# گزارش پروژه درس پردازش گفتار

**موضوع: بررسی و پیاده سازی مقاله**

**END-TO-END ATTENTION-BASED LARGE VOCABULARY SPEECH RECOGNITION**

**نویسنده: محمد امین بنائیان زاده (۹۸۲۰۸۸۳۵)**

## مقدمه

در این گزارش به طور کلی قصد داریم به بررسی و پیاده سازی مقاله END-TO-END ATTENTION-BASED LVCSR بپردازیم. هدف این مقاله حل مسئله Automatic Speech Reognition با استفاده از رویکردهای یادگیری عمیق و شبکه‌های عصبی است و در این مقاله یک معماری تازه به منظور حل چنین مسئله‌ای معرفی شده است. به طور دقیق‌تر، ساختاری که در این مقاله پیشنهاد می‌شود، یک ساختار مبتنی بر شبکه‌های بازگشتی عمیق و به فرم Seq2Seq می‌باشد. نکته‌ای که این مقاله را متمایز می‌کند، پیشنهاد یک مکانیزم اضافه به این نوع شبکه‌هاست که مکانیزم توجه (attention) نام دارد. لذا ما در این گزارش قصد داریم تا این معماری را بررسی کرده، کدهای آن را توسط کتابخانه PyTorch پیاده سازی کنیم و در نهایت با سایر روش‌های موجود مقایسه کنیم.

ساختار این گزارش به این صورت است: در بخش اول به معرفی دقیق‌تر صورت مسئله پرداخته و روش‌های مرسوم برای حل چنین مسئله‌ای را بررسی می‌کنیم. در بخش دوم به سراغ مجموعه دادگان رفته و نحوه کار با آن‌ها را شرح می‌دهیم. سپس در بخش بعد، توضیحاتی درباره معماری پیشنهادی و پیاده‌سازی آن ارائه می‌کنیم. همچنین در همین بخش درباره پیاده‌سازی روش‌های دیگر ASR مبتنی بر یادگیری عمیق صحبت می‌کنیم. در بخش چهارم مجموعه الگوریتم‌هایی که می‌تواند به بهبود عملکرد این معماری در زمان تست منجر شود را معرفی می‌کنیم. برای این منظور دو روش استفاده از beam decode و همچنین استفاده از مدل زبانی پیشنهاد می‌شود. در بخش بعدی به بررسی نتایج به دست آمده پرداخته و سعی می‌کنیم تحلیلی در این‌باره ارائه کنیم. در نهایت در بخش آخر یکی دو مقاله چدیدتر که مرتبط با همین مقاله هستند را معرفی می‌کنیم و به طور خلاصه آن‌ها را بررسی می‌کنیم.

## بخش اول) معرفی هدف و پیشینه روش‌های موجود برای حل آن

یکی از مسائل بسیار مهم حوزه پردازش گفتار، تبدیل اتوماتیک گفتار به نوشتار (ASR) می‌باشد. برای حل این مسئله کارهای زیادی صورت گرفته است و از قدیم روش‌های گوناگونی برای این منظور معرفی شده است. در میان همه این روش‌ها، روش‌های آماری که غالباً مبتنی HMM یا GMMها هستند به بهترین نتایج دست یافته بودند. با توجه به ذات آماری این روش‌ها، لازم است تا یک دیتاست به اندازه کافی بزرگ برای این منظور تهیه شود و مدل‌های مورد نظر توسط این روش آموزش داده شوند.

در چند سال اخیر استفاده از شبکه‌های عصبی برای حل بسیاری از مسائل سخت با موفقیت همراه بوده و لذا دانشمندان حوزه پردازش گفتار نیز به سمت استفاده از شبکه‌های عصبی برای حل مسئله ASR حرکت کرده‌اند. در روش‌های قدیمی‌تر، مسئله ASR به دو زیرمسئله ساخت مدل آکوستیکی و ساخت مدل زبانی شکسته می‌شود. شبکه‌های عصبی قادر هستند هم مدلسازی آکوستیکی و هم مدلسازی زبانی را انجام دهند. لذا اولین گامی که در جهت استفاده از شبکه‌های عصبی در حل مسئله ASR معرفی شد، روش‌های hybrid نام داشت که در این روش‌ها بخشی از سیستم توسط شبکه‌های عصبی و بخش دیگری توسط مدل‌های سنتی (مثلاً HMM) پیاده‌سازی می‌شد.

با پیشرفت یادگیری عمیق، این تمایل به وجود آمد که از شبکه‌های عصبی به صورت end-to-end برای حل این مسئله استفاده شود و لذا مدلسازی همه بخش‌ها توسط شبکه عصبی صورت پذیرد. اولین مقاله‌ای که در این زمینه یک گام جدی برداشت، مقاله Connectionist Temporal Classification (یا به طور خلاصه همان CTC) بود که توانست یک رویکرد عملی برای معرفی تابع ضرر و آموزش end-to-end یک شبکه بازگشتی معرفی می‌کند. [1]. در سال‌های بعدی از این روش‌ها استفاده شد تا یک مسئله واقعی با یک دیتاست بزرگ ( Wall Street Journal

corpus) حل شود. [2]. این تلاش‌ها به نتایج خیلی خوبی منجر شد و شبکه‌های عمیق توانستند تبدیل به state-of-the-art حوزه پردازش گفتار شوند. در واقع مسئله ASR را می‌توان به صورت یک مسئله many-to-many دید که در آن هدف تبدیل یک دنباله صوتی به دنباله حروف آن است. (البته مسئله از جنس .order sync می‌باشد چرا که ترتیب حروف و آواها متناظر یکدیگر می‌باشند.) با این دید می‌توان رویکردهای دیگری که برای حل مسائل many-to-many استفاده می‌شود را برای حل مسئله ASR نیز به کار برد.

همزمان با پیشرفت‌های ذکر شده در حوزه ASR، یک رویکرد دیگر در یادگیری عمیق پدیدار شد که به منظور حل مسائل Seq2Seq معرفی شد. به عبارت دیگر در این مسائل هدف آن بود که مدل با دریافت یک دنباله ورودی، بتواند یک دنباله را در خروجی پیش‌بینی کند. به عنوان مثال می‌توان وظیفه machine translation را معرفی کرد که در آن شبکه عمیق با دریافت یک جمله در دنباله مبدا، معادل ترجمه شده آن را در زبان مقصد تولید می‌کند. همچنین در سال ۲۰۱۵ یک مکانیزمی معرفی شد که دقت این شبکه‌ها را به شدت افزایش می‌داد. این مکانیزم، مکانیزم توجه نام دارد که در ابتدا توسط [3] برای انجام وظیفه machine translation معرفی شد. از آن جا که مسئله بازشناسی گفتار را هم می‌توان به صورت تبدیل دنباله آوایی به دنباله حروف دید، لذا نویسندگان همان مقاله به این فکر افتادند که این رویکرد را در این قالب نیز استفاده کنند.

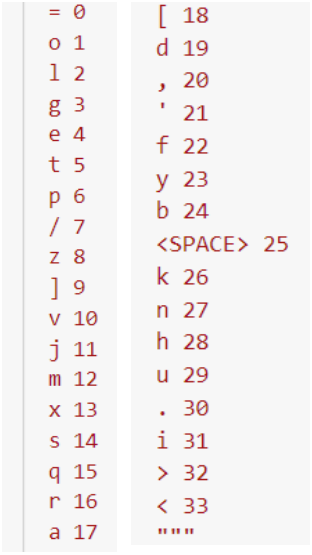
مقاله فعلی، تلاشی است در جهت استفاده از مدل‌های seq2seq و مکانیزم attention برای حل مسئله ASR به صورت یکپارچه و با رویکرد end-to-end. ما در این گزارش معماری معرفی شده در مقاله را توضیح داده و پیاده‌سازی کرده و با مدل‌های مبتنی بر CTC مقایسه می‌کنیم. پیش از ارائه توضیحات لازم است تا دیتاست مورد استفاده را توضیح داده و ویژگی‌های آن را مورد بررسی قرار دهیم.

## بخش دوم) معرفی دیتاست‌های مورد استفاده در این پروژه

دیتاستی که مقاله مرجع روی آن کار کرده است، دیتاست Wall Street Journal است که یک دیتاست رایگان نمی‌باشد. بنابراین در پیاده‌سازی این مقاله ما به جای این دیتاست، از دو جایگزین استفاده می‌کنیم. اول دیتاست **Librispeech** که یک دیتاست برچسب خورده انگلیسی و رایگان است. و دوم دیتاست **فارس‌دات** که توسط تیم آموزشی در اختیار ما قرار گرفته است.

دیتاست انگلیسی مورد استفاده تقریباً به صورت آماده موجود است و توسط کتابخانه pytorch می‌توان به سادگی از آن استفاده کرد. اما دیتاست فارسی نیاز به آماده سازی بیشتری و پردازش اولیه داشت که این مسئله به صورت دستی انجام شده است و در ادامه تا حدی مسائل مربوط به این دیتاست را توضیح خواهیم داد. ضمناً توجه کنید قابلیت تعوضی میان دیتاست فارسی و انگلیسی به سادگی و فقط با کمک مقداردهی به متغیر PERSIAN_FLAG در کد انجام می‌شود.

دیتاست فارسی به طور کلی حاوی دو پوشه CD1 و CD2 بود که در هرکدام مجموعه‌ای از فایل‌های صوتی به همراه سندهای مربوط به برچسب‌گذاری این فایل‌ها وجود داشت. فایل‌های صوتی در پوشه Wave موجود هستند و در هر فایل صوتی، چندین جمله به صورت پشت سر هم توسط یک فرد خاص ادا شده است. لذا در گام اول لازم است تا جملات از هم شکسته شوند. برای این منظور به سراغ پوشه sentences رفته و زمان شروع و پایان هر جمله در فایل صوتی مورد نظر را استخراج می‌کنیم (البته اطلاعات با اندیس بایت داده شده بود که تبدیل آن به زمان با یک ضریب تقسیم بر ۲ صورت می‌گیرد). از آن‌جا که مدل پیاده سازی شده در بخش‌های بعدی نیازی به برچسب زدن تک تک آواها ندارد، دسترسی به زمان دقیق شروع و پایان هر کلمه یا هر آوا در اینجا برای ما اهمیتی نداشت و لذا از اطلاعات آن‌ها استفاده‌ای نشده است. بعد از خواندن جملات، یک دیکشنری از روی نمادهای خوانده شده ساخته می‌شود که به هر نماد یک عدد را نسبت می‌دهد. در شکل زیر می‌توان این نگاشت از نماد به اعداد را مشاهده کرد:




مثلاً نماد x که نشان‌دهنده حرف «خ» می‌باشد، در جدول فوق با عدد ۱۳ نشان داده شده است. توجه کنید که نمادهای =، < و > در دیتاست فارسی وجود ندارد و به صورت دستی به این مجموعه اضافه شده‌اند تا مدلسازی padding، start token و stop token را انجام دهند.

توجه:

از آن جا که حجم داده‌های فارسی زیاد بود و پیاده‌سازی های انجام گرفته روی کولب صورت گرفته است، خواندن تمامی فایل‌های صوتی در ابتدای هربار اجرای کد بسیار زمانبر بود. برای حل این مشکل، یک بار تمامی این فایل‌ها را خوانده، برچسب گذاری کرده و سپس با کمک دستور torch.save آن را در گوگل درایو ذخیره می‌کنیم. لذا بار اول اجرای این کد می‌تواند خیلی زمانبر باشد. اما در زمان‌های بعدی، به جای انجام این کار صرفاً فایل ایندکس شده را با دستور torch.load بارگذاری می‌کنیم. اگر تمایل دارید تا فایل‌های ایندکس شده ساخته شوند، قسمتی که این فایل‌ها را تولید می‌کند از حالت کامنت خارج نموده و کد را اجرا کنید.

مراحل پیش‌پردازش داده:

به طور کلی پیش‌پردازش اولیه می‌تواند به بهبود عملکرد شبکه‌های عمیق کمک کند. برای این منظور، بخش‌هایی در کد پیاده‌سازی شده است که این مسئله را مدیریت می‌کند. یکی از این موارد استخراج ویژگی‌های Mel از روی سیگنال صوتی است. برای این منظور از کتابخانه torchaudio استفاده شده است و از هر سیکتال صوتی به تعداد ۱۲۸ ویژگی MFCC استخراج می‌شود. همچنین مسئله augmentation که به بهبود عملکرد شبکه‌ها و جلوگیری از overfitting منجر می‌شود نیز استفاده شده است. به عنوان نمونه از augmentation در حوزه صوت، می‌توان FrequencyMasking و TimeMasking را معرفی نمود.

توجه:

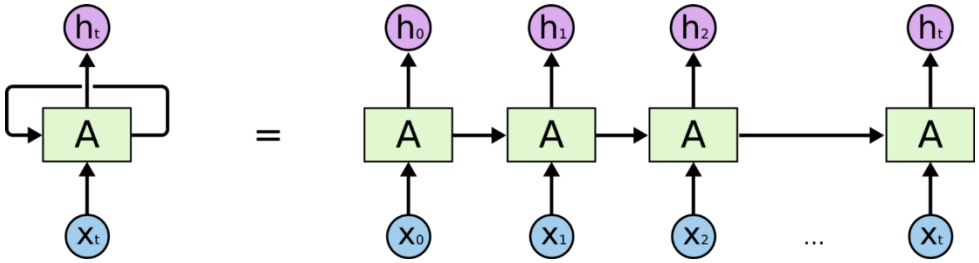
پیش از ورود به بخش سوم لازم است توجه شود که کدهای مربوط به این پیاده‌سازی به این گزارش ضمیمه شده است. همچنین باید اشاره کرد که برای پیاده سازی بعضی از بخش‌های این کد از جمله آماده سازی دیتاست انگلیسی، مدل CTC و متریک‌های مقایسه از آموزش موجود در سایت assemblyai.com استفاده شده است. (البته هیچ یک از این موارد موضوع مستقیم این مقاله نیستند و به علاوه این که تسلط کافی به بخش‌های مختلف این کد وجو دارد). اما بقیه قسمت‌ها اعم از خواندن و آماده سازی دیتاست فارسی، پیاده سازی مدل زبانی، پیاده سازی مدل Seq2Seq، پیاده‌سازی مکانیزم توجه، پیاده‌سازی جستجوی Beam Search و ... توسط اینجانب و صرفاً به کمک توضیحات درون مقاله پیاده‌سازی شده است. همچنین ذکر این نکته ضروری است که کدهای مربوط به مقاله مورد بررسی، در صفحه github آن‌ها موجود است اما این پیاده سازی با کتابخانه قدیمی Theano صورت گرفته است. ما در این پروژه از هیچ یک از کدهای آن‌ها استفاده نکردیم و ضمناً همه پیاده‌سازی‌ها صرفاً با کمک کتابخانه PyTorch انجام شده است.

## بخش سوم) معرفی ساختار شبکه عصبی و معماری پیاده‌سازی شده

در این پروژه دو شبکه عصبی عمیق با دو معماری متفاوت پیاده سازی شده و عملکرد این دو مدل روی دیتاست‌های معرفی شده سنجیده شده است. مدل اول مدل مبتنی بر CTC بوده که موضوع اصلی مقاله ما نیست اما به منظور مقایسه پیاده سازی شده است. مدل دوم که مدل seq2seq و attention based است در همین مقاله معرفی شده است. پیش از توضیح این دو مدل، به نظر می‌رسد خوب است اندکی درباره ساختارهای رایج پردازش یک دنباله توسط شبکه‌های عصبی توضیح داده شود. (البته توضیحاتی که اینجا ارائه می‌کنیم خیلی خلاصه هستند و مطالعه بیشتر درباره بلوک‌های سازنده شبکه‌های عمیق به خواننده واگذار می‌شود).

### شبکه‌های بازگشتی:

در بحث‌های یادگیری عمیق، یکی از ساختارهایی که برای پردازش سری‌های زمانی استفاده می‌شود، ساختار بازگشتی یا Recurrent نام دارد. عملکرد شبکه‌های بازگشتی به این صورت است که یک ورودی از دنباله مورد نظر دریافت می‌کند و پردازش‌های مورد نظر روی آن را انجام می‌دهد. این پردازش‌ها به عنوان متغیر حالت در ساختار شبکه ذخیره شده و همراه با عضو بعدی دنباله دوباره در ورودی ماژول بعدی وارد می‌شود. لذا این ساختار می‌تواند انجام پردازش هر ورودی را متناسب با تمام ورودی‌های از قبل دیده شده انجام دهد بدون آن که به طول دنباله حساس باشد. در شکل زیر یک نمونه از این چنین ساختاری دیده می‌شود.



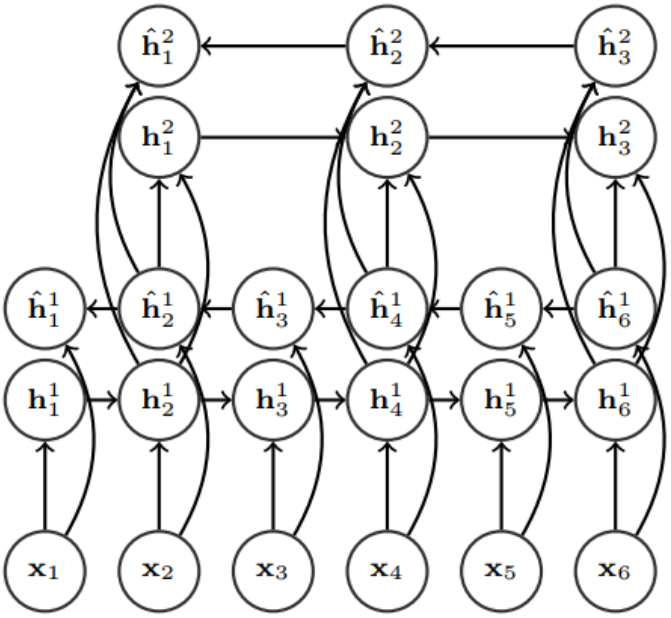

در مورد این که ساختار درونی ماژول A چه چیزی باشد، تحقیقات گسترده‌ای صورت گرفته است اما به طور کلی، دو ساختار مرسوم برای این ماژول عبارت است از LSTM و GRU.

در مقاله مرجع ما از ماژول GRU به منظور پیاده‌سازی‌های بازگشتی استفاده شده است. در این قسمت می‌توانید روابط به کار رفته در این ساختار را مشاهده نمایید.

$$
\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{U}_{hz}\mathbf{h}_{t-1}), \\
\mathbf{r}_t &= \sigma\left(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{U}_{hr}\mathbf{h}_{t-1}\right), \\
\tilde{\mathbf{h}}_t &= \tanh\left(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{U}_{rh}(\mathbf{r}_t \otimes \mathbf{h}_{t-1})\right) \\
\mathbf{h}_t &= (1 - \mathbf{z}_t)\mathbf{h}_{t-1} + \mathbf{z}_t\tilde{\mathbf{h}}_t,
\end{aligned}
$$

در روابط فوق منظور از $x_t$ همان ورودی لحظه $t$ام می‌باشد و $h_t$ نشان دهنده hidden state آن لحظه است.

البته به جای پردازش ورودی‌ها از چپ به راست، می‌توان علاوه بر آن از راست به چپ هم دنباله را مورد توجه قرار داد. در این صورت ساختار جدیدی معرفی می‌شود که biGRU نام دارد. همچنین می‌توان از GRU در یک ساختار چند طبقه استفاده نمود در نتیجه شکل زیر به عنوان مدل نهایی استفاده شده در بخش‌های بعدی به منظور انجام پردازش‌های اولیه روی صوت مورد استفاده قرار می‌گیرد:



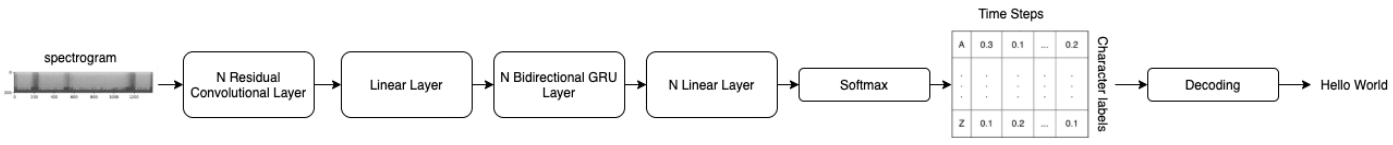
توجه کنید که در شکل فوق، هم bidirectional بودن و هم ۲ لایه بودن نشان داده شده است. البته به دلیل زیاد بودن اطلاعات صوت در ورودی، در لایه دوم یک downsampling صورت گرفته تا پردازش‌ها بهینه‌تر و سریع‌تر صورت بگیرد بدون آن که اطلاعات زیادی از دست برود.

با این توضیحات، حالا می‌توان معماری دو شبکه پیاده‌سازی شده را توضیح داد. این دو شبکه به عنوان مدل‌های آکوستیکی استفاده می‌شوند که با دریافت دنباله صوت در ورودی، توزیع احتمال حروف ادا شده را تولید می‌کنند:

**مدل اول) مدل CTC:**

در معماری اول پیاده‌سازی شده، سیگنال صوتی به ورودی یک GRU یک طرفه داده می‌شود و از مدل انتظار داریم تا در خروجی هر واحد دنباله واجی متناظر با آن را معرفی کند (یا معادلاً یک توزیع احتمالاتی روی واجها تولید نماید) البته این رویکرد مشکلاتی دارد از جمله این که طول دنباله ورودی (یعنی تعداد فریم‌های صوتی) خیلی بیشتر از دنباله هدف می‌باشد. مشکل دیگر این است که ما نمی‌دانیم از کدام خروجی باید انتظار کدام  واج را داشته باشیم چرا که ادای واجهای مختلف در سیگنال صوتی می‌تواند اندازه‌های زمانی متفاوتی داشته باشد. لذا رویکردی که در سال 2006 برای حل این مشکل ارائه شد یک تابع ضرر با نام CTC پیشنهاد داد [1] که سعی می‌کند ابتدا محتمل‌ترین مسیر را از خروجی مدل استخراج کند و بعد از آن مسئله بیشینه سازی بخت درست‌نمایی صورت پذیرد. با توجه به این که این مقاله موضوع اصلی ما نیست، از توضیح بیشتر درباره آن صرف نظر می‌شود. همچنین پیاده‌سازی تابع ضرر CTC، به کمک توابع آماده pytorch صورت گرفته که می‌توانید آن را در کدها ببینید.

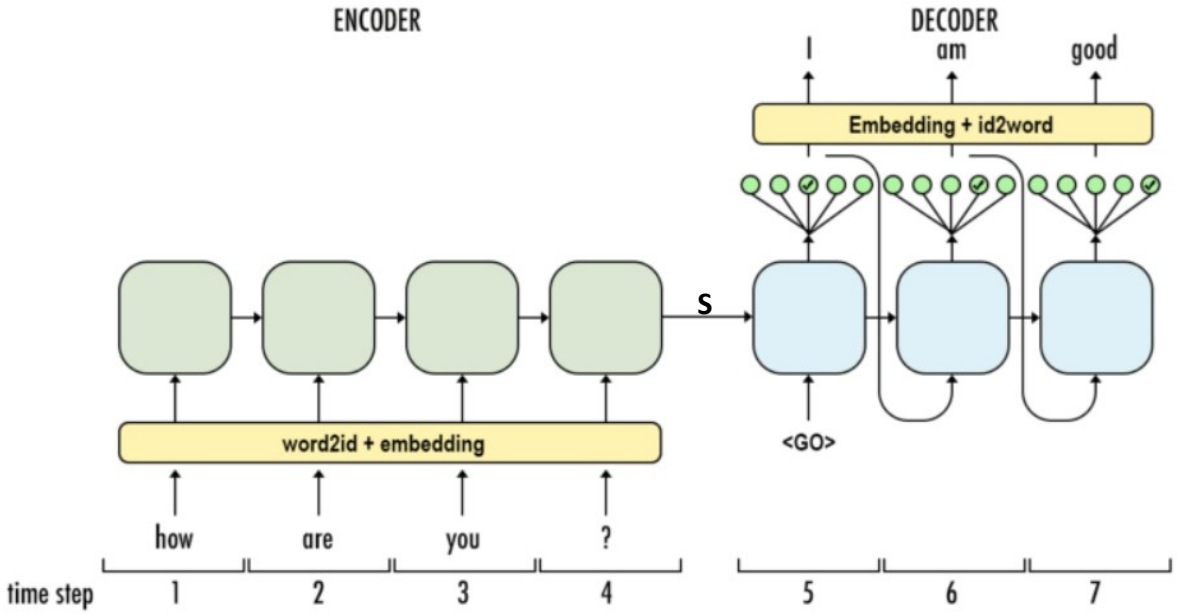
ذکر یک نکته دیگر در اینجا ضروریست. مدل یک شبکه RNN به تنهایی برای استخراج تمام اطلاعات صوت کافی نیست. لذا پیش از آن که سیگنال صوت وارد شبکه بازگشتی شود، چند لایه convolution روی آن اعمال می‌شود (توجه کنید که سیگنال صوتی یک بعد زمان دارد و یک بعد ویژگی‌های MFCC. لذا روی این دو بعد می‌توان Conv2D استفاده کرد.) البته برای آموزش بهتر وزن‌های کانولوشن، از معماری ResNet استفاده شده و یال‌های skip به منظور انتقال گرادیان اضافه شده است. نمایی از نحوه عملکرد کلی این مدل در تصویر زیر قابل مشاهده است:




کدهای مربوط به این مدل را می‌توانید در زیربخش CTC-Based model مشاهده کنید. توجه کنید که مدل‌های CTC برای انجام درست decode به یک نماد blank در مجموعه لغات نیاز دارند که این مسئله رعایت شده است. همچنین برای مدل CTC یک دیکودر greedy در بخش Metrics and Decoders پیاده‌سازی شده است که فقط در زمان تست به کار می‌آید و تاثیری در فرآیند آموزش ندارد.

**مدل دوم) مدل Attention based seq2seq:**

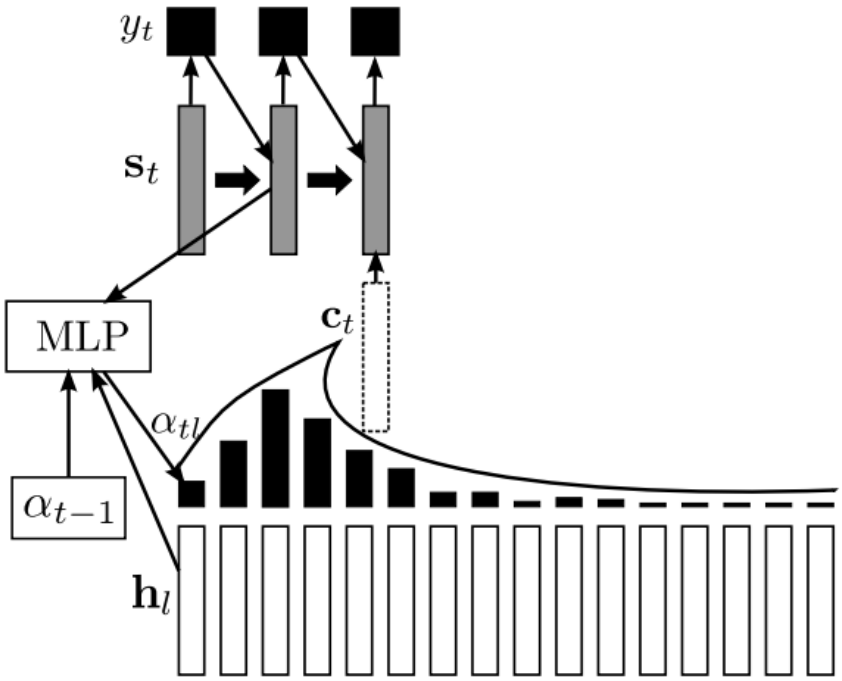
مدل seq2seq یک مدل شناخته شده در زمینه حل مسائل many-to-many می‌باشد. این معماری از دو بخش encoder و decoder تشکیل شده است. قسمت encoder دنباله ورودی را با یک ساختار بازگشتی مورد بررسی قرار داده و یک بردار ویژگی از آن استخراج می‌کند. مدل decoder که آن هم یک مدل بازگشتی است، با اتکا بر این بردار ویژگی، حروف دنباله خروجی را یکی یکی تولید می‌کند. در شماتیک زیر، می‌توان نمایی از نحوه کار این مدل را مشاهده نمود:




البته دقت کنید که شکل فوق برای یک وظیفه Question Answering طراحی شده است اما در مسئله ASR، در قسمت Encoder سیگنال صوتی قرار گرفته و در قسمت دیکودر دنباله واجی مورد نظر تولید می‌شود. آموزش چنین مدلی با رویکرد teacher forcing انجام می‌شود. یعنی در قسمت دیکودر، سیگنال حاصل از encoder به همراه واج درست به ورودی decoder داده می‌شود و از مدل انتظار می‌رود تا بتواند توزیع احتمال مناسب برای کلمه بعدی را در خروجی تولید نماید. به عبارت دیگر مدل باید قادر باشد تا با دریافت اطلاعات صوت به علاوه واج فعلی، واج بعدی را به درستی پیش‌بینی کند. تابع ضرر این شبکه چیزی نیست به جز cross entropy میان توزیع به دست آمده از مدل و خروجی‌های درست موجود در دیتاست. توجه کنید که در زمان تست لیبل‌های درست کلمات در دسترس نیست تا از آن‌ها برای generation واج‌های بعدی استفاده کنیم. لذا در این رویکرد به ابتدای تمامی جملات یک start token اضافه می‌شود تا در زمان تست مشکلی به وجود نیاید. همچنین برای آن که در زمان تست مدل متوجه شود چه زمانی باید generation را متوقف کند، یک لیبل stop token به انتهای تمامی جملات اضافه می‌شود. لذا در زمان تولید خروجی در زمان تست، هرگاه فرایند decoding به stop token برسد، تولید دنباله را متوقف می‌کند.

یکی از معایبی که این مدل دارد آن است که تلاش می‌کند تا کل اطلاعات صوت ورودی را در یک بردار ویژگی خلاصه کند (در شکل با S نمایش داده شده است). این مسئله می‌تواند بخش decoder را با مشکل مواجه کند. لذا در سال‌های اخیر مکانیزمی به نام مکانیزم توجه برای الگوریتم‌های seq2seq معرفی شده است. این مکانیزم توسط [3] برای بار اول در وظیفه Machine Translation معرفی شد و بعداً ورژن کامل شده دیگری از آن توسط [4] ارائه شد.

بر طبق این رویکرد، به جای آن که کل ورودی را در یک بردار S خلاصه کنیم، ورودی پردازش شده (یعنی خروجی encoder) را در تمامی لحظات در اختیار decoder می‌گذاریم. اما از آن جا که پردازش همه ورودی در هر سلول بازگشتی دیکودر ممکن نیست، لازم است تا آن را به نوعی خلاصه کنیم. برای این منظور، مدل یک توزیع احتمال روی دنباله ورودی یاد می‌گیرد که اهمیت هر قطعه زمانی از ورودی را نشان می‌دهد. بعد از یافتن این توزیع احتمال، دنباله ورودی در راستای بعد زمان به صورت وزن‌دار خلاصه می‌شود تا در نهایت یک بردار خلاصه به ازای کل دنباله ورودی حاصل شود. البته توجه کنید که توزیع احتمال یادگرفته شده توسط مدل، در هر گام زمانی تغییر می‌کند و لذا مدل به نوعی یاد می‌گیرد که در زمان تولید خروجی در هر لحظه، به کدام قسمت‌های دنباله ورودی توجه کند. از این رو این مکانیزم را به نام attention نیز می‌شناسند. در شکل زیر، نمونه‌ای از نحوه عملکرد دیکودر آورده شده است:




البته با توجه به این که توجه به کل دنباله ورودی بازهم کار مشکلی است، معمولاً به جای مکانیزم توجه global، یک پنجره در نظر می‌گیرند و مکانیزم توجه را در همان پنجره اعمال می‌کنند.

تا اینجای کار دو معماری شبکه‌های عصبی را برای ASR معرفی کردیم. توجه کنید که در کد تحویل داده شده، جابجایی میان مدل‌های CTC و seq2seq به سادگی انجام می‌شود. کافیست متغیر SEQ2SEQ_Training را از قسمت Flagها تغییر دهید تا آموزش شبکه مورد نظر انجام شود و نتایج آورده شده در گزارش تولید شوند. پیش از پرداختن به نتایج آموزش شبکه‌ها، قصد داریم در قسمت بعد دو الگوریتم را معرفی نماییم به تولید با کیفیت‌تر خروجی در زمان decoding کمک می‌کنند. این دو روش عبارتند از k-beam search و استفاده از language model.

## بخش چهارم) الگوریتم‌های decoding و مدل زبانی

چیزی که مدل‌های seq2seq در خروجی خود تولید می‌کنند، توزیع احتمال واج بعدی است به شرط دنباله صوتی ورودی و به شرط همه کلمات قبلی. در واقع اگر در گام $t$ام دیکودر قرار داشته باشیم، چیزی که دیکودر در خروجی خود می‌دهد، عبارت است از :

$$p(y_t|y_{t-1}, y_{t-2}, \dots, y_1, S)$$

اما ما در زمان دیکودینگ به دنبال یافتن دنباله‌ای هستیم که توزیع احتمال «جمله» را ماکزیمم کند. یعنی بع دنبال عبارت زیر هستیم:

$$\underset{y_1 \dots y_T}{\operatorname{argmax}} \prod_{t=1}^{T} p(y_t|y_{t-1}, y_{t-2}, \dots, y_1, S)$$

اما بدیهی است که امتحان کردن همه چینش‌های $y_1 \dots y_T$ برای یافتن جمله با بیشینه احتمال شدنی نیست. لذا پیشنهادی که در این بخش مطرح می‌شود، آن است که به صورت greedy عمل کنیم. لذا به جای ماکسیمم کردن احتمال فوق به صورت زیر عمل می‌کنیم:

$$\underset{y_1 \dots y_T}{\operatorname{argmax}} \prod_{t=1}^{T} p(y_t|y_{t-1}, y_{t-2}, \dots, y_1, S)$$
$$\approx \underset{y_2}{\operatorname{argmax}}\ p(y_2|y_1, S), \underset{y_3}{\operatorname{argmax}}\ p(y_3|y_1, y_2, S), \dots, \underset{y_T}{\operatorname{argmax}}\ p(y_T|y_{T-1}, y_{T-2}, \dots, y_1, S)$$

بنابراین در الگوریتم حریصانه از توزیع واج اول نمونه می‌گیرند و سپس آن را به مدل می‌دهند تا توزیع واج بعدی تولید شود و همینطور جلو می‌روند تا نهایتاً stop token ایجاد شود.

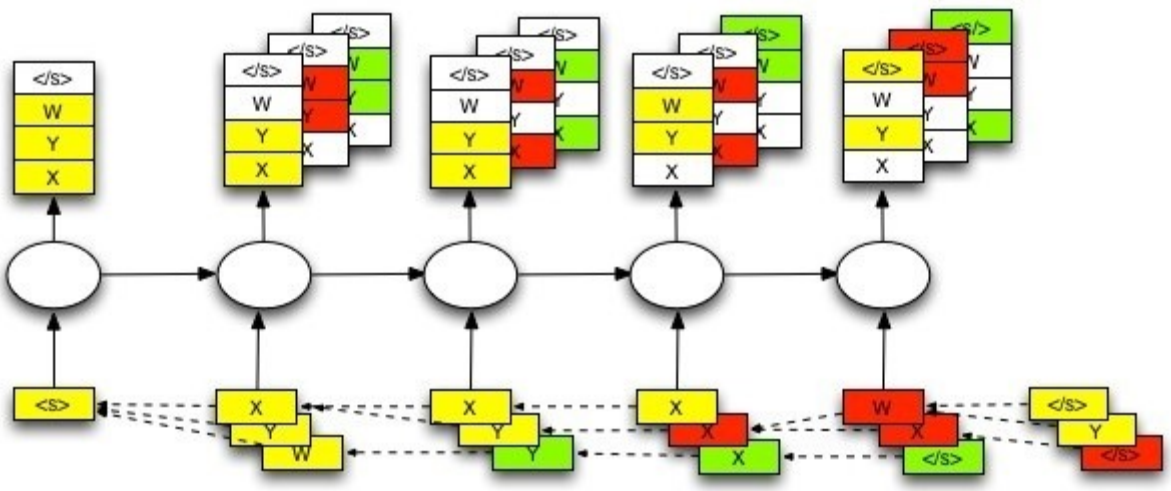
بدیهی است که روش حریصانه فوق ممکن است به جواب واقعاً خوبی منجر نشود. مثلاً فرض کنید هدف تولید جمله «من از دور می‌آیم» باشد. مدل وقتی به دیکود کردن کلمه «دور» می‌رسد، در انتخاب واج «د» و «ن» شک خواهد داشت چون این دو ساختار آوایی مشابهی دارند. اگر در اینجا به صورت حریصانه عمل کند، ممکن است کلمه «نور» را اشتباهاً انتخاب کند. با توجه به این که در گام‌های بعدی امکان اصلاح این اشتباه برای مدل وجود ندارد، توزیع مورد نظر روی واج‌ها و کلمه‌های بعدی نیز به هم می‌ریزد و در کل نمی‌توان انتظار خروجی مناسبی از این مدل داشت. در ادامه دو الگوریتم که می‌توانند جایگزین دیکود حریصانه شود را معرفی می‌کنیم:

**الگوریتم k-beam search :**

الگوریتم k-beam search یک الگوریتم مشهور است که در قسمت decoding جایگزین الگوریتم حریصانه شده و لذا تنها در زمان تست می‌تواند مفید باشد. در این الگوریتم به جای این که مانند greedy تنها واج با بیشترین احتمال انتخاب شود و دیگر انتخاب‌ها کاملاً رها شوند، همیشه یک صف kتایی نزد خود نگه می‌دارد و k مسیر را همزمان در دیکودر جلو می‌برد. این k مسیر در واقع مسیرهایی هستند که بیشترین احتمال را تولید می‌کنند. مسیرهایی که جزء kتای برتر نباشند، دور ریخته می‌شوند و وارد صف نخواهند شد. بعد از آن که تمامی k مسیر برتر به stop token خود رسیدند، حالا وقت آن می‌رسد که یکی از میان این‌ها انتخاب کنیم لذا در آخرین گام، یک جمله را از میان این kتا که بیشترین احتمال را به دست آورده انتخاب می‌کنیم و به عنوان خروجی باز می‌گردانیم. برای این منظور لازم است که احتمالات هر مسیر از نقطه فعلی تا شاخه اصلی را ذخیره کرده باشیم (این کار را می‌توان با کمک ضرب احتمالات در یکدیگر از اول تا اینجا یا با کمک جمع کردن log احتمالات انجام داد):

$$\log \prod_{t=1}^{T} p(y_t|y_{t-1}, y_{t-2}, \dots, y_1, S) = \sum_{t=1}^{T} \log p(y_t|y_{t-1}, y_{t-2}, \dots, y_1, S)$$

گرچه این الگوریتم از نظر بهینه‌بودن خیلی بهتر از الگوریتم حریصانه است، اما هنوز ممکن است به احتمال اندکی در تولید جواب بهینه، مسیر واقعی را از دست بدهد. نمونه‌ای از طرز کار این الگوریتم با k=3 در شکل زیر نشان داده شده است:

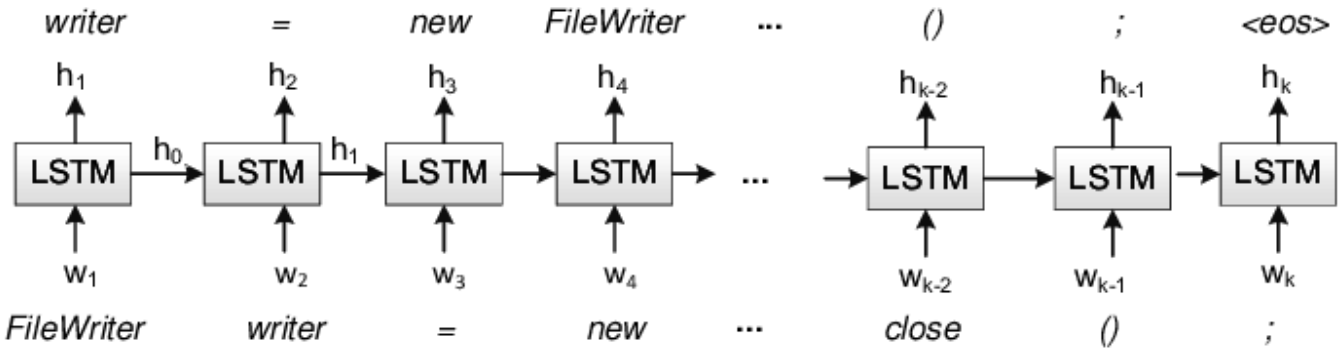



توجه : الگوریتم k beam search صرفاً برای مدل seq2seq پیاده سازی و مقایسه شده است (چرا که تمرکز در این مقاله فقط برروی مدل seq2seq بوده است). ورژنی از beam search وجود دارد که مختص پیاده‌سازی CTC است که در این پروژه به دلیل نامربوط بودن پیاده‌سازی نشده است.

**استفاده از مدل زبانی (Language Model) در بخش دیکودر:**

مدل زبانی، مدلی است که مستقل از ویژگی‌های آکوستیکی زبان بوده و صرفاً روی کلمات و واج‌های آن تعریف می‌شود. در واقع مدل زبانی قادر است تا با دریافت یک جمله، احتمال بروز چنین جمله‌ای را در زبان مورد نظر به عنوان خروجی تولید نماید. مدل زبانی می‌تواند روی مجموعه متون یک زبان تعریف شود و نیازی به داشتن فایل صوتی ندارد. استفاده از مدل زبانی در بخش دیکودر شبکه seq2seq می‌تواند به بهبود کیفیت جملات تولید شده کمک نماید.

در این پروژه یک LM با رویکرد استفاده از شبکه عصبی روی وظیفه next word prediction آموزش داده شده است. در واقع شبکه ما یک شبکه بازگشتی با واحدهای LSTM است که با دریافت یک جمله، احتمال وجود آن جمله در آن زبان را تولید می‌نماید. این مدل ساختاری مشابه ساختار زیر دارد:




توجه کنید که واحدهای ورودی نشان داده شده در شکل فوق از جنس کلمات هستند در صورتی که مدل LM ما در سطح واج تعریف شده و آموزش دیده است. همچنین نکته دیگری که باید اشاره شود آن است که مدل seq2seq به دلیل ساختاری که دارد، احتمالاً خودش می‌تواند بدون آموزش مستقیم و تنها از طریق دیدن داده‌های صوتی، مدل زبانی را به طور ضمنی یاد بگیرد بنابراین انتظار می‌رود عملکرد آن با حضور یا عدم حضور مدل زبانی خارجی تفاوت چندانی پیدا نکند. این مسئله در مقاله مرجع نیز ذکر شده است اما در نهایت تصمیم گرفته شده که از مدل زبانی به طور خارجی استفاده شود.

برای اعمال مدل زبانی در قسمت دیکودر شبکه seq2seq، کافیست تا در گام محاسبه احتمالات، به عبارت $\sum_{t=1}^{T} \log p_{acoustic}$ جمله دیگری به صورت $\sum_{t=1}^{T} \log p_{acoustic} + \log p_{LM}$ اضافه نمود که منظور از $p_{LM}$ احتمالی است که مدل زبانی با دیدن آن جمله تولید می‌کند.

توجه:

رویکرد درست در به کارگیری مدل زبانی آن است که مدل را روی یک corpus عظیم متنی آموزش داده و سپس در مسئله ASR استفاده کنیم. از آن جا که چنین دیتاستی فعلاً برای زبان فارسی در اختیار من نبود، مدل زبانی را روی جملات ادا شده موجود در دیتاست آموزش دادم که البته احتمالاً کار چندان درستی نیست (به این دلیل که تعداد جملات بسیار کم است) و احتمالاً پاسخ‌های تولید شده دیکودر را به شدت به سمت خوب شدن بایاس می‌کند. هرچند به نظر می‌رسد استفاده از این LM به نوعی کار درستی نیست، اما چون مقاله مرجع نیز این رویکرد را در پیش گرفته بود، من نیز تصمیم گرفتم تا از این مدل در پیاده سازی نهایی خودم استفاده کنم.

متغیری با نام USE_LM در کد وجود دارد که می‌توانید با تغییر آن استفاده از مدل زبانی را روشن یا خاموش کنید.

حال که تمام پیاده‌سازی‌ها و جزئیات الگوریتم‌ها شرح داده شد، نوبت به آن می‌رسد تا در بخش بعد نتایج آموزش برای هر دو ساختار معرفی شده آورده شود و مقایسه‌های لازم صورت پذیرد.

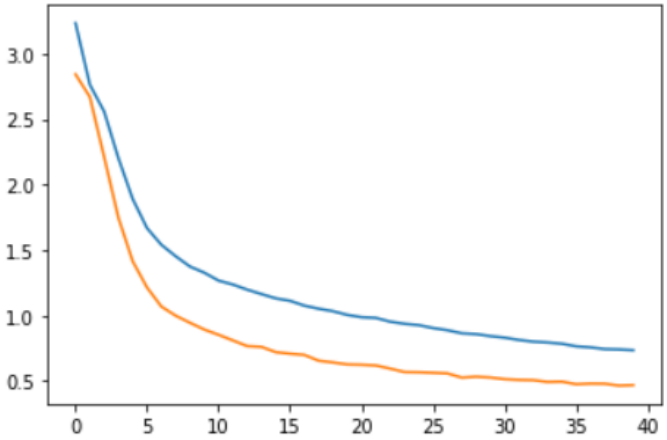
## بخش پنجم) نتایج آموزش شبکه‌ها

در این بخش می‌خواهیم عملکرد مدل‌های پیاده‌سازی شده را مقایسه کنیم. برای این منظور، سه معیار را مد نظر قرار می‌دهیم. معیار اول روند تغییرات تابع ضرر آموزش شبکه‌هاست. البته باید توجه کرد که معیارهای سنجش CTC و شبکه Seq2Seq با همدیگر متفاوت است. لذا کار درستی نیست اگر نمودار تابع ضرر این دو با یکدیگر مقایسه شود. نمودار تابع ضرر صرفاً می‌تواند نشانگر بهبود هر مدل به صورت جداگانه در طول زمان باشد.

معیار دیگری که در ASR مرسوم است، معیار Word Error Rate (WER) می‌باشد. این معیار نشانگر این است که در خروجی الگوریتم چند درصد کلمات به درستی تولید شده‌اند. همچنین معیار دیگری وجود دارد به نام معیار CER که عملکردی شبیه WER دارد، با این تفاوت که در CER هدف شمردن تعداد کاراکترهای تولید شده صحیح است.

### نتایج آموزش شبکه CTC روی زبان فارسی

برای آموزش شبکه CTC روی زبان فارسی، از بهینه ساز AdamW با نرخ یادگیری 5e-4 استفاده شده است. تابع ضرر آموزش این شبکه با توضیحات گفته شده، بعد از گذشت حدود ۴۰ ایپاک به مقداری کمتر از ۱ می‌رسد. در شکل زیر می‌توانید روند آموزش را روی داده‌های train و validation مشاهده کنید:




بعد از اتمام آموزش برای این شبکه مقدار WER چیزی حدود ۴۶٪ و مقدار CER چیزی حدود ۱۳٪ به دست آمده است. همچنین نمونه‌ای از خروجی مطلوب (target) و خروجی دیکود شده (preds) را می‌توانید در تصویر زیر مشاهده کنید:

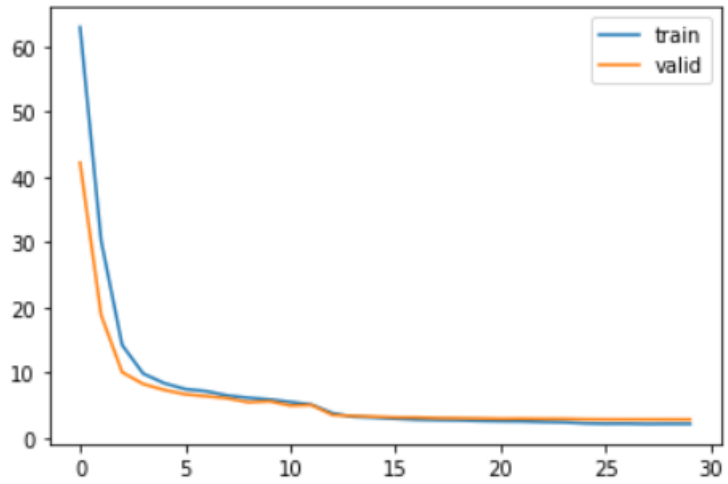
```
preds: <j g/ra, ]is/lne ]aravade kxubi d/.t>
targets: <j g/r/[ ]ems/l dar/made xubi d/.t>
Test set: Average loss: 0.4751, Average CER: 0.138627 Average WER: 0.4646
```

توجه:

فایل نوت بوک حاوی نمودارها و خروجی‌های نمونه به این گزارش پیوست شده است. اصلی‌ترین هدف ما در این پروژه پیاده‌سازی seq2seq روی زبان فارسی بود. لذا کدهای مربوط به این مورد در کنار این گزارش قرار دارد. در مورد قسمت‌های دیگر از جمله CTC_English، CTC_Persian و Seq2Seq_English می‌توانید کد آن‌ها را در پوشه Others مشاهده کنید. البته توجه کنید هر ۴ فایل ذکر شده دارای کدهای دقیقاً یکسان هستند. صرفاً در هر یک از فایل‌ها flagهای مربوط به ساختار مورد نظر تنظیم شده و آموزش صورت گرفته. همچنین نمودارها و نتایج آموزش هر مدل را می‌توانید در نوت بوک متناظر آن مشاهده کنید.

### نتایج آموزش شبکه Seq2Seq روی زبان فارسی

آموزش شبکه Seq2Seq روی دیتاست فارس‌دات نیز موفقیت آمیز بود و نتایج خیلی خوبی به دست آمده است. بهینه ساز مورد استفاده در اینجا نیز یک بهینه ساز AdamW، با نرخ یادگیری آغازین 5e-4 و Scheduler پله با ضریب 0.1 به ازای هر 12 گام می‌باشد. نمودار بهبود تابع ضرر را در 40 ایپاک آموزش می‌توانید مشاهده نمایید:



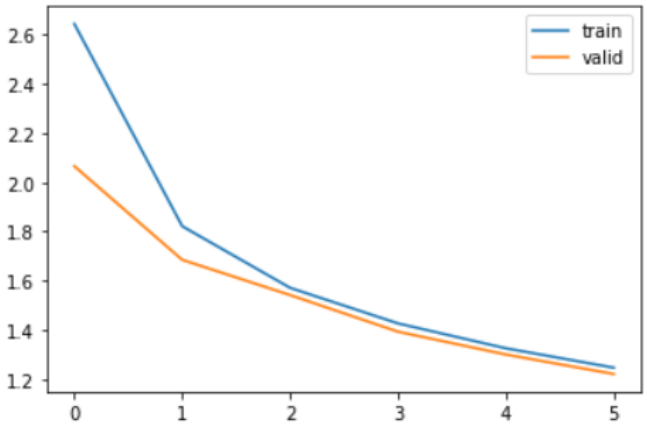

در مورد معیارهای WER و CER به ترتیب به امتیازهای ۲۷٪ و ۲۱٪ دست یافته‌ایم که نتایج خیلی خوبی محسوب می‌شوند. در تصویر زیر، می‌توانید یک نمونه از برچسب واقعی و پیش‌بینی مدل را مشاهده کنید:

```
preds: <j ]in lokomotiv r/nande nad/rad>
targets: <j ]in lokomotiv r/nande nad/rad>
Test set: Average loss: 2.8181, Average CER: 0.210651 Average WER: 0.2777
```

مجدداً ذکر می‌شود که نتایج این بخش در فایل Speech_ASR_Seq2Seq_persian ضمیمه شده است.

### نتایج آموزش شبکه CTC روی زبان انگلیسی

برای آموزش شبکه CTC روی زبان انگلیسی، از مشخصاتی شبیه به زبان فارسی استفاده کرده‌ایم (در بالا ذکر شد). توجه کنید که دیتاست انگلیسی ما خیلی خیلی بزرگتر از دیتاست فارسی بود و لذا انجام آموزش روی آن بسیار سنگین و زمانبر بود به طوری که هر ایپاک حدوداً ۲:۳۰ ساعت زمان نیاز داشت. با توجه به این که Colab محدودیت اجرا روی GPU دارد و بعد از مدت زمان مشخصی دسترسی به GPU را به صورت خودکار از بین می‌برد، تنها امکان آموزش مدل برای ۶ ایپاک وجود داشت. به هر حال در همین ۶ ایپاک هم می‌توان بهبود تابع ضرر و معیارهای CER و WER را مشاهده نمود:



```
preds: <the pron fwas ant fhreou long slops i had left a al together>
targets: <the proof was in three long slips i had left them all together>
Test set: Average loss: 1.2166, Average CER: 0.367458 Average WER: 0.8656
```

### نتایج آموزش شبکه Seq2Seq روی زبان انگلیسی

مجدداً به دلیل مشکلی که در بخش قبل ذکر شد و به دلیل سنگین بودن بیش از حد دیتاست انگلیسی، امکان آموزش مناسب شبکه Seq2Seq روی زبان انگلیسی وجود نداشت. البته توجه کنید که کدهای مربوط به این تنظیمات نوشته شده و در صورت در اختیار داشتن یک سرور مناسب می‌توان آموزش شبکه Seq2Seq روی زبان انگلیسی را به طور کامل انجام داد.

به دلیل آن که دیتاست انگلیسی مورد استفاده توسط ما (LibreSpeech) توسط مقاله اصلی استفاده نشده و به عنوان یک مورد مازاد در این پروژه پیاده‌سازی شده بود، بنابراین به دلیل کمبود امکانات از این قسمت صرف نظر شده است و نتیجه‌ای در اینجا ذکر نمی‌شود.

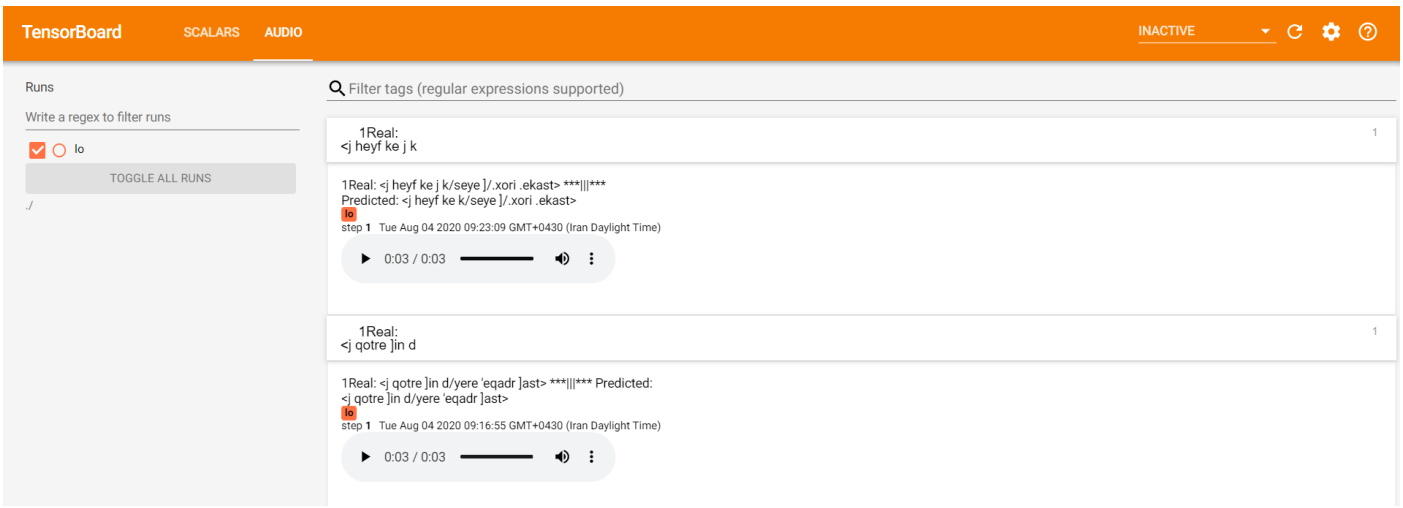
### استفاده از مدل زبانی

همانطور که بالاتر نیز اشاره شد، مدل زبانی به طور کلی می‌تواند باعث بهبود معیارهای CER و WER در زمان تست شود. البته پیشتر نیز اشاره شد که مدل Seq2Seq می‌تواند به صورت ضمنی مدل زبانی را هم از داده صوتی یادبگیرد لذا نمی‌توان بهبود چشمگیری را انتظار داشت. در این قسمت مدل زبانی روی Seq2Seq فارسی اعمال شد و همانطور که انتظار داشتیم در حد یکی دو درصد بهبود حاصل شد. یعنی CER به مقدار ۲۰٪ و WER به ۲۶٪ رسیده است:

```
preds: <j bim/riye sar] peyd/ karde]am>
targets: <j bim/riye sar] j peyd/ karde]am>
Test set: Average loss: 2.8067, Average CER: 0.204008 Average WER: 0.2680
```

توجه:

برای آن که بخش گزارش تکمیل شود، یکی دو نمونه از صوت‌های نمونه و دیکود شده آن‌ها ضمیمه شده است. این فایل‌های صوتی نمونه را می‌توانید در پوشه Results مشاهده کنید. البته توجه کنید که فایل‌های ارسالی توسط کتابخانه Tensorboard آماده شده‌اند و لذا برای باز کردن آن‌ها به این برنامه نیاز دارید. یک نمونه از محیط tensorboard و متن‌های مربوط به صورت را می‌توانید در تصویر زیر مشاهده کنید:



**مقایسه نتایج پیاده‌سازی شده با نتایج مقاله مرجع:**

در این قسمت دو جدول زیر را از مقاله مرجع (سمت راست) به همراه نتایج به دست آمده از پیاده‌سازی‌های صورت گرفته (جدول سمت چپ) ذکر می‌کنیم. توجه کنید که دیتاست این دو جدول متفاوت است (یکی wall street journal و دیگری Farsdot) و لذا احتمالاً مقایسه این دو جدول با هم کار درستی نیست. با این حال نتایج به صورت زیر است. همچنین توجه کنید که نام دیگر مدل seq2seq، همان encoder decoder می‌باشد:

| Model | CER | WER |
|---|---|---|
| Seq2Seq | 21 | 27 |
| Seq2Seq + LM | 20 | 26 |
| CTC | 13 | 46 |

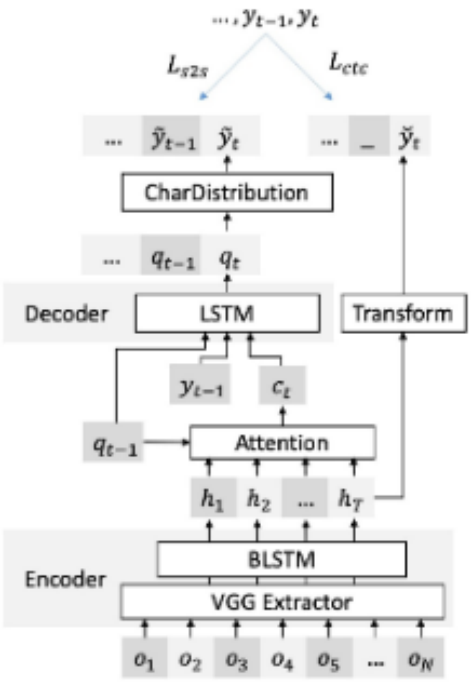
| Model | CER | WER |
|---|---|---|
| Encoder-Decoder | 6.4 | 18.6 |
| Encoder-Decoder + bigram LM | 5.3 | 11.7 |
| Encoder-Decoder + trigram LM | 4.8 | 10.8 |
| Encoder-Decoder + extended trigram LM | 3.9 | 9.3 |
| Graves and Jaitly (2014) | | |
| CTC | 9.2 | 30.1 |
| CTC, expected transcription loss | 8.4 | 27.3 |
| Hannun et al. (2014) | | |
| CTC | 10.0 | 35.8 |
| CTC + bigram LM | 5.7 | 14.1 |
| Miao et al. (2015), | | |
| CTC for phonemes + lexicon | - | 26.9 |
| CTC for phonemes + trigram LM | - | 7.3 |
| CTC + trigram LM | - | 9.0 |

## بخش ششم) بررسی مقالات مرتبط

در روند توضیحات فوق، چند مقاله به طور مفصل توضیح داده شد. از جمله [1] و [2] که متمرکز بر پیاده‌سازی *CTC* بودند. همچنین در مورد مقاله [3] صحبت شد که هدف آن پیاده‌سازی *Seq2Seq* و مدل Attention برای وظیفه *machine translation* بود.

در این قسمت یک مقاله مرتبط دیگر با بحث‌های صورت گرفته معرفی می‌شود:

مدل تصویر روبه‌رو خلاصه‌ای است از مدلی که در دو مقاله [5] و [6] پیشنهاد شده است. در این مدل به نوعی هر دو روش *CTC* و *Seq2Seq* ترکیب شده‌اند و همانطور که دیده می‌شود، دو تابع ضرر در خروجی به دست آمده و هر دو تابع به صورت همزمان بهینه‌سازی می‌شوند.




Example Architecture of End-to-End ASR

برای این منظور، در ورودی ابتدا توسط یک معماری *VGG* اطلاعات صوتی از روی فریم‌ها استخراج می‌شود و سپس به یک *LSTM* دو طرفه داده می‌شود. خروجی حاصل از یک مسیر مستقیماً به سمت خروجی رفته و در محاسبه ضرر *CTC* استفاده می‌شود. در مسیر دیگر *BLSTM* نقش *encoder* را به خود گرفته و در قسمت دیکودر با یک مکانیزم توجه خروجی‌ها تولید می‌شود. خروجی‌های تولید شده از این مسیر هم تابع ضرر *Seq2Seq* را تشکیل می‌دهند و جمع دو تابع ضرر روی هم می‌تواند مدل را به خوبی آموزش دهد.

# References


[1] Graves, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural," *ICML,* 2006.

[2] Hannun, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns," *arXiv,* 2014b.

[3] Bahdanau, "Neural machine translation by jointly learning to align and translate," *ICLR,* 2015.

[4] Luong, "Effective Approaches to Attention-based Neural Machine Translation," *arXiv,* 2015.

[5] A. H. Liu, "Adversarial Training of End-to-end Speech Recognition Using a Criticizing Language Model," *arXiv,* 2018.

[6] A. H. Liu, "Sequence-to-sequence Automatic Speech Recognition with Word Embedding Regularization and Fused Decoding," *arXiv,* 2019.