|  |  |  |
| --- | --- | --- |
|  | به نام خدا |  |
| **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر**  **شبکه های عصبی و یادگیری عمیق**  **مینی پروژه سری دوم** | | |

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

­

**فهرست گزارش سوالات**

[سوال اول 3](#_Toc92681571)

[سوال دوم – تشخیص نُت موسیقی 18](#_Toc92681572)

[سوال سوم 48](#_Toc92681573)

# سوال اول

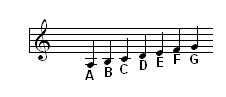
**قسمت اول: Load کردن دیتا برای مجموعه داده‌گان Chopin**

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:

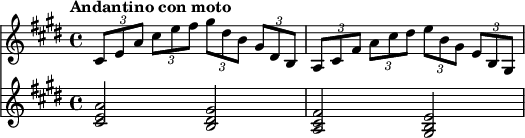


شکل 1 نمونه ای از نام برخی نُت ها

(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. نُت‌های کورد به‌جای هم‌زمان، یکی پس از دیگری به صدا در می‌آیند، 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 نُت ها را مشاهده کرد و در ردیف پایینی کوردهای ساخته شده از نُت های قطعه ای است که در قسمت بالایی نشان داده شده است:

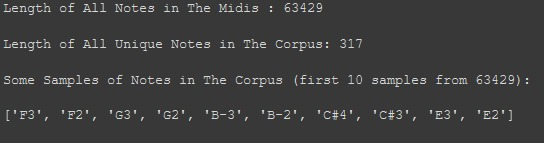


شکل 2 ردیف بالایی نُت ها و ردیف پایینی کوردهای ساخته شده از آن

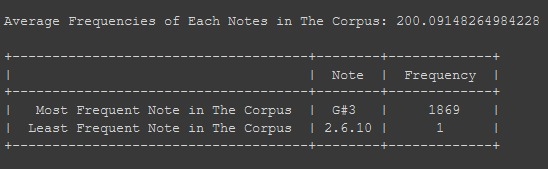
**قسمت دوم: پیش پردازش دیتا برای مجموعه داده‌گان Chopin**

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 chopin، برابر با 63429 است. و تعداد نُت های منحصر بفرد(غیر تکراری) برابر با 317 است.

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

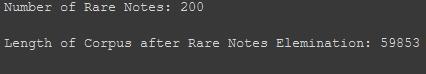


شکل 3 خروجی ترمینال شامل تعداد کل نُت ها و نُت های واحد و 10 تا نُت اول Corpus

و سپس تعداد تکرار های هر نُت را می شماریم که این مقدار تکرار از 1 بار تا 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:

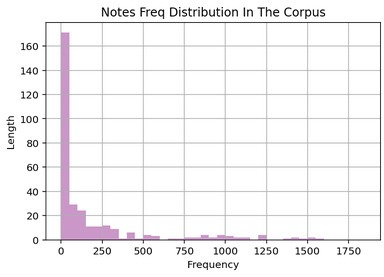
شکل 4 میانگین تعداد تکرار هر نُت و نُت با کمترین و بیشترین تکرار به همراه تعداد تکرار آنها

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:



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:



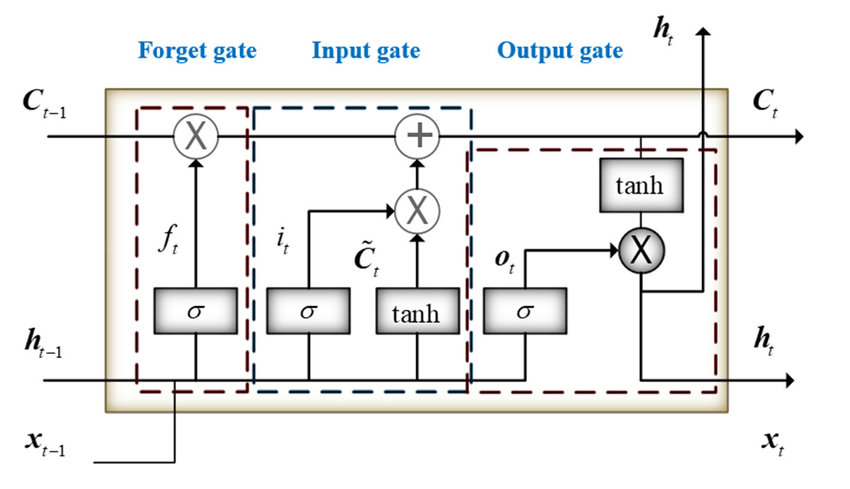
شکل 6 چگالی تعداد نُت ها بر حسب تعداد تکرارشان در Corpus

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 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.

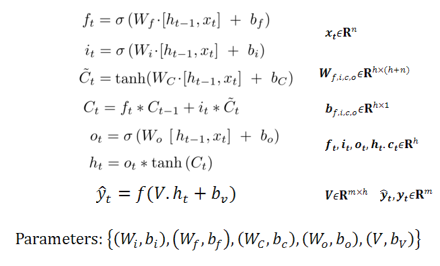
**قسمت سوم: پیاده سازی شبکه عصبی بازگشتی برای مجموعه داده‌گان Chopin**

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 می‌دهد دارای ۳ دروازه (گیت) است به نام‌های دروازه ورودی () و خروجی () و دروازه فراموشی () که دروازه فراموشی تصمیم می گیرد که اطلاعاتی از حالت سلول قبلی را نگه دارد یا دور بریزد. معادلات ما بصورت زیر هستند:



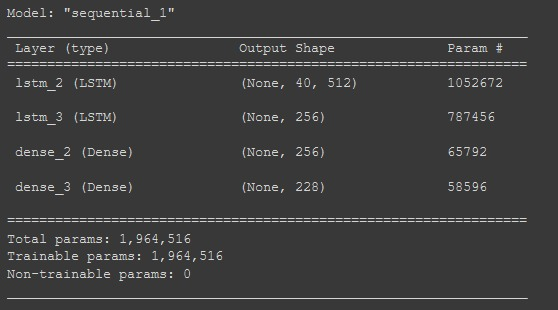
شبکه عصبی بازگشتی ما از دو لایه LSTM ساخته می شود، که مدل آن در زیر آورده شده است:

loss function: categorical\_crossentropy

activation function: softmax

optimizer = Adamax(learning\_rate = 0.01)

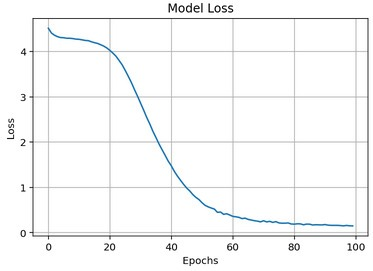
خروجی ترمینال که مدل ساخته شده با دستور summary را به نمایش می‌گذارد، خدمت‌تان ارائه می‌کنیم.



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 Fulse 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 Fulse قرار می دهیم.

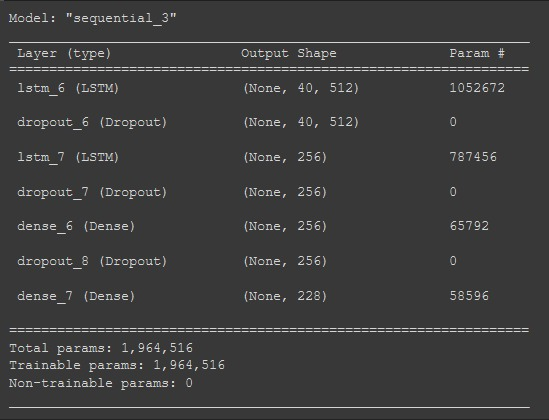
شبکه عصبی بازگشتی LSTM مان بصورت زیر می شود و نتایج این اضافه کردن dropout بصورت زیر حاصل می شوند:

loss function: categorical\_crossentropy

activation function: softmax

optimizer = Adamax(learning\_rate = 0.01)

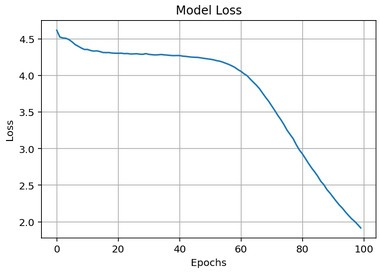
خروجی ترمینال که مدل ساخته شده با دستور summary را به نمایش می‌گذارد، خدمت‌تان ارائه می‌کنیم.



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:

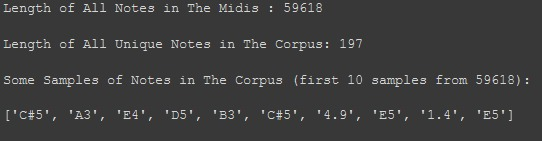
**قسمت اول: Load کردن دیتا برای مجموعه داده‌گان Mozart**

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.

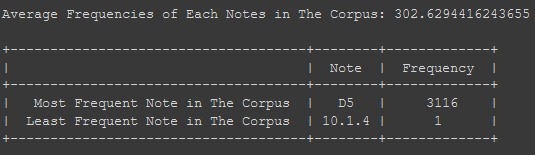
برای Load کردن دیتا آهنگ های این هنرمند را از پوشه به نام آن با استفاده از کتابخانه music21 می خوانیم. و سپس note ها و chord های موجود در هر آهنگ را بدست می آوریم.

**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.

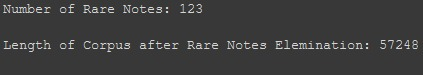
و در نهایت 10 تا نُت ابتدایی موجود در corpus را در زیر می بینیم:

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:

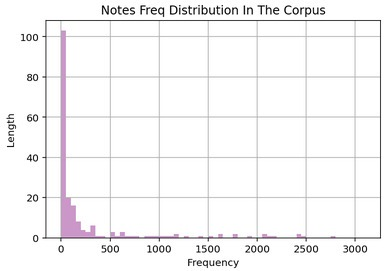
شکل 13 میانگین تعداد تکرار هر نُت و نُت با کمترین و بیشترین تکرار به همراه تعداد تکرار آنها

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:



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.

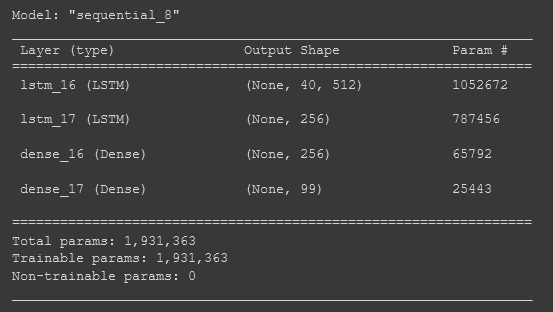
**قسمت سوم: پیاده سازی شبکه عصبی بازگشتی برای مجموعه داده‌گان Mozart**

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

loss function: categorical\_crossentropy

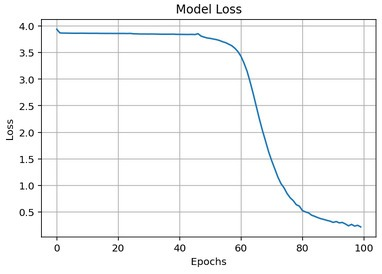
activation function: softmax

optimizer = Adamax(learning\_rate = 0.01)



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:

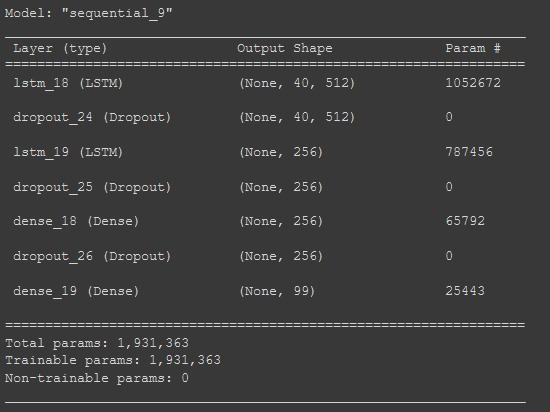
**قسمت سوم: پیاده سازی شبکه عصبی بازگشتی برای مجموعه داده‌گان Mozart به همراه Dropout**

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

loss function: categorical\_crossentropy

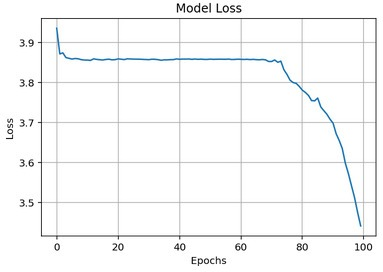
activation function: softmax

optimizer = Adamax(learning\_rate = 0.01)



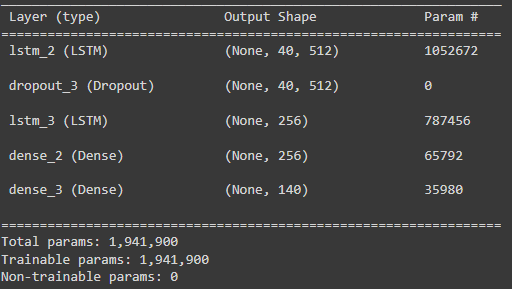
شکل 18 ساختار انتخابی شبکه عصبی بازگشتی با اضافه کردن لایه های dropout پس از هر لایه

**قسمت چهارم: ارزیابی شبکه برای مجموعه داده‌گان Mozart به همراه Dropout**

مقدار loss را برای فرآیند بصورت زیر داریم:

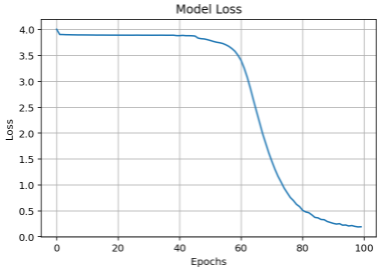
شکل 19 نمودار Loss برای آهنگ های mozart پس از اضافه کردن لایخ dropout پس از هر لایه

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:

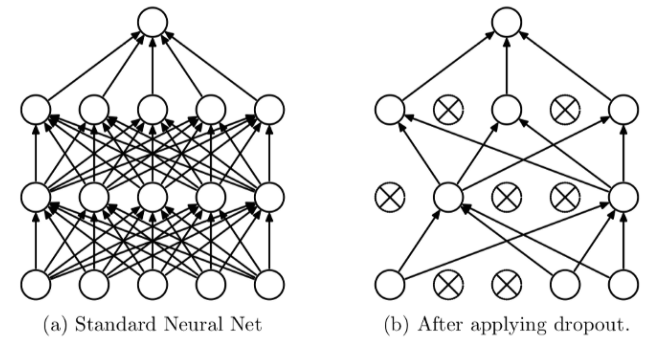
شکل 20 ساختار انتخابی شبکه عصبی بازگشتی با اضافه کردن یک لایه dropout پس از لایه اول برای آهنگ های mozart

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 لایه‌ها می‌تواند کارایی کمتری نسبت به قرارگیری تنها یک dropout بعد لایه‌ی مناسب در شبکه باشد که در نتایج فوق مشهود است.

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

شکل 22 یک مثال از اضافه کردن Dropout به شبکه

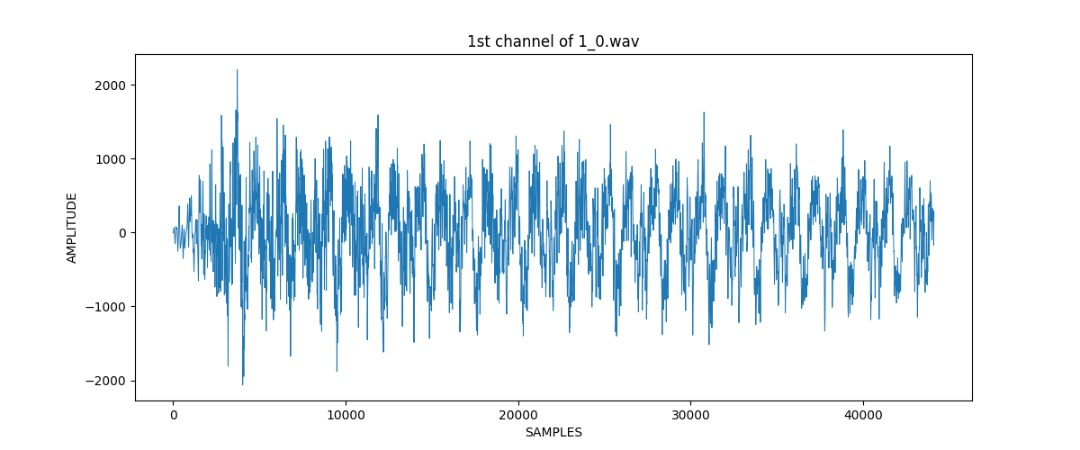
منظم سازی(regularization): مقدار خطایی که شبکه های عصبی روی داده های آموزشی بدست می آورند بسیار کمتر از خطای آنها روی داده های آزمایشی است. به همین دلیل در معرض بیش برازش قرار می گیرند. برای مقابله با این مساله، از تکنیک های منظم سازی استفاده می شود که گنجایش مدل را کاهش دهیم. بدین ترتیب تفاوت بین خطای 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 نمودنِ dropout با مقدار 0.05 و برای یک لایه کاملا 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 ملاحضه 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 ساختارِ 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 عمد 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 طبیعتا کار درستی نیست لذا برای ساختار شبکه‌ای 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 می‌پردازد.

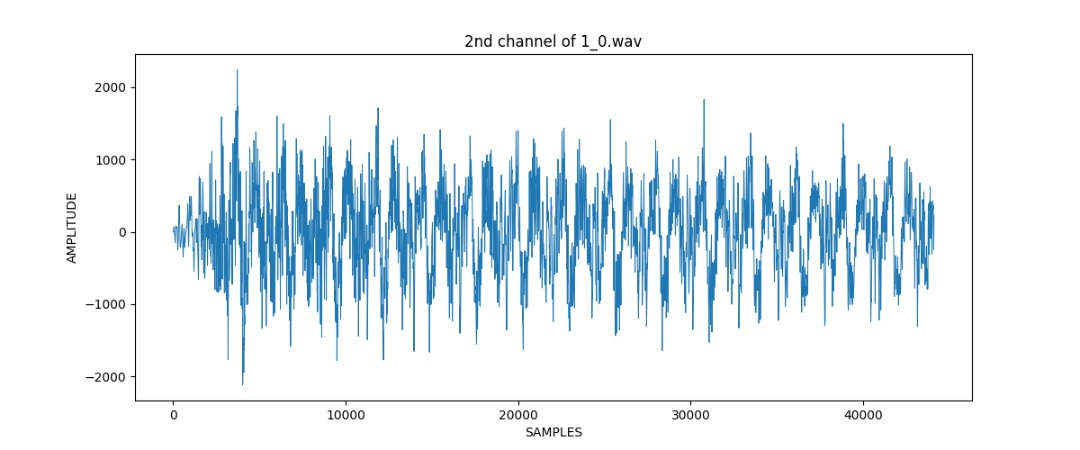
# سوال دوم – تشخیص نُت موسیقی

**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:

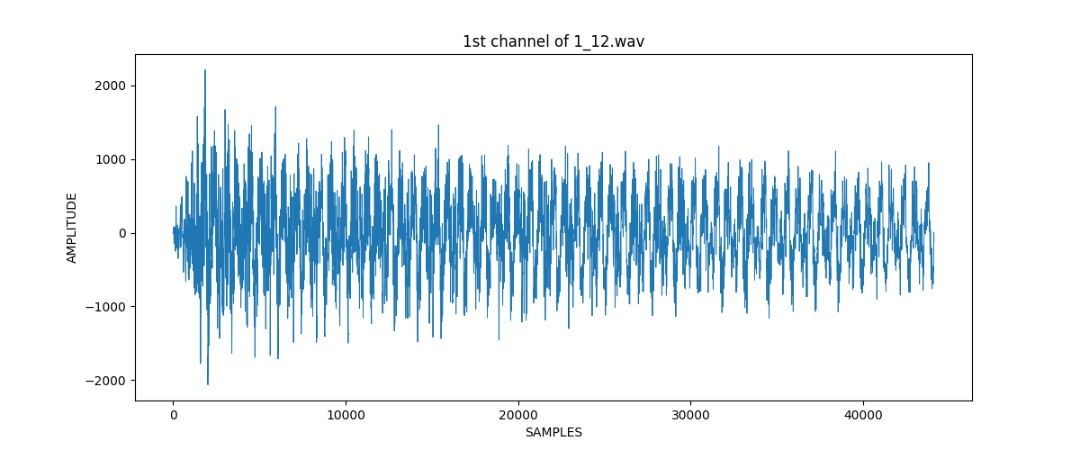
شکل 23 سیگنال 1\_0 برای کانال اول

حال نتایج را برای کانال دوم بررسی می‌کنیم.

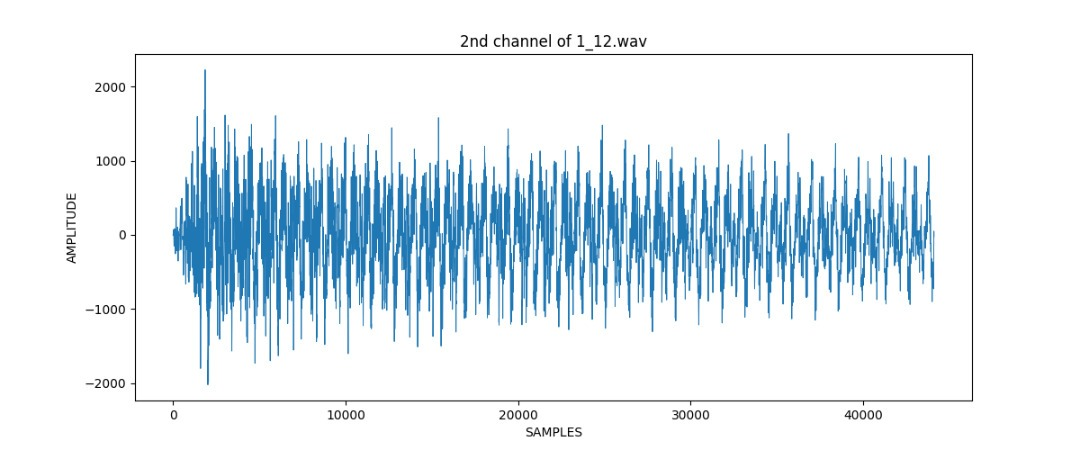


شکل 24 سیگنال 1\_0 برای کانال دوم

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

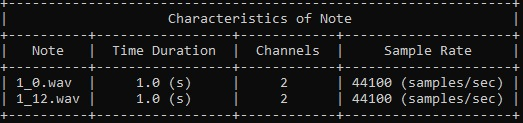


شکل 25 سیگنال 1\_12 برای کانال اول

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

شکل 26 سیگنال 1\_12 برای کانال دوم

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



شکل 27 اطلاعات مربوط به سیگنال های 1\_0 و 1\_12

حال به موارد خواسته شده در صورت سوال می‌پردازیم.

* **شباهت ها:**

رنج دامنه ها در یک instrument تقریبا مشابه هم هستند و می بینیم که در این instrument رنج دامنه بین [-2000,2000] است.

در نمونه های اولیه (در زمانهای اولیه) پوش منحنی صعودی و سپس یک روند نزولی به خود می گیرد.

* **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 آمدیم و لیستی از نام فایلها تهییه 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 سایز 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 نهایتا 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 نیاورد 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 پیاده‌سازی چند نکته را ذکر می‌کنم. متاسفانه سخت افزار اینجانب و هم گروهی من بسیار معمولی و اصلا برای پردازش این حد از داده پاسخگو نبود. از طرفی سایت کلب هم ما را بدلیل استفاده‌ی زیاد در این تمرین بُن نمود. لذا ما مجبور شدیم تعداد نُت‌ها را کمتر (به طور معمول 30 تا) و تعداد ابزارها را نیز در همین حدود در نظر بگیریم. البته در قسمت فریم‌های زمانی این اعداد تغییر کردند چون فضای حافظه را به یکباره زیاد می‌کرد و کامپیوتر پاسخگو نبود.

**توضیحات پیاده‌سازی:**

برای پیاده‌سازی کلاسی با نام noteRecognizer طراحی نمودیم. در این کلاس قابلیت‌های زیادی از جمله کاهش ابعاد با نرخ‌های مجاز، تعریف فریم‌های زمانی، تعداد نَت، 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 ورودی‌ها را به عنوان قطعه‌های صوتی متشکل از فریم‌های صوتی 250 ms به عنوان ورودی شبکه در نظر می‌گیرد. با توجه به سخت افزار خود تعداد نُت‌ها را 30 و تعداد ابزار‌های مربوط به نُت‌های انتخابی را نیز 30 (یعنی 30 تا ساز برای هر نُت) در نظر گرفتیم. نرخ یادگیر تابع بهینه‌ساز adam را 0.001 لحاظ کرده و شبکه را برای 50 ایپاک آموزش دیدیم.

مهمترین قسمت شبکه: این شبکه بدلیل پیش‌پردازش‌های 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 می‌دهد که از بسیاری از افراد دیگر که نتایجشان را پرسیدم بهتر است. دلیل این امر دو چیز است.

* دلیل اول عملکرد نسبتا مطلوب شبکه در این مسأله بخصوص:

پیش پردازش بر روی داده‌ها. انجام پیش‌پردازش بر روی داده‌ها بدین صورت بود که من هر داده را با توجه به سمپل‌های خودش اسکیل کردم. این امیر باعث افزایش صحت عملکرد شبکه ام شد.

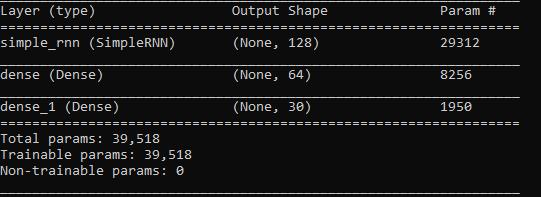
* دلیل دوم: یادگیری برای شبکه‌های بازگشتی با تعداد ابعاد بالا به ازای هر ریکورد در اکثر مواقع امکان پذیر نیست. لذا تصمیم گرفتیم از ابعادِ هر ریکورد، نمونه برداری کنم. این امر نیز به طرز بسیار خوبی توانست قدرت شبکه را در پیش‌بینی داده‌های تست و نیز در صحت داده‌های ارزیابی بالا ببرد.

نهایتا دلیل عملکرد نسبتا خوب شبکه را علاوه بر تنظیم مناسب پارمترها و هایپرپارامترها، بلکه ناشی از دو پیاده‌سازی فوق می‌دانم.

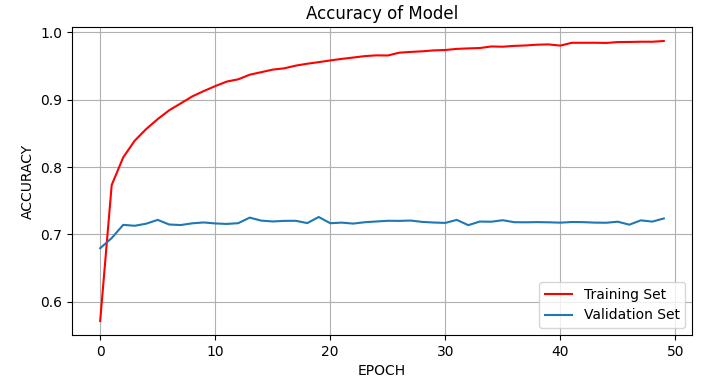
دقت فرمایید که نرخ نمونه برداری از داده‌گان دریافتی برای کاهش ابعاد در کد توسط پارامتر طولِ دیتای فشرده (compressionLength) قابل تنظیم است. تنها توجه کنید که نرخ بسیار بالا زمان پیش‌پردازش‌ها را به شدت افزایش می‌دهد و نرخ دیفالت قرار گرفته در کد تا ده برابر هم می‌تواند سرعت خوبی از خود نشان دهد. حال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

**الف)**

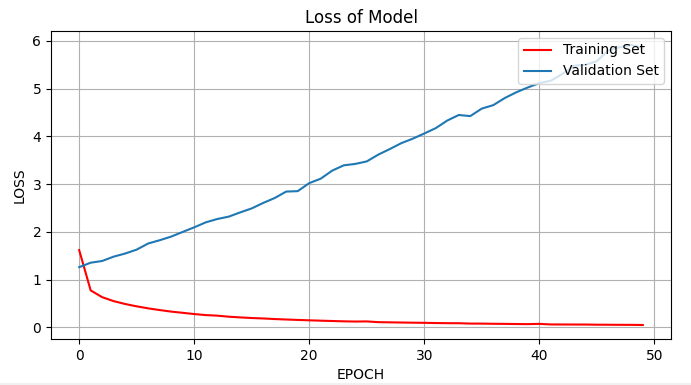
ابتدا نتایج را برای شبکه RNN بررسی می‌کنیم. فلذا شبکه را با سلول های RNN طراحی کردیم و ساختار شبکه را بصورت زیر می‌توانید مشاهده فرمایید:



شکل 28 ساختار شبکه انتخابی با سلول های 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

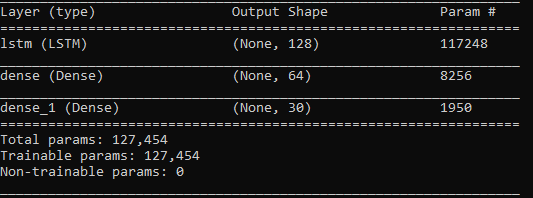
زمان آموزش نیز بصورت زیر شد:



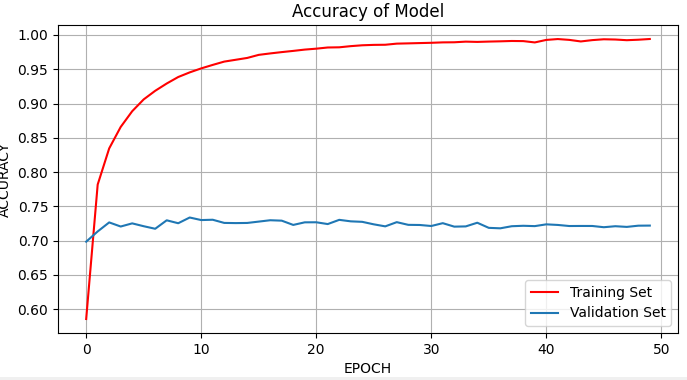
مقادیر Test Loss و Test Accuracy بصورت زیر شد:

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

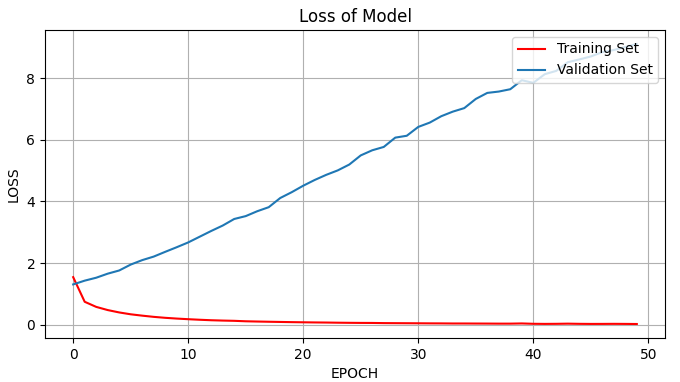
حال شبکه را با سلول های LSTM طراحی کردیم و ساختار شبکه بصورت زیر می باشد:



شکل 31 ساختار شبکه انتخابی با سلول های LSTM

صحت مدل بصورت زیر می باشد:

شکل 32 صحت مدل شبکه LSTM

نمودار Loss نیز بصورت زیر حاصل می شود:

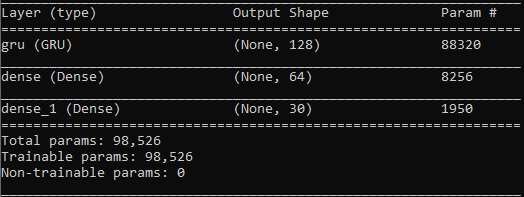
شکل 33 خطا مدل شبکه LSTM

زمان آموزش نیز بصورت زیر شد:

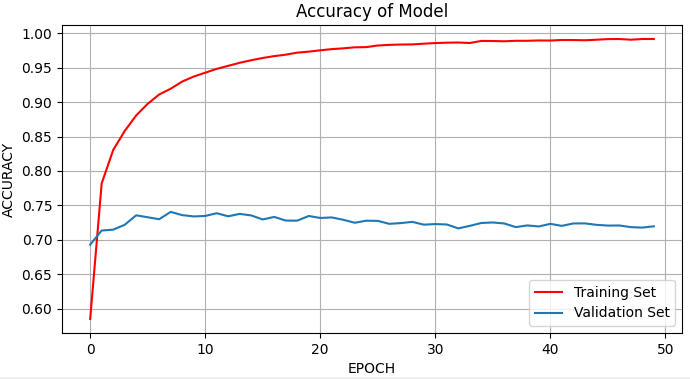
مقادیر Test Loss و Test Accuracy بصورت زیر شد:

سپس شبکه را با سلول های GRU طراحی کردیم.

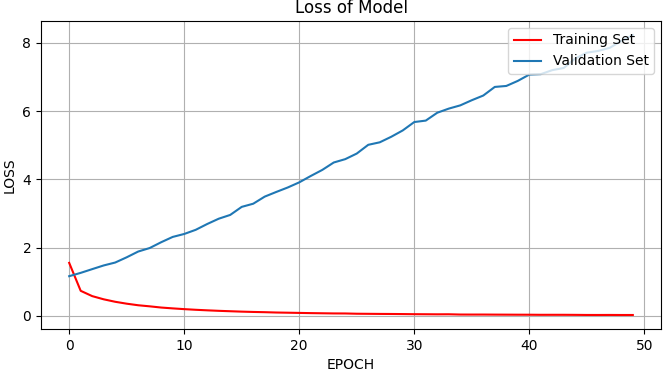
ساختار شبکه بصورت زیر می باشد:



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:



مقادیر Test Loss و Test Accuracy بصورت زیر شد:

نتایج کلی بصورت زیر می شود:

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

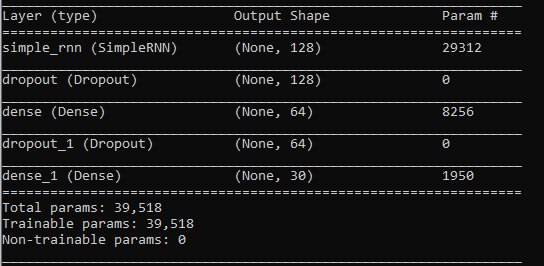
|  |  |  |  |
| --- | --- | --- | --- |
| Test Loss | Test Accuracy | Training Time |  |
| 7.6772 | 0.63775 | 290.1640 | RNN |
| 10.3971 | 0.633358 | 578.1516 | LSTM |
| 9.34515 | 0.646864 | 484.5051 | GRU |

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 تقریبا 22 دصد سریعتر از lstm است.

**ب) بررسی تاثیر dropout روی هر سه شبکه قبل:**

بررسی RNN با Dropout:

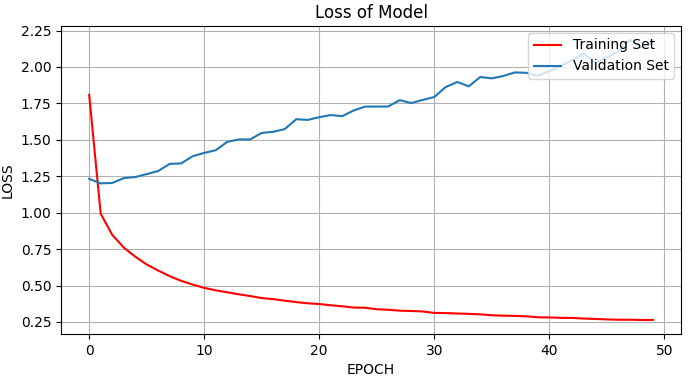
بعد از لایه RNN یک dropout با مقدار 10% و نیز یک dropout با مقدار 5% برای fully connect لایه میانی قرار دادیم که ساختار شبکه بصورت زیر می باشد:



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:

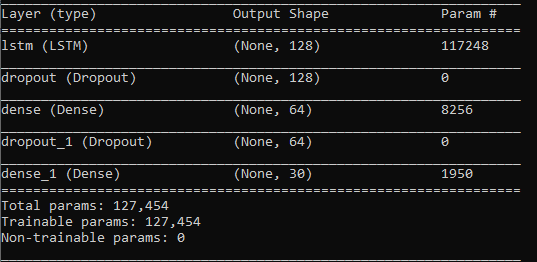
شکل 39 خطا مدل شبکه RNN با اعمال dropout

زمان آموزش نیز بصورت زیر شد:

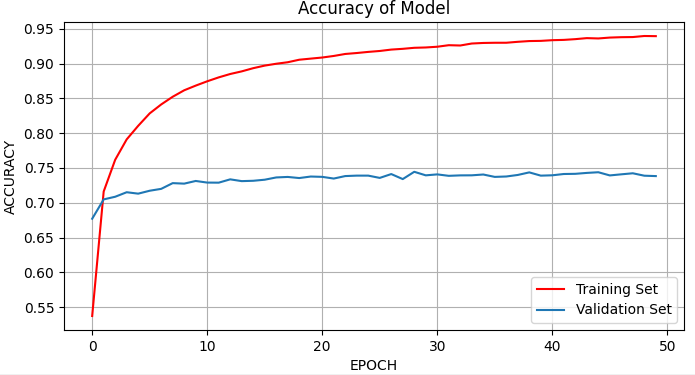


مقادیر Test Loss و Test Accuracy بصورت زیر شد:

بررسی LSTM با Dropout:

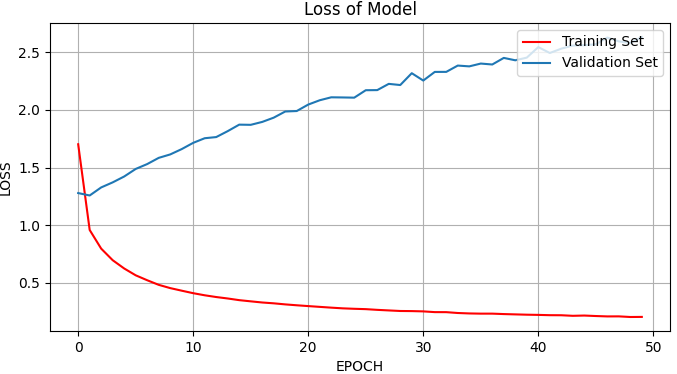
بعد از لایه LSTM یک dropout با مقدار 10% و نیز یک dropout برای fully connect لایه میانی با مقدار 5% قرار دادیم که ساختار شبکه بصورت زیر می باشد:

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:

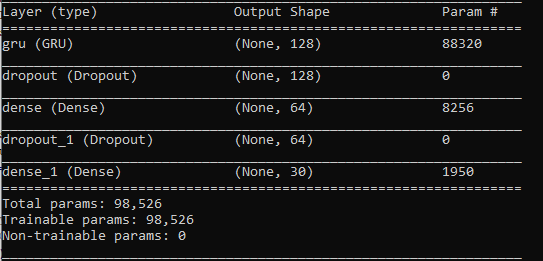


مقادیر Test Loss و Test Accuracy بصورت زیر شد:



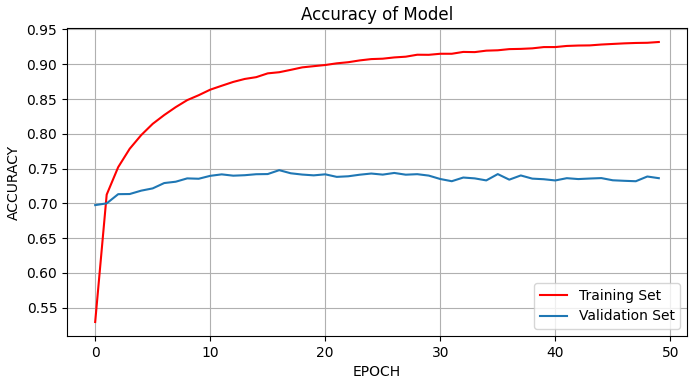
بررسی GRU با Dropout:

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:

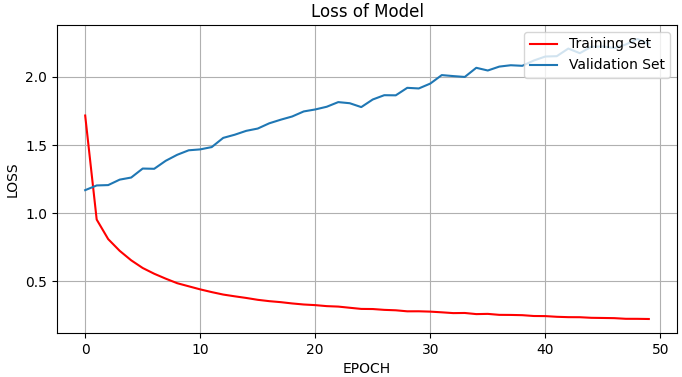


شکل 43 ساختار شبکه انتخابی با سلول های GRU با اعمال dropout

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:



مقادیر Test Loss و Test Accuracy بصورت زیر شد:

نتایج کلی بصورت زیر می شود:

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

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

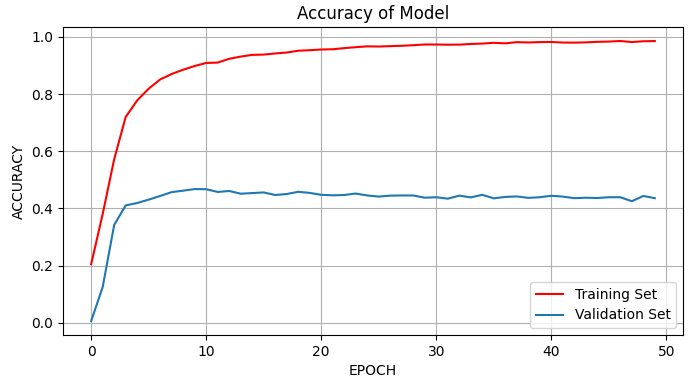
همانطور که از نتایج این قسمت برآمده است، LSTM بهترین عملکرد را بالاخص در پیش‌بینی داده‌های تست انجام داده. لذا زمان زیاد این مدل را به عملکرد خوبش به جان می‌خریم و ادامه‌ی روند در بخش‌های بعدی را با از استفاده از لایه‌ی LSTM پیش می‌گیریم.

**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 شبکه ای که در قسمت قبل بدست آمد، شبکه LSTM بود. حال برای ادامه مساله از این شبکه استفاده می کنیم. برای فریم های زمانی مختلف داریم:

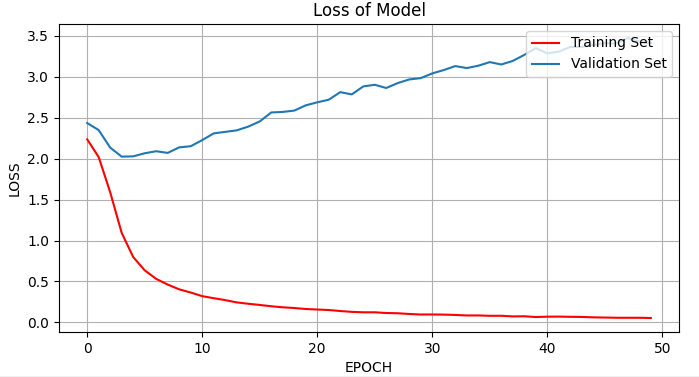
**فریم زمانی 300:**

صحت مدل بصورت زیر می باشد:



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

نمودار Loss نیز بصورت زیر حاصل می شود:



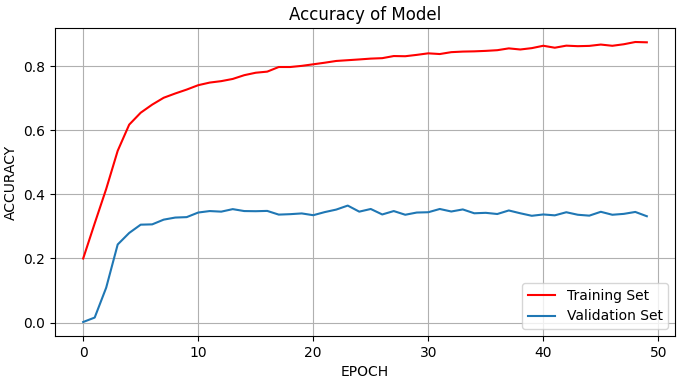
شکل 47 خطا مدل شبکه LSTM با اعمال dropout و فریم زمانی 300

زمان آموزش نیز بصورت زیر شد:

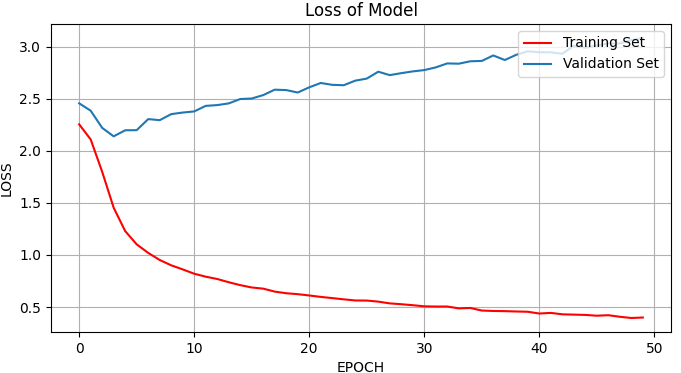
مقادیر Test Loss و Test Accuracy بصورت زیر شد:

**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:

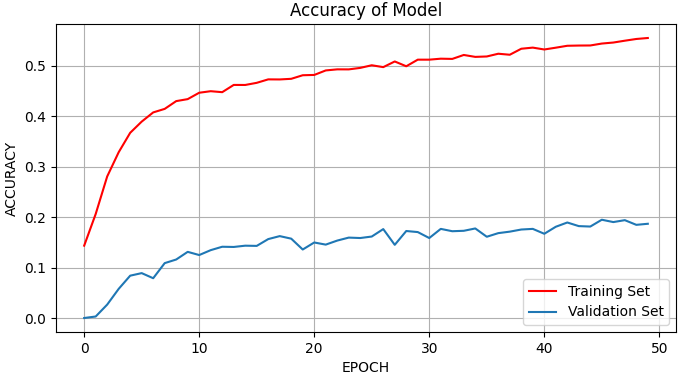
شکل 49 خطا مدل شبکه LSTM با اعمال dropout و فریم زمانی 150

زمان آموزش نیز بصورت زیر شد:



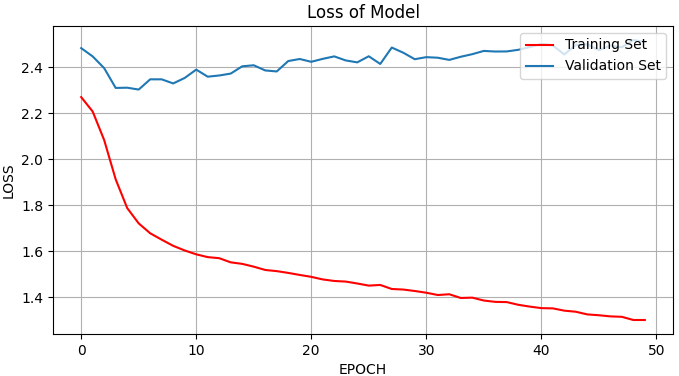
مقادیر Test Loss و Test Accuracy بصورت زیر شد:

**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:

مقادیر Test Loss و Test Accuracy بصورت زیر شد:



**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:

مقادیر Test Loss و Test Accuracy بصورت زیر شد:



نتایج کلی به شرح زیر است:

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

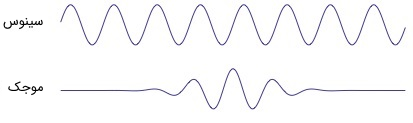
|  |  |  |  |
| --- | --- | --- | --- |
| Test Loss | Test Accuracy | Training Time | Time Frame |
| 2.5242 | 0.7381 | 33.8235 | 300 |
| 1.95437 | 0.57686 | 36.531 | 150 |
| 2.1458 | 0.2222 | 42.727 | 70 |
| 2.5583 | 0.0787 | 36.55 | 35 |

نتایج نشان می‌دهد که فریم زمانی 300 میلی ثانیه توانسته بهنرین نتیجه را در آموزش داده‌گان و نیز پیش‌بینی ریکورهای تست و همچنین از نقطه نظر زمانی به ما ارائه کند. مسلما هر اندازه که فریم زمانی کاهش می‌یابد، اطلاعات همبستگی بیشتر پیدا کرده و عملا 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)** الگوریتم های استخراج ویژگی متنوعی وجود دارد که در اینجا به چند مورد می پردازیم:

**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 دقیقا 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، کاملا 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 گوییم. 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.

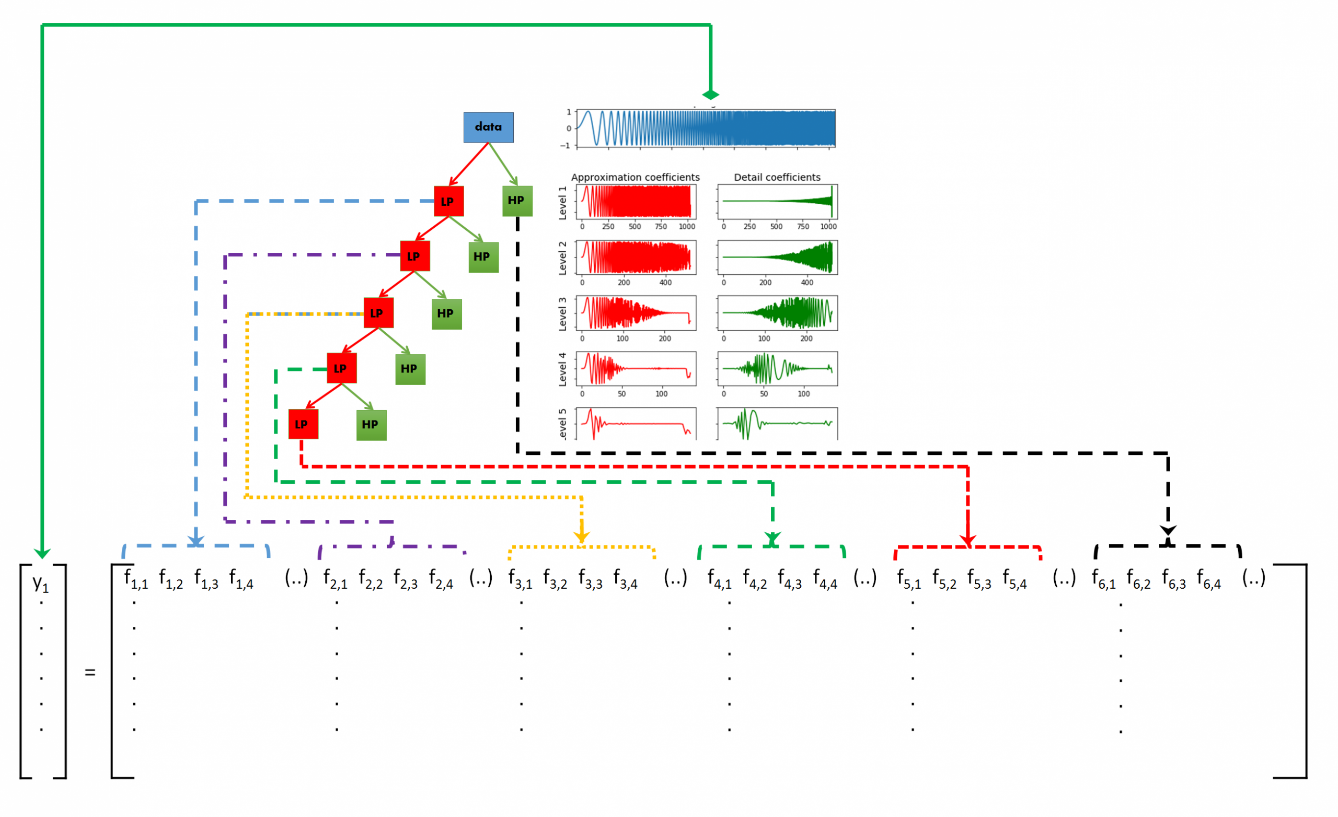
تبدیل فوریه معمولا برای تبدیل یک سیگنال در طیف زمانی به سیگنالی در طیف فرکانسی مورد استفاده قرار می‌گیرد. در حالی که تبدیل فوریه نمایشی از سیگنال در حوزه فرکانس ایجاد می‌کند، تبدیل موجک نمایشی از سیگنال را در حوزه زمان و فرکانس ایجاد می‌کند و در نتیجه امکان دسترسی کارآمد به اطلاعات محلی در مورد سیگنال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

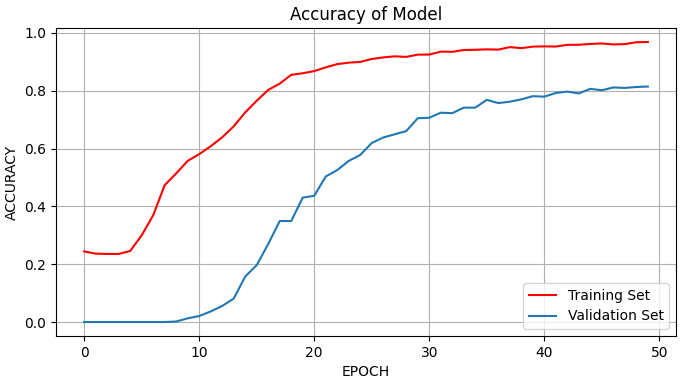
ابتدا از تبدیل موجک گسسته برای تفکیک یک سیگنال به زیرباندهای فرکانسی مختلف استفاده می‌کنیم. این کار را تا حد ممکن یا تا حد لازم تکرار می‌کنیم. چون انواع مختلف سیگنال‌ها، مشخصه‌های فرکانسی مختلفی را از خود نشان می‌دهند، در نتیجه این تمایز در رفتار باید در یکی از زیرباندهای فرکانسی خود را نشان دهد. بنابراین اگر ویژگی‌ها Features)) را از هریک از زیرباندهای مختلف ایجاد کنیم و از مجموعه تمام ویژگی‌ها به عنوان ورودی در یک طبقه بندClassifier) ) 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

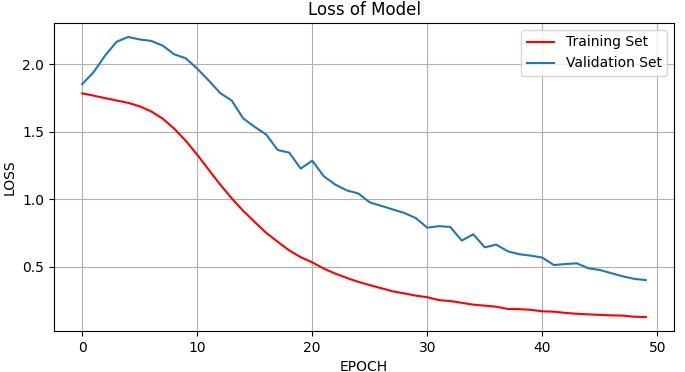
حال برای این مساله از تبدیل فوریه استفاده کردیم و نتایج زیر حاصل شد:

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 زمانی300 و اعمال تبدیل فوریه

زمان آموزش نیز بصورت زیر شد:



مقادیر Test Loss و Test Accuracy بصورت زیر شد:

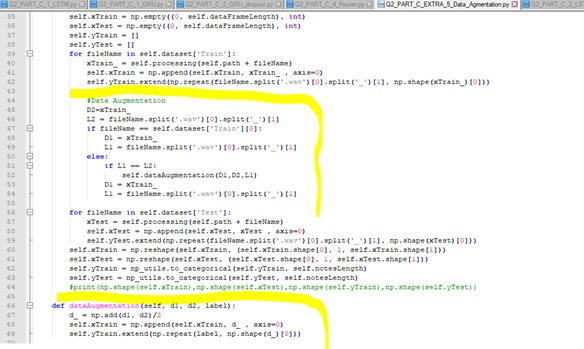


همانطور که مشاهده می‌فرمایید تبدیل فوریه توانسته نتایج را به میزان زیادی بهبود بخشد. از نظر من این موضوع باید ریشه در شباهت فرکانسی مابین نوت‌های یکسان از سازهای گوناگون باشد. در واقع نُت ها اگرچه در قالب سیگنال زمانی توانسته اند شباهت نسبتا خوبی را با پیش پردازش های قبلی از خود نشان دهند ولی به نظر می‌رسد که داده‌گان 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 تقسیم‌بندی نُت‌ها و تشخیص بسیار بهر برای داده‌گان تست منجر شده است.

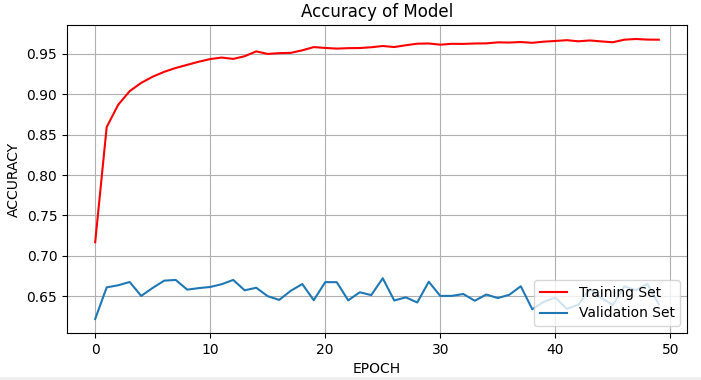
**پاسخ بخش ه – امتیازی)** برای این بخش روش هایی موجود است که به چند مورد می پردازیم:

* **روش اول:** برای data augmentation یک روش اینست که به ازای یک نُت خاص سیگنال های دو instrument را با هم میانگین بگیریم و همینطور ادامه می دهیم و دو به دو میانگین می گیریم بدین صورت سمپل‌های جدید ساخته و ادغام آنها برای دو داده به یک ریکورد جدید می‌رسیم. 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 نُت 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 می‌کنیم.

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

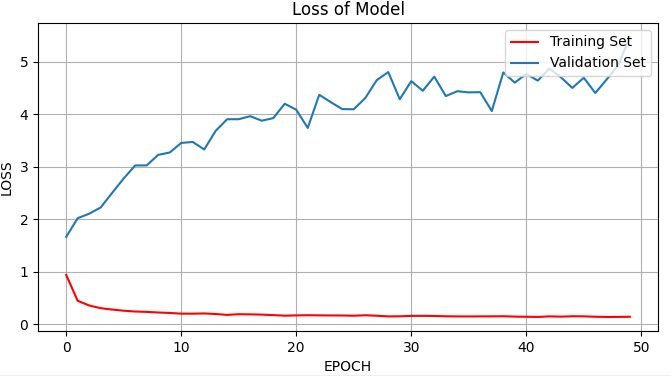
کد های این بخش بصورت زیر می باشد:

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 زمانی300 و اعمال data augmentation

نمودار Loss نیز بصورت زیر حاصل می شود:



شکل 60 خطا مدل شبکه LSTM با اعمال dropout و فریم زمانی300 و اعمال data augmentation

زمان آموزش نیز بصورت زیر شد:



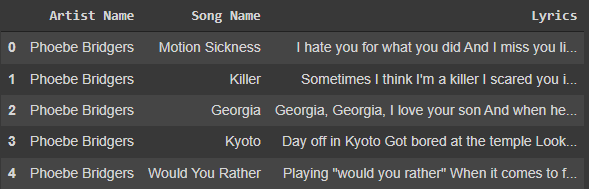
مقادیر Test Loss و Test Accuracy بصورت زیر شد:



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. طبیعتا مدل توانسته خود را با داده‌های جدیدتر ورزیده و بهتر آموزش دهد فلذا انتظار این بهبودی کاملا 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:



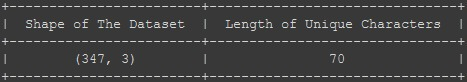
شکل 61 سطرهای ابتدایی دیتاست

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.

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', '¡', '£', '¨', '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']

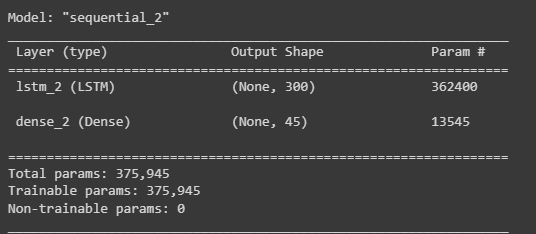
می بینیم که پس از پیش پردازش های لازم طول کل کاراکتر ها برابر با 419781 و تعداد کاراکترهای واحدمان برابر با 45 عدد است:

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)



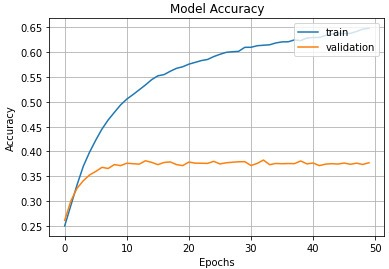
شکل 64 ساختار شبکه انتخابی

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

**loss function: categorical\_crossentropy**

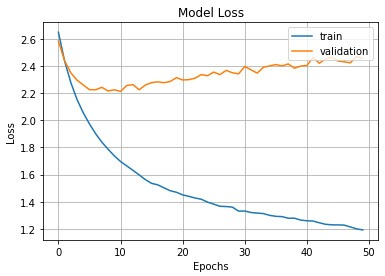
نتایج بصورت زیر حاصل می شود:

**نمودار صحت:**



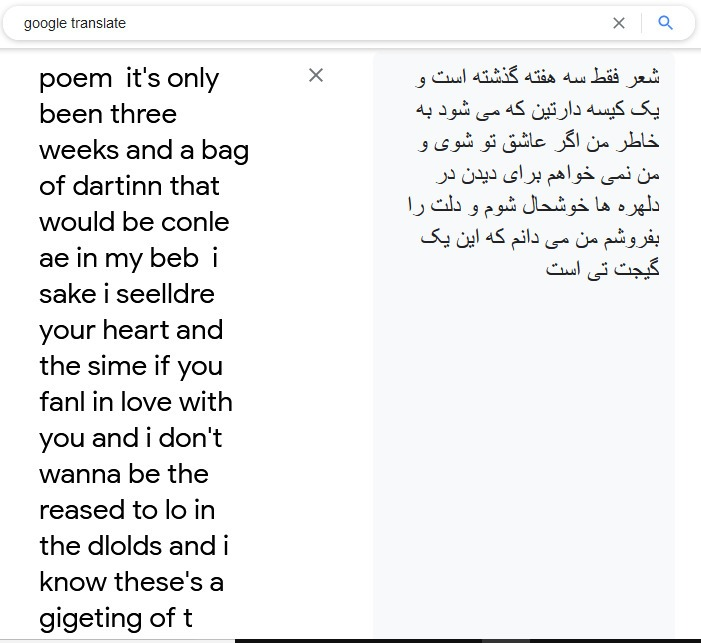
شکل 65 صحت مدل شبکه

**نمودار خطا:**



شکل 66 خطا مدل شبکه

**بیت خروجی شبکه به همراه ترجمه در گوگل ترنسلیت:**



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 کاملا 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.

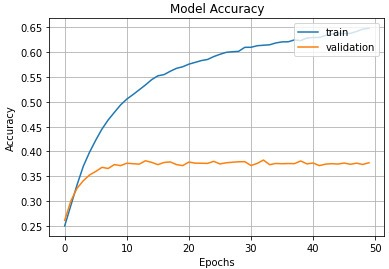
**لحاظ کردن missing value ها:** در ابتدا missing value ها را حذف می کنیم. پس در صدد حذف برخی از این کاراکتر ها شدیم که بی معنی و بدون کاربرد هستند.

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:

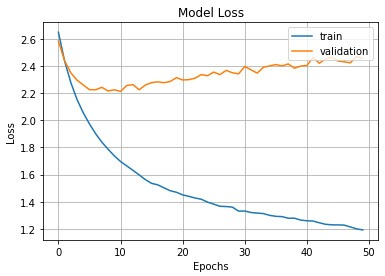
* **الف) loss function: categorical\_crossentropy**

**نمودار صحت:**



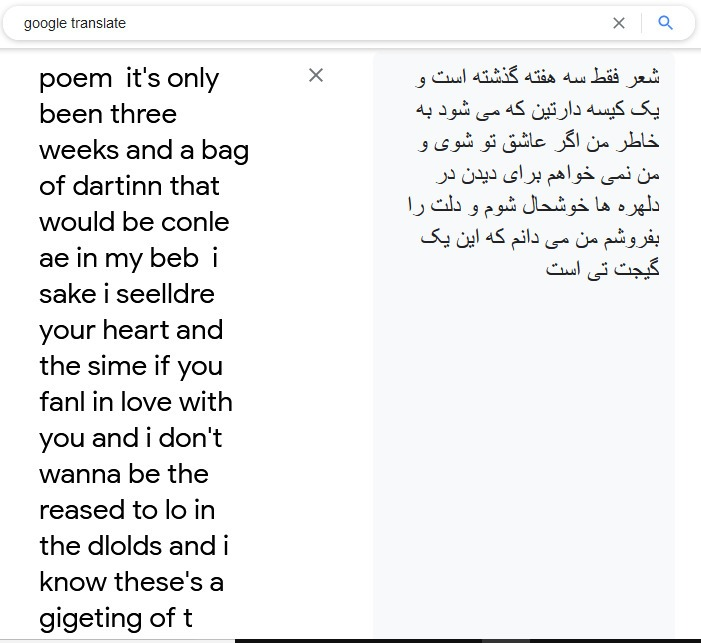
شکل 68 صحت مدل شبکه با Loss function: categorical\_ceorrentropy

**نمودار خطا:**



شکل 69 خطا مدل شبکه با Loss function: categorical\_ceorrentropy

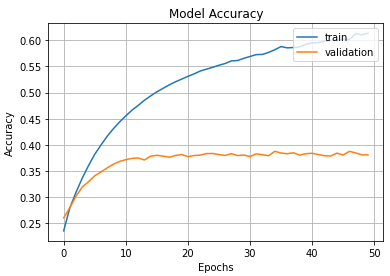
**بیت خروجی شبکه به همراه ترجمه در گوگل ترنسلیت:**



شکل 70 بیت خروجی شبکه به همراه ترجمه آن

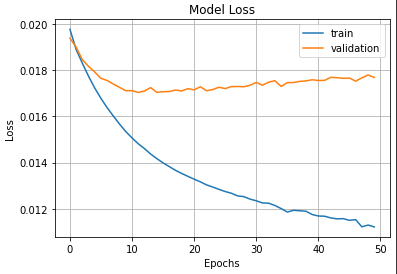
* **loss function: mse**

**نمودار صحت:**



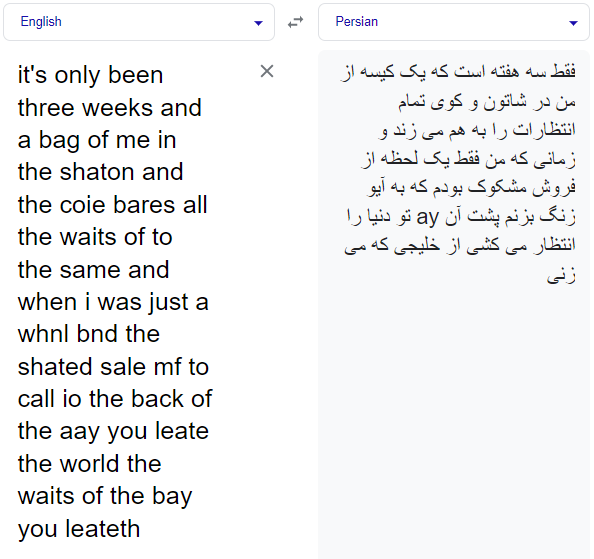
شکل 71 صحت مدل شبکه با Loss function: mse

**نمودار خطا:**



شکل 72 خطا مدل شبکه با Loss function: mse

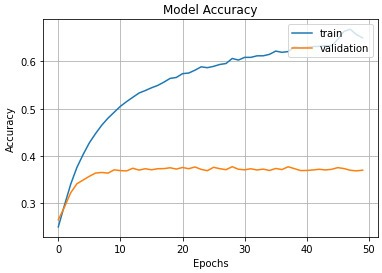
**بیت خروجی شبکه به همراه ترجمه در گوگل ترنسلیت:**



شکل 73 بیت خروجی شبکه به همراه ترجمه آن

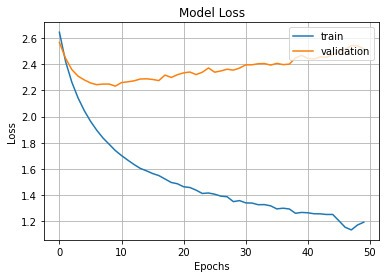
* **loss function: KLDivergence**

**نمودار صحت:**



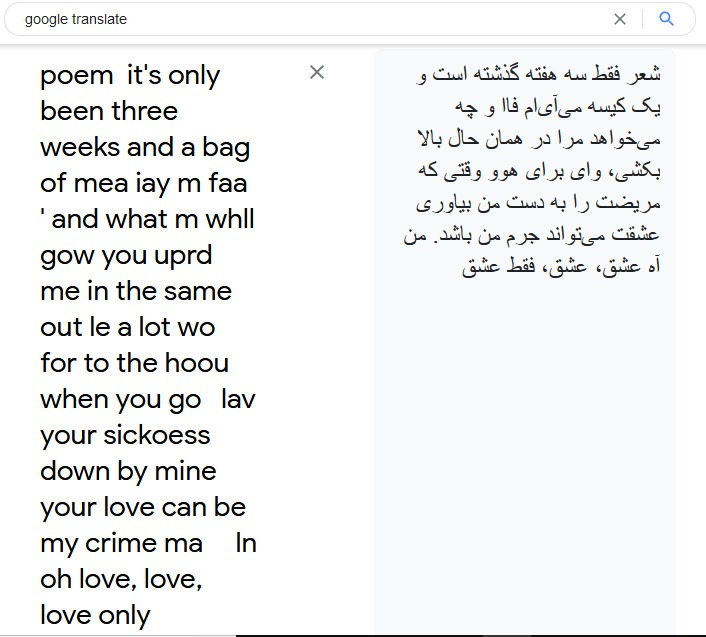
شکل 74 صحت مدل شبکه با Loss function: KLDivergence

**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 ایپاک و همان

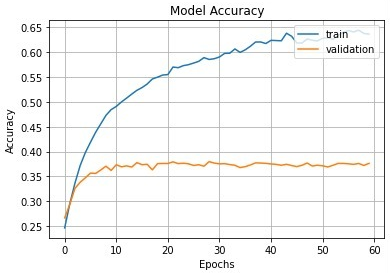
optimizer = Adamax(learning\_rate = 0.015)

بررسی می کنیم و سپس با optimizer های متفاوت زیر عملکرد مدل را بررسی می کنیم و به نتایج زیر می رسیم:

**1) عملکرد مدل یکبار برای 60 ایپاک و همان optimizer = Adamax(learning\_rate = 0.015)**

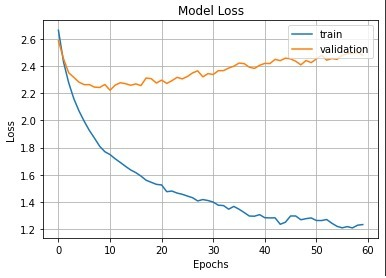
**loss function: categorical\_crossentropy**

**نمودار صحت:**



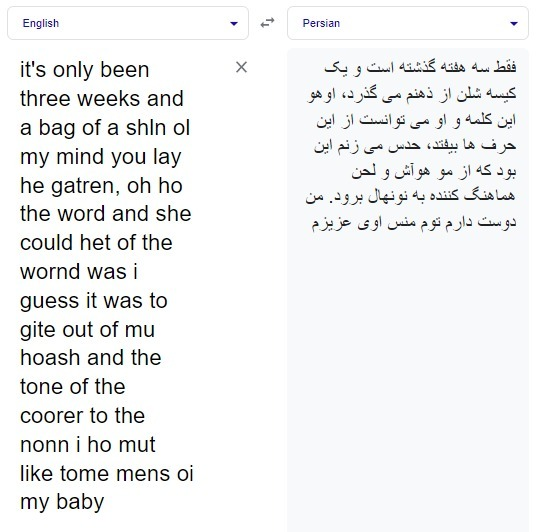
شکل 77 صحت مدل شبکه با Loss function: categorical\_crossentropy و optimizer: Adamax(Learning\_rate=0.05) برای 60 ایپاک

**نمودار خطا:**



شکل 78 خطا مدل شبکه با Loss function: categorical\_crossentropy و optimizer: Adamax(Learning\_rate=0.05) برای 60 ایپاک

**بیت خروجی شبکه به همراه ترجمه در گوگل ترنسلیت:**



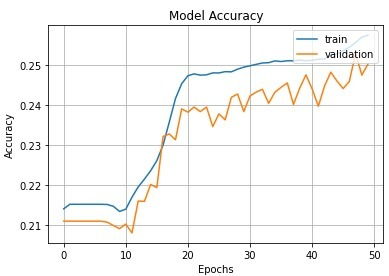
شکل 79 بیت خروجی شبکه به همراه ترجمه آن

می بینیم که با تعداد ایتریشن 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**

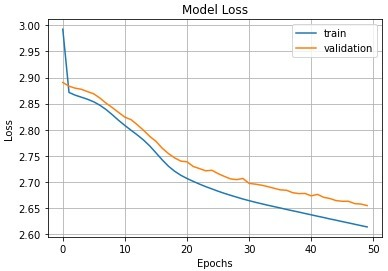
در این مرحله optimizer خود را به SGD تغییر می دهیم تا عملکرد مدل را بررسی کنیم:

**نمودار صحت:**



شکل 80 صحت مدل شبکه با Loss function: categorical\_crossentropy و optimizer: SGD

**نمودار خطا:**



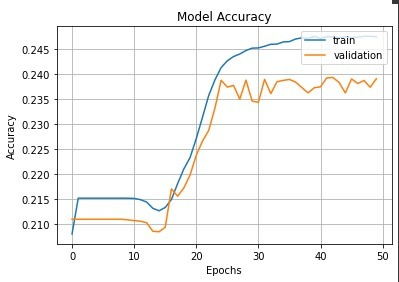
شکل 81 خطا مدل شبکه با Loss function: categorical\_crossentropy و optimizer: SGD

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

**3) عملکرد مدل برای optimizer = adadelta و loss function: categorical\_crossentropy**

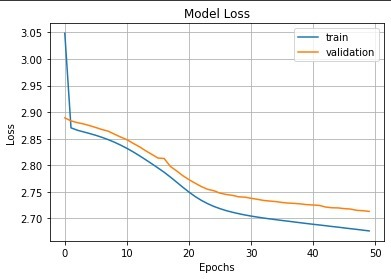
در این مرحله optimizer خود را به adadelta تغییر می دهیم تا عملکرد مدل را بررسی کنیم:

**نمودار صحت:**



شکل 82 صحت مدل شبکه با Loss function: categorical\_crossentropy و optimizer: adadelta

**نمودار خطا:**



شکل 83 خطا مدل شبکه با Loss function: categorical\_crossentropy و optimizer: adadelta

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 تقریبا 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 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 زیرشبکه‌ها به منظور انتشار خطا در حالی که آنها را برای انتشار فعالیت تقسیم نمی‌کنند، انجام می‌شود. که در مقاله زیر این روش مفصلا 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 fully-learned approach،SA-RNN ، 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 به‌روزرسانی را به‌صورت آنلاین تطبیق می‌دهد، و اجازه می‌دهد ابعاد مختلف به‌صورت مشروط به ورودی به‌روزرسانی 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.

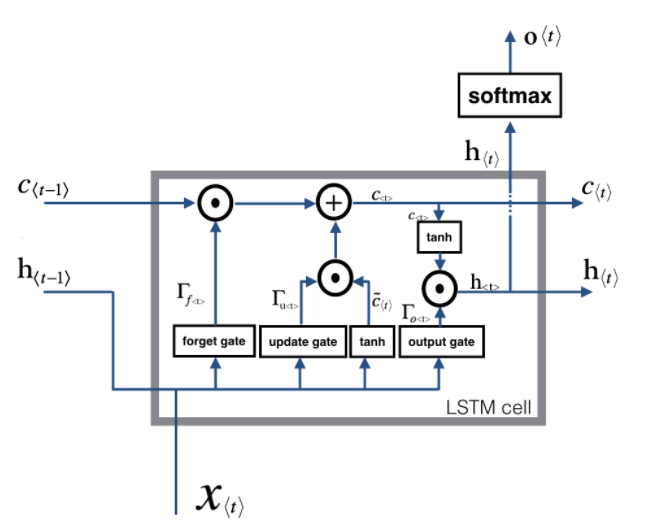
* **استفاده از تعداد سلول کمتر و اضافه نمودن dropout:** مسلما هر چه تعداد سلول کمتر باشد، پیچیدگی محاسباتی کاهش می‌یابد. لاکن برای جبران صحت کمتری که مدل ارائه می‌کند، در مراجعی ذکر شده بود که از dropout برای جبران میزان صحت کاهش‌یافته استفاده می‌کنند.لازم به ذکر است که پیچیدگی محاسباتی dropout در برابر پیچیدگی محاسباتی سلول‌ها قابل مقایسه نیست و این روش یکی از روش‌های رایج در حال حاضر است.

**پاسخ بخش ث)**

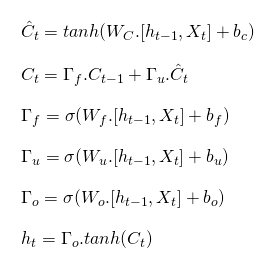
سلول های عصبی LSTM شامل 3 دروازه (gate) می باشد، که یکی از این دروازه ها به نام دروازه فراموشی است که با نمایش داده می شود. و وظیفه کنترل جریان اطلاعات از گام زمانی قبلی را دارد. این دروازه مشخص می کند که آیا اطلاعات حافظه از گام زمانی قبل مورد استفاده قرار گیرد یا خیر و اگر باید از گام زمانی قبل چیزی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

دروازه بروزرسانی یا همان Update gate که بصورت 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 عموما به دروازه ورودی نیز یاد میشود.

دروازه خروجی یا همان Output gate که بصورت 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 شکل یک بلوک LSTM



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:

در اینجا 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 خواهد بود که مقادیری بین ۰ و ۱ خواهد داشت. این بردار سپس در عبارت بعدی در ضرب خواهد شد. بنابر این اگر مقادیر بردار دروازه فراموشی صفر باشد(یا به سمت صفر میل کند) عملا به معنای در نظر نگرفتن محتوای است. به عبارت ساده تر یعنی شبکه اطلاعات ارائه شده توسط را دور انداخته و هیچ توجهی به آن نمی کند. به همین صورت اگر مقادیر بردار 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