

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

گروه مهندسی نرم افزار

پایان نامه کارشناسی رشته‌ی مهندسی کامپیوتر گرایش هوش مصنوعی و نرم افزار

آموزش شبکه مدل زبان ترکیبی برای ساخت موسیقی lo-fi

استاد راهنما:

دکتر زهرا زجاجی

دانشجو:

سهیل سلیمی

شهریور ۱۴۰۳

چکیده

افزایش تولید محتوا منجر به تقاضای بی‌سابقه‌ای برای موسیقی با کیفیت بالا و بدون حق کپی‌رایت برای استفاده در محتوای چندرسانه‌ای شده است. موسیقی لوفای، با ویژگی‌های آرامش‌بخش و تسکین‌دهنده‌اش، به یکی از عناصر اصلی در تولید ویدئو، پادکست و پخش زنده تبدیل شده است. با این حال، دستیابی به موسیقی لوفای با کیفیت بالا و بدون حق کپی‌رایت می‌تواند فرآیندی چالش‌برانگیز و پرهزینه باشد. این مقاله رویکردی نوین برای تولید موسیقی لوفای با استفاده از یک مدل زبانی کوچک ارائه می‌دهد که نیاز به یک راه‌حل مقرون‌به‌صرفه و مقیاس‌پذیر برای تولیدکنندگان محتوا را برطرف می‌کند. ما از معماری RWKV [؟] استفاده می‌کنیم، مدلی پیشرفته که کارایی یک ترانسفورمر را با انعطاف‌پذیری یک شبکه عصبی بازگشتی ترکیب می‌کند. در این مقاله، ما بر جنبه‌های فنی تولید موسیقی لوفای تمرکز می‌کنیم و چالش‌های آموزش یک مدل برای تولید موسیقی با کیفیت بالا و طول نامحدود را بررسی می‌کنیم. ما دو مدل جداگانه، یکی برای پیانو و دیگری برای سازهای درام، آموزش دادیم تا موسیقی لوفای را به صورت درخواستی تولید کنند. رویکرد ما به تولیدکنندگان محتوا این امکان را می‌دهد که بر روی دیدگاه خلاقانه خود تمرکز کنند، به جای اینکه زمان و منابع خود را صرف جستجوی موسیقی مناسب کنند. با خودکارسازی فرآیند تولید موسیقی، ما قصد داریم اهمیت انتخاب موسیقی در تولید محتوا را کاهش دهیم و تولیدکنندگان را آزاد کنیم تا بر روی مهارت‌های اصلی خود تمرکز کنند.

کلیدواژه‌ها: ۱- تولید موسیقی lo-fi ۲- موسیقی تولید شده توسط هوش مصنوعی ۳- هوش مصنوعی

مولد

فهرست مطالب

صفحه	عنوان
۱	۱: مقدمه
۱	۱-۱ پیش‌گفتار
۳	۲: بررسی پیشینه و ادبیات
۳	۱-۲ ادبیات موضوع
۳	۱-۱-۲ RWKV
۴	۲-۱-۲ MIDI file format
۸	۲-۲ روشهای پیشین
۸	۱-۲-۲ استفاده از معماری VAE
۹	۲-۲-۲ استفاده از معماری RNN LSTM
۱۱	۳: آموزش مدل و معماری
۱۱	۱-۳ معماری کلی پروژه
۱۲	۲-۳ دیتاسیت ها
۱۲	۱-۲-۳ مدل پیانو
۱۳	۲-۲-۳ مدل دارم
۱۳	۳-۳ تبدیل MIDI به متن
۱۳	۴-۳ رویکرد برای توکن‌سازی فایل‌های MIDI
۱۵	۱-۴-۳ توکن‌سازی
۱۷	۵-۳ آموزش مدل
۱۷	۱-۵-۳ پارامترهای آموزش مدل
۱۷	۲-۵-۳ نحوه آموزش مدل
۲۰	۶-۳ ارزیابی عملکرد مدل
۲۰	۱-۶-۳ ارزیابی مدل پیانو
۲۰	۲-۶-۳ ارزیابی مدل دارم
۲۲	۳-۶-۳ ارزیابی عملکرد مدل با معیارهای موسیقی
۲۶	۷-۳ ساخت موسیقی نهایی
۲۹	۴: ساخت خط لوله متن به lo-fi
۲۹	۱-۴ معماری خط لوله
۳۰	۱-۱-۴ کمبودهای استفاده مستقیم از مدل MusicGen برای تولید موسیقی
۳۱	۲-۴ ارزیابی خط لوله
۳۳	۵: نتیجه‌گیری و پیشنهادها

صفحه	عنوان
۳۳	۱-۵ نتیجه گیری.....
۳۴	۲-۵ پیشنهادها.....
۳۵	منابع و مآخذ.....
۳۶	پیوست‌ها
۳۶	پ-۱ دسترسی به کد ها.....

عنوان

صفحه

فهرست تصاویر

عنوان	صفحه
شکل ۱-۲: معماری RWKV برای مدل های زبان	۵
شکل ۲-۲: عناصر موجود در یک بلوک RWKV (سمت چپ) و بلوک باقیمانده کامل RWKV	۶
مجهز به یک سر نهایی برای مدل سازی زبان (سمت راست)	۷
شکل ۳-۲: ساختار فایل MIDI	۱۲
شکل ۱-۳: ساختار pipeline	۲۱
شکل ۲-۳: نمودار های پیشرفت یادگیری مدل پیانو	۲۱
شکل ۳-۳: نمودار های پیشرفت یادگیری مدل درام	۲۱

فهرست جداول

عنوان	صفحه
جدول ۱-۳: خلاصه‌ای از مجموعه داده	۱۳
جدول ۲-۳: نتایج این معیار ها برای مدل پیانو	۲۵
جدول ۱-۴: نتایج نظرسنجی ارزیابی خط لوله	۳۲

فهرست الگوریتم‌ها

صفحه	عنوان
۱۶	۱-۳ توکن کردن فایل های MIDI
۱۸	۲-۳ آموزش مدل
۲۲	۳-۳ هماهنگی ریتم
۲۳	۴-۳ شباهت ملودی
۲۴	۵-۳ ثبات تونال
۲۵	۶-۳ هماهنگی هارمونیک
۲۷	۷-۳ تبدیل توکن به MIDI
۲۸	۸-۳ متن به MIDI message
۲۸	۹-۳ نحوه ساخت موسیقی نهایی

فصل اول

مقدمه

۱-۱ پیش‌گفتار

موسیقی لو-فای که با صداهای آرام خود شناخته می‌شود، در سال‌های اخیر محبوبیت زیادی پیدا کرده است. این ژانر که اغلب با لیست‌های پخش مطالعه و آرامش مرتبط است، ترکیبی از ضرب‌آهنگ‌های ملایم، صداهای محیطی و کیفیت تولید خام و متمایز را به نمایش می‌گذارد. با افزایش تعداد استریمرها و اینفلوئنسرها که به دنبال موسیقی بدون حق کپی‌رایت برای محتوای خود هستند، نیاز به آهنگ‌های لو-فای رایگان بیشتر احساس می‌شود. این مدل می‌تواند برای پخش نامحدود موسیقی لو-فای استفاده شود. ظهور هوش مصنوعی^۱ و یادگیری ماشین^۲ راه‌های جدیدی برای خلق موسیقی باز کرده است و امکان توسعه مدل‌هایی را فراهم کرده که می‌توانند به طور خودکار موسیقی لو-فای تولید کنند.

در این مقاله، فرآیند آموزش یک مدل زبان کوچک را که به طور خاص برای تولید موسیقی لو-فای طراحی شده است، بررسی می‌کنیم. با استفاده از قابلیت‌های مدل rWkv، قصد داریم الگوهای ریتمیک و ملودیک منحصر به فرد موجود در موسیقی لو-فای را به دست آوریم. رویکرد ما شامل آموزش دو

^۱ Artificial intelligence

^۲ Machine learning

مدل جداگانه برای سازهای انفرادی: پیانو و درام، هر کدام با ۱۰۰ میلیون پارامتر است. این امر به ما امکان می‌دهد تا بر جزئیات دقیق هر ساز تمرکز کنیم و تولید موسیقی با کیفیت بالا و اصیل را تضمین کنیم.

هدف اصلی این تحقیق نشان دادن امکان استفاده از یک مدل زبان کوچک و کارآمد برای تولید موسیقی است که می‌تواند به ویژه برای موسیقی‌دانان مستقل و علاقه‌مندان با منابع محدود مفید باشد. ما به معماری مدل rwkv می‌پردازیم و مزایای آن در پردازش داده‌های ترتیبی و مناسب بودن آن برای وظایف تولید موسیقی را برجسته می‌کنیم. علاوه بر این، فرآیند جمع‌آوری داده‌ها، از جمله انتخاب و پیش‌پردازش قطعات موسیقی لو-فای برای ایجاد یک مجموعه داده آموزشی قوی را مورد بحث قرار می‌دهیم.

در طول فرآیند آموزش، با چالش‌های مختلفی مانند نیاز به داده‌های آموزشی متنوع مواجه شدیم. ما این مسائل را با اجرای تکنیک‌هایی مانند افزایش داده‌ها و منظم‌سازی حل می‌کنیم تا قابلیت تعمیم و عملکرد مدل را تضمین کنیم. علاوه بر این، موسیقی تولید شده را با استفاده از معیارهای کمی و ارزیابی‌های کیفی ارزیابی می‌کنیم و بینش‌هایی در مورد اثربخشی مدل و زمینه‌های بهبود ارائه می‌دهیم. در پایان، هدف ما ارائه یک راهنمای جامع در مورد نحوه آموزش یک مدل زبان کوچک برای تولید موسیقی لو-فای است و پتانسیل هوش مصنوعی در دموکراتیزه کردن تولید موسیقی و تقویت خلاقیت در عصر دیجیتال را برجسته می‌کنیم. همچنین، جهت‌های آینده این تحقیق را که شامل ادغام سازهای اضافی و بررسی ساختارهای موسیقی پیچیده‌تر برای افزایش قابلیت‌های موسیقی لو-فای تولید شده توسط هوش مصنوعی است، بررسی می‌کنیم.

فصل دوم

بررسی پیشینه و ادبیات

۱-۲ ادبیات موضوع

۱-۱-۲ RWKV

معماری RWKV [۱] به گونه‌ای طراحی شده است که از مزایای هر دو معماری RNN^۱ [۲] و ترانسفورمر [۳]^۲ بهره‌برداری کند.

چگونگی کار معماری RWKV:

۱. مکانیزم توجه خطی: در RWKV، از مکانیزم توجه خطی استفاده می‌شود که به مدل اجازه می‌دهد تا اطلاعات را به صورت موازی پردازش کند. این مکانیزم به جای استفاده از ماتریس‌های بزرگ توجه، از وزن‌دهی خطی استفاده می‌کند که باعث کاهش پیچیدگی محاسباتی می‌شود.

۲. ترکیب RNN و ترانسفورمر: RWKV از ساختار RNN برای حفظ وابستگی‌های طولانی‌مدت در داده‌ها استفاده می‌کند، در حالی که از مکانیزم توجه ترانسفورمر برای پردازش موازی و کارایی بهتر بهره می‌برد. این ترکیب باعث می‌شود که مدل بتواند دنباله‌های طولانی را با کارایی بالا پردازش کند.

^۱ Recurrent Neural Networks

^۲ Transformer

مزایای RWKV نسبت به ترانسفورمرها

۱. کارایی محاسباتی بهتر: ترانسفورمرها به دلیل پیچیدگی محاسباتی و حافظه‌ای که به صورت مربعی با طول دنباله افزایش می‌یابد، در پردازش دنباله‌های طولانی مشکل دارند. در مقابل، RWKV با استفاده از مکانیزم توجه خطی، این مشکل را حل می‌کند و پیچیدگی محاسباتی و حافظه‌ای خطی دارد.

۲. استنتاج سریع‌تر: RWKV می‌تواند در طول استنتاج به صورت کارآمدتری عمل کند، زیرا از مزایای RNN‌ها در این زمینه بهره می‌برد.

۳. این ترکیب باعث می‌شود RWKV یک گزینه جذاب برای کاربردهایی باشد که نیاز به پردازش دنباله‌های طولانی با کارایی بالا دارند.

همانطور که در ۱-۲ نشان داده شده است، مدل با یک لایه embedding شروع می‌شود که پس از آن، چندین residual blocks مشابه به صورت متوالی قرار گرفته‌اند. این بلوک‌ها در شکل‌های ۲-۲ نشان داده شده‌اند. پس از آخرین بلوک، یک سر خروجی ساده شامل یک لایه نرمال‌سازی^۱ و یک پروجکشن خطی برای تولید لاجیت‌ها^۲ جهت پیش‌بینی توکن بعدی و محاسبه‌ی خطای متقاطع^۳ در طول آموزش استفاده می‌شود.

۲-۱-۲ MIDI file format

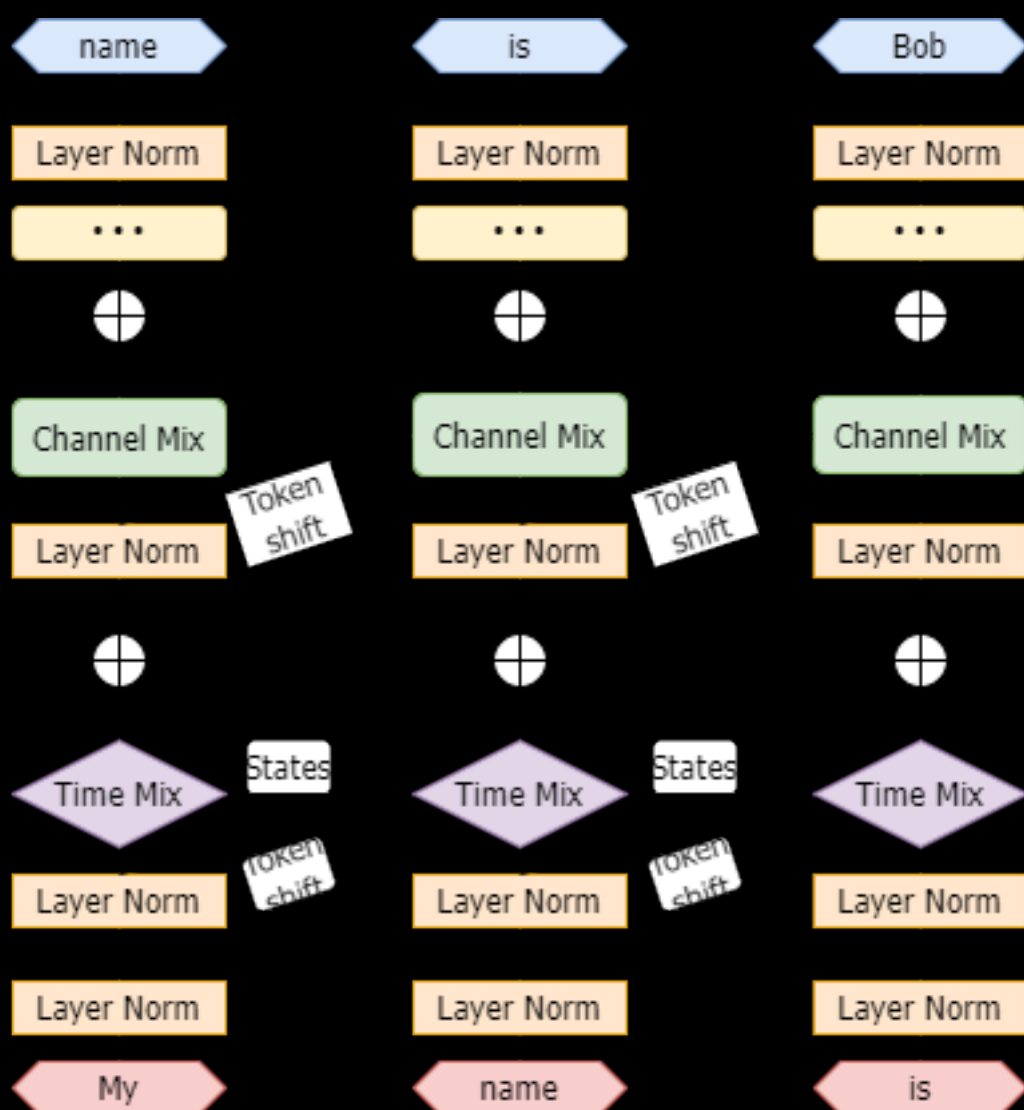
فرمت MIDI^۴ [۴] یک استاندارد فنی برای ارتباط بین ابزارهای موسیقی الکترونیکی، کامپیوترها و دیگر دستگاه‌های مرتبط با موسیقی است. برخلاف فایل‌های صوتی معمولی مانند MP3 یا WAV، فایل‌های MIDI حاوی داده‌های صوتی واقعی نیستند. در عوض، آن‌ها شامل اطلاعاتی مانند نت‌های موسیقی، زمان‌بندی، مدت زمان و شدت صدا برای هر نت هستند^{۱۲}. این فرمت به موسیقی‌دانان و تولیدکنندگان موسیقی اجازه می‌دهد تا داده‌های موسیقی را به صورت دیجیتالی ضبط و پخش کنند و به راحتی بین نرم‌افزارها و سخت‌افزارهای مختلف به اشتراک بگذارند. به دلیل اندازه کوچک فایل‌های MIDI، انتقال و ذخیره‌سازی آن‌ها بسیار آسان است.

^۱(LayerNorm)

^۲(logits)

^۳(cross-entropy loss)

^۴Musical Instrument Digital Interface



شکل ۱-۲ - معماری RWKV برای مدل های زبان

از دیدگاه کامپیوتری، فایل های MIDI به عنوان مجموعه ای از پیام های دیجیتالی ذخیره می شوند که هر پیام شامل اطلاعاتی درباره نحوه پخش موسیقی است. این پیام ها به صورت باینری کدگذاری می شوند و شامل سه بخش اصلی هستند:

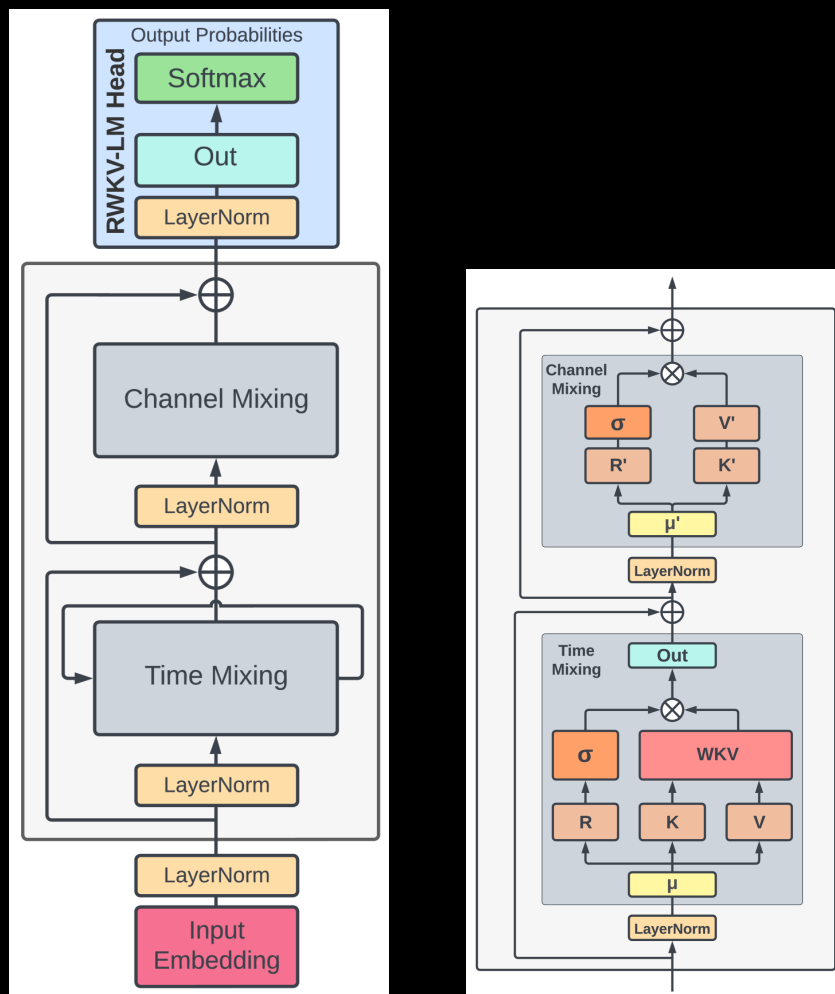
۱. **پیام های وضعیت**^۱: این پیام ها نوع عملیاتی که باید انجام شود را مشخص می کنند، مانند

نواختن یک نت، تغییر شدت صدا، یا تغییر ابزار موسیقی.

۲. **پیام های داده**^۲: این پیام ها اطلاعات دقیق تری درباره عملیات مشخص شده در پیام های وضعیت

^۱Status Messages

^۲Data Messages



شکل ۲-۲- عناصر موجود در یک بلوک RWKV (سمت چپ) و بلوک باقیمانده کامل RWKV، مجهز به یک سر نهایی برای مدل سازی زبان (سمت راست).

ارائه می‌دهند، مانند شماره نت، شدت صدا، و مدت زمان.

۳. **زمان‌بندی**^۱: این بخش زمان دقیق اجرای هر پیام را مشخص می‌کند، که به دستگاه‌ها اجازه می‌دهد تا موسیقی را با دقت زمانی بالا پخش کنند.

پیام وضعیت: نواختن نت^۲ پیام داده: شماره نت (مثلاً C4)، شدت صدا (مثلاً ۶۴) زمان‌بندی: زمان شروع (مثلاً ۵۰۰ میلی‌ثانیه پس از شروع) این پیام‌ها به ترتیب در یک فایل MIDI ذخیره می‌شوند و هنگام پخش، دستگاه‌های MIDI این پیام‌ها را تفسیر کرده و موسیقی را تولید می‌کنند. این ساختار به کامپیوترها و دستگاه‌های موسیقی اجازه می‌دهد تا به صورت هماهنگ و دقیق موسیقی را پخش کنند.

time message time message time message time message
time message time message time message time message
time message time message time message time message
time message time message time message time message
time message time message time message time message

شکل ۲-۳ - ساختار فایل MIDI

- استفاده از فرمت MIDI برای آموزش مدل‌های زبانی نسبت به فرمت WAV مزایای متعددی دارد:
۱. **اندازه فایل کوچکتر**: فایل‌های MIDI بسیار کوچکتر از فایل‌های WAV هستند. این امر باعث می‌شود که پردازش و انتقال داده‌ها سریع‌تر و کارآمدتر باشد.
 ۲. **داده‌های ساختاریافته**: فایل‌های MIDI شامل اطلاعات دقیق و ساختاریافته‌ای درباره نت‌های موسیقی، زمان‌بندی، و شدت صدا هستند. این داده‌ها به مدل‌های زبانی کمک می‌کنند تا الگوهای موسیقی را بهتر درک کنند و پیش‌بینی‌های دقیق‌تری انجام دهند.
 ۳. **انعطاف‌پذیری بیشتر**: با استفاده از MIDI، می‌توان به راحتی تغییرات مختلفی در موسیقی اعمال کرد، مانند تغییر تمپو، کلید، و ابزار موسیقی. این انعطاف‌پذیری به مدل‌های زبانی کمک می‌کند تا با شرایط مختلف سازگار شوند و عملکرد بهتری داشته باشند.
 ۴. **کاهش نویز**: فایل‌های WAV شامل داده‌های صوتی خام هستند که ممکن است نویز و اختلالات

^۱Timing

^۲Note On

زیادی داشته باشند. در مقابل، فایل‌های MIDI تنها شامل داده‌های دیجیتالی هستند که نويز ندارند و این امر باعث می‌شود که مدل‌های زبانی با داده‌های تمیزتر و دقیق‌تری آموزش ببینند. یک مزیت دیگر استفاده از فرمت MIDI برای آموزش مدل‌های زبانی این است که موسیقی چندلایه را به خوبی پشتیبانی می‌کند. فایل‌های MIDI می‌توانند چندین ترک^۱ را به صورت همزمان ذخیره کنند، که هر ترک می‌تواند نمایانگر یک ابزار موسیقی مختلف باشد. این ویژگی به مدل‌های زبانی اجازه می‌دهد تا تعاملات پیچیده بین ابزارهای مختلف را درک کنند و تحلیل کنند که چگونه این ابزارها با هم ترکیب می‌شوند تا یک قطعه موسیقی کامل را تشکیل دهند. این قابلیت به ویژه برای آموزش مدل‌های زبانی که هدفشان تولید یا تحلیل موسیقی پیچیده است، بسیار مفید است. با داشتن داده‌های چندلایه، مدل‌ها می‌توانند به درک عمیق‌تری از ساختار موسیقی برسند و پیش‌بینی‌های دقیق‌تری انجام دهند.

۲-۲ روش‌های پیشین ۱-۲-۲ استفاده از معماری VAE

پروژه jacobz/Lofi^[۵] با استفاده از معماری VAE کار مشابهی را انجام می‌دهد. استفاده از معماری RWKV، معماری که ما در این پروژه استفاده کرده‌ایم. برای ساخت موزیک لوفای^۲ مزایای متعددی نسبت به VAE^۳ دارد:

۱. **حفظ ساختار زمانی:** RWKV به دلیل استفاده از مکانیزم‌های بازگشتی، قادر است ساختار زمانی و توالی‌های طولانی را بهتر حفظ کند. این ویژگی برای موزیک لوفای که اغلب دارای الگوهای تکراری و ریتمیک است، بسیار مهم است.
۲. **کیفیت بازسازی بهتر:** RWKV به دلیل استفاده از کلیدها و مقادیر وزنی، می‌تواند جزئیات بیشتری از داده‌های ورودی را حفظ کند و بازسازی دقیق‌تری ارائه دهد.
۳. **انعطاف‌پذیری بیشتر:** این معماری به دلیل استفاده از مکانیزم‌های توجه، می‌تواند به طور دینامیک به بخش‌های مختلف داده توجه کند و این امر باعث می‌شود که در تولید موزیک‌های پیچیده‌تر و متنوع‌تر عملکرد بهتری داشته باشد.

¹Track

²Lo-Fi

³Variational Autoencoder

۱-۲-۲-۱ محدودیت‌های VAE

۱. محدودیت در اندازه آهنگ: یکی از محدودیت‌های اصلی VAE این است که به دلیل استفاده از فضای نهان با ابعاد کمتر، ممکن است در بازسازی آهنگ‌های طولانی‌تر دچار مشکل شود. این امر می‌تواند منجر به از دست رفتن جزئیات مهم و کاهش کیفیت بازسازی شود.^۱

۲. کیفیت بازسازی پایین‌تر: VAE به دلیل استفاده از توزیع‌های احتمالاتی برای بازسازی داده‌ها، ممکن است در بازسازی جزئیات دقیق دچار مشکل شود و کیفیت نهایی موزیک کاهش یابد.

به طور کلی، معماری RWKV به دلیل توانایی بهتر در حفظ ساختار زمانی و جزئیات داده‌ها، برای ساخت موزیک لوفای مناسب‌تر است. از طرف دیگر، VAE به دلیل محدودیت‌های ذاتی خود در بازسازی آهنگ‌های طولانی و پیچیده، ممکن است کیفیت نهایی موزیک را کاهش دهد.

۲-۲-۲ استفاده از معماری RNN LSTM

استفاده از معماری RWKV برای ساخت موزیک لوفای مزایای متعددی نسبت به LSTM^۱ دارد. RWKV به دلیل استفاده از مکانیزم‌های کلید-مقدار وزنی، قادر است ساختار زمانی و توالی‌های طولانی را بهتر حفظ کند. این ویژگی برای موزیک لوفای که اغلب دارای الگوهای تکراری و ریتمیک است، بسیار مهم است. همچنین، RWKV به دلیل استفاده از کلیدها و مقادیر وزنی، می‌تواند جزئیات بیشتری از داده‌های ورودی را حفظ کند و بازسازی دقیق‌تری ارائه دهد. این معماری به دلیل استفاده از مکانیزم‌های Attention، می‌تواند به طور دینامیک به بخش‌های مختلف داده توجه کند و این امر باعث می‌شود که در تولید موزیک‌های پیچیده‌تر و متنوع‌تر عملکرد بهتری داشته باشد.

از سوی دیگر، یکی از محدودیت‌های اصلی LSTM این است که به دلیل استفاده از حافظه کوتاه مدت، ممکن است در بازسازی آهنگ‌های طولانی‌تر دچار مشکل شود. این امر می‌تواند منجر به از دست رفتن جزئیات مهم و کاهش کیفیت بازسازی شود. همچنین، LSTM به دلیل استفاده از توزیع‌های احتمالاتی برای بازسازی داده‌ها، ممکن است در بازسازی جزئیات دقیق دچار مشکل شود و کیفیت نهایی موزیک کاهش یابد.

به طور کلی، معماری RWKV به دلیل توانایی بهتر در حفظ ساختار زمانی و جزئیات داده‌ها، برای ساخت موزیک لوفای مناسب‌تر است. از طرف دیگر، LSTM به دلیل محدودیت‌های ذاتی خود در

^۱Long Short-Term Memory

بازسازی آهنگ‌های طولانی و پیچیده، ممکن است کیفیت نهایی موزیک را کاهش دهد.

فصل سوم

آموزش مدل و معماری

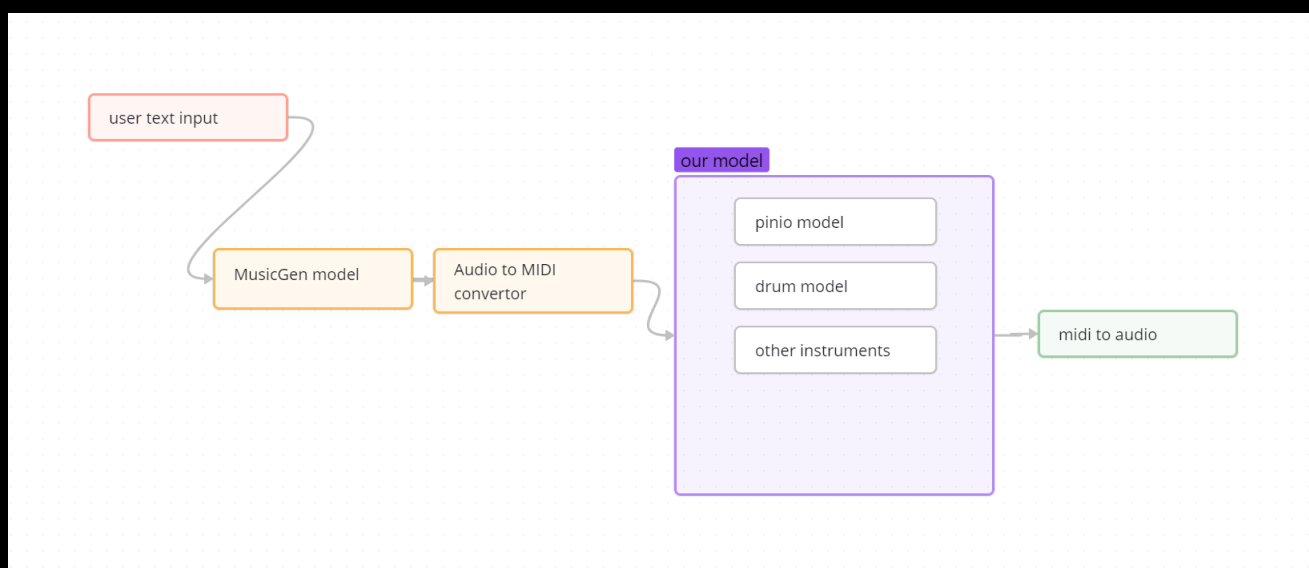
۳-۱ معماری کلی پروژه

در رویکرد ما، هدف این است که مدل‌های جداگانه‌ای برای هر ساز که قصد استفاده در آهنگ خود داریم، آموزش دهیم. به عنوان مثال، ما مدل‌هایی برای پیانو و درام آموزش داده‌ایم. خروجی‌های این مدل‌ها سپس ترکیب می‌شوند تا ترکیب نهایی ایجاد شود و اطمینان حاصل شود که سهم هر ساز به درستی نمایانده شده است.

ورودی مدل‌های ما می‌تواند یک فایل MIDI یا هر فایل WAV باشد. اگر ورودی یک فایل WAV باشد، ابتدا با استفاده از یک الگوریتم تبدیل به نت‌های MIDI تبدیل می‌شود. سپس این نت‌های MIDI برای پردازش به مدل‌های ما ارسال می‌شوند. این مرحله تبدیل بسیار مهم است زیرا به ما امکان می‌دهد با یک فرمت استاندارد کار کنیم و مدیریت ورودی‌های صوتی مختلف را آسان‌تر می‌کند.

از آنجا که ما از مدل زبان RWKV استفاده می‌کنیم، نیاز به یک توکنایزر داریم تا فایل‌های MIDI را به فرمت متنی تبدیل کند که مدل بتواند آن را درک کند. توکنایزر فایل‌های MIDI را به قطعات کوچکتر و قابل مدیریت تقسیم می‌کند که سپس به مدل RWKV تغذیه می‌شوند. این فرآیند به مدل

امکان می‌دهد تا به طور مؤثر توالی‌های موسیقی را یاد بگیرد و تولید کند.



شکل ۳-۱ - ساختار pipeline

علاوه بر آموزش مدل‌ها، ما یک pipeline توسعه داده‌ایم که تجربه کاربری را بهبود می‌بخشد. این خط لوله همانطور که در ۳-۱ نشان داده شده است، ورودی متنی کاربر را دریافت کرده و آن را از طریق یک مدل تولید موسیقی (MusicGen) [۶] پردازش می‌کند. مدل MusicGen یک فایل MIDI بر اساس ورودی متنی کاربر ایجاد می‌کند. این فایل MIDI تولید شده سپس از طریق مدل‌های آموزش دیده ما برای هر ساز عبور می‌کند. در نهایت، خروجی این مدل‌ها ترکیب شده و به عنوان ترکیب نهایی موسیقی ذخیره می‌شود.

با آموزش مدل‌های جداگانه برای هر ساز و ترکیب خروجی‌های آن‌ها، می‌توانیم به یک ترکیب موسیقایی دقیق‌تر و پویاتر دست یابیم. این روش انعطاف‌پذیری و خلاقیت بیشتری در تولید موسیقی فراهم می‌کند، زیرا هر ساز می‌تواند به صورت جداگانه تنظیم شود و سپس در قطعه نهایی ادغام شود.

۳-۲ دیتاسیت‌ها

۳-۲-۱ مدل پیانو

برای انجام آموزش مدل پیانو خود، ما از مجموعه داده گسترده‌ای به نام مجموعه داده MIDI موسیقی ایرلندی IrishMMD [۷] استفاده کردیم. این مجموعه داده شامل ۲۱۶،۲۸۴ قطعه موسیقی ایرلندی

به فرمت MIDI است. این مجموعه داده به دو بخش تقسیم شده است: ۹۹٪ (۲۱۴،۱۲۲ قطعه) برای آموزش مدل و ۱٪ (۲،۱۶۲ قطعه) برای ارزیابی آن.

قطعات موسیقی این مجموعه داده از وبسایت‌های thesession.org و abcnnotation.com جمع‌آوری شده‌اند. برای اطمینان از یکپارچگی داده‌ها، ممکن است برخی از قطعات موسیقی که به صورت متن بودند به فرمت MIDI تبدیل شده باشند. همچنین، اطلاعات غیرموسیقی مانند عنوان و متن ترانه‌ها حذف شده است.

۲-۲-۳ مدل دارم

مجموعه داده گسترده EGMD [۸]

برای تحقیق خود، ما از نسخه گسترده‌تری از مجموعه داده EGMD استفاده کردیم که به عنوان **مجموعه داده گسترده EGMD** شناخته می‌شود. GMD یک مجموعه داده از اجراهای درام انسانی است که به صورت MIDI بر روی یک درام کیت الکترونیکی Roland TD-11 ضبط شده است.

بخش	تعداد توالی‌های منحصربه‌فرد	تعداد کل توالی‌ها	مدت زمان (ساعت)
آموزشی	۸۱۹	۳۵،۲۱۷	۴،۳۴۱
آزمایشی	۱۲۳	۵،۲۸۹	۹،۵۰
اعتبارسنجی	۱۱۷	۵،۰۳۱	۲،۵۲
کل	۱،۰۵۹	۴۵،۵۳۷	۵،۴۴۴

جدول ۳-۱ - خلاصه‌ای از مجموعه داده

ما تقسیم‌بندی‌های آموزشی، آزمایشی و اعتبارسنجی را که در GMD وجود داشت، حفظ کردیم. نکته مهم این است که از آنجایی که هر کیت برای هر توالی ضبط شده است، تمام ۴۳ کیت در بخش‌های آموزشی، آزمایشی و اعتبارسنجی وجود دارند. که به طور خلاصه در ۳-۱ نشان داده شده است.

۳-۳ تبدیل MIDI به متن

۴-۳ رویکرد برای توکن‌سازی فایل‌های MIDI

- پیش‌پردازش

□ **فیلتر کردن:** حذف پیام‌های متا ناشناخته برای اطمینان از پردازش فقط داده‌های MIDI مرتبط.

□ **ادغام ترک‌ها:** اگر فایل MIDI شامل چندین ترک باشد، آن‌ها را به یک ترک واحد ادغام کنید تا پردازش ساده‌تر شود.

- مدیریت وضعیت

□ **وضعیت کانال‌ها:** نگهداری دیکشنری‌هایی برای پیگیری وضعیت هر کانال MIDI شامل تغییرات برنامه، حجم، بیان، نوت‌های فعال و وضعیت پدال.

□ **زمان‌بندی:** پیگیری زمان سپری شده بین رویدادهای MIDI برای نمایش دقیق زمان‌بندی در توالی توکن‌ها.

- بافر توکن

□ **بافرینگ:** استفاده از یک بافر برای ذخیره موقت داده‌های توکن قبل از تبدیل آن‌ها به توکن‌های رشته‌ای. این کار به مدیریت زمان‌بندی و توالی توکن‌ها کمک می‌کند.

- پردازش رویدادها

□ **رویدادهای نوت:** پردازش رویدادهای `note_on` و `note_off` برای شروع و توقف نوت‌ها، با در نظر گرفتن سرعت، حجم و بیان.

□ **تغییرات کنترل:** پردازش پیام‌های تغییر کنترل برای به‌روزرسانی وضعیت کانال‌ها، مانند حجم، بیان و وضعیت پدال.

- تولید توکن

□ **تبدیل توکن:** تبدیل داده‌های نوت بافر شده به توکن‌های رشته‌ای با استفاده از فرمت‌های از پیش تعریف شده. این شامل نگاشت رویدادهای MIDI به نمایش‌های خاص توکن است.

□ **توکن‌های زمان‌بندی:** تولید توکن‌هایی که زمان سپری شده بین رویدادها را نمایش می‌دهند تا ساختار زمانی موسیقی حفظ شود.

- ساخت خروجی

□ **تقسیم قطعات:** تقسیم خروجی به قطعات بر اساس سکوت یا معیارهای دیگر برای ایجاد بخش‌های قابل مدیریت از توکن‌ها.

□ **نهایی سازی:** افزودن توکن‌های شروع و پایان به هر قطعه و ترکیب لیست نهایی توالی‌های توکن.

این رویکرد شامل پیش‌پردازش داده‌های MIDI، مدیریت وضعیت کانال‌های MIDI، بافر کردن داده‌های توکن، پردازش رویدادهای مختلف MIDI، تولید توکن‌ها و ساخت خروجی نهایی است. این روش اطمینان می‌دهد که ساختار زمانی و موسیقایی فایل MIDI به‌طور دقیق در توالی توکن‌ها نمایش داده می‌شود.

مثال ۳-۱. برای مثال `<start> 26:2 t8 26:0 t5 26:2 <end>` می‌تواند خروجی الگوریتم ۳-۱ باشد.

۳-۴-۱ توکن سازی

در کار ما، از یک رویکرد ساده برای آماده‌سازی و آموزش مدل با استفاده از کتابخانه `Tokenizer` [۹] استفاده کردیم. در اینجا توضیح دقیقی از این فرآیند آورده شده است:

۳-۴-۱-۱ توکن سازی سریع با کتابخانه `Tokenizer`

ما از کتابخانه `Tokenizer` برای انجام توکن‌سازی سریع و کارآمد داده‌ها استفاده کردیم. این کتابخانه برای پردازش مجموعه داده‌های بزرگ و تبدیل متن خام به توکن‌ها به سرعت طراحی شده است. مزایای کلیدی استفاده از این کتابخانه شامل موارد زیر است: - **سرعت:** کتابخانه `Tokenizer` برای عملکرد بهینه‌سازی شده است و به ما امکان می‌دهد حجم زیادی از داده‌ها را در زمان کوتاهی پردازش کنیم. - **انعطاف‌پذیری:** این کتابخانه از استراتژی‌های مختلف توکن‌سازی پشتیبانی می‌کند و به راحتی می‌توان آن را برای نیازهای خاص پروژه سفارشی کرد.

۳-۴-۱-۲ تبدیل به فرمت `JSONL`

پس از توکن‌سازی، داده‌های توکن‌شده را به فرمت `JSON Lines (JSONL)` تبدیل کردیم. این فرمت به خصوص برای پردازش مجموعه داده‌های بزرگ مناسب است زیرا: - **پردازش خط به خط:** هر خط در یک فایل `JSONL` نمایانگر یک شیء `JSON` جداگانه است که پردازش داده‌ها را خط به خط بدون نیاز به بارگذاری کل مجموعه داده در حافظه آسان می‌کند. - **سادگی:** `JSONL` به راحتی خوانده

```

1: function mix_volume(velocity, volume, expression)
2:   return  $velocity \times \left(\frac{volume}{127.0}\right) \times \left(\frac{expression}{127.0}\right)$ 
3: function convert_midi_to_str(cfg, filter_cfg, mid, augment=None)
4:   Initialize state variables
5:   function flush_token_data_buffer
6:     Convert token data buffer to token data
7:     Append formatted tokens to output
8:     Clear token data buffer
9:   function consume_note_program_data(prog, chan, note, vel)
10:    if token is valid then
11:      if delta_time_ms > threshold then
12:        Check if any notes are held
13:        if no notes are held then
14:          Call flush_token_data_buffer()
15:          Append "<end>" to output
16:          Reset output and state variables
17:        Generate wait tokens and append to output
18:        Reset delta_time_ms
19:        Append token data to buffer
20:        Set started_flag to True
21:    for each msg in mid.tracks[0] do
22:      Update delta_time_ms with msg.time
23:      function handle_note_off(ch, prog, n)
24:        if pedal is on then
25:          Set pedal event
26:        else
27:          Call consume_note_program_data(prog, ch, n, 0)
28:          Remove note from channel_notes
29:      if msg.type is "program_change" then
30:        Update channel_program
31:      else if msg.type is "note_on" then
32:        if velocity is 0 then
33:          Call handle_note_off
34:        else
35:          Remove pedal event if exists
36:          Call consume_note_program_data with mixed volume
37:          Add note to channel_notes
38:      else if msg.type is "note_off" then
39:        Call handle_note_off
40:      else if msg.type is "control_change" then
41:        Update channel state based on control type
42:      else
43:        pass
44:    Call flush_token_data_buffer()
45:    Append "<end>" to output
46:    return output_list

```

و نوشته می‌شود و با بسیاری از ابزارهای پردازش داده‌ها به خوبی ادغام می‌شود.

۳-۴-۱-۳ تبدیل به فرمت binidx برای آموزش سریع

برای بهینه‌سازی بیشتر فرآیند آموزش، داده‌های JSONL را به فرمت binidx تبدیل کردیم. binidx یک فرمت باینری است که چندین مزیت برای آموزش مدل‌های یادگیری ماشین ارائه می‌دهد: - **کارایی:** فرمت‌های باینری به طور کلی فشرده‌تر و سریع‌تر برای خواندن/نوشتن نسبت به فرمت‌های متنی هستند و سر بار I/O را در طول آموزش کاهش می‌دهند. - **سازگاری:** فرمت binidx با بسیاری از چارچوب‌های یادگیری ماشین سازگار است و ادغام بدون مشکل در خط لوله آموزش ما را تسهیل می‌کند. ما برای تبدیل فایل‌های JSONL به فرمت binidx از بخشی از کد های کتابخانه gpt-neox [۱۰] استفاده کردیم.

با استفاده از کتابخانه Tokenizer برای توکن‌سازی سریع و تبدیل داده‌ها به فرمت JSONL و سپس به فرمت binidx، ما به طور قابل توجهی کارایی فرآیندهای آماده‌سازی داده و آموزش را بهبود بخشیدیم. این رویکرد به ما امکان داد تا مجموعه داده‌های بزرگ را به طور مؤثر مدیریت کنیم و زمان کلی آموزش را تسریع کنیم که منجر به توسعه کارآمدتر مدل شد.

۳-۵ آموزش مدل

۳-۵-۱ پارامترهای آموزش مدل

در این بخش، پارامترهای مورد استفاده برای آموزش مدل توضیح داده شده‌اند: ما از معماری `rwkv-0.6` [۱۱] استفاده کردیم. همچنین امکان استفاده از مدل دارد. مدل ما شامل ۲۰ لایه و embedding برابر با ۵۱۲ است. Context Length^۱ مدل برابر با ۵۱۲ است. مقدار نرخ یادگیری اولیه و نهایی برابر با 6×10^{-4} و 6×10^{-5} است.

۳-۵-۲ نحوه آموزش مدل

الگوریتم ۲-۳ برای مدیریت و بهینه‌سازی فرآیند آموزش مدل با استفاده از PyTorch Lightning [۹] طراحی شده است.

ابتدا، تابع `save_pth` که برای ذخیره دیکشنری حالت مدل در یک مسیر فایل مشخص استفاده

^۱ Context Length بی‌نهایت فقط در هنگام اجرای مدل معنا می‌دهد.

```

1: Function save_pth(dd, ff)
2: torch.save(dd, ff)
3: Class ResetValDataloader(Callback)
4: Function on_validation_start(trainer, pl_module)
5: trainer.reset_val_dataloader(pl_module)
6: Class TrainCallback(Callback)
7: Function __init__(self, args)
8: self.args = args
9: Function on_train_batch_start(self, trainer, pl_module, batch, batch_idx)
10: Adjust learning rate based on global step and schedule
11: for param_group in trainer.optimizers[0].param_groups do
12:     param_group['lr'] = lr * param_group['my_lr_scale'] if args.layerwise_lr > 0 else
        lr
13: trainer.my_lr = lr
14: Function on_train_batch_end(self, trainer, pl_module, outputs, batch, batch_idx)
15: Log metrics and update loss
16: Function on_train_epoch_start(self, trainer, pl_module)
17: Update dataset attributes
18: Function on_train_epoch_end(self, trainer, pl_module)
19: if trainer.is_global_zero and (args.epoch_save > 0 and trainer.current_epoch %
    args.epoch_save == 0) or (trainer.current_epoch == args.epoch_count - 1) then
20:     save_pth(pl_module.state_dict(), f'args.proj_dir/self.prefix_args.epoch_begin +
    trainer.current_epoch.pth')

```

می‌شود. این قابلیت برای ایجاد نقاط بازرسی در طول آموزش ضروری است و به مدل اجازه می‌دهد تا ذخیره و بعداً بارگذاری شود. این امر تضمین می‌کند که فرآیند آموزش می‌تواند از آخرین حالت ذخیره شده در صورت وقفه‌ها از سر گرفته شود و پیشرفت‌های حاصل شده در طول آموزش حفظ شود.

کلاس `TrainCallback` با آرگومان‌های مختلفی که فرآیند آموزش را کنترل می‌کنند، مقداردهی اولیه می‌شود. این شامل تنظیم برنامه‌های نرخ یادگیری، لاگ‌گیری و سایر پارامترهای خاص آموزش است. در متد `on_train_batch_start`، نرخ یادگیری به صورت پویا بر اساس مرحله فعلی آموزش تنظیم می‌شود. برنامه نرخ یادگیری به دو مرحله اصلی تقسیم می‌شود: مرحله گرم‌کردن^۱ و مرحله کاهش. در مرحله گرم‌کردن، نرخ یادگیری به تدریج از یک مقدار کوچک به نرخ یادگیری اولیه افزایش می‌یابد. این کار به فرآیند آموزش در مراحل اولیه کمک زیادی می‌کند. در مرحله کاهش، نرخ یادگیری بر اساس کاهش خطی یا نمایی تنظیم می‌شود. این تنظیم پویا نرخ یادگیری به بهینه‌سازی فرآیند آموزش و بهبود همگرایی کمک می‌کند.

لاگ‌گیری و نظارت گسترده بخش‌های جدایی‌ناپذیر این بخش هستند. متدهای `on_train_batch_start` و `on_train_batch_end` شامل مکانیزم‌های لاگ‌گیری دقیقی برای دیدن پیشرفت آموزش هستند. این شامل لاگ‌گیری نرخ یادگیری و `Loss Function`، پیگیری تعداد توکن‌های پردازش شده در هر ثانیه و لاگ‌گیری به سرویس‌های مانند `Weights & Biases (wandb)` [۹] برای پیگیری آزمایش‌ها است. متدهای `on_train_epoch_start` و `on_train_epoch_end` وظایفی را که باید در ابتدای هر دوره آموزش و پایان آن انجام شوند، مدیریت می‌کنند. این شامل تنظیم پارامترهای مجموعه داده مانند رتبه جهانی و دوره واقعی و ذخیره نقاط بازرسی مدل در فواصل مشخص یا در پایان آموزش است. ذخیره منظم نقاط بازرسی مدل تضمین می‌کند که فرآیند آموزش می‌تواند از آخرین حالت ذخیره شده از سر گرفته شود و یک محافظت در برابر وقفه‌ها فراهم می‌کند.

۳-۵-۲-۱ کتابخانه‌ها استفاده شده

چندین کتابخانه در این روش برای ساده‌سازی فرآیند آموزش استفاده می‌شوند. کتابخانه `torch` [۹] قابلیت‌های اصلی برای ساخت و آموزش شبکه‌های عصبی را فراهم می‌کند. `PyTorch Lightning` [۹] فرآیند آموزش را با انتزاع کدهای تکراری ساده می‌کند، حلقه‌های آموزش، اعتبارسنجی و تست را

^۱warm up

از طریق کلاس Trainer مدیریت می‌کند و از کلاس‌های Callback برای افزودن رفتار سفارشی در مراحل مختلف آموزش استفاده می‌کند. کتابخانه wandb برای لاگ‌گیری و پیگیری آزمایش‌ها استفاده می‌شود.

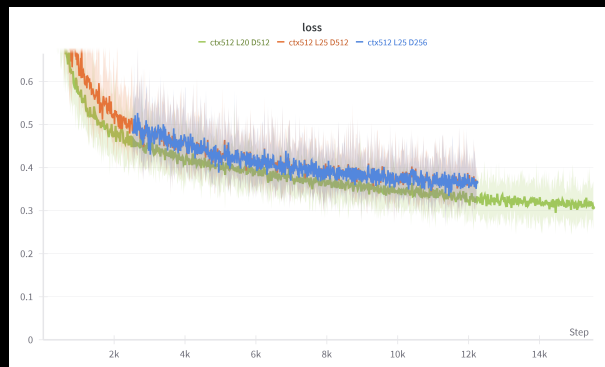
۳-۶ ارزیابی عملکرد مدل ۳-۶-۱ ارزیابی مدل پیانو

با توجه به شکل ۳-۲ می‌توان گفت که مدل سبز^۱ به دلیل شروع با مقدار اولیه کمتری از خطا، عملکرد بهتری دارد. این موضوع می‌تواند به دلیل پیش‌آموزش مؤثرتر یا وزن‌های اولیه بهتر باشد. برخلاف مدل‌های قرمز و آبی که کاهش سریعی در خطا نشان می‌دهند و سپس به یک سطح ثابت می‌رسند، مدل سبز به تدریج و به طور پیوسته کاهش می‌یابد. این نشان‌دهنده یک فرآیند یادگیری پایدارتر است که خطر بیش‌برازش را کاهش می‌دهد و اطمینان می‌دهد که مدل به داده‌های جدید بهتر تعمیم می‌یابد. مدل‌های قرمز و آبی، در حالی که بهبودهای اولیه سریعی نشان می‌دهند، تمایل دارند در کمینه‌های محلی گیر کنند که عملکرد بلندمدت آن‌ها را محدود می‌کند. برای تولید موسیقی لو-فای، که در آن ظرافت‌ها و ثبات اهمیت دارند، فرآیند یادگیری پایدار و پیوسته مدل سبز آن را به گزینه بهتری تبدیل می‌کند.

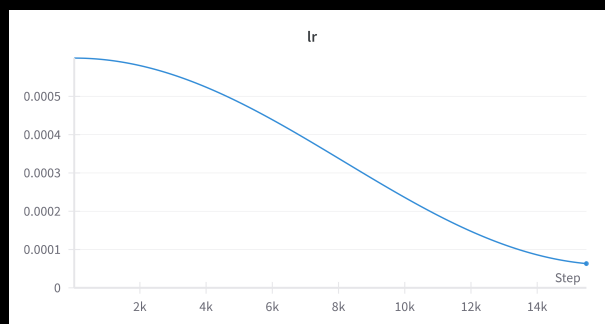
۳-۶-۲ ارزیابی مدل درام

مطابق با نمودار ۳-۳ می‌توان گفت که مدل بنفش^۲ با مقدار خطای اولیه کمتری شروع می‌شود و به تدریج در طول زمان کاهش می‌یابد. این نشان‌دهنده یک فرآیند یادگیری پایدار است. کاهش تدریجی مقدار خطا نشان می‌دهد که مدل در حال یادگیری و تطبیق خوب است که این یک نشانه مثبت است. با تنظیمات بیشتر و دوره‌های آموزشی اضافی، مدل بنفش پتانسیل دستیابی به عملکرد حتی بهتر را دارد. ولی به احتمال زیاد ادامه آموزش این مدل باعث overfit شدن خواهد شد زیرا حجم داده‌های درام خیلی بالا نیست و ادامه بیش از این احتمالاً باعث overfit می‌شود.

^۱ در اینجا مدل سبز به مدل ctx512 L20 D512 اشاره می‌کند. که مدل نهایی انتخاب شده برای ارزیابی است.
^۲ در اینجا مدل بنفش به مدل ctx512 L20 D512 اشاره می‌کند. که مدل نهایی انتخاب شده برای ارزیابی است.

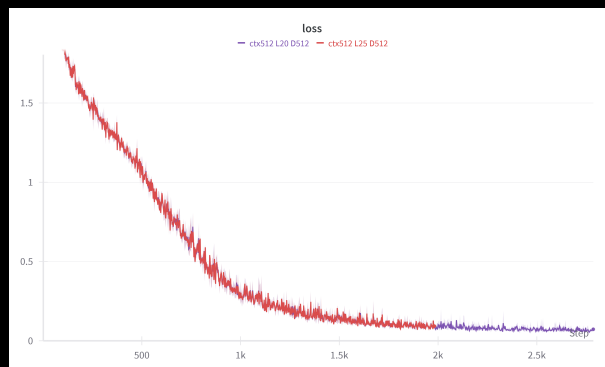


(الف) تغییر مقدار تابع خطا

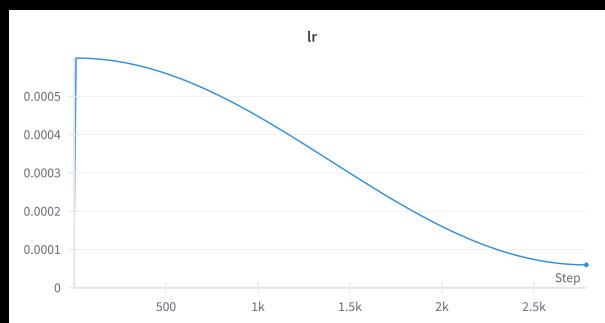


(ب) تغییر نرخ یادگیری

شکل ۳-۲ - نمودار های پیشرفت یادگیری مدل پیانو



(الف) تغییر مقدار تابع خطا



(ب) تغییر نرخ یادگیری

شکل ۳-۳ - نمودار های پیشرفت یادگیری مدل درام

۳-۶-۳ ارزیابی عملکرد مدل با معیار های موسیقی

برای ارزیابی عملکرد مدل زبان کوچک ما در تولید موسیقی لو-فی، از مجموعه ای از معیارهای ارزیابی استفاده می شود که کیفیت های مختلف موسیقی مانند ریتم، ملودی و توالی را ارزیابی می کنند. بخش های زیر معیارهای مورد استفاده برای ارزیابی عملکرد مدل را به تصویر می کشند. اکثر این معیار ها در مقاله A Comprehensive Survey for Evaluation Methodologies of AI-Generated Music [۹] معرفی شده اند. و ما آن ها را با کتابخانه music21 [۹] پیاده سازی کردیم.

۱-۳-۶-۳ هماهنگی ریتم (Rhythm Consistency)

هماهنگی ریتم یک متریک است که به بررسی نوسان یا یکنواختی طول نت ها در یک قطعه موسیقی می پردازد. این متریک نشان می دهد که قطعه های موسیقی چه میزان یکنواختی و یا چه میزان نوسانی دارند.

فرمول و محاسبه

فرمول هماهنگی ریتم به صورت زیر است: $RC = \frac{\mu}{\mu + SD}$

در این فرمول:

- SD معیار انحراف معیار طول نت ها است
- μ میانگین طول نت ها است

الگوریتم ۳-۳ نشان دهنده نحوه انجام این کار برای یک فایل MIDI است.^۱

الگوریتم ۳-۳ هماهنگی ریتم

Require: MIDI file

Ensure: rhythm consistency measure

Parse the MIDI file using a converter to obtain a score

Extract notes from the score

Extract durations of notes

Calculate the mean duration of notes

Calculate the variance of note durations

Calculate the standard deviation of note durations as the square root of the variance

Return the standard deviation of note durations

^۱کد پایتون این الگوریتم را میتوانید در مشاهده کنید.

این فرمول برای نرمال سازی معیار نقطه کمره طول نت ها توسط میانگین، امکان مقایسه و تفسیر بهتر هماهنگی ریتم را فراهم می کند. نتیجه به صورت یک عدد بین ۰ و ۱ است:

- هماهنگی ریتم = ۱ نشان دهنده هماهنگی کامل (همه نت ها طول یکسانی دارند)
- هماهنگی ریتم = ۰ نشان دهنده عدم هماهنگی کامل (نت ها طول های بسیار متفاوت دارند)
- هماهنگی ریتم = ۵.۰ نشان دهنده هماهنگی متوسط (نت ها طول هایی یکنواخت اما با کمی نوسان دارند)

۲-۳-۶-۳ شباهت ملودی (Melodic Similarity Metric)

متحلیل شباهت ملودی، یک متد برای اندازه گیری شباهت بین دو ملودی بر اساس ترتیب نت هایشان است. این متد ساده و مستقیماً از نسبت نت های مشابه دو ملودی برای محاسبه شباهت استفاده می کند. فرمول:

$$\text{شباهت ملودی} = (\text{تعداد نت های مشابه}) / (\text{اندازه کوتاه ترین ملودی})$$

در این فرمول: تعداد نت های مشابه، تعداد نت هایی است که در همان موقعیت در دو ملودی وجود دارند. اندازه کوتاه ترین ملودی، طول کوتاه ترین ملودی بین دو ملودی است. الگوریتم ۳-۴ نشان دهنده نحوه انجام این کار برای یک فایل MIDI است.^۱

الگوریتم ۳-۴ شباهت ملودی

Require: MIDI file, Reference MIDI file

Ensure: melodic similarity measure

Parse the input MIDI file using a converter to obtain a score

Parse the reference MIDI file using a converter to obtain a reference score

Extract generated melody from the score

Extract reference melody from the reference score

Extract pitch sequences from the generated and reference melodies

Calculate the number of matching pitches between the generated and reference pitch sequences

Calculate the similarity as the ratio of matching pitches to the length of the shorter pitch sequence

Return the melodic similarity measure

۳-۳-۶-۳ ثبات تونال (Tonal Stability Metric)

متحلیل ثبات تونال، یک متد برای اندازه گیری ثبات تونال یک ملودی است. این متد از تعداد تغییرات کلید در یک ملودی برای محاسبه ثبات تونال استفاده می کند.

فرمول:

ثبات تونال = ۱ - (تعداد تغییرات کلید)

به عبارت دیگر، هرچه تعداد تغییرات کلید در یک ملودی کمتر باشد، ثبات تونال آن بیشتر است.

الگوریتم ۳-۵ نشان دهنده نحوه انجام این کار برای یک فایل MIDI است.^۱

الگوریتم ۳-۵ ثبات تونال

Require: MIDI file

Ensure: tonal stability measure

Parse the MIDI file using a converter to obtain a score

Extract key changes from the score

Count the number of key changes

Return the number of key changes

۴-۳-۶-۳ هماهنگی هارمونیک (Harmonic Coherence Metric)

هماهنگی هارمونیک یک متد برای اندازه گیری هماهنگی هارمونیک یک ملودی است. این متد از نسبت هارمونی سازگار (ملایمت) و غیرسازگار (ناهم خوانی) در یک ملودی برای محاسبه هماهنگی هارمونیک استفاده می کند.

فرمول:

هماهنگی هارمونیک = (نسبت ملایمت + نسبت ناهم خوانی)

نسبت ملایمت: نسبت تعداد هارمونی های سازگار به کل تعداد هارمونی ها نسبت دیشناسه: نسبت

تعداد هارمونی های غیرسازگار به کل تعداد هارمونی ها

به عبارت دیگر، هرچه نسبت ملایمت در یک ملودی بیشتر باشد، هماهنگی هارمونیک آن بیشتر

است. الگوریتم ۳-۶ نشان دهنده نحوه انجام این کار برای یک فایل MIDI است.^۱

برای کد های الگوریتم های گفته می توانید به ؟؟ مراجعه کنید.

```

1: procedure analyzeHarmonicCoherence(midi_file)
2:   Parse MIDI file:  $score \leftarrow converter.parse(midi\_file)$ 
3:   Analyze key:  $key\_analysis \leftarrow score.analyze('key')$ 
4:   Chordify score:  $chords \leftarrow score.chordify()$ 
5:   Get chord list:  $chord\_list \leftarrow [ch \text{ for } ch \text{ in } chords.recurse().Chord]$ 
6:   Count consonant chords:
7:      $consonance\_count \leftarrow \sum_{ch \in chord\_list} 1 \text{ if } ch.isConsonant() \text{ else } 0$ 
8:   Calculate ratios:  $dissonance\_count \leftarrow len(chord\_list) - consonance\_count$ 
9:    $total\_chords \leftarrow len(chord\_list)$ 
10:   $consonance\_ratio \leftarrow consonance\_count / total\_chords$ 
11:   $dissonance\_ratio \leftarrow dissonance\_count / total\_chords$ 
12:  return  $consonance\_ratio, dissonance\_ratio$ 

```

جدول ۳-۲ - نتایج این معیار ها برای مدل پیانو

مقدار	متد
۳۴.۰	هماهنگی ریتم
۸۷.۰	هماهنگی هارمونیک (ملایمت)
۱۳.۰	هماهنگی هارمونیک (ناهم خوانی)
۲.۰	تغییرات کلید

۳-۶-۵ نتایج این معیار ها برای مدل پیانو

تحلیل جدول ۴۴ به صورت زیر است:

هماهنگی ریتم مقدار ۳۴.۰ نشان دهنده این است که ریتم ملودی دارای تناوب های نسبتاً مشابه است، اما همچنان

دارای برخی از تغییرات و نوسانات است.

هماهنگی هارمونیک مقدار ۸۷.۰ نشان دهنده این است که هارمونی های ملودی در مجموع سازگار هستند و دارای

هماهنگی بالایی هستند.

تغییرات کلید مقدار ۲.۰ نشان دهنده این است که ملودی تقریباً هیچ تغییراتی در کلید ندارد و در کلید ثابت

باقی می ماند.

ملودی های ساخته شده توسط مدل دارای هماهنگی بالایی در هارمونی و ریتم است، اما هنوز دارای

بعضی از ناهمسانی ها و عدم هماهنگی است که می تواند بهبود یابد.

۷-۳ ساخت موسیقی نهایی

در کد ما که در الگوریتم ۳-۹ نشان داده شده است، برای تولید موسیقی لو-فای، چندین روش به کار گرفته شده است تا یک قطعه موسیقی منحصر به فرد و هماهنگ ایجاد شود. این فرآیند با تولید یک تمپوی تصادفی آغاز می‌شود که به هر قطعه موسیقی تنوع و یگانگی می‌بخشد. هسته اصلی تولید موسیقی شامل تبدیل خروجی مدل به فایل‌های MIDI با استفاده از تابع `convert_str_to_midi` است که مربوط به بخش ۳-۳ است که در الگوریتم ۳-۷ و الگوریتم ۳-۸ نشان داده شده است. این فایل‌های MIDI سپس با استفاده از FluidSynth، یک نرم‌افزار سینتت‌سایزر که برای تولید صدا به فونت‌های صوتی متکی است، به فایل‌های صوتی تبدیل می‌شوند. فونت‌های صوتی، مانند فونت مشخص شده در کد ما (SGM-v2.01-NicePianosGuitarsBass-V1.2.sf2)، مجموعه‌ای از نمونه‌های صوتی هستند که صدای سازهای واقعی و با کیفیت بالا را فراهم می‌کنند. این امر برای موسیقی لو-فای بسیار مهم است، زیرا بافت و طنین سازها به طور قابل توجهی به زیبایی کلی موسیقی کمک می‌کند. استفاده از فونت‌های صوتی چندین مزیت دارد. آن‌ها صدای سازهای واقعی را فراهم می‌کنند که اصالت موسیقی تولید شده را افزایش می‌دهد. همچنین امکان سفارشی‌سازی را فراهم می‌کنند و به شما اجازه می‌دهند فونت‌های صوتی را انتخاب کنید که با سبک و احساس مورد نظر موسیقی مطابقت داشته باشند. یکنواختی در کیفیت صدا نیز یکی دیگر از مزایا است که اطمینان می‌دهد موسیقی دارای صدای یکنواختی است. علاوه بر این، فونت‌های صوتی فرآیند سینتت‌سایز را کارآمد می‌کنند، زیرا FluidSynth نمونه‌های صوتی از پیش ضبط شده را طبق دستورالعمل‌های MIDI پخش می‌کند.

پس از تبدیل فایل‌های MIDI به فایل‌های صوتی، ما از `pydub` برای بارگذاری و دستکاری این قطعات صوتی استفاده می‌کنند. این شامل رول پیانو اصلی، لوپ‌های درام که توسط مدل‌های ما ساخته شده است و `sfx sound` است. یک `sfx sound` تصادفی انتخاب می‌شود تا به موسیقی بافت و تصادفی بودن اضافه کند. توالی درام به طور مشابه با ملودی اصلی پردازش می‌شود تا هماهنگی با موسیقی تولید شده را تضمین کند. قطعات صوتی مختلف سپس برای ایجاد لایه‌دار روی هم قرار می‌گیرند. حجم رول پیانو تنظیم می‌شود و لوپ‌های `sfx sound` و درام برای مطابقت با مدت زمان رول پیانو تکرار می‌شوند. قطعه موسیقی نهایی با تکرار قطعات مخلوط شده به طول مورد نظر گسترش می‌یابد و با یک افکت محو شدن در پایان، یک پایان نرم ایجاد می‌شود. برای دسترسی به کدها و اجرا آنها به ؟؟ مراجعه کنید.

```

1: Input: utils, token, state, channel, end_token_pause
2: Output: Iterator of (Optional MIDI Message, DecodeState)
3: if state is None then
4:     Initialize state
5: token ← token.strip()
6: if token is empty or starts with "<" then
7:     yield (None, state) return
8: if token is "<end>" then
9:     Update state with end_token_pause
10:    if utils.cfg.decode_end_held_note_delay ≠ 0.0 then
11:        for (channel, note), start_time in state.active_notes do
12:            Convert delta_accum to ticks
13:            Remove (channel, note) from active_notes
14:            yield (note_off message, state)
15:    yield (None, state) return
16: if token is a wait token then
17:    Update state with wait token delta
18:    if utils.cfg.decode_end_held_note_delay ≠ 0.0 then
19:        for (channel, note), start_time in state.active_notes do
20:            if note held too long then
21:                Convert delta_accum to ticks
22:                Remove (channel, note) from active_notes
23:                yield (note_off message, state) return
24: else
25:    (bin, note, velocity) ← utils.note_token_to_data(token)
26:    Convert delta_accum to ticks
27:    if velocity > 0 then
28:        if utils.cfg.decode_fix_repeated_notes and (channel, note) in active_notes then
29:            Remove (channel, note) from active_notes
30:            yield (note_off message, state)
31:        Add (channel, note) to active_notes
32:        yield (note_on message, state)
33:    else
34:        Remove (channel, note) from active_notes
35:        yield (note_off message, state)
36: yield (None, state)

```

```

1: Input: utils, data, channel
2: Output: Iterator of MIDI Messages
3: state  $\leftarrow$  None
4: for token in data.split(" ") do
5:     for msg, new_state in token_to_midi_message(utils, token, state, channel) do
6:         state  $\leftarrow$  new_state
7:         if msg is not None then
8:             yield msg

```

```

1: procedure GenMusic(data, dataDrum)
2:     tempo  $\leftarrow$  5 18) random.randint(14,                                 $\triangleright$  Set tempo
3:     convert_str_to_midi(__join(data), tempo)                              $\leftarrow$ 
4:     save(relpath("./soundFont/mdiOut.mid"))                              $\triangleright$  Generate MIDI file
5:     fs  $\leftarrow$  FluidSynth(relpath("./soundFont/SGM-v2.01-NicePianosGuitarsBass-V1.2.sf2"))
6:      $\triangleright$  Load sound font
7:     fs.midi_to_audio(relpath("./soundFont/mdiOut.mid"), relpath("./soundFont/output.flac"))
8:      $\triangleright$  Convert MIDI to audio
9:     pianoRoll  $\leftarrow$  AudioSegment.from_file(relpath("./soundFont/output.flac"))  $\triangleright$  Load
10:    piano roll
11:    fillName  $\leftarrow$  random.choice(os.listdir(relpath("./loops/vinyl")))  $\triangleright$  Select fill name
12:    fill  $\leftarrow$  AudioSegment.from_file(relpath("./loops/vinyl/fillName"))  $\triangleright$  Load fill
13:    convert_str_to_midi(__join(data), 400, 9)                              $\leftarrow$ 
14:    save(relpath("./soundFont/mdiOutDrum.mid"))                              $\triangleright$  Generate drum MIDI file
15:    fs.midi_to_audio(relpath("./soundFont/mdiOutDrum.mid"), relpath("./soundFont/output.flac"))
16:     $\triangleright$  Convert drum MIDI to audio
17:    drum  $\leftarrow$  AudioSegment.from_file(relpath("./soundFont/output.flac"))  $\triangleright$  Load drum
18: procedure mix_lines(music_len)
19:    pianoRoll  $\leftarrow$  pianoRoll + 10  $\triangleright$  Adjust piano roll
20:    fill  $\leftarrow$  fillmath.ceil(pianoRoll.duration_seconds/fill.duration_seconds)  $\triangleright$  Repeat
21:    fill
22:    drum  $\leftarrow$  drummath.ceil(pianoRoll.duration_seconds/drum.duration_seconds)  $\triangleright$ 
23:    Repeat drum
24:    music  $\leftarrow$  pianoRoll.overlay(fill - 10).overlay(drum + 3)  $\triangleright$  Mix audio
25:    music  $\leftarrow$  musicmath.ceil(music_len/music.duration_seconds)  $\triangleright$  Repeat music
26:    music  $\leftarrow$  music.fade_out(2SECOND)  $\triangleright$  Fade out music
27: return music

```

فصل چهارم

ساخت خط لوله متن به lo-fi

۴-۱ معماری خط لوله

این pipeline شامل چند مرحله ای برای تولید موسیقی است که از مدل های مختلف و تکنیک ها برای تولید موسیقی با کیفیت بالا از متون بنام استفاده می کند. این روش را می توان به چهار مرحله اصلی تقسیم کرد:

- تولید موسیقی از متن
- تبدیل نت به MIDI
- پیش بینی نت ها
- پردازش پس از تولید موسیقی

در مرحله اول، بستر از یک مدل تولیدی Musicgen [۶]، برای تولید موسیقی از متن استفاده می کنیم. مدل در داده مورد نظر، ”facebook/musicgen-small”، ذخیره شده است که یک مدل پیش آموزش داده شده^۱ را برای تولید موسیقی فراهم می کند. مدل، یک دنباله از مقادیر صوتی را بر اساس متن ورودی تولید می کند و صوت تولید شده را به عنوان فایل WAV ذخیره می کند.

^۱pre-trained

در مرحله دوم، بستر از یک مدل دیگری با نام basic pitch [۴] برای پیش بینی نت های صوتی تولید شده استفاده می کند. مدل پیش بینی نت های صوتی، فایل صوتی را تحلیل می کند و اطلاعات نت ها را استخراج می کند که سپس برای تولید فایل MIDI استفاده می شود. فایل MIDI، یک دنباله از نت ها با pitches، طول کلامات و داده های موسیقی دیگر را دارد.

در مرحله سوم، از الگوریتم ۱-۳ استفاده می کنیم تا فایل MIDI را به یک متن قابل فهم مدل تبدیل کنیم.

در مرحله چهارم، از الگوریتم ۶-۳ استفاده می کنیم تا آهنگ نهایی خود را بسازیم. فایل صوتی تولید شده سپس با استفاده از IPython display module پخش می شود و کاربر می تواند صوت تولید شده را بشنود.

۱-۱-۴ کمبودهای استفاده مستقیم از مدل MusicGen برای تولید موسیقی

شاید این سوال ایجاد شود چرا باید مدل قوی مانند Musicgen با این مدل ها pipe شود تا بتواند این موسیقی ساده را تولید کند. مگر به تنها قادر به انجام اینکار نیست؟ جواب این سوال این است که با اینکه مدل Musicgen، مانند بسیاری از مدل های تولید موسیقی دیگر، یک سیستم پیچیده است که می تواند موسیقی با کیفیت بالا تولید کند، اما همچنین دارای محدودیت هایی است. در اینجا چند دلیل وجود دارد که چرا استفاده مستقیم از آن برای تولید موسیقی ممکن است ایده آل نباشد:

۱. **کیفیت پایین و نویز:** همانطور که اشاره کردید، خروجی مدل Musicgen می تواند نویزی و با کیفیت پایین باشد. این به این دلیل است که مدل بر روی مقدار زیادی داده آموزش دیده است که شامل موسیقی نویزی و با کیفیت پایین است. مدل این الگوها را یاد می گیرد و می تواند منجر به خروجی نویزی و با کیفیت پایین شود.

۲. **سختی در تغییر و ویرایش موسیقی تولید شده:** مدل Musicgen موسیقی را به صورت فایل WAV تولید می کند که یک فرمت باینری است. این باعث می شود تغییر و ویرایش موسیقی تولید شده دشوار باشد. سازندگان آهنگ ممکن است بخواهند یک نت، آکورد یا ملودی خاص را تغییر دهند، اما انجام این کار با یک فایل WAV چالش برانگیز است.

۳. **انعطاف پذیری محدود در سبک و ژانر موسیقی:** مدل Musicgen بر روی یک مجموعه داده خاص آموزش دیده است، به این معنی که درک محدودی از سبک ها و ژانر های مختلف موسیقی

دارد. سازندگان آهنگ ممکن است بخواهند موسیقی را در یک سبک یا ژانر خاص ایجاد کنند، اما مدل Musicgen ممکن است نتواند موسیقی‌ای تولید کند که انتظارات آنها را برآورده کند.

۴. **سختی در همکاری با سایر موسیقی‌دانان:** مدل Musicgen موسیقی را به صورت فایل WAV تولید می‌کند که همکاری با سایر موسیقی‌دانان را دشوار می‌کند. سازندگان آهنگ ممکن است بخواهند موسیقی تولید شده خود را با مشارکت‌های سایر موسیقی‌دانان ترکیب کنند، اما انجام این کار با یک فایل WAV چالش‌برانگیز است.

به همین دلیل است که رویکردی که ما توصیف کردیم، که از مدل Musicgen برای تولید موسیقی استفاده می‌کنیم و سپس از یک مدل جداگانه برای پیش‌بینی گام صدای تولید شده و تبدیل آن به MIDI استفاده می‌کند و در نهایت ساخت موسیقی lo-fi با یک مدل دیگر می‌تواند ایده خوبی باشد. این رویکرد مزایای زیر را می‌دهد:

۱. بهبود کیفیت و کاهش نویز در موسیقی تولید شده

۲. کنترل بیشتر بر فرآیند تولید موسیقی

۳. تغییر و ویرایش آسان‌تر موسیقی تولید شده

۲-۴ ارزیابی خط لوله

متن کاربران اغلب شامل نکات احساسی و انتظاراتی است که به سختی می‌توان آن‌ها را به صورت ریاضی اندازه‌گیری کرد. وقتی کاربران درخواست‌های خود را برای تولید موسیقی ارائه می‌دهند حال و هوا، جو و تأثیر احساسی مورد نظر خود را با متن می‌خواهند منتقل کنند. معیارهای ریاضی می‌توانند جنبه‌هایی مانند دقت نت‌ها یا زمان‌بندی را اندازه‌گیری کنند، اما در ثبت تأثیر احساسی و ارزش زیبایی‌شناختی موسیقی نمی‌توانند خیلی خوب عمل کنند. پس برای اینکه بتوانیم این pipeline را ارزیابی کنیم یک نظر سنجی انجام داریم.

ما یک نظرسنجی با پنج کاربر، که به طوری حرفه‌ای در زمینه ساخت این نوع موسیقی هستند یا شنونده این نوع موسیقی هستند، برای ارزیابی اثربخشی خط لوله تولید موسیقی خود انجام دادیم. این نظرسنجی شامل سوالاتی در مورد کیفیت صدا، رضایت کاربر بود. متأسفانه، نتایج امیدوارکننده نبود. در جدول؟؟ خلاصه‌ای از بازخوردهای دریافت شده آمده است:

جدول ۴-۱ - نتایج نظرسنجی ارزیابی خط لوله

سوال	پاسخ
وضوح صدای تولید شده	۲۰٪ (یک نفر) آن را ضعیف ارزیابی کردند و به مشکلات نویز و اعوجاج اشاره کردند.
غنای صدای تولید شده	۴۰٪ (دو نفر) احساس کردند که صدا عمق و غنای کافی ندارد و آن را تخت توصیف کردند.
تطابق با ورودی‌ها	کاربران به عدم تطابق بین انتظارات و خروجی اشاره کردند.

چندین کاربر اشاره کردند که موسیقی تولید شده با ورودی‌های داده شده به خوبی تطابق ندارد، که منجر به عدم تطابق بین انتظارات و خروجی شد.

ایده تبدیل متن به آهنگ لوفای (lo-fi) ممکن است به دلیل وجود افکت‌های مختلف و نودهای متفاوت و همچنین عدم توانایی گفتار در تشبیه دقیق آن‌ها، به خوبی عمل نکند. موسیقی لوفای شامل عناصر مختلفی مانند افکت‌های صوتی، تغییرات در ریتم و تمپو، و استفاده از صداها محیطی است که به سختی می‌توان آن‌ها را به صورت متنی توصیف کرد.

علاوه بر این، نودهای موسیقی لوفای معمولاً دارای حس و حال خاصی هستند که به سختی می‌توان آن‌ها را با کلمات بیان کرد. این نودها ممکن است شامل حس آرامش، نوستالژی، یا حتی غم باشند که انتقال این احساسات از طریق متن به موسیقی نیازمند درک عمیق و دقیق از موسیقی و احساسات انسانی است.

بنابراین، به دلیل پیچیدگی‌های احساسی و توصیف درست احساسات موجود در موسیقی لوفای، تبدیل متن به آهنگ لوفای ممکن است نتواند به طور کامل و دقیق این عناصر را بازتاب دهد. و مدل بتواند موسیقی تولید کند که مطابق ورودی باشد که کاربر وارد کرده است. همچنین این نتیجه در پروژه jacobz/Lofi [۹] نیز به دست آمده است.^۱

^۱نتیجه این کار در اینجا قابل مشاهده است.

فصل پنجم

نتیجه‌گیری و پیشنهادها

۵-۱ نتیجه‌گیری

‘ در این مقاله، ما نشان دادیم که استفاده از کانتینر ها برای اجرای سریع و محلی مدل های زبانی بزرگ مزایای قابل توجهی دارد. کانتینر ها به ما امکان می دهند تا مدل های زبانی را بدون نیاز به نصب پیش نیاز های پیچیده و تنظیمات سخت افزاری، در هر سیستم عامل و پلتفرمی اجرا کنیم. این کار باعث افزایش قابلیت استفاده، کارایی و امنیت مدل های زبانی می شود. همچنین، کانتینر ها به ما کمک می کنند تا مدل های زبانی را به راحتی به صورت توزیع شده و مقیاس پذیر در شبکه های ابری یا لوکال اجرا کنیم. در نهایت، کانتینر ها به ما اجازه می دهند تا مدل های زبانی را با استفاده از فن آوری های جدید و بهینه سازی شده برای عملکرد بالاتر، بروز رسانی و توسعه دهیم. بنابراین، استفاده از کانتینر ها برای اجرای سریع و محلی مدل های زبانی بزرگ یک روش جذاب و قابل اعتماد است که در آینده بسیار پرکاربرد خواهد بود. برخی دیگر از مزایای این روش به صورت زیر است:

- با استفاده از کانتینرها، ما می‌توانیم مدل مدل های زبانی بزرگ را در هر محیطی که دارای اجراکننده کانتینر باشد، به راحتی اجرا کنیم. این کانتینرها سرعت و کارایی بالایی دارند و نیاز

به نصب و تنظیمات اضافی ندارند.

- این روش هیچ هزینه ای برای کاربران ندارد. کاربران فقط کافی است کانتینر مدل های زبانی بزرگ را دانلود و اجرا کنند و از آن بهره ببرند. همچنین کاربران می توانند کانتینر را با دیگران به اشتراک بگذارند و یا از کانتینرهای دیگران استفاده کنند.

- این روش امنیت داده های کاربر را حفظ می کند. کاربران نیازی ندارند که داده های خود را به سرورهای خارجی یا ابری ارسال کنند و یا از سرویس های پرداختی استفاده کنند. کانتینر مدل های زبانی بزرگ روی رایانه یا شبکه داخلی کاربر اجرا می شود و داده ها در دسترس کاربر می مانند.

- این روش به ما امکان می دهد که به یک API واحد برای مدل های زبانی دسترسی پیدا کنیم. ما می توانیم از پروتوکول های HTTP و gRPC برای ارتباط با مدل های زبانی بزرگ استفاده کنیم و داده ها را با فرمت های مختلفی مانند JSON، XML، Protobuf و غیره ارسال و دریافت کنیم. این API واحد ما را از پیچیدگی های مربوط به مدل های زبانی مختلف مستقل می کند.

۲-۵ پیشنهادها

یکی از چالش های موجود در استفاده از مدل های زبانی بزرگ، نیاز به منابع محاسباتی زیاد و پیچیده است. این مدل ها معمولاً نیاز به تعداد زیادی از پردازنده های گرافیکی یا تنسوری دارند که همه آنها باید با هم همگام سازی شوند. این کار باعث می شود که اجرای این مدل ها در محیط های محلی یا کوچک بسیار دشوار و گران قیمت باشد. برای حل این مشکل، یک راه حل ممکن استفاده از کانتینر هاست. کانتینر ها روشی برای بسته بندی و اجرای نرم افزار ها به صورت جدا و مستقل از سخت افزار و سیستم عامل زیرین هستند. با استفاده از کانتینر ها، می توان یک محیط یکنواخت و قابل حمل برای اجرای مدل های زبانی بزرگ فراهم کرد. در این مقاله، ما پروژه ای ساختیم که هدف عمده آن روی اجرای سریع محلی برای کاربر عادی یا سرور ها بود ولی مقایس پذیری این کانتینر ها در این مقاله بررسی نشده. و می توان این مورد را هم بررسی دقیق تر نمود که چگونه کانتینر های خود را بتوانیم روی چندین سرور در یک شبکه محلی قرار دهیم [۱].

- [1] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, M. Grella, K. K. GV, X. He, H. Hou, J. Lin, P. Kazienko, J. Koccon, J. Kong, B. Koptyra, H. Lau, K. S. I. Mantri, F. Mom, A. Saito, G. Song, X. Tang, B. Wang, J. S. Wind, S. Wozniak, R. Zhang, Z. Zhang, Q. Zhao, P. Zhou, Q. Zhou, J. Zhu, and R.-J. Zhu, "Rwkv: Reinventing rnns for the transformer era," 2023.
- [2] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," 2018.
- [3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.
- [4] H. M. de Oliveira and R. de Oliveira, "Understanding midi: A painless tutorial on midi format," *arXiv preprint arXiv:1705.05322*, 2017.
- [5] J. Zhang, "Lofi: ML-supported lo-fi music generator,"
- [6] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [7] S. Wu, X. Li, F. Yu, and M. Sun, "Tunesformer: Forming irish tunes with control codes by bar patching," in *Proceedings of the 2nd Workshop on Human-Centric Music Information Retrieval 2023 co-located with the 24th International Society for Music Information Retrieval Conference (ISMIR 2023), Milan, Italy, November 10, 2023* (L. Porcaro, R. Batlle-Roca, and E. Gómez, eds.), vol.3528 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023.
- [8] L. Callender, C. Hawthorne, and J. Engel, "Improving perceptual quality of drum transcription with the expanded groove midi dataset," 2020.
- [9] A. Moi and N. Patry, "HuggingFace's Tokenizers," Apr. 2023.
- [10] A. Andonian, Q. Anthony, S. Biderman, S. Black, P. Gali, L. Gao, E. Hallahan, J. Levy-Kramer, C. Leahy, L. Nestler, K. Parker, M. Pieler, J. Phang, S. Purohit, H. Schoelkopf, D. Stander, T. Songz, C. Tigges, B. Thérien, P. Wang, and S. Weinbach, "GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch," 9 2023.
- [11] B. Peng, D. Goldstein, Q. Anthony, A. Albalak, E. Alcaide, S. Biderman, E. Cheah, T. Ferdinan, H. Hou, P. Kazienko, *et al.*, "Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence," *arXiv preprint arXiv:2404.05892*, 2024.

پیوست‌ها

پ-۱ دسترسی به کد ها

؟؟ می توانید کد استفاده شده در این پروژه را در آدرس زیر دسترسی پیدا کنید: [لینک گیت‌هاب](#)

برای اجرای مدل میتوانید از اسکریپت زیر استفاده کنید [لینک اسکریپت گیت‌هاب](#)