باسمه تعالی

دانشگاه صنعتی شریف

دانشکده مهندسی برق

۲۵۷۴۲ _ سیگنال‌ها و سیستم‌ها _ ترم بهار ۱۴۰۰-۲

پروژه نهایی

تشخیص آهنگ با استفاده از اثر انگشت صوتی

موعد تحویل: مطابق cw

پرسش: در صورت داشتن هر گونه ابهام با آیدی تلگرام @arminpanjehpour یا @ali_arasteh ارتباط برقرار کنید.

نحوه تحویل: کدها و فایل گزارش خود را در CW آپلود کنید. گزارش و نتایج تنها در صورتی معتبر هستند که اجرای کدها با خطا همراه نباشد. زبان برنامه‌نویسی برای انجام تکلیف Matlab است.

توجه: تلاش شما و دریافت مفاهیم، مهم‌تر از خروجی معتبر در نمره‌دهی هستند. با این اوصاف، مشاهده تقلب باعث از دست دادن کل یا بخشی از نمره هر دو طرف می‌شود.

مقدمه

ممکن است برایتان اتفاق افتاده باشد که در یک مهمانی، رستوران و ...، آهنگی زیبایی بشنوید و دلتان بخواهد نام آهنگ یا خواننده آن را بدانید تا بعدا بتوانید، هر زمان که خواستید، دوباره به آن آهنگ گوش دهید. خوشبختانه در دنیای کنونی همه ما گوشی‌های هوشمندی داریم که تقریبا در بسیاری از مشکلات، از جمله مشکل مذکور، به کمک ما می‌آیند. اگر ۱۰ تا ۲۰ ثانیه از آهنگ مورد نظر را ضبط کنید، اپلیکیشن‌های بسیاری وجود دارند که می‌توانند با دریافت این فایل صوتی با دقت خوبی آهنگ مورد نظر را تشخیص دهند و اطلاعات آن را به شما اعلام کنند. سوال اصلی این است که این کار چگونه توسط چنین اپلیکیشن‌هایی انجام می‌پذیرد؟ به وضوح برای تشخیص یک آهنگ از میان میلیون‌ها آهنگ منتشر شده در طول سالیان طولانی، آن هم تنها با استفاده از ۱۰ تا ۲۰ ثانیه صدای نویزی و با کیفیت پایین که در یک محیط غیر ایده‌آل ضبط شده است، باید از ویژگی‌هایی استفاده کنیم که نسبت به نویز و تغییرات جزئی دامنه صدا بسیار مقاوم باشند. چگونگی ایجاد یک مجموعه ویژگی مناسب که بتواند یک آهنگ را به طور یکتا مشخص کند و دارای خصوصیات دلخواه باشد، برای مدت زیادی مورد بحث بوده است که معمولا به عنوان اثر انگشت صوتی شناخته می‌شود. الگوریتم‌های متنوعی برای تولید این نوع از اثر انگشت پیشنهاد شده است. اولین و معروف‌ترین الگوریتم در این راستا، Shazam است که مقاله مربوط به آن در فایل‌های پروژه موجود است. در این پروژه قصد داریم با این الگوریتم به طور کامل آشنا شویم.

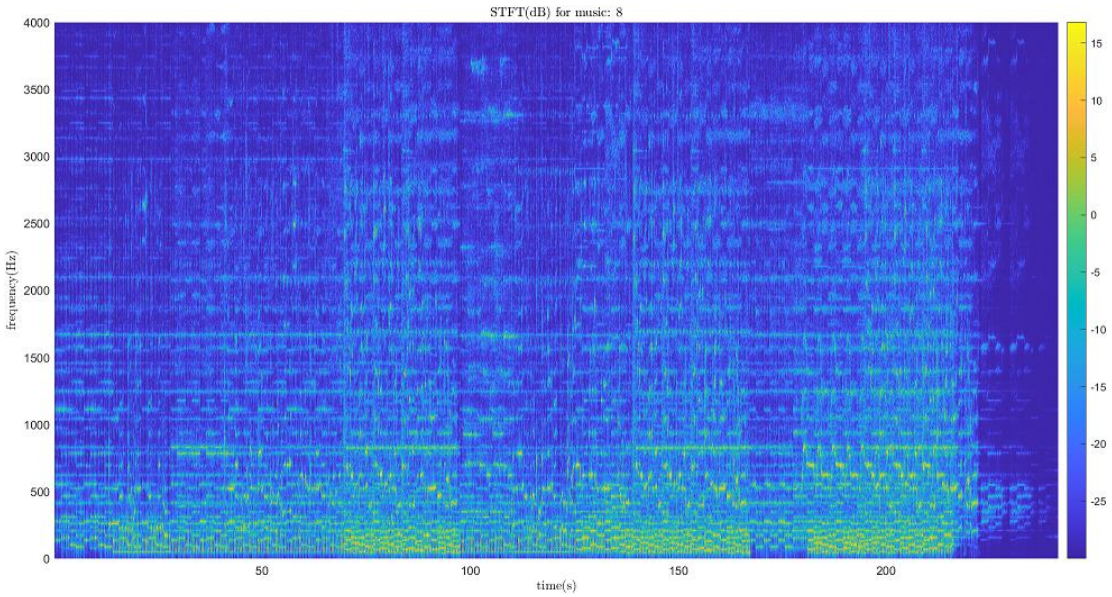
الگوریتم Shazam

در ابتدا به توضیح مرحله به مرحله نحوه ساخت مجموعه ویژگی مناسب برای یک آهنگ دلخواه در این الگوریتم می‌پردازیم.

۱) نرخ نمونه‌برداری آهنگ را کاهش می‌دهیم. نرخ نمونه‌برداری معمول در فایل‌های صوتی، ۴۴۱۰۰ هرتز است. داشتن همچین نرخی برای حفظ تمام بازه فرکانسی که توسط گوش انسان شنیده می‌شود، لازم است؛ اما برای ساخت اثر انگشت آهنگ به این میزان دقت نیاز نیست و می‌توان با استفاده از فرآیند downsampling و کاهش تعداد نمونه‌های موجود، پیچیدگی محاسباتی الگوریتم را کاهش داد. لذا پیش از شروع پردازش آهنگ، نرخ نمونه برداری آن را به ۸۰۰۰ هرتز کاهش می‌دهیم.

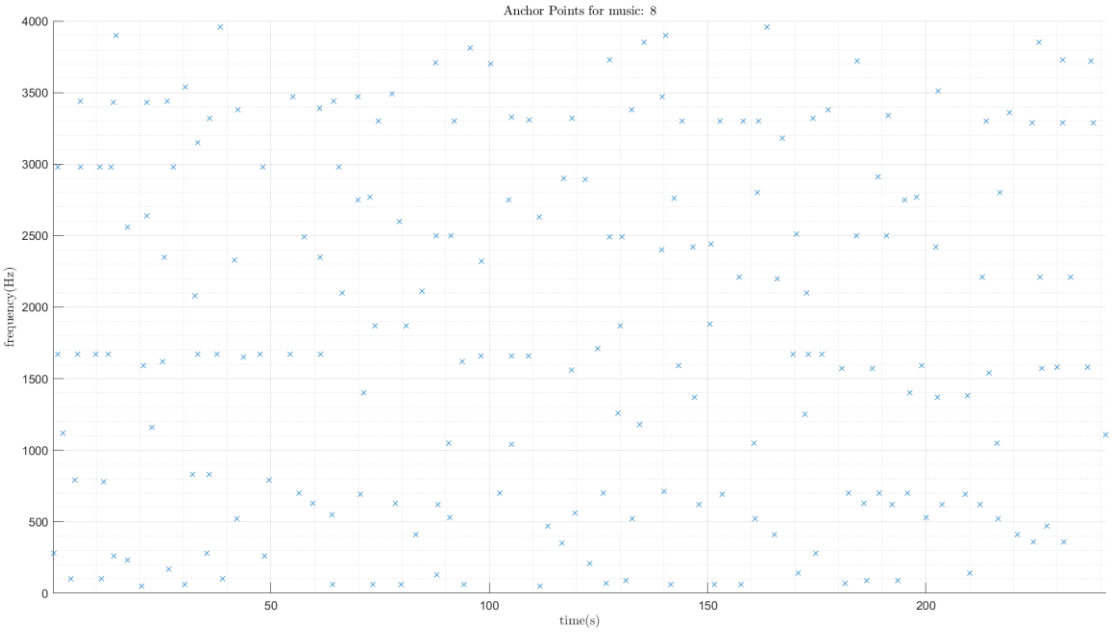
۲) از سیگنال بدست آمده در مرحله قبل، Short Time Fourier Transform یا به اختصار STFT می‌گیریم. پیاده‌سازی ساده STFT بدین صورت است که یک پنجره لغزان را که طول آن توسط کاربر مشخص شده است، با هم‌پوشانی مورد نظر (در این پروژه هم‌پوشانی ۵۰ درصد در نظر گرفته شده است) روی سیگنال بدست آمده در قسمت قبل حرکت می‌دهیم و هر بار تبدیل فوریه گسسته تکه سیگنالی که داخل پنجره قرار گرفته است را محاسبه می‌کنیم و اندازه آن را بدست می‌آوریم. حال اگر تنها فرکانس‌های مثبت را نگه داریم و مقدار فرکانس‌های غیر صفر را ۲ برابر کنیم، تا توان سیگنال ثابت بماند، با کنار هم قرار دادن اندازه تبدیل فوریه گسسته پنجره‌های متوالی، STFT سیگنال بدست آمده، حاصل می‌شود.

نمونه‌ای از STFT برای آهنگی دلخواه را در زیر مشاهده می‌کنید:



۳) درایه‌های anchor point را برای STFT حاصل از مرحله قبل بدست می‌آوریم. ابتدا مقادیر df و dt را به صورت مناسب تعریف می‌کنیم و سپس STFT حاصل از مرحله قبل را در هر دو محور زمان و فرکانس sweap می‌کنیم. عملیات sweap کردن بدین صورت است که به ازای هر درایه STFT، پنجره‌ای به ابعاد 2df*2dt به مرکز آن درایه قرار می‌دهیم و در صورتی که درایه مورد نظر در آن پنجره بیشینه مقدار را داشت، آن را به عنوان یک درایه anchor point انتخاب می‌نماییم و زمان و فرکانس آن را ذخیره می‌کنیم.

خروجی این مرحله برای STFT قسمت قبل را در زیر مشاهده می‌کنید:



۴) اثر انگشت آهنگ مورد نظر را با استفاده از درایه‌های anchor point حاصل از قسمت قبل تولید می‌کنیم. ابتدا مقادیر df_hash و dt_hash را به صورت مناسب تعریف می‌کنیم. برای ساخت اثر انگشت به ازای هر anchor point، پنجره‌ای به ابعاد 2df_hash*dt_hash جلوی درایه مورد نظر قرار می‌دهیم و سپس به ازای هر یک از درایه‌های anchor point دیگری که در این پنجره قرار می‌گیرند، یک hash tag می‌سازیم که در آن hash_key و hash_value، آرایه‌های کاراکتری با ساختارهای زیر هستند.

آرایه hash_key شامل فرکانس درایه anchor point مرجع، فرکانس درایه anchor ponit داخل پنجره و اختلاف زمانی درایه‌های anchor point مرجع و داخل پنجره است که با کاراکتر * از یکدیگر جدا شده اند.

آرایه hash_value شامل شماره آهنگ و زمان درایه anchor point مرجع است که با کاراکتر * از یکدیگر جدا شده اند.

در انتها hash tag ساخته شده به database، که داده ساختاری از نوع hash map است، اضافه می‌شود. شایان ذکر است برای جلوگیری از ایجاد مشکل، در صورت ایجاد یک hash_key تکراری، hash_value آن به hash_value قبلی اضافه می‌شود. در این حالت دو hash_value، با یک کاراکتر + از یکدیگر جدا می‌شوند.

در الگوریتم Shazam، مجموعه hash tag های بدست آمده برای یک آهنگ، به عنوان اثر انگشت آن آهنگ شناخته می‌شوند.

با تکرار مراحل فوق برای همه آهنگ‌های موجود، database را کامل می‌کنیم.

حال اگر یک تکه صدای ۱۰ تا ۲۰ ثانیه‌ای وارد الگوریتم شود، برای تشخیص آهنگ، به صورت زیر عمل می‌شود:

در ابتدا با طی کردن مراحل مشابه، hash tag های تکه صدای بدست آمده را تولید می‌کنیم. سپس به ازای هر hash tag، وجود hash_key مربوط به آن در database را بررسی می‌کنیم. اگر این hash_key در database موجود بود، به ازای هر یک از hash_value های ذخیره شده برای آن hash_key، یک سه‌تایی شامل شماره آهنگ موجود در database، زمان درایه anchor point متناظر در آهنگ موجود در database و زمان درایه anchor point متناظر در تکه صدای ورودی، می‌سازیم. با کنار هم قرار دادن سه‌تایی‌های مذکور لیستی از موارد تشابه بدست می‌آید. در ادامه با استفاده از این لیست، برای آهنگ‌هایی که حداقل یک تشابه داشته اند، احتمال آن که تکه صدای ورودی از آن آهنگ باشد را بدست می‌آوریم. برای این کار ابتدا برای هر آهنگ، دو پارامتر تعداد تشابه و انحراف از میعار اختلاف زمان نرمالیزه شده درایه‌های anchor point در آهنگ موجود در database و تکه صدای ورودی را محاسبه می‌کنیم و سپس از فرمول زیر امتیازی به آن آهنگ اختصاص می‌دهیم.

$$score = \frac{\log(num) * \left(1 - e^{\frac{1-num}{10}}\right)}{std + \epsilon}$$

در انتها با اعمال تابع softmax بر روی امتیازهای بدست آمده، احتمال وقوع آن که تکه صدای ورودی از آن آهنگ باشد، را بدست می‌آوریم و لیست مرتب شده‌ای از آهنگ‌هایی که حداقل یک تشابه داشته اند و احتمال هر یک را به عنوان خروجی تولید می‌کنیم.

در ادامه به توضیح تابع‌های موجود در فایل‌های پروژه می‌پردازیم:

تابع import_audio: این تابع دارای دو ورودی و دو خروجی است. ورودی اول، مسیر دسترسی به پوشه آهنگ و ورودی دوم، شماره آهنگ است. خروجی اول، نرخ نمونه‌برداری در سیگنال downsample شده و خروجی دوم سیگنال downsample شده تک کاناله است.

در این تابع ابتدا آهنگ مورد نظر از حافظه خوانده می‌شود، سپس با میانگین گرفتن از کانال‌های چپ و راست، سیگنال تک کاناله تولید می‌گردد و در آخر سیگنال تک کاناله با نرخ تعریف شده در تابع، downsample می‌شود.

تابع FFT: این تابع دارای یک ورودی و یک خروجی است. ورودی اول، سیگنال مورد نظر است. خروجی اول، اندازه تبدیل فوریه گسسته سیگنال ورودی به ازای فرکانس‌های مثبت است که در آن مقدار فرکانس‌های غیر صفر ۲ برابر شده اند.

تابع STFT: این تابع دارای سه ورودی و سه خروجی است. ورودی اول، سیگنال صوتی، ورودی دوم نرخ نمونه‌برداری این سیگنال و ورودی سوم طول زمانی پنجره برای ساخت STFT است. خروجی اول برداری حاوی زمان‌های انتهای پنجره‌های متوالی، خروجی دوم برداری حاوی فرکانس‌های محاسبه تبدیل فوریه گسسته و خروجی سوم ماتریس حاوی اطلاعات STFT است.

در این تابع ابتدا با استفاده از نرخ نمونه‌برداری و طول زمانی پنجره، تعداد نمونه‌های موجود در یک پنجره بدست می‌آید و سپس بر اساس آن، تعداد پنجره‌های STFT محاسبه می‌شود. در ادامه برای هر پنجره، بخشی از سیگنال صوتی ورودی که داخل پنجره قرار می‌گیرد، جدا می‌شود و با استفاده از تابع FFT، تبدیل فوریه گسسته محاسبه می‌گردد. در انتها بردارهای time و freq به طور مناسب تعریف می‌شوند.

تابع find_anchor_points: این تابع دارای سه ورودی و یک خروجی است. ورودی اول ماتریس حاوی اطلاعات STFT، ورودی دوم dt و ورودی سوم df است. خروجی اول، لیست دوتایی‌های فرکانس و زمان درایه‌های anchor point است.

در این تابع به ازای هر درایه ماتریس حاوی اطلاعات STFT، پنجره‌ای به ابعاد 2df*2dt به مرکز آن درایه قرار داده می‌شود و در صورتی که درایه مورد نظر در آن پنجره بیشینه مقدار را داشت، به عنوان یک درایه anchor point انتخاب می‌گردد و زمان و فرکانس آن ذخیره می‌شود.

تابع create_hash_tags: این تابع دارای چهار ورودی و ۲ خروجی است. ورودی اول لیست دوتایی‌های فرکانس و زمان درایه‌های anchor point، ورودی دوم dt_hash، ورودی سوم df_hash و ورودی چهارم شماره آهنگ است. خروجی اول لیست hash_key های بدست آمدهو خروجی دوم لیست hash_value های ساخته شده است.

در این تابع به ازای هر anchor point، پنجره‌ای به ابعاد 2df_hash*dt_hash جلوی درایه مورد نظر قرار داده می‌شود و سپس به ازای هر یک از درایه‌های anchor point دیگری که در این پنجره قرار می‌گیرند، یک hash tag تولید می‌گردد که در آن hash_key یک سه‌تایی مرتب شامل فرکانس درایه anchor point مرجع، فرکانس درایه anchor ponit داخل پنجره و اختلاف زمانی درایه‌های anchor point مرجع و داخل پنجره و hash_value یک دوتایی مرتب شامل شماره آهنگ و زمان درایه anchor point مرجع است.

تابع scoring: درای یک ورودی و یک خروجی است. ورودی اول لیست موارد تشابه است. خروجی اول لیست مرتب شده آهنگ‌های محتمل و احتمال هر یک است.

در این تابع ابتدا با استفاده از لیست ورودی، برای هر آهنگی که حداقل یک تشابه داشته است، دو پارامتر تعداد تشابه و انحراف از میعار اختلاف زمان نرمالیزه شده درایه‌های anchor point در آهنگ موجود در database و تکه صدای ورودی محاسبه می‌شود و سپس از فرمول زیر امتیازی به آن آهنگ اختصاص می‌یابد:

$$score = \frac{\log(num) * \left(1 - e^{\frac{1-num}{10}}\right)}{std + \epsilon}$$

در انتها با اعمال تابع softmax بر روی امتیازهای بدست آمده، احتمال آن که تکه صدای ورودی از آن آهنگ باشد، محاسبه می‌گردد و لیستی دوتایی مرتب شده از آهنگ‌هایی که حداقل یک تشابه داشته اند و احتمال هر یک به عنوان خروجی تولید می‌شود.

خواسته‌های پروژه

۱ تابع import_audio را کامل کنید. در این تابع نیاز است بعد از بدست آوردن میانگین کانال‌های چپ و راست، سیگنال مورد نظر downsample شود. برای این منظور می‌توانید از دستورهای rat و resample استفاده کنید.

۲ تابع FFT را کامل کنید. توجه داشته باشید که باید طیف توان یک طرفه را به عنوان خروجی برگردانید. برای این کار ابتدا باید اندازه تبدیل فوریه را به طول سیگنال تقسیم کنید و سپس خروجی را به توان دو برسانید. با این کار طیف توان دو طرفه بدست می‌آید. سپس تنها فرکانس‌های مثبت را نگه دارید و مقدار فرکانس‌های غیر صفر را ۲ برابر کنید تا طیف توان یک طرفه بدست آید.

۳ تابع STFT را کامل کنید. برای این منظور باید در هر بار تکرار حلقه موجود، پنجره مناسبی از سیگنال را جدا کنید و با استفاده از تابع FFT، طیف توان یک طرفه آن را بدست آورید. در انتها باید خروجی در ستون مناسبی از ماتریس mat_freq_time ذخیره شود.

۴ فایل create_database.m را مطالعه کنید و نحوه عملکرد آن را به اختصار توضیح دهید. سپس با استفاده از این تابع، اثر انگشت آهنگ‌های داده شده در پوشه musics را بدست آورید. این اطلاعات در متغیر database که از نوع hashmap است، ذخیره می‌شوند.

۵ فایل search_databse.m را مطالعه کنید و نحوه دقیق ساخت متغیر list را شرح دهید.

۶ در صورتی که این الگوریتم بخواهد در شرایط واقعی استفاده شود، تعداد آهنگ‌های موجود در database در حدود چند میلیون خواهد بود. در چنین حالتی جستجو برای یافتن تطابق به ازای هر hash tag امری بسیار زمان‌بر خواهد بود. در مورد داده ساختار hashmap و مرتبه زمانی جستجو در آن، مطالعه کنید و بر این اساس، استفاده از این نوع داده ساختار در مسئله مذکور را توجیه نمایید.

۷ با استفاده از فایل search_databse.m، مشخص کنید هر یک از تکه آهنگ‌های موجود درپوشه test_musics، مربوط به کدام آهنگ است و سپس با استفاده از متغیر list، زمان تقریبی شروع این تکه آهنگ در آهنگ مذکور را بدست آورید. با گوش کردن به این قسمت از آهنگ شناسایی شده، از صحت عملکرد الگوریتم مطمئن شوید.

۸ آهنگی از آهنگ‌های موجود در پوشه musics را به دلخواه انتخاب کنید. به صورت تصادفی قسمتی از این آهنگ به طول ۲۰ ثانیه جدا کنید. در ادامه با استفاده از دستور randn، آهنگ بدست آمده را به نویز سفید گوسی با میانگین صفر و واریانس ۰/۰۱ آغشته کنید. حال عملکرد فایل search_databse.m را بر روی خروجی بررسی کنید. این کار را به ازای SNR های مختلف تکرار کنید و کمترین SNR ممکن برای تشخیص صحیح آهنگ را بدست آورید.

۹ آهنگی از آهنگ‌های موجود در پوشه musics را به دلخواه انتخاب کنید. در ادامه به ازای تغییر مقدار SNR از ۱۰ تا ۱ با دقت ۰/۱ و با استفاده از دستور randn، آهنگ مورد نظر را به نویز سفید گوسی با میانگین صفر و واریانس مناسب آغشته کنید. حال به ازای هر مقدار SNR، به صورت تصادفی ۱۰۰ قسمت از این آهنگ به طول ۲۰ ثانیه انتخاب کنید و نمودار میانگین احتمال نسبت داده شده به آهنگ انتخاب شده را بر حسب SNR، رسم کنید. این کار را برای چند آهنگ دیگر نیز تکرار کنید و نتایج را مقایسه کنید.

۱۰ آهنگی از آهنگ‌های موجود در پوشه musics را به دلخواه انتخاب کنید. تکه‌ای از این آهنگ را با لپتاپ پخش کنید و با گوشی خود صدای محیط را ضبط کنید. حال عملکرد فایل search_databse.m را بر روی خروجی بررسی کنید (نمودار مربوط به STFT و درایه‌های anchor point را در گزارش قرار دهید). با تکرار این آزمایش و ایجاد سر و صدا در هنگام ضبط آهنگ، میزان دقت الگوریتم مورد استفاده را در کاربردهای واقعی بسنجید.

۱۱ دو آهنگ از آهنگ‌های موجود در پوشه musics را به دلخواه انتخاب کنید. به صورت تصادفی قسمتی از هر یک به طول ۲۰ ثانیه انتخاب کنید. دو تکه آهنگ بدست آمده را با ضرایب α و $1-\alpha$ به ازای $\alpha = \frac{1}{2}$ با یکدیگر جمع کنید. حال عملکرد فایل search_databse.m را بر روی خروجی بررسی کنید. با تغییر ضریب α بین صفر تا یک، نمودار احتمال نسبت داده شده به هر یک از دو آهنگ بر حسب ضریب α را رسم کنید و تغییر تصمیم الگوریتم را مشاهده نمایید.