وقتی optimizer را از adamax به adadelta و sgd تبدیل می کنیم می بینیم که مقادیر صحت به اندازه adamax بالا نمی رود و مقدار Loss به اندازه adamax پایین نمی رود. پس در نتیجه می بینیم که بهترین مدل حاصل شده در این سوال همان مدل شبکه با

loss function: categorical_crossentropy

optimizer: Adamax(learning_rate = 0.015)

می باشد.

پاسخ بخش پ) با توجه به ابیات مختلف تولید شده در مرحله قبل در خروجی شبکه دیدیم که ابیات بسیار پر محتوا و کامل هستند اما با توجه به ران کردن چندین و چند باره و با loss function ها و optimizer های مختلف میتوان پی برد که اولین کلمات تولید شده معنادارتر و بیشتر به پیکره ی شعر ارتباط دارند و بسیار منظم تر و پر معنی هستند و پی می بریم که هر چه جلوتر می رویم و به آخر بیت نزدیک می شویم به کلماتی بر می خوریم که شاید ارتباط کمتری با پیکره ی اصلی شعر دارند و یا شاید مفهوم خاصی نداشته باشند. دلیل این مساله همان حالت recurrent بودن شبکه است که به عقب بر می گردد و ترین می شود و از کلمات قبلی استفاده می کند و همینطور ادامه پیدا می کند، که هر چه به انتهای بیت می رسیم چون به کلمات قبلی وابستگی دارد، باعث می شود که کلمات انتهایی نسبت به کلمات ابتدایی کم معنا تر شوند.

پاسخ بخش ت)

❖ **الگوریتمی به نام RTRL:** برای آموزش شبکه های عصبی fully recurrent نشان داده شده

است که RTRL دارای قدرت و عمومیت بالایی است، اما این مضرات این است که به زمان

محاسباتی زیادی نیاز دارد. این تکنیک برای کاهش مقدار محاسبات مورد به نام RTRL بدون تغییر اتصال شبکه ها است و این امر با تقسیم شبکه اصلی به زیرشبکه ها به منظور انتشار خطا در حالی که آنها را برای انتشار فعالیت تقسیم نمی کنند، انجام می شود. که در مقاله زیر این روش مفصلاً توضیح داده شده است:

Zipser, David. "A subgrouping strategy that reduces complexity and speeds up learning in recurrent networks." *Neural Computation* 1.4 (1989): 552-558.

❖ کاهش محاسبات در شبکه های recurrent با به روز رسانی انتخابی نورون های حالت:

یک رویکرد SA-RNN ، fully-learned approach، که هر RNN را با پیش بینی الگوهای به روز رسانی گسسته در دانه بندی دقیق ابعاد حالت پنهان مستقل از طریق پارامترسازی توزیع احتمالات به روز رسانی که به طور کامل توسط داده های ورودی هدایت می شود، تقویت می کند. این روش الگوهای به روز رسانی را به صورت آنلاین تطبیق می دهد، و اجازه می دهد ابعاد مختلف به صورت مشروط به ورودی به روز رسانی شوند. برای یادگیری اینکه کدام به روز رسانی شود، این مدل یک مسئله بهینه سازی چند هدفه را حل می کند و دقت را به حداکثر می رساند و در عین حال تعداد به روز رسانی ها را بر اساس یک کنترل یکپارچه به حداقل می رساند. علاوه بر این، این روش می تواند به طور مستقیم برای طیف گسترده ای از مدل های حاوی معماری RNN اعمال شود. در مقاله زیر به طور مفصل توضیح داده می شود:

Hartvigsen, Thomas, et al. "Reducing Computation in Recurrent Networks by Selectively Updating State Neurons." (2019).

❖ استفاده از GPU به جای LSTM: برای کاهش هزینه محاسباتی مدل که در اینجا مدل ما

LSTM است یک راه اینست که از GRU به جای LSTM استفاده کنیم، زیرا که این نوع سلول نسبت به LSTM یک دروازه (Gate) کمتر دارد و بنابراین از نظر محاسباتی کارآمدتر است. علاوه بر این، GPU ها نسبت به LSTM هایپر پارامترهای کمتری دارند، بنابراین فضای جستجوی ممکن، کوچکتر است.

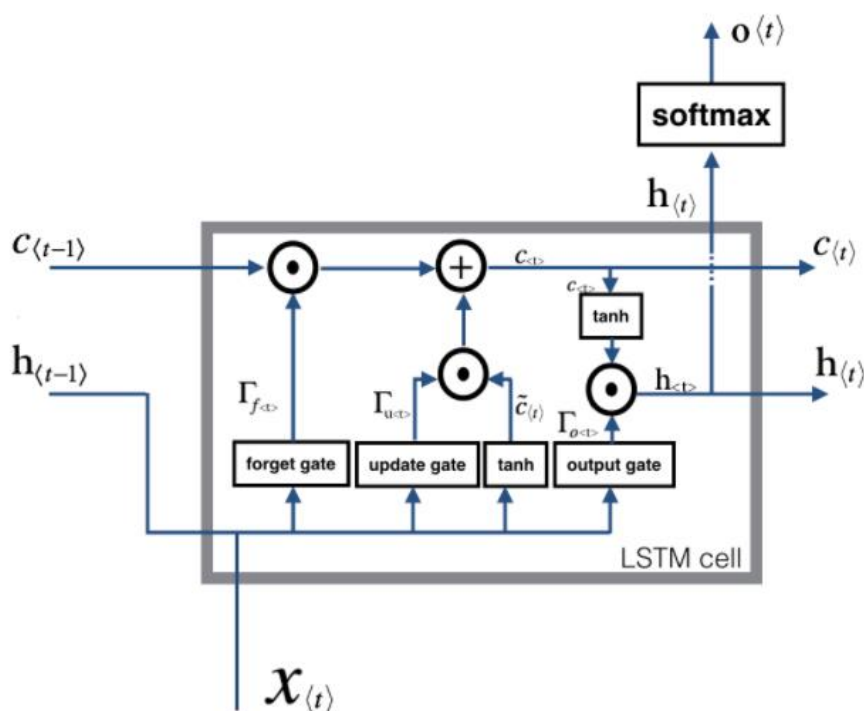
❖ استفاده از تعداد سلول کمتر و اضافه نمودن dropout: مسلماً هر چه تعداد سلول کمتر

باشد، پیچیدگی محاسباتی کاهش می یابد. لاکن برای جبران صحت کمتری که مدل ارائه می کند، در مراجعی ذکر شده بود که از dropout برای جبران میزان صحت کاهش یافته استفاده می کنند. لازم به ذکر است که پیچیدگی محاسباتی dropout در برابر پیچیدگی محاسباتی سلول ها قابل مقایسه نیست و این روش یکی از روش های رایج در حال حاضر است.

پاسخ بخش ث

سلول های عصبی LSTM شامل 3 دروازه (gate) می باشد، که یکی از این دروازه ها به نام دروازه فراموشی است که با Γ_f نمایش داده می شود. و وظیفه کنترل جریان اطلاعات از گام زمانی قبلی را دارد. این دروازه مشخص می کند که آیا اطلاعات حافظه از گام زمانی قبل مورد استفاده قرار گیرد یا خیر و اگر باید از گام زمانی قبل چیزی وارد شود به چه میزان باشد.

دروازه بروزرسانی یا همان Update gate که بصورت Γ_u نمایش داده شده است، وظیفه کنترل جریان اطلاعات جدید را بر عهده دارد. این دروازه مشخص میکند آیا در گام زمانی فعلی باید از اطلاعات جدید مورد استفاده قرار گیرد یا خیر و اگر بلی به چه میزان. از این دروازه عموماً به دروازه ورودی نیز یاد میشود. دروازه خروجی یا همان Output gate که بصورت Γ_o نمایش داده شده است، نیز مشخص میکند چه میزان از اطلاعات گام زمانی قبل با اطلاعات گام زمانی فعلی به گام زمانی بعد منتقل شود. وجود این دروازه ها به این شکل است که مکانیزم کنترلی بسیار دقیقی را ایجاد می کند.



شکل 84 یک بلوک LSTM

$$\hat{C}_t = \tanh(W_C.[h_{t-1}, X_t] + b_c)$$

$$C_t = \Gamma_f.C_{t-1} + \Gamma_u.\hat{C}_t$$

$$\Gamma_f = \sigma(W_f.[h_{t-1}, X_t] + b_f)$$

$$\Gamma_u = \sigma(W_u.[h_{t-1}, X_t] + b_u)$$

$$\Gamma_o = \sigma(W_o.[h_{t-1}, X_t] + b_o)$$

$$h_t = \Gamma_o.\tanh(C_t)$$

همان طور که در بالا مشاهده میفرمایید فرض کنید ما چند کلمه از یک بیت را از ورودی می خوانیم و می خواهیم از یک LSTM برای چک و کنترل ساختار گرامری استفاده کنیم (مثلا ببینیم آیا فاعل مفرد است یا جمع است). اگر فاعل از مفرد به جمع تغییر پیدا کرد (یا بلعکس) ما باید راهی پیدا کنیم تا مقدار ذخیره شده قبلی در حافظه را با حالت جدید تعویض کنیم. در LSTM این کار از طریق دروازه forget بصورت زیر انجام میشود:

$$\Gamma_f = \sigma(W_f.[h_{t-1}, X_t] + b_f)$$

در اینجا W_f ماتریس وزنی است که رفتار دروازه فراموشی را کنترل می کند. اگر ما عملیات فوق را انجام دهیم چون از تابع فعالسازی سیگموید استفاده می کنیم نتیجه برداری بنام Γ_f خواهد بود که مقادیری بین ۰ و ۱ خواهد داشت. این بردار سپس در عبارت بعدی در C_{t-1} ضرب خواهد شد. بنابر این اگر مقادیر بردار دروازه فراموشی Γ_f صفر باشد (یا به سمت صفر میل کند) عملاً به معنای در نظر نگرفتن محتوای C_{t-1} است. به عبارت ساده تر یعنی شبکه اطلاعات ارائه شده توسط C_{t-1} را دور انداخته و هیچ توجهی به آن نمی کند. به همین صورت اگر مقادیر بردار Γ_f برابر با 1 باشد این اطلاعات توسط شبکه حفظ می شوند. مقادیر مابینی نیز موجب می شود شبکه به همان میزان از محتوای ارائه شده از گام زمانی قبل استفاده کند (یعنی بخشی را دور ریخته و از بخش دیگر استفاده کند).

که این عمل موجب عملکرد موثر این شبکه است.

با تشکر