دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی
مهندسی کامپیوتر

پیاده سازی یک سرویس تبدیل صدا مبتنی بر یادگیری عمیق

نگارش

کسرا دماوندی اصلی

استاد راهنما

حسین صامتی

تیر ۱۴۰۲

به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی

این پایان‌نامه به عنوان تحقق بخشی از شرایط دریافت درجه کارشناسی است.

عنوان: پیاده سازی یک سرویس تبدیل صدا مبتنی بر یادگیری عمیق

نگارش: کسرا دماوندی اصلی

کمیته ممتحنین

استاد راهنما: حسین صامتی امضاء:

استاد مشاور: استاد مشاور امضاء:

استاد مدعو: استاد ممتحن امضاء:

تاریخ:

سپاس

از استاد بزرگوارم جناب دکتر صامتی که با کمک‌ها و راهنمایی‌های بی‌دریغشان، مرا در به سرانجام رساندن این پایان‌نامه یاری داده‌اند، تشکر و قدردانی می‌کنم. همچنین از آقای سروش گوران که در تمام فرایند انجام این پروژه مرا مشاوره دادند و سایر عزیزانی که مرا یاری نمودند صمیمانه سپاسگزارم.

چکیده

تبدیل صدا فرایند تبدیل صدای یک گوینده به گوینده دیگر است در حالی که محتوای زبانی گفتار باید حفظ شود. از این فناوری در حوزه‌هایی چون رسانه، تقویت ارتباط گفتاری و سرگرمی استفاده می‌شود. امرو غالبا پروژه‌های تبدیل صدا مبتنی بر شبکه‌های عصبی می‌باشند. در این پروژه، هدف ما پیاده سازی یک سرویس تبدیل صدا می‌باشد که قابلیت دریافت فایل صوتی از کاربر و ارسال نتیجه به او را دارد. Backend این سرویس با Django پیاده‌سازی شده است. همچنین با استفاده از MongoDB یک پایگاه داده برای ذخیره اطلاعات مورد نیاز ساخته شده است. در نهایت با ایجاد یک بات تلگرام این سرویس را به بهره‌برداری رساندیم.

کلیدواژه‌ها: تبدیل صدا، شبکه‌های عصبی، پایگاه داده، Backend

فهرست مطالب

فهرست شکل‌ها

فصل ۱

مقدمه

۱ـ۱ تعریف مسئله

امروزه هوش مصنوعی به یکی از حوزه‌های پرطرفدار تکنولوژی بدل شده است که به مسائل گوناگونی از جمله پردازش گفتار، متن، تصویر و... می‌پردازد. یکی از مسائلی که در این حوزه به آن پرداخته می‌شود تبدیل صدا است. تبدیل صدا فرایند تغییر صدای یک گفتار به صدای شخصی دیگر(گوینده مقصد) است به گونه‌ای که طبیعی بودن آن و همچنین محتوای زبانی مانند کلمات، لحن، ریتم و تاکیدها حفظ شود. این فرایند معمولا در ۳ مرحله صورت می‌گیرد:

۱. استخراج اطلاعات مربوط به محتوای گفتار

۲. استخراج اطلاعات مربوط به صدای گوینده مقصد

۳. ساخت موج صوتی جدید با استفاده از اطلاعات به دست آمده.

۱ـ۲ کاربردها

از تبدیل صدا می‌توان در رسانه‌ها استفاده کرد. برای مثال در صنعت دوبله فیلم و سریال می‌توان از صدای صداپیشگان قدیمی که دیگر فعالیت نمی‌کنند استفاده کرد. از دیگر کاربردهای تبدیل صدا می‌توان به تقویت ارتباط گفتاری اشاره نمود که از یک موج صوتی کم کیفیت(دارای نویز) یک موج صوتی با کیفیت بالاتر به دست آورد. همچنین این حوزه کاربرد فراوانی برای سرگرمی و استفاده در فضاهای مجازی دارد. برای

مثال افراد می‌توانند صدای خود را به صدای افراد معروف یا صداهای ترسناک یا خنده‌دار تبدیل نمایند. بنابراین وجود بستری برای ارائه سرویس تبدیل صدا بسیار مفید و پرطرفدار خواهد بود و به راحتی می‌توان از آن استفاده تجاری نمود.

۱ـ۳ انوع تبدیل صدا

مدل‌های تبدیل صدا با توجه به روشی که استفاده می‌کنند در نهایت به صورت صفر شات[۱] یا تک شات[۲] انجام می‌شوند. تک شات به معنای آن است که مدل در فرایند یادگیری باید گوینده مقصد را دیده باشد که نوعی محدودیت به حساب می‌آید. ولی مدل‌های صفر شات این محدودیت را ندارند. همچنین برخی مدل‌ها قابلیت تبدیل صدا به صورت بی‌درنگ[۳] را دارند. در این پروژه ما از یک مدل تک شات استفاده کردیم که قابلیت بی‌درنگ بودن را نیز ندارد.

۱ـ۴ اهداف پژوهش

هدف از این پروژه پیاده سازی بستری برای ارائه سرویس تبدیل صدا می‌باشد که با توجه به کیفیت آن می‌توان در حوزه‌های گوناگون از جمله صنعت دوبله آن را به کار برد و تنها لازم است یک دیتاست از صدای گوینده یا صداپیشه موردنظر جمع‌آوری نموده و مدل را با آن آموزش دهیم. ما با توجه به امکاناتی که در اختیار بود فرایند یادگیری را برروی داده‌های صوتی خانم صارمی(گوینده) انجام دادیم.

۱ـ۵ ساختار پایان‌نامه

این پایان‌نامه در شش فصل به شرح زیر ارائه می‌شود. در فصل دوم به شرح مختصر دو مدل بسیار مهم در حوزه پردازش گفتار می‌پردازیم که از این دو مدل در مقاله‌ی اصلی مورد استفاده در این پایان‌نامه استفاده شده است. درفصل سوم به شرح ساختار و عملکرد مدل FreeVC که مبنای اصلی این پردوژه است می‌پردازیم. در فصل چهارم به نحوه‌ی آموزش و استنتاج مدل پرداخته شده است. در فصل پنجم هم پیاده‌سازی سرویس با استفاده از Django و ساختار پایگاه داده را شرح داده‌ایم. فصل ششم به جمع‌بندی کارهای انجام شده در این پژوهش و ارائه‌ی پیشنهادهایی برای انجام کارهای آتی خواهد پرداخت.

[1] Zero Shot
[2] One Shot
[3] Real Time

فصل ۲

کارهای پیشین

در این پروژه از یک مدل تبدیل صدا به نام FreeVC استفاده شده است. این مدل خود از مدل‌های دیگری درون ساختار خود استفاده کرده است که در این فصل به بررسی مختصر این مدل‌ها می‌پردازیم و در فصل بعد مدل FreeVC را شرح خواهیم داد.

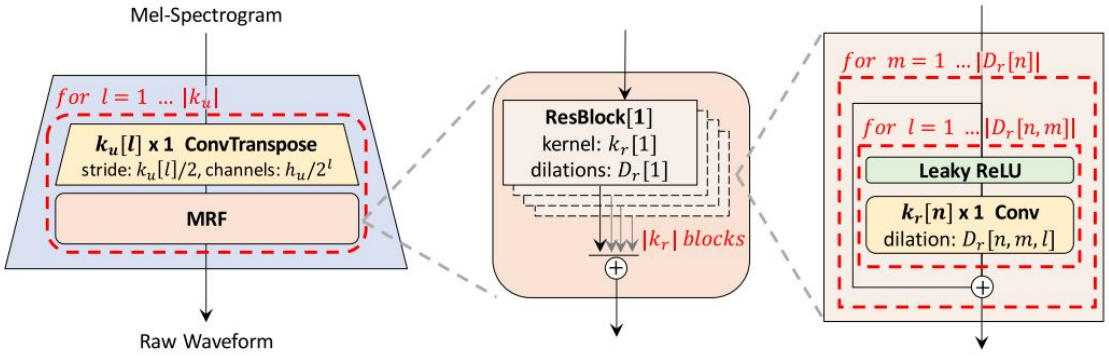
HiFi GAN ۲ـ۱

یکی از مدل‌هایی که در زمینه تولید گفتار نتایج خیلی خوبی گرفته است مدل HiFi GAN می‌باشد. این مدل که در سال ۲۰۲۰ ارائه شده است هم از لحاظ محاسباتی و هم از لحاظ کیفیت و کارایی نسبت مدل‌های پیشین خود عملکرد بهتری دارد. HiFi GAN یک مدل تولید کننده[۱] می‌باشد که بر اساس یادگیری تخاصمی[۲] کار می‌کند به این صورت که یک تولید کننده(Generator) و دو جدا کننده(Discriminator) در مقابل هم قرار می‌گیرند که در زیر به شرح عملکرد هریک می‌پردازیم:

Generator ۲ـ۱ـ۱

شبکه تولید کننده در HiFi GAN بر اساس CNN[۳] می‌باشد. ساختار این تولید کننده در شکل ۲ـ۱ آمده است.

[1] Generative
[2] Adversarial
[3] Convolutional Neural Network



شکل ۲ـ۱: ساختار Generator در HiFi GAN

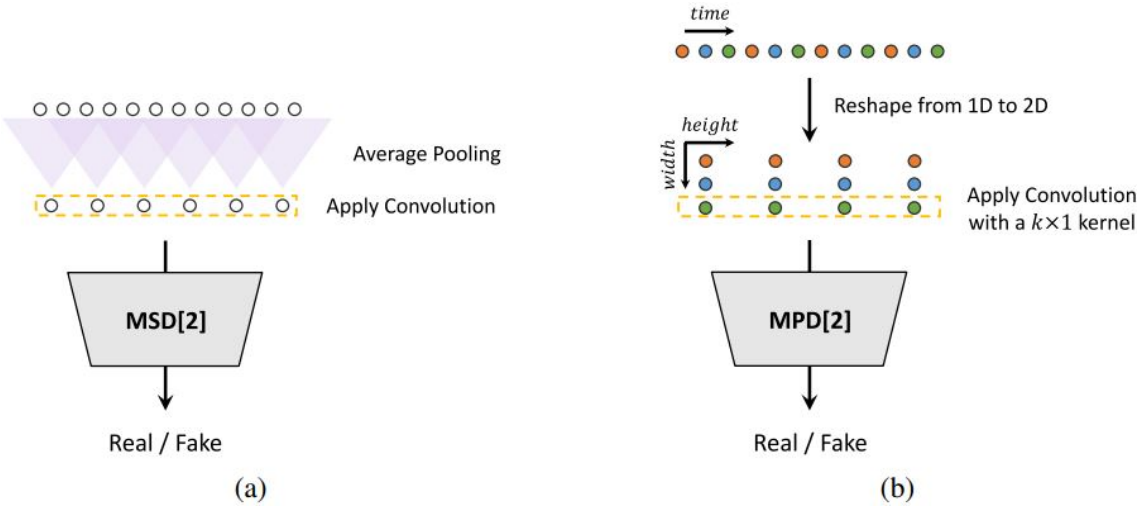
ورودی این ماژول، اسپکتروگرام مل[۴] می‌باشد و به عنوان خروجی یک موج صوتی می‌دهد. بر روی هر اسپکتروگرام ورودی تعدادی تبدیل Convolution اجرا می‌شود و پس از هر Convolution یک ماژول MRF[۵] قرار دارد که شامل تعدادی بلاک[۶] می‌باشد که به طور موازی اجرا می‌شوند که هر یک مقداری را به حاصل Convolution اضافه می‌کند.

۲ـ۱ـ۲ Discriminator

سیگنال صوتی یک گفتار شامل تعداد زیاد موج سینوسی با طول موج‌های متفاوت است. یکی از مسائلی که در پردازش گفتار با آن روبرو هستیم شناسایی تمام الگوهای متناوب موجود در یک سیگنال صوتی می‌باشد. همچنین برای مدل کردن هر چه بهتر یک سیگنال صوتی لازم است وابستگی‌های بلند مدت آن را شناسایی کنیم. برای مثال یک واج[۷] ممکن است بیش از ۱۰۰ میلی‌ثانیه طول بکشد که باعث ایجاد همبستگی میان بیش از ۲۲۰۰ نمونه[۸] مجاور می‌شود. در این مدل برای حل این مشکلات دو Discriminator قرار داده شده است که به شرح هر یک از آنان می‌پردازیم.

Multi-Period Discriminator: MPD خود شامل تعدادی جداکننده(Sub-Discriminator) می‌باشد که هر کدام از آن‌ها تعداد p نمونه صوتی را دریافت می‌کند. هر کدام از این جداکننده‌ها ویژگی‌های متفاوتی از یک سیگنال صوتی را استخراج می‌کنند زیرا با قسمت‌های متفاوتی از این سیگنال مواجه شده‌اند. همانطور که در شکل ۲ـ۲ مشخص است یک سیگنال صوتی یک بعدی با طول T به یک داده‌ساختار

[4] Mel Spectrogram
[5] Multi-Receptive Field Fusion
[6] Residual Block
[7] Phoneme
[8] Sample



شکل ۲ـ۲: ساختار Discriminator ها در HiFi GAN

دوبعدی با طول T/p و عرض p تبدیل می‌شود. همچنین برای اینکه نمونه‌های متناوب، مستقل از یکدیگر پردازش شوند در هر کانولوشن مقدار Kernel Size را در محور عرض برابر ۱ قرار می‌دهیم.

Multi-Scale Discriminator: همانطور که گفته شد در MPD هر یک از Sub-Discriminator ها نمونه‌های صوتی را به صورت جدا از هم دریافت می‌کند. برای اینکه کل سیگنال صوتی ارزیابی شود یک Discriminator دیگر قرار داده شده‌است. MSD شامل ۳ Sub-Discriminator می‌باشد که هریک ورودی‌های متفاوتی دریافت می‌کنند:

- Raw Audio
- ۲×۲ Averaged-pool Audio
- ۴×۴ Averaged-pool Audio

هرکدام از این Discriminator ها شامل دنباله‌ای از لایه‌های کانولوشنی گروه‌بندی شده و همراه با گام[۹] می‌باشد. لازم به ذکر است که ساختار این Discriminator برگرفته از مدل MelGAN می‌باشد.

[9] Stride

۲ـ۱ـ۳ خطای یادگیری

خطای یادگیری تخاصمی: در یادگیری تخاصمی Generator سعی می‌کند تا سیگنالی تولید کند که Discriminator آن را سیگنال صوتی واقعی تشخیص دهد و Discriminator نیز سعی بر آن دارد تا سیگنال واقعی را واقعی و سیگنال تولید شده توسط Generator را غیر واقعی تشخیص دهد. بنابراین خطای یادگیری تخاصمی برای تولیدکننده و جداکننده به صورت زیر است:

$$L_{adv}(D;G) = \mathbb{E}_{(x,s)}[(D(x)-۱)^۲ + (D(G(s)))^۲] \quad (۱ـ۲)$$

$$L_{adv}(G;D) = \mathbb{E}_{s}[(D(G(s))-۱)^۲] \quad (۲ـ۲)$$

در عبارات بالا x نشان‌دهنده سیگنال واقعی و s نشان‌دهنده اسپکتروگرام مل ورودی تولید کننده است.

خطای اسپکتروگرام مل: علاوه بر خطای یادگیری می‌توان خطای دیگری اضافه نمود تا بهره‌وری یادگیری و همچنین کیفیت سیگنال خروجی تولیدکننده افزایش یابد. بدین منظور از خطای اسپکتروگرام مل استفاده می‌شود. به گونه‌ای که تولیدکننده سعی می‌کند تا سیگنالی تولید کند که فاصله $L۱$ میان اسپکتروگرام آن و اسپکتروگرام سیگنال واقعی کمینه شود. این خطا به صورت زیر تعریف می‌شود:

$$L_{Mel}(G) = \mathbb{E}_{(x,s)}[\| \phi(x) - \phi(G(s)) \|_۱] \quad (۳ـ۲)$$

در عبارت بالا ϕ تابع تبدیل کننده موج صوتی به اسپکتروگرام مل آن می‌باشد.

خطای تطابق ویژگی‌ها[۱۰]: این خطا مربوط به تفاوت میان ویژگی‌های استخراج شده در هر لایه از Discriminator میان سیگنال واقعی و سیگنال تولید شده توسط Generator می‌باشد و به صورت زیر تعریف می‌شود:

$$L_{FM}(G;D) = \mathbb{E}_{(x,s)}[\sum_{i=۱}^{T} \frac{۱}{N_i} \| D^i(x) - D^i(G(s)) \|_۱] \quad (۴ـ۲)$$

در عبارت بالا T نشان‌دهنده تعداد لایه‌های Discriminator و D^i و N_i به ترتیب نشان‌دهنده ویژگی‌ها و تعداد ویژگی‌های لایه iام می‌باشد.

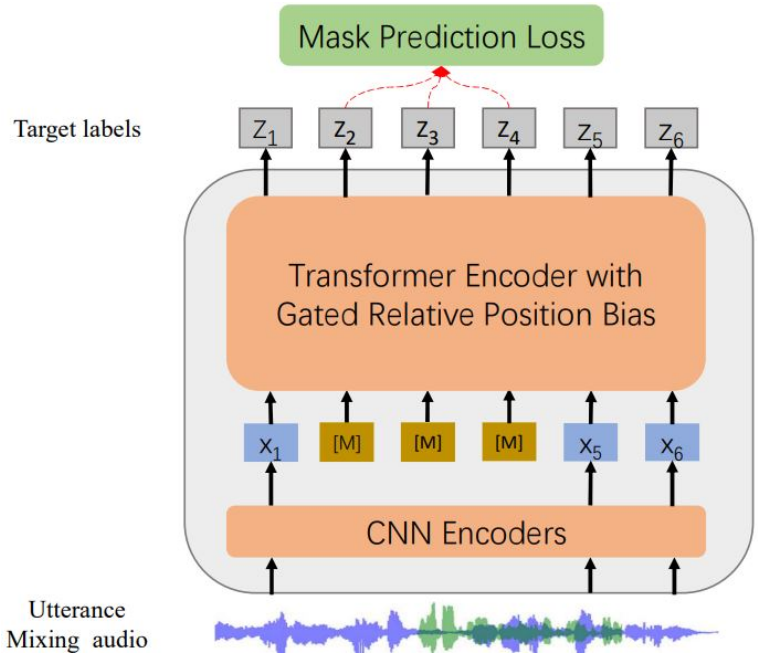
در نهایت خطای کل به شکل زیر درمی‌آید:

$$L_G = L_{Adv}(G;D) + \lambda_{fm} L_{FM}(G;D) + \lambda_{mel} L_{Mel}(G) \quad (۵ـ۲)$$

$$L_D = L_{Adv}(D;G) \quad (۶ـ۲)$$

که قرار می‌دهیم: $\lambda_{fm} = ۲$ و $\lambda_{mel} = ۴۵$

[10] Feature Matching



شکل ۲ـ۳: ساختار WavLM

WavLM ۲ـ۲

در سال‌های اخیر روش‌های یادگیری خودنظارتی[۱۱] در حوزه‌های پردازش زبان‌های طبیعی و پردازش گفتار پیشرفت‌های بسزایی رقم زده‌اند. البته در حوزه‌ی پردازش گفتار توجه محققین بیشتر برروی تشخیص گفتار[۱۲] معطوف بوده‌است. بنابراین وجود یک مدل از پیش آموزش داده‌شده برروی حجم زیادی داده برای حوزه‌های مختلف پردازش گفتار بسیار مفید و البته لازم می‌باشد. مدل WavLM که در سال ۲۰۲۲ منتشر شده است، برروی حجم زیادی از داده بدون برچسب[۱۳] آموزش دیده است و قابلیت دریافت سیگنال یک گفتار و ارائه نمایش گفتار[۱۴] را دارد که این نمایش شامل تمام ویژگی‌های مهم گفتار می‌باشد. همچنین WavLM در حذف نویز از سیگنال صوتی هم پیشرفت قابل ملاحظه‌ای رقم زده‌است.

همانطور که در شکل ۲ـ۳ ملاحظه می‌فرمایید ورودی ماژول WavLM سیگنال صوتی یک گفتار می‌باشد و ابتدا این سیگنال به یک انکودر[۱۵] CNN داده می‌شود. این انکودر شامل ۷ بلاک کانولوشنی می‌باشد و بعد از هرکدام نیز یک لایه Normalization وجود دارد. خروجی این انکودر به یک انکودرِ ترانسفورمر داده می‌شود و در نهایت به عنوان خروجی یک نمایش از گفتار، شامل تمام ویژگی‌های مهم اعم

[11] Self-supervised Learning
[12] Speech Recognition
[13] Unlabeled
[14] Speech Representation
[15] Encoder

از ویژگی‌های مربوط به گوینده و ویژگی‌های مربوط به محتوای گفتار می‌باشد و البته این خروجی بدون نویز می‌باشد و برای کارهای مختلف پردازش گفتار مانند تشخیص گفتار، تشخیص گوینده و تبدیل صدا قابل استفاده است.

فصل ۳

مدل FreeVC

در این پروژه ما از مدل تبدیل صدای FreeVC[۱] استفاده کرده‌ایم. این مقاله که در سال ۲۰۲۲ منتشر شده‌است به نتایج بسیار خوبی دست‌یافته است و برای گفتارهای به زبان فارسی نیز به خوبی عمل می‌کند. در این فصل به شرح ساختار این مدل می‌پردازیم و در فصل بعد گزارش‌های فنی مربوط به آموزش مدل و استنتاج[۲] را ذکر خواهیم کرد.

۳ـ۱ مدل FreeVC

مدل FreeVC یک مدل تبدیل صدای تک‌شات می‌باشد. رویه‌ی معمولی که در مدل‌های تک‌شات به کار می‌رود اینست که اطلاعات مربوط به محتوا را از گفتار مبدا و اطلاعات مربوط به صدای گوینده را از گفتار مقصد استخراج کرده و با استفاده از یک دکودر[۳] یک موج صوتی جدید بسازیم. بنابراین کیفیت یک مدل به طور کلی به توانایی استخراج اطلاعات مورد نظر از گفتار و توانایی ساخت موج صوتی وابسته است. برخی مدل‌های تبدیل صدا از داده‌های برچسب‌دار استفاده می‌کنند[۴]. در این روش معمولا از یک مدل شناسایی خودکار گفتار[۵] استفاده کرده و اطلاعات مربوط به محتوای گفتار را از این طریق بدست می‌آورند. مشکلی که این روش‌ها دارند برچسب‌گذاری داده‌ها می‌باشد که فرایندی پرهزینه است. بنابراین روش‌های بدون برچسب‌گذاری[۶] طرفدار زیادی پیدا کرده‌اند. از لحاظ ساختار، بسیاری از مدل‌های تبدیل صدا در دو مرحله

[1] FREEVC: TOWARDS HIGH-QUALITY TEXT-FREE ONE-SHOT VOICE CONVERSION
[2] Inference
[3] Decoder
[4] Text-Based Models
[5] Automatic Speech Recognition(ASR)
[6] Text-Free

به انجام می‌رسند. در مرحله اول، اطلاعات آکوستیکی گفتار به اطلاعات مربوط به گوینده مقصد تبدیل می‌شود و در مرحله دوم یک وُکودر[7] اطلاعات تبدیل شده را به موج صوتی تبدیل می‌کند. مشکلی که در این حالت به وجود می‌آید اینست که این دو مرحله جدا از هم آموزش داده می‌شوند و اطلاعاتی که مدل اول پیش‌بینی می‌کند توزیع احتمال متفاوتی نسبت به داده‌هایی دارد که مدل دوم با آن آموزش داده می‌شود. این مشکل که به آن ناهمخوانی ویژگی‌ها[8] گفته می‌شود باعث کاهش کیفیت موج صوتی ساخته‌شده می‌شود. مدل FreeVC از داده‌های بدون برچسب استفاده می‌کند و همچنین از طریق اتصال دو مرحله ذکر شده به وسیله بردارهای پنهان[9] سعی در حل مشکل ناهمخوانی ویژگی‌ها دارد. همچنین علی‌رغم اینکه این مدل تک‌شات می‌باشد ویژگی مثبتی که دارد اینست که لازم نیست مدل با گوینده گفتار مبدا در حین آموزش مواجه شده باشد.

۳ـ۲ ساختار مدل FreeVC

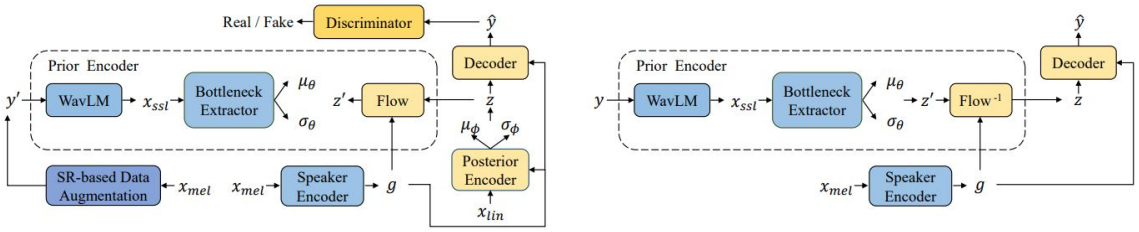
مدل FreeVC شامل اجزای: انکودر پیشین[10]، انکودر پسین[11]، انکودر گوینده[12]، دکودر و جداکننده می‌باشد که در زیر به شرح عملکرد هریک می‌پردازیم.

۳ـ۲ـ۱ انکودر پیشین

انکودر پیشین همانطور که در شکل ۳ـ۱ مشخص است، خود شامل ۳ بخش می‌باشد: ماژول WavLM، یک گلوگاه اطلاعاتی[13] و یک ماژول نرمال‌کننده[14]. ماژول WavLM و گلوگاه، مسئول استخراج اطلاعات مربوط به محتوای گفتار به صورت توزیع $N(z' : \mu_\theta, \sigma_\theta^2)$ می‌باشند.

ماژول WavLM به عنوان ورودی یک موج صوتی دریافت کرده و یک بردار ۱۰۲۴ بعدی از ویژگی‌های SSL[15]، شامل اطلاعات مربوط به محتوای گفتار و اطلاعات مربوط به گوینده‌ی گفتار را به عنوان خروجی می‌دهد(x_{ssl}). به دلیل اینکه نیازی به اطلاعات مربوط به گوینده نداریم باید این اطلاعات را از خروجی WavLM از بین ببریم. به همین دلیل این بردار به یک گلوگاه اطلاعاتی داده می‌شود و بردار ۱۰۲۴ بعدی به

[7] Vocoder
[8] Feature Mismatch
[9] Latent Vectors
[10] Prior Encoder
[11] Posterior Encoder
[12] Speaker Encoder
[13] Information Bottleneck
[14] Normalizing Flow
[15] Self-Supervised Learning



شکل ۳ـ۱: سمت راست: استنتاج، سمت چپ: یادگیری

یک بردار با بعد بسیار کوچکتر(d) تبدیل می‌شود.[۱۶] به دلیل کاهش بعد زیادی که رخ می‌دهد به این قسمت گلوگاه می‌گویند که باعث می‌شود تا اطلاعات مربوط به گوینده و همچنین نویزها و اطلاعات اضافی از بین بروند. در نهایت این بردار d بعدی به دو بردار d بعدیِ μ_θ و σ_θ تبدیل می‌شود. سپس برای اینکه بر پیچیدگی توزیع احتمال بدست آمده بیافزاییم، این توزیع را به یک ماژول Normalizing Flow می‌دهیم که شامل تعدادی تبدیل آفین می‌باشد. بدین صورت توزیع احتمال ساده‌ی ابتدایی(نرمال) به یک توزیع پیچیده‌تر تبدیل می‌شود.

۳ـ۲ـ۲ انکودر گوینده

این انکودر وظیفه استخراج اطلاعات مربوط به گوینده را دارد. در این مقاله از دو نوع انکودر برای این امر استفاده شده: یک انکودر از پیش آموزش داده شده که مخصوص شناسایی گوینده[۱۷] می‌باشد. یک بار هم از یک انکودر آموزش داده نشده استفاده شده است که معماری آن براساس LSTM[۱۸] می‌باشد و باید همراه سایر اجزای مدل آموزش داده شود. در این پروژه ما از انکودر دوم استفاده کردیم تا همه اجزا با یکدیگر آموزش داده شوند.

۳ـ۲ـ۳ انکودر پسین

این انکودر که فقط در فرایند یادگیری حضور دارد به عنوان ورودی اسپکتروگرام خطی موج صوتی و خروجی انکودر گوینده را دریافت کرده و یک توزیع احتمال نرمال با پارامترهای μ_ϕ و σ_ϕ به عنوان خروجی می‌دهد. خروجی این انکودر به دکودر داده می‌شود تا از آن برای ساخت موج صوتی استفاده کند.

در ادامه همین فصل توزیع خواهیم داد که چگونه حضور انکودر پسین در کنار انکودر پیشین باعث

[۱۶] .

[17] Speaker Verification
[18] Long Short-Term Memory

برطرف کردن مشکل ناهمخوانی ویژگی‌ها می‌شود.

۳ـ۲ـ۴ دکودر و جداکننده

در این مدل از دکودر به کار رفته در مدل HiFi-GAN استفاده شده‌است. همانطور که در فصل قبل توضیح دادیم این دکودر شامل چند عملیات کانولوشنی می‌باشد که بعد از هرکدام یک ماژول MRF[۱۹] قرار دارد.

به عنوان جداکننده[۲۰] هم از جداکننده MPD[۲۱] به کار رفته در مدل HiFi-GAN استفاده شده است که توضیحات مربوط به آن را می‌توانید در فصل قبل ملاحظه فرمایید.

۳ـ۲ـ۵ داده‌افزایی

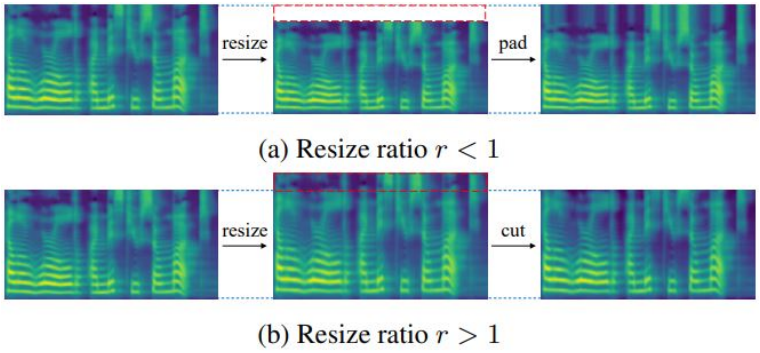
یکی از مشکلاتی که بسیاری از مدل‌های یادگیری ماشین با آن مواجه هستند کمبود داده می‌باشد که منجر به یادگیری ناقص می‌شود. همچنین در چنین مدل‌هایی که یک گلوگاه اطلاعاتی دارند، تعیین اندازه‌ی این گلوگاه بسیار مهم است به گونه‌ای که اگر اندازه آن خیلی کم باشد مدل اطلاعات مهمی را ممکن است از دست بدهد و اگر اندازه آن بسیار بزرگ باشد اطلاعات ناخواسته نیز وارد می‌شوند. به جای آنکه اندازه گلوگاه را با دقت تنظیم کنیم،می‌توانیم حجم دادگان خود را با استفاده از روش‌های داده‌افزایی، افزایش داده و مدل در فرایند یادگیری به تدریج بیاموزد که اطلاعات مهم را حفظ و اطلاعات زائد را حذف کند.

در این مدل برای داده‌افزایی از روش تغییر اندازه اسپکتروگرام[۲۲] استفاده شده است.

در این روش همانطور که در شکل ۳ـ۲ ملاحظه می‌فرمایید اسپکتروگرام مل یک موج صوتی را دریافت کرده و با یک ضریب r آن را در راستای محور زمان یا محور فرکانس فشرده یا کشیده می‌کنیم. اگر $r > ۱$ اسپکتروگرام، کشیده می‌شود و باید بخش اضافی آن را cut کرد. و اگر $r < ۱$ اسپکتروگرام، فشرده می‌شود و باید جای خالی را pad کرد.

بدین ترتیب می‌توان حجم دادگانی را که در اختیار داریم افزایش داریم و در نهایت کیفیت خروجی مدل را بهبود ببخشیم.

[19] Multi-Receptive Field Fusion
[20] Discriminator
[21] Multi-Period Discriminator
[22] Spectrogram Resize



(a) Resize ratio $r < 1$

(b) Resize ratio $r > 1$

شکل ۳ـ۲: داده‌افزایی مبتنی بر تغییر اندازه اسپکتروگرام

۳ـ۳ خطای مدل

خطای مدل FreeVC بسیار به خطای مدل HiFi-GAN شباهت دارد. خطای جداکننده که در دو مدل عینا مثل هم است:

$$L_{adv}(D) = \mathbb{E}_{(y,z)}[(D(y) - ۱)^۲ + (D(G(z)))^۲] \quad (۱ـ۳)$$

در مدل HiFi-GAN همانطور که فصل قبل توضیح داده شد برای تولید کننده ۳ خطا داریم:

$$L_{adv}(G;D) = \mathbb{E}_z[(D(G(z)) - ۱)^۲] \quad (۲ـ۳)$$

$$L_{Mel}(G) = \mathbb{E}_{(y,z)}[\| \phi(y) - \phi(G(z)) \|_۱] \quad (۳ـ۳)$$

$$L_{FM}(G;D) = \mathbb{E}_{(y,z)}[\sum_{i=۱}^{T} \frac{۱}{N_i} \| D^i(y) - D^i(G(z)) \|_۱] \quad (۴ـ۳)$$

در این مدل یک خطای دیگر هم داریم:

$$L_{kl} = \log q_\phi(z \mid x_{lin}) - \log p_\theta(z \mid c) \quad (۵ـ۳)$$

که در عبارت بالا c اطلاعات مربوط به محتوای گفتار می‌باشد و داریم:

$$q_\phi(z \mid x_{lin}) = N(z; \mu_\phi, \sigma_\phi^2) \tag{۶ـ۳}$$

$$p_\theta(z \mid c) = N(z'; \mu_\theta, \sigma_\theta^2) \mid det\frac{\partial z'}{\partial z} \mid \tag{۷ـ۳}$$

این خطا در واقع دیورژانش کولباک‌ـلیبلر می‌باشد. این دیورژانس دو توزیع احتمال دریافت کرده و می‌گوید این دو توزیع از لحاظ اطلاعاتی که در اختیار می‌گذارند چقدر با یکدیگر تفاوت دارند. در اینجا ما دو توزیع احتمال پیشین و پسین را به این دیورژانس داده‌ایم در نتیجه با کمینه کردن این خطا، توزیع احتمالی که انکودر پیشین پیش‌بینی می‌کند و توزیع احتمالی که دکودر با آن آموزش داده می‌شود به یکدیگر نزدیک خواهند شد در نتیجه مشکل ناهمخوانی ویژگی‌ها تا حدودی برطرف می‌شود.

در نهایت خطای کل مدل به صورت زیر درمی‌آید:

$$L(D) = L_{Adv}(D) \tag{۸ـ۳}$$

$$L(G) = L_{Adv}(G) + L_{Mel}(G) + L_{FM}(G) + L_{kl}(G) \tag{۹ـ۳}$$

فصل ۴

گزارش‌های فنی آموزش و استنتاج مدل

در این فصل به شرح چگونگی آموزش مدل و استنتاج آن می‌پردازیم و تنظیمات مربوط به آن را بیان می‌کنیم.
نکته‌ی لازم به ذکر آنست که این مدل به ۳ صورت قابل استفاده‌است:

- بدون داده‌افزایی w/o sr
- استفاده از انکودر گوینده‌ی آموزش ندیده و همراه با داده‌افزایی(FreeVC-s)
- استفاده از انکودر گوینده‌ی از پیش آموزش داده و همراه با داده‌افزایی(FreeVC)

نتایج ارزیابی هرکدام از این ۳ حالت را در پیوست آ می‌توانید ملاحظه فرمایید.
همچنین لازم به ذکر است که ما در این پروژه از حالت دوم(FreeVC-s) استفاده کردیم.

۴ـ۱ تنظیمات مدل

پس از دانلود مدل از صفحه گیت‌هاب[1] می‌توان به فایل‌های آن دسترسی پیدا کرد. در پوشه configs سه فایل آمده است و باتوجه به اینکه ما از دومین حالت مدل استفاده کردیم فایل freevc-s را باز می‌کنیم. حال می‌توان تنظیمات مربوط به مدل را مشاهده و در صورت نیاز تغییر داد. تنها موردی که تغییر داده‌شد مقدار batch size می‌باشد که به دلیل کمبود حافظه مجبور شدیم آن را کاهش دهیم و برابر ۸ قرار دهیم. همچنین همانطور که ملاحظه می‌فرمایید مقدار epochs برابر ۱۰۰۰۰ می‌باشد که آن را تغییر ندادیم ولی

[1] github.com/olawod/freevc

با توجه به checkpoint هایی که در هر مرحله بدست می‌آمد تقریبا پس از گذشت ۲۰۰۰ epoch تغییر چندانی ملاحظه نشد. بنابراین احتمالا با کاهش این مقدار مدل باز هم به خوبی کار کند. همچنین باید فایل‌های مربوط به Generator و WavLM را از آدرس ذکر شده در صفحه گیت‌هاب دانلود کرده و در پوشه مورد نظر قرار دهیم.

۴ـ۲ آموزش مدل

ابتدا یک پوشه به نام dataset ایجاد می‌کنیم. در این پوشه نیز به تعداد افرادی که از صدایشان داده داریم پوشه ساخته و نام هر شخص را برروی یک پوشه می‌گذاریم. برای مثال اگر از ۳ گوینده به نام‌های spk۱ و spk۲ و spk۳ داریم، درون پوشه dataset سه پوشه به نام‌های spk۱، spk۲ و spk۳ می‌سازیم. ما چون از یک گوینده(خانم صارمی) داده‌ی صوتی داشتیم ۱ پوشه به نام saremi ساختیم و دادگان صوتی را در این پوشه قرار دادیم. (لازم به ذکر است تمام فایل‌های صوتی باید ترخ نمونه‌برداری[۲] ۱۶KHz داشته باشند.) ابتدا قبل از شروع فرایند آموزش باید مدل را آماده کنیم. اگر می‌خواهیم از انکودر گوینده‌ی از پیش آموش داده شده استفاده کنیم اید تکه کد یر را اجرا کنیم:

```
# run this if you want to use pretrained speaker encoder
CUDA_VISIBLE_DEVICES=0 python preprocess_spk.py
```

شکل ۴ـ۱: یادگیری با انکودر از پیش آموزش دیده

اگر نمی‌خواهیم از داده‌افزایی استفاده کنیم باید تکه کد زیر را اجرا کنیم:

```
# run this if you want to train without SR-based augmentation
CUDA_VISIBLE_DEVICES=0 python preprocess_ssl.py
```

شکل ۴ـ۲: یادگیری بدون داده‌افزایی

و اگر می‌خواهیم از داده‌افزایی استفاده کنیم ابتدا باید در پوشه اصلی یک پوشه به نام `training_set` بسازیم و سپس تکه کد زیر را اجرا کنیم:

در پوشه filelists سه فایل متنی به نام‌های train و test و val وجود داردکه در هر کدام باید مسیر فایل‌های صوتی مربوط به یادگیری یا ارزیابی یا تست را قرار دهیم. بنابراین اگر فایل صوتی با نام `file_name.wav` از گوینده با نام spk۱ را می‌خواهیم در یادگیری شرکت دهیم باید در فایل train وارد

[2] Sample Rate

```
CUDA_VISIBLE_DEVICES=1 python preprocess_sr.py --in_dir "dataset/" --wav_dir "training-set/" --ssl_dir "training-set/" --min 68 --max 72
CUDA_VISIBLE_DEVICES=1 python preprocess_sr.py --in_dir "dataset/" --wav_dir "training-set/" --ssl_dir "training-set/" --min 73 --max 76
CUDA_VISIBLE_DEVICES=2 python preprocess_sr.py --in_dir "dataset/" --wav_dir "training-set/" --ssl_dir "training-set/" --min 77 --max 80
CUDA_VISIBLE_DEVICES=2 python preprocess_sr.py --in_dir "dataset/" --wav_dir "training-set/" --ssl_dir "training-set/" --min 81 --max 84
CUDA_VISIBLE_DEVICES=3 python preprocess_sr.py --in_dir "dataset/" --wav_dir "training-set/" --ssl_dir "training-set/" --min 85 --max 88
CUDA_VISIBLE_DEVICES=3 python preprocess_sr.py --in_dir "dataset/" --wav_dir "training-set/" --ssl_dir "training-set/" --min 89 --max 92
```

شکل ۴ـ۳: یادگیری با داده‌افزایی

کنیم: `training_set/spk۱/file_name.wav` اگر می‌خواستیم این فایل را در ارزیابی(validation) شرکت دهیم همین عبارت را باید در فایل val قرار می‌دادیم. لازم به ذکر است که در هر خط از این فایل باید مسیر یک فایل مشخص شود. با توجه به این که تعداد فایل‌ها زیاد می‌باشد وارد کردن مسیر همه آن‌ها به صورت دستی کاری طاقت‌فرسا است بنا بر این می‌توان از تکه کد زیر استفاده کرد:

```python
import os
# folder path
dir_path = r'training_set/'

# list to store files
res = []
# Iterate directory
for path in os.listdir(dir_path):
    # check if current path is a file
    if os.path.isfile(os.path.join(dir_path, path)):
        f = open("filelists/train.txt", "a")
        f.write("training_set/" + path + "\n")
        f.close()
```

شکل ۴ـ۴: وارد کردن آدرس فایل‌های صوتی

در نهایت فایل متنی باید به شکل زیر در آید:

```
training_set/saremi/559.wav
training_set/saremi/13.wav
training_set/saremi/107.wav
```

شکل ۴ـ۵: فایل متنی

۴ـ۳ استنتاج

ابتدا باید آخرین checkpoint حاصل از فرایند یادگیری را گرفته و در پوشه f۰ قرار دهیم و نام این checkpoint را ckpt.pth فرض می‌کنیم. یک فایل صوتی از گوینده مقصد نیاز داریم. نام این فایل را target_speaker.wav در نظر می‌گیریم و این فایل را در پوشه f۱ قرار می‌دهیم.

حال فرض کنید می‌خواهیم فایل صوتی source_speaker.wav را که در پوشه f۲ قرار دارد به صدای گوینده target_speaker تبدیل کنیم. بدین منظور باید در فایل convert.txt که در پوشه اصلی قرار دارد عبارت زیر را وارد کنیم:

```
name|f2/source_speaker.wav|f1/target_speaker.wav
```

شکل ۴ـ۶: فایل convert.txt

که name هر عبارت دلخواه می‌تواند باشد و نام فایل خروجی برابر آن خواهد بود. همچنین در فایل convert.txt می‌توان چند خط وارد کرد و مدل همه‌ی آن‌ها را به ترتیب اجرا خواهد کرد.

حال برای اجرای فرایند استنتاج اگر از FreeVC-s استفاده می‌کنیم باید تکه کد زیر را اجرا کرد:

```
#!/bin/bash
echo "inference begins!"
CUDA_VISIBLE_DEVICES=0 python3 convert.py --hpfile configs/freevc-s.json --ptfile f0/ckpt.pth --txtpath convert.txt --outdir outputs/freevc-s
```

شکل ۴ـ۷: استنتاج

و اگر از FreeVC استفاده می‌شود باید تکه کد زیر را اجرا نمود:

```
#!/bin/bash
echo "inference begins!"
CUDA_VISIBLE_DEVICES=0 python3 convert.py --hpfile configs/freevc.json --ptfile f0/ckpt.pth --txtpath convert.txt --outdir outputs/freevc
```

شکل ۴ـ۸: استنتاج

در نهایت فایل خروجی در پوشه‌ی output/freevc یا output/freevc-s قرار خواهد گرفت

فصل ۵

پیاده‌سازی سرویس تبدیل صدا

پس از آموزش مدل و آشنایی با نحوه استنتاج لازم است تا بستری فراهم شود که سرویس تبدیل صدا را ارائه دهد. بدین منظور ما از بات تلگرام استفاده کردیم. پس از ساخت بات تلگرامی لازم بود تا یک سیستم Backend ساخته شود و بات از طریق این سیستم با مدل تبدیل صدا ارتباط برقرار کند. برای پیاده‌سازی Backend از Django و REST Framework استفاده نمودیم.

همچنین به منظور استفاده تجاری از این بات، لازم بود تا یک پایگاه داده نیز برای این سرویس درنظر گرفته شود. بدین منظور از MongoDB استفاده کردیم و در ادامه فصل به شرح آن خواهیم پرداخت.

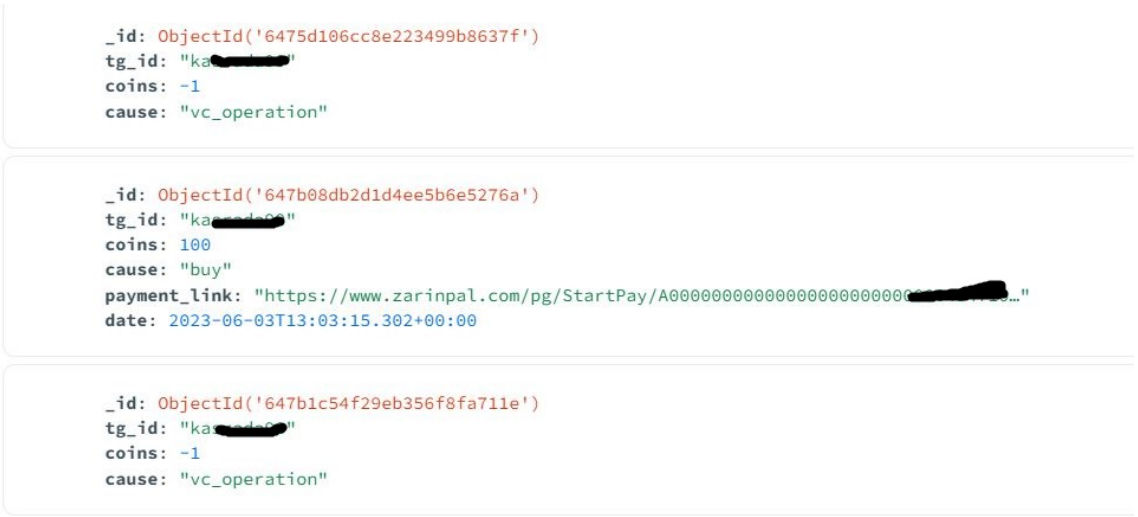
۵ـ۱ پایگاه داده

برای پایگاه داده‌ی این سرویس دو مجموعه[۱] درنظر گرفته‌ایم. یک مجموعه user که اطلاعات هر کاربر را ذخیره می‌کنیم. در حال حاضر برای این مجموعه دو field درنظر گرفته شده‌است: `tg_id` که آیدی تلگرام شخص را ذخیره می‌کند و `join_date` که تاریخ اولین استفاده کاربر از بات را نشان می‌دهد.

مجموعه دومی که در این پایگاه داده داریم `user_account` می‌باشد که حساب کاربر را نشان می‌دهد. هر بار که کاربر عملیات تبدیل صدا یا خرید سکه را انجام می‌دهد یک سند[۲] به این مجموعه اضافه می‌شود.این مجموعه سه field دارد: `tg_id` که آیدی تلگرام کاربر را مشخص می‌کند، coins که تعداد سکه‌های کم شده یا اضافه شده به حساب کاربر را نشان می‌دهد(علامت منفی به معنای کم شدن است) و cause که علت این افزایش یا کاهش سکه را نشان می‌دهد و ۳ حالت دارد: `initial_charge` که

[1] Collection
[2] Document

زمانی است که کاربر برای اولین بار از بات استفاده می‌کند و ۲۰ سکه به طور رایگان به عنوان شارژ اولیه به او داده می‌شود(مقدار coins در این حالت برابر ۲۰ است.)، vc_operation زمانی است که کاربر یک عملیات تبدیل صدا انجام داده‌است و یک سکه از حسابش کسر می‌شود(مقدار coins برابر ۱ـ قرار می‌گیرد) و buy زمانی است که کاربر سکه خریده است. در این حالت سند شامل دو field دیگر نیز می‌باشد: payment_link که لینک درگاه پرداختی که کاربر استفاده کرده است را ذخیره می‌کند و date که زمان(تاریخ و ساعت) تراکنش را نشان می‌دهد. مقدار coins نیز در این حالت برابر تعداد سکه خریداری شده خواهد بود که در حال حاضر در منوی بات، ۴ امکان وجود دارد: خرید ۱۰۰، ۲۰۰، ۴۰۰ و ۵۰۰ سکه. در شکل ۵ـ۱ تصویری از چند سند موجود در این مجموعه را ملاحظه می‌فرمایید.



شکل ۵ـ۱: مجموعه حساب کاربر

۲ـ۵ Backend

همانطور که در ابتدای فصل ذکر شد Backend این سیستم را با استفاده از Django پیاده‌سازی کردیم. به طور کلی این سیستم دو امکان را فراهم می‌کند: تبدیل صدا و خرید سکه.

برای تبدیل صدا کاربر کافی‌است یک فایل صوتی آپلود نماید. ابتدا بات با ارسال یک Query به پایگاه داده بررسی می‌کند که آیا کاربر حداقل یک سکه دارد یا خیر. اگر سکه‌های کاربر تمام شده‌بود پیغام خطا برای او ارسال می‌شود. در غیر این صورت این فایل به آدرس */api/audio/* ارسال می‌شود(درخواست[۳] POST) و Backend این فایل را به مدل می‌دهد و پس از اتمام عملیات تبدیل صدا یک url برای بات

[3] Request

ارسال می‌شود و بات بار ارسال یک درخواست GET به url داده شده فایل را دریافت می‌کند و برای کاربر ارسال می‌کند. پس از ارسال فایل برای کاربر یک سند برای مجموعه‌ی حساب کاربر در پایگاه داده ارسال می‌شود که این ستد حاوی اطلاعات زیر است:

برای خرید سکه کافی‌است از بات یک درخواست POST به $/api/buy/id/coins$ ارسال شود که id آیدی تلگرام کاربر و coins تعداد سکه‌هایی است که می‌خواهد بخرد می‌باشد. کاربر کافی است از منوی خرید سکه یک گزینه را انتخاب کند و خود بات این درخواست را ارسال می‌کند و در جواب لینک درگاه پرداخت داده می‌شود که برای کاربر ارسال می‌شود. در صورت موفقیت آمیز بودن پرداخت یک سند مانند شکل زیر به پایگاه داده ارسال می‌شود و در غیر این صورت کاربر پیغام خطا دریافت می‌کند.

هنگامی هم که کاربر برای اولین بار از بات استفاده می‌کند با کلیک برروی دکمه start بیست سکه به حساب او اضافه می‌شود و پیغام شروع به او ارسال می‌شود:

فصل ۶

جمع‌بندی

در این پایان‌نامه به شرح ساختار و عملکرد یک مدل تبدیل صدا مبتنی بر یادگیری عمیق پرداخته شد که بر اساس مدل FreeVC می‌باشد و در نهایت به پیاده‌سازی بستری برای ارائه این سرویس و استفاده تجاری از آن پرداخته شد.

البته این سیستم قابلیت گسترش نیز دارد. به راحتی با جمع‌آوری یک دیتاست[1] از صدای یک گوینده می‌توان مدل را آموش داد و گویندگان مختلف را به این سرویس اضافه نمود که قطعا بر جذابیت آن خواهد افزود.

همچنین با توجه به این که برای این سیستم یک Backend تعبیه شده است می‌توان در بسترهای دیگری مانند وب‌سایت یا نرم‌افزار موبایل این سرویس را ارائه داد.

[1] Dataset

پیوست آ

مطالب تکمیلی

در این بخش نتایج ارزیابی‌های Subjective و Objective مدل را آورده‌ایم که در زیر می‌توانید ملاحظه فرمایید.

	seen-to-seen		unseen-to-seen		unseen-to-unseen	
	MOS	SMOS	MOS	SMOS	MOS	SMOS
VQMIVC	2.31±0.09	2.10±0.08	1.50±0.08	1.71±0.08	1.49±0.08	1.29±0.05
BNE-PPG-VC	2.80±0.12	2.95±0.12	2.89±0.10	2.83±0.10	3.44±0.08	2.63±0.10
YourTTS	3.46±0.10	3.25±0.09	2.54±0.10	2.50±0.10	2.87±0.09	1.97±0.09
FreeVC	3.99±0.09	**3.80±0.09**	4.06±0.08	**3.77±0.09**	**4.06±0.08**	**2.83±0.08**
FreeVC (w/o SR)	3.85±0.10	3.50±0.10	3.88±0.08	3.58±0.08	3.97±0.09	2.80±0.09
FreeVC-s	**4.01±0.09**	3.75±0.09	**4.08±0.08**	3.68±0.09	4.02±0.09	2.78±0.09
Source	4.32±0.08	-	4.11±0.10	-	4.17±0.09	-

همانطور که در جدول بالا مشاهده می‌کنید بر اساس ارزیابی Subjective که بر اساس نظر مردم می‌باشد در همه‌ی حالات FreeVC عملکرد بهتری داشته است. لازم به ذکر است که MOS نمره‌ای بین ۱ تا ۵ بر اساس کیفیت فایل صوتی خروجی می‌باشد.

SMOS نیز نمره‌ای بین ۱ تا ۵ بر اساس شباهت بین صدای تولید شده و صدای مقصد است. همچنین در حالت سوم که مدل حین یادگیری با صدای مقصد مواجه نشده بود عملکرد خوبی ندارد(از لحاظ شباهت) که نشان‌دهنده‌ی تک شات بودن این مدل است. ولی در حالت دوم که صدای مبدا در حین یادگیری دیده نشده بود همچنان مدل عملکرد خوبی دارد.

در زیر هم ارزیابی Objective مدل آورده شده است که بر اساس نرخ خطای کلمات[۱] و حروف[۲] می‌باشد.

همچنین F۰-PCC که عددی بین ۱ـ و ۱ است، نشان می‌دهد که مدل FreeVC موفقیت بیشتری

[1] Word Error Rate
[2] Character Error Rate

	WER	CER	F0-PCC
VQMIVC	50.68%	29.61%	0.665
BNE-PPG-VC	6.54%	2.50%	0.718
YourTTS	12.87%	5.70%	0.736
FreeVC	4.35%	1.53%	**0.778**
FreeVC (w/o SR)	4.92%	1.77%	0.762
FreeVC-s	**4.23%**	**1.46%**	0.768

در همخوانی تغییرات فرکانس بین صدای تولید شده و گفتار مبدا دارد.