
بسم الله الرحمن الرحيم

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

گروه مهندسی نرم‌افزار

پایان‌نامه کارشناسی رشته‌ی مهندسی کامپیوتر گرایش هوش مصنوعی و نرم افزار

آموزش یک مدل زبانی کوچک برای ساخت موسیقی lo-fi

استاد راهنما:

دکتر زهرا زجاجی

دانشجو:

سهیل سلیمی

شهریور ۱۴۰۳

تقدیم به:

من، مغز متفکر پشت ایده‌ی آموزش یک مدل زبان کوچک برای تولید موسیقی لو-فای، Chat GPT، دستیار هوش مصنوعی که همیشه آماده است تا متنی تولید کند که تقریباً منطقی باشد، و LLaMA3، که بیشتر اوقات واقعاً منطقی است. بدون کمک شما، این مقاله به یک حلقه بی‌پایان از سکوت تبدیل می‌شد و جهان از صداهای شیرین و دلنشین موسیقی لو-فای تولید شده توسط این مدل محروم می‌ماند. از کمک‌هایتان متشکرم و امیدوارم آماده باشید تا به برخی از بیت‌های تولید شده توسط هوش مصنوعی گوش دهید و لذت ببرید :)

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 [۱] 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

عنوان **صفحه**

۱: مقدمه ۱
۱-۱ پیش‌گفتار ۱

۲: بررسی پیشینه و ادبیات ۳
۲-۱ ادبیات موضوع ۳
۲-۱-۱ چگونگی کار معماری RWKV: ۳
۲-۱-۲ فرمت فایل MIDI ۵
۲-۲ روشهای پیشین ۹
۲-۲-۱ استفاده از معماری VAE ۹
۲-۲-۲ استفاده از معماری RNN LSTM ۱۰

۳: آموزش مدل و معماری ۱۱
۳-۱ معماری کلی پروژه ۱۱
۳-۲ دیتاسیت ها ۱۲
۳-۳ تبدیل MIDI به متن ۱۴
۳-۳-۱ رویکرد های موجود برای توکن‌سازی فایل‌های MIDI ۱۴
۳-۳-۲ توکن سازی ۱۹
۳-۴ آموزش مدل ۲۱
۳-۴-۱ پارامترهای آموزش مدل ۲۱
۳-۴-۲ نحویه آموزش مدل ۲۲
۳-۵ ارزیابی عملکرد مدل ۲۴
۳-۵-۱ ارزیابی مدل پیانو ۲۴
۳-۵-۲ ارزیابی مدل درام ۲۵
۳-۵-۳ ارزیابی عملکرد مدل با معیار های موسیقی ۲۵
۳-۶ ساخت موسیقی نهایی ۳۰

۴: ساخت خط لوله متن به lo-fi ۳۴
۴-۱ معماری خط لوله ۳۴
۴-۱-۱ کمبودهای استفاده مستقیم از مدل MusicGen برای تولید موسیقی ۳۵
۴-۲ ارزیابی خط لوله ۳۶

۵: نتیجه‌گیری و پیشنهادها ۳۸
۵-۱ نتیجه‌گیری ۳۸
۵-۲ پیشنهادها ۳۹

عنوان **صفحه**

منابع و مآخذ ۴۰

پیوست‌ها ۴۲
پ-۱ دسترسی به کد ها ۴۲

عنوان صفحه

فهرست تصاویر

عنوان **صفحه**

شکل ۲-۱: معماری RWKV برای مدل های زبان ۵
شکل ۲-۲: عناصر موجود در یک بلوک RWKV ۶
شکل ۲-۳: ساختار فایل MIDI ۷

شکل ۳-۱: ساختار pipeline ۱۲
شکل ۳-۲: نمودار های پیشرفت یادگیری مدل پیانو ۲۴
شکل ۳-۳: نمودار های پیشرفت یادگیری مدل درام ۲۵

فهرست جداول

عنوان **صفحه**

جدول ۳-۱: خلاصه‌ای از مجموعه داده ۱۳
جدول ۳-۲: نگاشت های فرمت MIDICompact ۱۶
جدول ۳-۳: نتایج این معیار ها برای مدل پیانو ۲۹
جدول ۴-۱: نتایج نظرسنجی ارزیابی خط لوله ۳۷

فهرست الگوریتم‌ها

عنوان **صفحه**

۳-۱ کوانتایز کردن سرعت ۱۸
۳-۲ توکن کردن فایل های MIDI ۲۰
۳-۳ آموزش مدل ۲۲
۳-۴ هماهنگی ریتم ۲۶
۳-۵ شباهت ملودی ۲۷
۳-۶ ثبات تونال ۲۸
۳-۷ هماهنگی هارمونیک ۲۹
۳-۸ تبدیل توکن به MIDI ۳۱
۳-۹ متن به MIDI message ۳۲
۳-۱۰ نحویه ساخت موسیقی نهایی ۳۳

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 [1] و یادگیری ماشین [2] 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

[1] Artificial intelligence
[2] 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

۲-۱ ادبیات موضوع

۲-۱-۱ چگونگی کار معماری RWKV:

RWKV [۲] [۱۲] یک معماری شبکه عصبی است که ترکیبی از مزایای شبکه‌های عصبی بازگشتی [۳][1] و ترانسفورمرها [۴][2] را به کار می‌گیرد. این معماری برای پردازش کارآمد دنباله‌های داده طراحی شده است، مانند RNN ها، اما همچنین از قابلیت‌های پردازش موازی ترانسفورمرها بهره می‌برد. این رویکرد ترکیبی به RWKV اجازه می‌دهد تا وابستگی‌های بلندمدت در داده‌ها را حفظ کند، که این مسئله برای RNNs سنتی به دلیل مشکلاتی مانند مشکل ناپدید شدن گرادیان چالش‌برانگیز است. در عین حال، از مقیاس‌پذیری و عملکرد ترانسفورمرها، به ویژه در طول آموزش، بهره‌مند می‌شود. به طور کلی، RWKV می‌تواند به صورت موازی مانند ترانسفورمرها آموزش ببیند اما در زمان استنتاج به صورت دنباله‌ای عمل کند، که این ویژگی آن را هم کارآمد و هم قدرتمند برای وظایف مختلف پردازش زبان طبیعی می‌سازد.

---

[1] Recurrent Neural Networks
[2] Transformers

۲-۱-۱-۱ RWKV در مقایسه با transformer

در مورد آموزش یک مدل زبانی برای تولید موسیقی لو-فای، ما تصمیم گرفتیم از معماری RWKV به جای معماری ترانسفورمر استفاده کنیم. یکی از دلایل اصلی این انتخاب این است که RWKV برای کارایی بیشتر و اجرای آسان‌تر روی CPU طراحی شده است که برای پروژه ما مفید است.

به طور خاص، معماری RWKV به صورت خطی و ترتیبی است که زمان‌های استنتاج سریع‌تری نسبت به معماری ترانسفورمر فراهم می‌کند. این به این دلیل است که مدل RWKV دنباله ورودی را به صورت ترتیبی، یک توکن در هر زمان، پردازش می‌کند، مشابه شبکه‌های عصبی بازگشتی [1]. این پردازش ترتیبی به RWKV اجازه می‌دهد تا از روابط زمانی در داده‌های ورودی بهره‌برداری کند و زمان‌های استنتاج آن را به طور قابل توجهی سریع‌تر کند.

علاوه بر این، RWKV یک مکانیزم توجه [2] را در خود جای داده است که به مدل اجازه می‌دهد هنگام تولید توکن بعدی، بر روی بخش‌های خاصی از دنباله ورودی تمرکز کند. این مکانیزم به RWKV امکان می‌دهد تا وابستگی‌ها و روابط بلندمدت بین توکن‌ها را به دست آورد، در حالی که همچنان کارایی یک مدل ترتیبی را حفظ می‌کند. مکانیزم توجه در RWKV به گونه‌ای طراحی شده است که از نظر محاسباتی کارآمد باشد و از ترکیبی از تبدیل‌های خطی و ضرب نقطه‌ای برای محاسبه وزن‌های توجه استفاده کند.

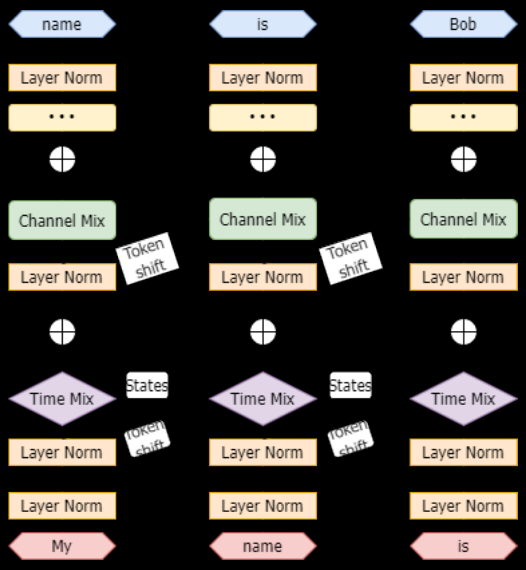
علاوه بر این، معماری RWKV به طور خاص برای پردازش داده‌های جریانی طراحی شده است که یک نیاز کلیدی برای بسیاری از برنامه‌های تولید موسیقی است. RWKV داده‌های ورودی را به صورت جریانی پردازش می‌کند، به طوری که هر توکن ورودی به محض ورود پردازش می‌شود، بدون نیاز به بارگذاری کل دنباله ورودی در حافظه. این به RWKV اجازه می‌دهد تا دنباله‌های ورودی بزرگ، مانند آهنگ‌ها یا فایل‌های صوتی طولانی، را بدون تمام شدن حافظه مدیریت کند. این قابلیت جریانی همچنین به RWKV اجازه می‌دهد تا موسیقی را به صورت بلادرنگ تولید کند، که آن را برای برنامه‌هایی مانند تولید موسیقی زنده یا ابزارهای ترکیب موسیقی مناسب می‌سازد.

علاوه بر این، RWKV همچنین زمان‌های آموزش سریع‌تری را فراهم می‌کند، زیرا می‌تواند داده‌های ورودی را به صورت جریانی، یک توکن در هر زمان، پردازش کند، به جای نیاز به بارگذاری کل دنباله

[1] RNNs
[2] Attention mechanism

ورودی در حافظه به صورت یکجا.

به طور کلی، معماری RWKV تعادلی بهتر بین عملکرد، کارایی و سهولت استقرار فراهم می‌کند و آن را به یک انتخاب ایده‌آل برای تولید موسیقی لو-فای تبدیل می‌کند.



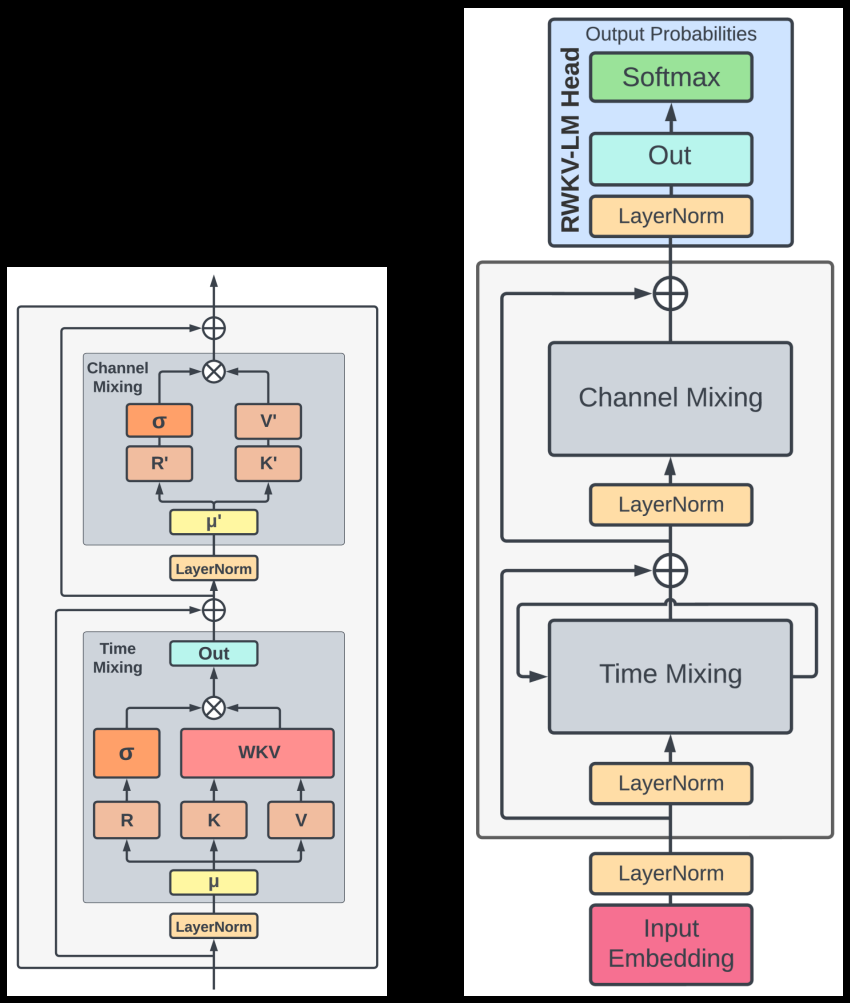
شکل ۲-۱ - معماری RWKV برای مدل های زبان

همانطور که در ۲-۱ 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 ؟؟ نشان داده شده‌اند. پس از آخرین بلوک، یک سر خروجی ساده شامل یک لایه نرمال‌سازی [1] و یک پروجکشن خطی برای تولید لاجیت‌ها [2] جهت پیش‌بینی توکن بعدی و محاسبه‌ی خطای متقاطع [3] در طول آموزش استفاده می‌شود.

۲-۱-۲ فرمت فایل MIDI

فرمت MIDI [4] [۵] 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

[1] (LayerNorm)
[2] (logits)
[3] (cross-entropy loss)
[4] Musical Instrument Digital Interface



(الف) Residual block

(ب) Final head of language model

شکل ۲-۲ - عناصر موجود در یک بلوک RWKV

موسیقی، زمان‌بندی، مدت زمان و شدت صدا برای هر نت هستند.

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.

از دیدگاه کامپیوتری، فایل‌های MIDI به عنوان مجموعه‌ای از پیام‌های دیجیتالی ذخیره می‌شوند که هر پیام شامل اطلاعاتی درباره نحوه پخش موسیقی است. این پیام‌ها به صورت باینری کدگذاری می‌شوند و شامل سه بخش اصلی هستند:

۱. **پیام‌های وضعیت** [1]: این پیام‌ها نوع عملیاتی که باید انجام شود را مشخص می‌کنند، مانند نواختن یک نت، تغییر شدت صدا، یا تغییر ابزار موسیقی.
۲. **پیام‌های داده** [2]: این پیام‌ها اطلاعات دقیق‌تری درباره عملیات مشخص شده در پیام‌های وضعیت ارائه می‌دهند، مانند شماره نت، شدت صدا، و مدت زمان.
۳. **زمان‌بندی** [3]: 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 [4] 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.

شکل ۲-۳ - ساختار فایل MIDI

[1] Status Messages
[2] Data Messages
[3] Timing
[4] Note On

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 زیادی داشته باشند. در مقابل، فایل‌های MIDI تنها شامل داده‌های دیجیتالی هستند که نویز 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 [1] 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.

[1] Track

۲-۲ روشهای پیشین

۲-۲-۱ استفاده از معماری VAE

پروژه jacbz/Lofi [۶] با استفاده از معماری VAE کار مشابهی را انجام می دهد. استفاده از معماری RWKV، معماری که ما در این پروژه استفاده کرده‌ایم، برای ساخت موزیک لوفای [۱] مزایای متعددی نسبت به VAE [۲] 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 مکانیزم توجه، 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 [۳] ، 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.

۲-۲-۱-۱ محدودیت‌های پروژه jacbz/Lofi

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.

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.

---

[1] Lo-Fi
[2] Variational Autoencoder
[3] Attention mechanism

۲-۲-۲ استفاده از معماری RNN LSTM

استفاده از معماری RWKV برای ساخت موزیک لوفای مزایای متعددی نسبت به LSTM [1] 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 توجه [2]، 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 بازسازی آهنگ‌های طولانی و پیچیده، ممکن است کیفیت نهایی موزیک را کاهش دهد.

---

[1] Long Short-Term Memory

[2] Attention mechanism

فصل سوم
آموزش مدل و معماری

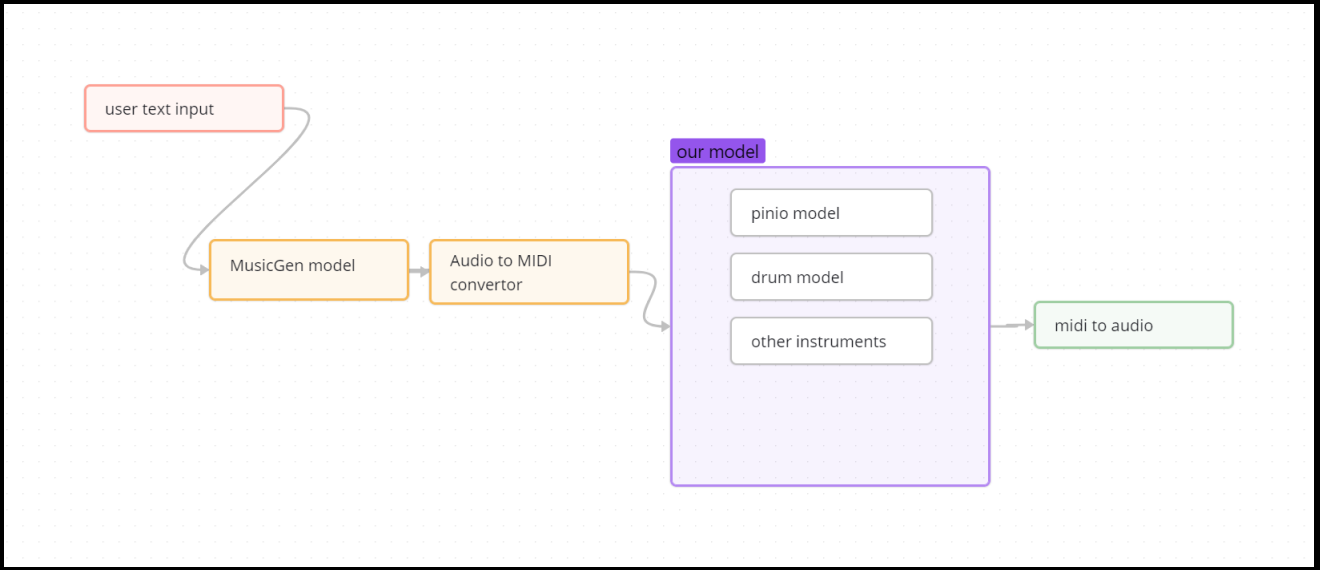
۳-۱ معماری کلی پروژه

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) [۷] 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 موسیقیایی 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 دیتاسیت ها

۳-۲-۰-۱ مدل پیانو

برای انجام آموزش مدل پیانو خود، ما از مجموعه داده گسترده‌ای به نام مجموعه داده MIDI موسیقی ایرلندی IrishMMD [۸] 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 abcnotation.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.

۳-۲-۰-۲ مدل دارم

مجموعه داده گسترده EGMD [۹]

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

۳-۳-۱ رویکرد های موجود برای توکن‌سازی فایل‌های MIDI

در نمایش موسیقی به متن، چندین فرمت مختلف وجود دارد که می‌توان برای نمایش اطلاعات موسیقی استفاده کرد. یکی از رایج‌ترین فرمت‌ها، نوت‌نویسی ABC است که یک فرمت قابل خواندن توسط انسان است و موسیقی را با استفاده از ترکیبی از حروف و نمادها برای نشان دادن ارتفاع صدا، مدت زمان و سایر عناصر موسیقی نمایش می‌دهد. نوت‌نویسی ABC به طور گسترده‌ای در نظریه موسیقی و آموزش موسیقی استفاده می‌شود.

با این حال، برای آموزش یک مدل زبانی برای تولید موسیقی لو-فای، باید تعادل بین پیچیدگی داده‌های ورودی و توانایی مدل در یادگیری از آن‌ها را در نظر بگیریم. در این مورد، ما تصمیم گرفتیم فایل‌های MIDI را به فرمت MIDICompact توکن‌سازی کنیم که یک نمایش ساده‌تر از اطلاعات موسیقی است.

فرمت MIDICompact

در فرمت MIDICompact، هر توکن یک رویداد موسیقی را نشان می‌دهد که ساختار آن به شرح زیر است:

در اینجا تجزیه و تحلیل اجزای فرمت آمده است: اگر یک نوت را به صورت

```
<Note>:<velocity> <Wait time>
```

در نظر بگیریم خواهیم داشت

<Note> این نشان دهنده نوتی است که نواخته می‌شود، جایی که <Note> یک مقدار هگزادسیمال بین ۰ و ۲۵ است. در این نمایش، هر نوت یک مقدار هگزادسیمال منحصر به فرد دارد که در جدول ؟؟ نشان داده شده است.

<velocity> این یک جداکننده بین نوت و شدت صدا است.[۱] این نشان دهنده شدت صدای نوت است، جایی که <velocity> یک مقدار هگزادسیمال بین ۰ و ۱۵ است. شدت صدا به ۱۶ مقدار ممکن تقسیم می‌شود که در جدول ؟؟ نشان داده شده است.

[۱] در اینجا برای سادگی از ۲۵ نوت استفاده شده ولی در کد های نوشته شده از ۱۲۸ نوت استفاده شده که کیفیت یادگیری را افزایش می‌دهد.

<Wait time> این نشان دهنده یک توکن انتظار است که نشان می‌دهد چه مدت باید قبل از نواختن نوت بعدی صبر کرد. توکن انتظار به صورت یک مقدار دهدهی بین ۱ و ۱۰۰ نمایش داده می‌شود، جایی که ۱ نشان دهنده زمان انتظار بسیار کوتاه و ۱۰۰ نشان دهنده زمان انتظار بسیار طولانی است.

این فرمت اجازه می‌دهد تا اطلاعات موسیقی به صورت فشرده و کارآمد نمایش داده شود، که آن را برای استفاده در پروژه ما به خوبی عمل کند. تعداد کل توکن هایی که می‌تواند ساخته شود به صورت زیر است:

$$(Note * velocity) + Waittime + pad + start + end = (128 * 16) + 125 + 3 = 2176$$

مثال ۳-۱. برای مثال `<start> t1 3:5 t2 10:2 t5 1:14 t3 d:11 <end>` می تواند خروجی الگوریتم ۳-۲ باشد.

A (نوت ۳) با شدت صدای ۵ (نسبتاً سخت) و زمان انتظار ۱

C# (نوت ۱۰) با شدت صدای ۲ (نسبتاً نرم) و زمان انتظار ۲

B (نوت ۱) با شدت صدای ۱۴ (حداکثر) و زمان انتظار ۵

A# (نوت ۱۳) با شدت صدای ۱۱ (بسیار بسیار سخت) و زمان انتظار ۳

در حالی که فرمت ABC بیانگرتر است و می‌تواند طیف گسترده‌ای از ظرافت‌های موسیقی را نمایش دهد، همچنین پیچیدگی اضافی را معرفی می‌کند که ممکن است برای پروژه ما ضروری نیست. به ویژه، فرمت ABC نیاز دارد که مدل تعداد بیشتری از توکن‌ها و روابط بین آن‌ها را یاد بگیرد که می‌تواند منجر به بیش‌برازش و کاهش قابلیت تعمیم‌دهی شود.

در تولید موسیقی، مقادیر سرعت [1] اغلب پیوسته هستند، اما در تولید صدا که در بخش ۳-۶ توضیح داده شده است، تنها می‌تواند مقادیر گسسته تولید کند. با کوانتایز کردن مقادیر سرعت به تعداد ثابتی از بین‌ها [2]، سیستم می‌تواند خروجی‌ای سازگارتر و قابل پیش‌بینی‌تر تولید کند. این روش به‌ویژه در موسیقی لو-فای مفید است، جایی که هدف ایجاد صدایی سازگار است. کد یک تابع کوانتایز کردن سرعت را پیاده‌سازی می‌کند که می‌تواند خطی یا نمایی باشد. این تابع یک مقدار سرعت پیوسته را به یک بین گسسته نگاشت می‌کند، که یک تکنیک رایج در تولید موسیقی، به‌ویژه در موسیقی لو-فای است.

[1] velocity
[2] bins

Hex	نوت
۰	A0
۱	A#
۲	B
۳	C
۴	C#
۵	D
۶	D#
۷	E
۸	F
۹	F#
۱۰	G
۱۱	G#
۱۲	A
۱۳	A#
۱۴	B
۱۵	C
۱۶	C#
۱۷	D
۱۸	D#
۱۹	E
۲۰	F
۲۱	F#
۲۲	G
۲۳	G#
۲۴	A
۲۵	B

(الف)
نگاشت
نوت ها

Hex	سرعت
۰	۰-۱۰٪ (بسیار نرم)
۱	۱۱-۲۰٪ (نرم)
۲	۲۱-۳۰٪ (نسبتاً نرم)
۳	۳۱-۴۰٪ (متوسط)
۴	۴۱-۵۰٪ (نسبتاً سخت)
۵	۵۱-۶۰٪ (سخت)
۶	۶۱-۷۰٪ (بسیار سخت)
۷	۷۱-۸۰٪ (بسیار بسیار سخت)
۸	۸۱-۹۰٪ (حداکثر)
۹	۹۱-۱۰۰٪ (حداکثر)
۱۰	۱۰۱-۱۱۰٪ (حداکثر)
۱۱	۱۱۱-۱۲۰٪ (حداکثر)
۱۲	۱۲۱-۱۳۰٪ (حداکثر)
۱۳	۱۳۱-۱۴۰٪ (حداکثر)
۱۴	۱۴۱-۱۵۰٪ (حداکثر)
۱۵	۱۵۱-۱۶۰٪ (حداکثر)

(ب) نگاشت سرعت

جدول ۳-۲ - نگاشت های فرمت MIDICompact

۳-۳-۱-۱ کوانتایز کردن خطی

در کوانتایز کردن خطی، مقدار سرعت پیوسته به تعداد ثابتی از بین‌های گسسته تقسیم می‌شود. اندازه بین با تقسیم حداکثر مقدار سرعت بر تعداد بین‌ها تعیین می‌شود. شاخص بین با تقسیم مقدار سرعت بر اندازه بین و گرد کردن به نزدیک‌ترین عدد صحیح محاسبه می‌شود.

۳-۳-۱-۲ کوانتایز کردن نمایی

در کوانتایز کردن نمایی، مقدار سرعت پیوسته با استفاده از یک تابع نمایی به یک بین گسسته نگاشت می‌شود. تابع نمایی توسط پارامتر velocity_exp کنترل می‌شود که شکل منحنی را تعیین می‌کند. مقدار بالاتر velocity_exp منجر به منحنی نمایی‌تر می‌شود، در حالی که مقدار پایین‌تر منجر به منحنی خطی‌تر می‌شود.

۳-۳-۱-۳ ریاضیات پشت کوانتایز کردن نمایی

فرمول کوانتایز کردن نمایی به صورت زیر است:

$$\text{bin\_index} = \left\lceil \frac{\text{velocity\_events} \cdot (\exp(\frac{\text{velocity}}{\text{velocity\_events}}) - 1)}{\exp(1) - 1} \right\rceil$$

که در آن velocity_events حداکثر مقدار سرعت، velocity مقدار سرعت ورودی و exp تابع نمایی است.

۳-۳-۱-۴ چرا کوانتایز کردن نمایی؟

کوانتایز کردن نمایی در تولید موسیقی بهتر است زیرا اجازه می‌دهد صدایی ظریف‌تر و دقیق‌تر ایجاد شود. در این پروژه ما مقدار پارامتر velocity_exp را برابر 0.33 قرار دادیم.

توضیحات الگوریتم ۳-۲ که به تبدیل داده ها به متن می‌کند به صورت زیر است:

- 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

الگوریتم ۳-۱ کوانتایز کردن سرعت

procedure velocity_to_bin(velocity)
$bin\_size \leftarrow \frac{\text{velocity\_events}}{\text{velocity\_bins}-1}$
if velocity_exp $==$ 1.0 **then**
$bin \leftarrow \lceil \frac{\text{velocity}}{bin\_size} \rceil$
else
$$bin \leftarrow \left\lceil \left(\text{velocity\_events} \cdot \left(\left(\text{velocity\_exp}^{\frac{\text{velocity}}{\text{velocity\_events}}} - 1 \right) \right) / (\text{velocity\_exp} - 1) \right) / bin\_size \right\rceil$$
return bin
procedure bin_to_velocity(bin)
$bin\_size \leftarrow \frac{\text{velocity\_events}}{\text{velocity\_bins}-1}$
if velocity_exp $==$ 1.0 **then**
$velocity \leftarrow \max(0, \lceil bin \cdot bin\_size - 1 \rceil)$
else
$$velocity \leftarrow \max(0, \lceil \text{velocity\_events} \cdot \log\left(\frac{(\text{velocity\_exp}-1) \cdot bin \cdot bin\_size}{\text{velocity\_events}} + 1, \text{velocity\_exp} \right) - 1 \rceil)$$
return $velocity$

- **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.
- **نهایی‌سازی**: 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 موسیقیایی فایل MIDI به‌طور دقیق در توالی توکن‌ها نمایش داده می‌شود.

۳-۳-۲ توکن سازی

در کار ما، از یک رویکرد ساده برای آماده‌سازی و آموزش مدل با استفاده از کتابخانه Tokenizer [۱۰] استفاده کردیم. در اینجا توضیح دقیقی از این فرآیند آورده شده است:

۳-۳-۲-۱ توکن‌سازی 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.

الگوریتم ۳-۲ توکن کردن فایل های MIDI

```
function convert_midi_to_str(cfg, filter_cfg, mid, augment=None)
    Initialize state variables
    function flush_token_data_buffer
        Convert token data buffer to token data
        Append formatted tokens to output
        Clear token data buffer
    function consume_note_program_data(prog, chan, note, vel)
        if token is valid then
            if delta_time_ms > threshold then
                Check if any notes are held
                if no notes are held then
                    Call flush_token_data_buffer()
                    Append ”<end>” to output
                    Reset output and state variables
            Generate wait tokens and append to output
            Reset delta_time_ms
            Append token data to buffer
            Set started_flag to True
    for each msg in mid.tracks[0] do
        Update delta_time_ms with msg.time
        function handle_note_off(ch, prog, n)
            if pedal is on then
                Set pedal event
            else
                Call consume_note_program_data(prog, ch, n, 0)
                Remove note from channel_notes
        if msg.type is ”program_change” then
            Update channel_program
        else if msg.type is ”note_on” then
            if velocity is 0 then
                Call handle_note_off
            else
                Remove pedal event if exists
                Call consume_note_program_data with mixed volume
                Add note to channel_notes
        else if msg.type is ”note_off” then
            Call handle_note_off
        else if msg.type is ”control_change” then
            Update channel state based on control type
        else
            pass
    Call flush_token_data_buffer()
    Append ”<end>” to output
    return output_list
```

۳-۳-۲-۲ 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 و نوشته می‌شود و با بسیاری از ابزارهای پردازش داده‌ها به خوبی ادغام می‌شود.

۳-۳-۲-۳ 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 سربار 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 فایل های JSONL به فرمت binidx از بخشی از کد های کتابخانه gpt-neox [۱۱] 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.

۳-۴ آموزش مدل

۳-۴-۱ پارامترهای آموزش مدل

در این بخش، پارامترهای مورد استفاده برای آموزش مدل توضیح داده شده‌اند:

ما از معماری `rwkv-0.6` [۱۲] 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

[۱] Context Length بی‌نهایت فقط در هنگام اجرای مدل معنا می دهد.

یادگیری اولیه و نهایی برابر با 6×10^{-4} و 6×10^{-5} است.

۳-۴-۲ نحویه آموزش مدل

الگوریتم ۳-۳ آموزش مدل

```
Function save_pth(dd, ff)
torch.save(dd, ff)
Class ResetValDataloader(Callback)
Function on_validation_start(trainer, pl_module)
trainer.reset_val_dataloader(pl_module)
Class TrainCallback(Callback)
Function __init__(self, args)
self.args = args
Function on_train_batch_start(self, trainer, pl_module, batch, batch_idx)
Adjust learning rate based on global step and schedule
for param_group in trainer.optimizers[0].param_groups do
    param_group['lr'] = lr * param_group['my_lr_scale'] if args.layerwise_lr > 0 else lr
trainer.my_lr = lr
Function on_train_batch_end(self, trainer, pl_module, outputs, batch, batch_idx)
Log metrics and update loss
Function on_train_epoch_start(self, trainer, pl_module)
Update dataset attributes
Function on_train_epoch_end(self, trainer, pl_module)
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
    save_pth(pl_module.state_dict(), f'args.proj_dir/self.prefix_args.epoch_begin + trainer.current_epoch.pth')
```

الگوریتم ۳-۳ برای مدیریت و بهینه‌سازی فرآیند آموزش مدل با استفاده از PyTorch Lightning [۱۳] طراحی شده است.

ابتدا، تابع `save_pth` که برای ذخیره دیکشنری حالت مدل در یک مسیر فایل مشخص استفاده 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 [1] 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.

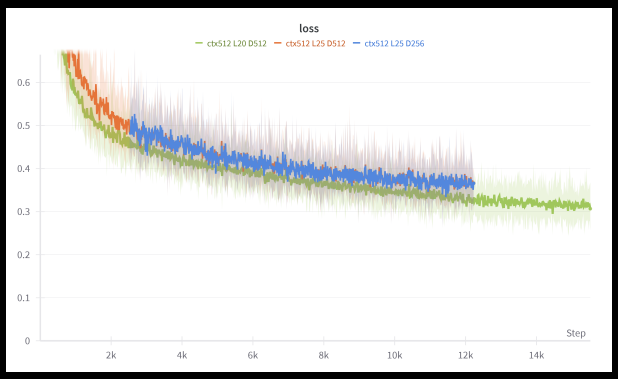
لاگ‌گیری و نظارت گسترده بخش‌های جدایی‌ناپذیر این بخش هستند. متدهای on_train_batch_start و on_train_batch_end شامل مکانیزم‌های لاگ‌گیری دقیقی برای دیدن پیشرفت آموزش هستند. این شامل لاگ‌گیری نرخ یادگیری و Loss Function، پیگیری تعداد توکن‌های پردازش شده در هر ثانیه و لاگ‌گیری به سرویس‌های مانند Weights & Biases (wandb) [۱۴] 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.

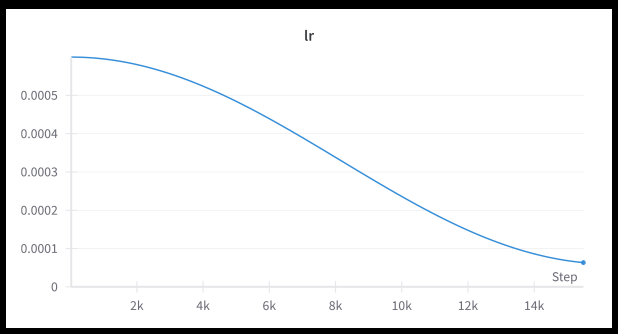
۳-۴-۲-۱ کتابخانه‌ها استفاده شده

چندین کتابخانه در این روش برای ساده‌سازی فرآیند آموزش استفاده می‌شوند. کتابخانه torch [۱۵] قابلیت‌های اصلی برای ساخت و آموزش شبکه‌های عصبی را فراهم می‌کند. PyTorch Lightning [۱۳] فرآیند آموزش را با انتزاع کدهای تکراری ساده می‌کند، حلقه‌های آموزش، اعتبارسنجی و تست را 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.

[1] warm up



(الف) تغییر مقدار تابع خطا



(ب) تغییر نرخ یادگیری

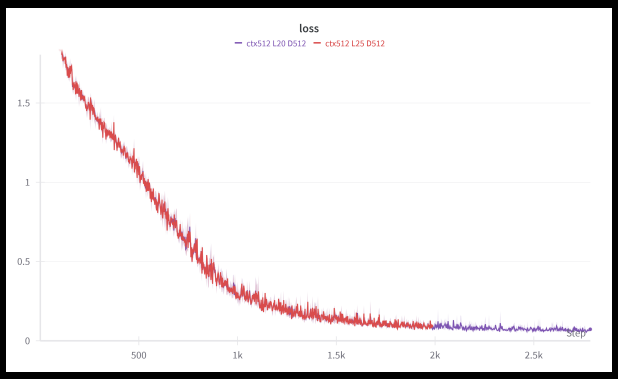
شکل ۳-۲ - نمودار های پیشرفت یادگیری مدل پیانو

۳-۵ ارزیابی عملکرد مدل

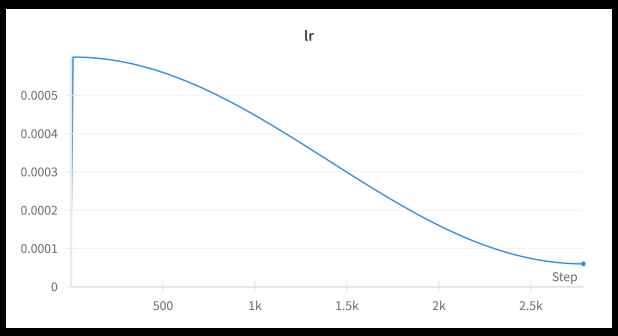
۳-۵-۱ ارزیابی مدل پیانو

با توجه به شکل ۳-۲ می توان 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.

[۱] در اینجا مدل سبز 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 ۳-۳ - نمودار های پیشرفت یادگیری مدل درام

۳-۵-۲ ارزیابی مدل درام

مطابق با نمودار ۳-۳ میتوان گفت که مدل بنفش [۱] 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 داده های دارم خیلی بالا نیست و ادامه بیش از این احتمالا باعث overfit می شود.

۳-۵-۳ 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 می‌کشند. اکثر این

[۱] در اینجا مدل بنفش به مدل ctx512 L20 D512 اشاره می‌کند. که مدل نهایی انتخاب شده برای ارزیابی است.

میعار ها در مقاله A Comprehensive Survey for Evaluation Methodologies of AI-Generated Music [۱۶] معرفی شده اند. و ما آن ها را با کتابخانه music21 [۱۷] پیاده سازی کردیم.

۳-۵-۳-۱ 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 ۳-۴ نشان دهنده نحویه انجام این کار برای یک فایل MIDI است.[۱]

الگوریتم ۳-۴ 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

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)

[۱] کد پاینون این الگوریم را میتوانید در مشاهده کنید.

- هماهنگی ریتم = ۵.۰ نشان دهنده هماهنگی متوسط (نت ها طول هایی یکنواخت اما با کمی نوسان دارند)

۳-۵-۳-۲ شباهت ملودی (Melodic Similarity Metric)

متحلیل شباهت ملودی، یک متد برای اندازه گیری شباهت بین دو ملودی بر اساس ترتیب نت‌هایشان است. این متد ساده و مستقیماً از نسبت نت‌های مشابه دو ملودی برای محاسبه شباهت استفاده می‌کند.

فرمول:

شباهت ملودی = (تعداد نت‌های مشابه) / (اندازه کوتاه‌ترین ملودی)

در این فرمول: تعداد نت‌های مشابه، تعداد نت‌هایی است که در همان موقعیت در دو ملودی وجود دارند. اندازه کوتاه‌ترین ملودی، طول کوتاه‌ترین ملودی بین دو ملودی است. الگوریتم ۳-۵ نشان دهنده نحویه انجام این کار برای یک فایل MIDI است. [۱]

الگوریتم ۳-۵ 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

۳-۵-۳-۳ 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 می‌کند.

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 ۳-۶ نشان دهنده نحویه انجام این کار برای یک فایل MIDI است.[۱]

الگوریتم ۳-۶ 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

۳-۵-۳-۴ 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 (ناهم‌خوانی) در یک ملودی برای محاسبه هماهنگی هارمونیک استفاده می‌کند.

فرمول:

هماهنگی هارمونیک = (نسبت ملایمت + نسبت ناهم‌خوانی)

نسبت ملایمت: نسبت تعداد هارمونی‌های سازگار به کل تعداد هارمونی‌ها نسبت دیشنانسه: نسبت تعداد هارمونی‌های غیرسازگار به کل تعداد هارمونی‌ها

به عبارت دیگر، هرچه نسبت ملایمت در یک ملودی بیشتر باشد، هماهنگی هارمونیک آن بیشتر است. الگوریتم ۳-۷ نشان دهنده نحویه انجام این کار برای یک فایل MIDI است.[۱]

برای کد های الگوریتم های گفته می‌توانید به پ-۱ مراجعه کنید.

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

تحیل جدول ۳-۳ به صورت زیر است:

هماهنگی ریتم مقدار ۳۴.۰ نشان‌دهنده این است که ریتم ملودی دارای تناوب‌های نسبتاً مشابه است، اما همچنان دارای برخی از تغییرات و نوسانات است.

هماهنگی هارمونیک مقدار ۸۷.۰ نشان‌دهنده این است که هارمونی‌های ملودی در مجموع سازگار هستند و دارای

الگوریتم ۳-۷ هماهنگی هارمونیک

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

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

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

هماهنگی بالایی هستند.

تغییرات کلید مقدار ۲.۰ نشان‌دهنده این است که ملودی تقریبا هیچ تغییراتی در کلید ندارد و در کلید ثابت باقی می‌ماند.

ملودی های ساخته شده توسط مدل دارای هماهنگی بالایی در هارمونی و ریتم است، اما هنوز دارای بعضی از ناهمسانی‌ها و عدم هماهنگی است که می تواند بهبود یابد.

۳-۶ ساخت موسیقی نهایی

در کد ما که در الگوریتم ۳-۱۰ 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 ۳-۸ و الگوریتم ۳-۹ نشان داده شده است. این فایل‌های MIDI سپس با استفاده از FluidSynth، یک نرم‌افزار سینتی‌سایزر 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 سینتی‌سایز 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 می‌کند. این شامل رول پیانو اصلی، لوپ‌های درام که توسط مدل های ما ساخته

الگوریتم ۳–۸ تبدیل توکن به MIDI

```
Input: utils, token, state, channel, end_token_pause
Output: Iterator of (Optional MIDI Message, DecodeState)
if state is None then
    Initialize state
token ← token.strip()
if token is empty or starts with "<" then
    yield (None, state) return
if token is "<end>" then
    Update state with end_token_pause
    if utils.cfg.decode_end_held_note_delay ≠ 0.0 then
        for (channel, note), start_time in state.active_notes do
            Convert delta_accum to ticks
            Remove (channel, note) from active_notes
            yield (note_off message, state)
    yield (None, state) return
if token is a wait token then
    Update state with wait token delta
    if utils.cfg.decode_end_held_note_delay ≠ 0.0 then
        for (channel, note), start_time in state.active_notes do
            if note held too long then
                Convert delta_accum to ticks
                Remove (channel, note) from active_notes
                yield (note_off message, state) return
else
    (bin, note, velocity) ← utils.note_token_to_data(token)
    Convert delta_accum to ticks
    if velocity > 0 then
        if utils.cfg.decode_fix_repeated_notes and (channel, note) in active_notes then
            Remove (channel, note) from active_notes
            yield (note_off message, state)
        Add (channel, note) to active_notes
        yield (note_on message, state)
    else
        Remove (channel, note) from active_notes
        yield (note_off message, state)
yield (None, state)
```

الگوریتم 3-9 متن به MIDI message

```
1: Input: utils, data, channel
2: Output: Iterator of MIDI Messages
3: state ← 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 ← new_state
7:         if msg is not None then
8:             yield msg
```

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 کد ها و اجرا آنها به پیوست پ-۱ مراجعه کنید.

الگوریتم ۳-۱۰ نحویه ساخت موسیقی نهایی

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

فصل چهارم
ساخت خط لوله متن به lo-fi

۴-۱ معماری خط لوله

این خط لوله [1] که در شکل ۳-۱ نمایش داده شده است، 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، از مدل Musicgen [۷] ، برای تولید موسیقی از متن استفاده می کنیم، که یک مدل از پیش آموزش داده شده [2] را برای تولید موسیقی است.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.

---

[1] pipeline
[2] pre-trained

در مرحله دوم، از یک مدل دیگری با نام basic pitch [۱۸] 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 ۳-۲ استفاده می کنیم تا فایل MIDI را به یک متن قابل فهم برای مدل تبدیل کنیم.

در مرحله چهارم، از الگوریتم ۳-۱۰ 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.

۴-۱-۱ 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 مدل‌های تولید موسیقی دیگر، یک سیستم پیچیده است که می‌تواند موسیقی با کیفیت بالا تولید کند، اما همچنین دارای محدودیت‌هایی است. در اینجا چند دلیل وجود دارد که چرا استفاده مستقیم از آن برای تولید موسیقی ممکن است ایده‌آل نباشد:

۱. **کیفیت پایین و نویز**: همانطور که اشاره کردید، خروجی مدل Musicgen می‌تواند 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 می‌گیرد و می‌تواند 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 می‌کند که یک فرمت باینری است. این باعث می‌شود 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 چالش‌برانگیز است.
۳. **انعطاف‌پذیری محدود در سبک و ژانر موسیقی**: مدل Musicgen بر روی یک مجموعه داده خاص آموزش دیده است، به این معنی که درک محدودی از سبک‌ها و ژانرهای 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**

۴-۲ 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 حرفه ای 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 ۴-۱ 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	همه نظر دهندگان به عدم تطابق بین انتظارات و خروجی اشاره کردند.

چندین نظر دهنده 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 صداهای 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 jacbz/Lofi [۶] نیز به دست آمده است.[۱]

[۱] نتیجه این کار در اینجا قابل مشاهده است.

فصل پنجم
نتیجه‌گیری و پیشنهادها

۵-۱ نتیجه‌گیری

در این پروژه، ما یک روش جدید برای تولید موسیقی لو-فای با استفاده از یک مدل زبان کوچک ارائه کرده‌ایم. مدل ما که بر روی چمد مجموعه داده مختلف و ملودی‌های موسیقی لو-فای آموزش دیده است، نتایج امیدوارکننده‌ای در ایجاد آهنگ‌های منحصر به فرد و جذاب با ملودی‌های دلپذیر نشان داده است. معیارهای ارزیابی، از جمله نمره نوآوری و نمره شباهت ملودی، نشان می‌دهند که مدل ما می‌تواند به طور مؤثر آهنگ‌های جدید لو-فای تولید کند که از آهنگ‌های موجود متمایز هستند و در عین حال ساختار موسیقایی منسجمی را حفظ می‌کنند.

با این حال، ما همچنین مشاهده کردیم که خط لوله ما با چالش‌های قابل توجهی در به‌دقت گرفتن جوهره موسیقی لو-فای از طریق توصیفات متنی مواجه است. روش فعلی به شدت به کیفیت و انسجام داده‌های متنی متکی است که می‌تواند منجر به عدم تطابق بین موسیقی تولید شده و زیبایی‌شناسی مورد نظر شود. این موضوع دشواری‌های ترجمه ویژگی‌های پیچیده صوتی به نمایه متنی را برجسته می‌کند، که یک چالش رایج در وظایف تولید موسیقی است.

با وجود این چالش‌ها، نتایج ما پتانسیل استفاده از مدل‌های زبان کوچک برای تولید موسیقی لو-فای را نشان می‌دهد. کارهای آینده باید بر بهبود خط لوله متن به موسیقی تمرکز کنند، احتمالاً از طریق استفاده از تکنیک‌های پیشرفته‌تر پردازش زبان طبیعی یا ادغام ویژگی‌های صوتی در توصیف متنی. علاوه بر این، ما پیشنهاد می‌کنیم که بهبودهای زیر را بررسی کنید:

۵-۲ پیشنهادها

یکی از کار هایی که می‌توان انجام داد پردازش جامع سازها است. روش فعلی ما هر ساز را به صورت جداگانه پردازش می‌کند که می‌تواند منجر به یک ترکیب غیرطبیعی شود. کارهای آینده می‌توانند شامل پردازش همه سازها به صورت یکجا باشند، که به مدل اجازه می‌دهد روابط بین سازها را یاد بگیرد و موسیقی واقعی‌تر و منسجم‌تری تولید کند. این می‌تواند با استفاده از تکنیک‌های پردازش چند ساز یا استفاده از یک معماری پیشرفته‌تر که می‌تواند تعاملات بین سازها را به تصویر بکشد، محقق شود.

با پرداختن به این محدودیت‌ها، می‌توانیم پتانسیل کامل مدل‌های زبان کوچک را برای تولید موسیقی لو-فای با کیفیت بالا که انتظارات علاقه‌مندان به موسیقی را برآورده می‌کند، باز کنیم.

منابع و مآخذ

[1] B. PENG, “RWKV-LM,” Aug. 2021.

[2] 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. Kocon, 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.

[3] A. Sherstinsky, “Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network,” 2018.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.

[5] H. M. de Oliveira and R. de Oliveira, “Understanding midi: A painless tutorial on midi format,” *arXiv preprint arXiv:1705.05322*, 2017.

[6] J. Zhang, “Lofi: Ml-supported lo-fi music generator,”

[7] 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.

[8] 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.

[9] L. Callender, C. Hawthorne, and J. Engel, “Improving perceptual quality of drum transcription with the expanded groove midi dataset,” 2020.

[10] A. Moi and N. Patry, “HuggingFace’s Tokenizers,” Apr. 2023.

[11] 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.

[12] 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.

[13] W. Falcon and The PyTorch Lightning team, “PyTorch Lightning,” Mar. 2019.

[14] L. Biewald, “Experiment tracking with weights and biases,” 2020. Software available from wandb.com.

[15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[16] Z. Xiong, W. Wang, J. Yu, Y. Lin, and Z. Wang, "A comprehensive survey for evaluation methodologies of ai-generated music," *arXiv preprint arXiv:2308.13736*, 2023.

[17] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data.," in *ISMIR* (J. S. Downie and R. C. Veltkamp, eds.), pp.637–642, International Society for Music Information Retrieval, 2010.

[18] R. M. Bittner, J. J. Bosch, D. Rubinstein, G. Meseguer-Brocal, and S. Ewert, "A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, (Singapore), 2022.

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 نوت بوک زیر استفاده کنید. لینک نوت بوک در گوگل Colab

Abstract

The rise of content creation has led to an unprecedented demand for high-quality, copyright-free music for use in multimedia content. Lo-fi music, with its calming and soothing properties, has become a staple in video, podcast, and live stream production. However, obtaining high-quality, copyright-free lo-fi music can be a challenging and costly process. This paper presents a novel approach to generating lo-fi music using a small language model, addressing the need for a cost-effective and scalable solution for content creators. We utilize the RWKV architecture, a state-of-the-art model that combines the efficiency of a transformer with the flexibility of a recurrent neural network. In this paper, we focus on the technical aspects of generating lo-fi music, exploring the challenges of training a model to produce high-quality music of unlimited length. We trained two separate models, one for piano and one for drum instruments, to generate lo-fi music on demand. Our approach enables content creators to focus on their creative vision, rather than spending time and resources searching for suitable music. By automating the process of music generation, we aim to reduce the importance of music selection in content creation, freeing creators to focus on their core competencies. Our work has far-reaching implications for the music industry, enabling the efficient creation of high-quality content and paving the way for new forms of creative expression.

Keywords: 1- generate lo-fi music 2- music generating ai 3- generate ai

University of Isfahan

Faculty of Computer Engineering

BS Thesis

Train a small language model for generating lo-fi music

Supervisor:

Dr. Zahra Zojaji

By:

Soheil Salimi

August 2024