
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

چکیده

در این پروژه، ما قصد داریم از معماری RWKV (Reinventing RNNs for the Transformer Era) [؟] استفاده کنیم تا یک مدل زبانی کوچک برای تولید آهنگ lo-fi بسازیم. موسیقی lo-fi که با صدای گرم و نوستالژیک و نقص‌های عمدی خود شناخته می‌شود، چالشی منحصر به فرد برای مدل‌های تولیدی ایجاد می‌کند. رویکرد ما شامل آموزش مدل RWKV بر روی مجموعه داده‌های متنوعی از نمونه‌های موسیقی lo-fi است که بر روی ویژگی‌های متمایز این ژانر مانند تمپوی آرام، نویز محیطی و سادگی ملودیک تمرکز دارد. یکی از مزایای کلیدی استفاده از معماری RWKV توانایی آن در مدیریت طول نامحدود زمینه است که به مدل اجازه می‌دهد تا انسجام را در طول توالی‌های موسیقی طولانی حفظ کند. این قابلیت برای تولید آهنگ‌های lo-fi که جریان طبیعی و پیوستگی دارند و تجربه شنیداری کلی را بهبود می‌بخشند، بسیار مهم است. هدف اصلی ما ایجاد مدلی است که بتواند آهنگ‌های lo-fi تولید کند که نه تنها از نظر فنی دقیق باشند بلکه از نظر هنری نیز جذاب باشند. ما بر اهمیت تعادل بین دقت فنی و بیان خلاقانه تأکید می‌کنیم تا اطمینان حاصل کنیم که موسیقی تولید شده از نظر احساسی با شنوندگان ارتباط برقرار می‌کند. نتایج ما نشان می‌دهد که مدل مبتنی بر RWKV می‌تواند با موفقیت موسیقی lo-fi تولید کند که هم معیارهای فنی و هم هنری را برآورده می‌کند. آهنگ‌های تولید شده دارای گرما و جذابیت خاص موسیقی lo-fi هستند و در عین حال سطح بالایی از انسجام و ساختار موسیقی را حفظ می‌کنند. این کار به حوزه در حال رشد موسیقی تولید شده توسط هوش مصنوعی کمک می‌کند و امکانات جدیدی برای استفاده از مدل‌های زبانی کوچک در تولید موسیقی باز می‌کند.

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

فهرست مطالب

عنوان **صفحه**

عنوان صفحه

عنوان صفحه

فهرست تصاویر

عنوان **صفحه**

فهرست جداول

عنوان **صفحه**

فصل اول
مقدمه

۱-۱ پیش‌گفتار

موسیقی لو-فای که با صداهای آرام و نوستالژیک 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. 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 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

[1] Artificial intelligence
[2] Machine learning

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 بیش‌برازش و 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 RWKV

معماری RWKV [؟] به گونه‌ای طراحی شده است که از مزایای هر دو معماری RNN [1] [؟] و ترانسفورمر [؟] [2] 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.

---

[1] Recurrent Neural Networks
[2] 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 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 ۲-۲ 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.

MIDI file format ۲-۱-۲

فرمت MIDI [4] [؟] 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 هستند۱۲.

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.

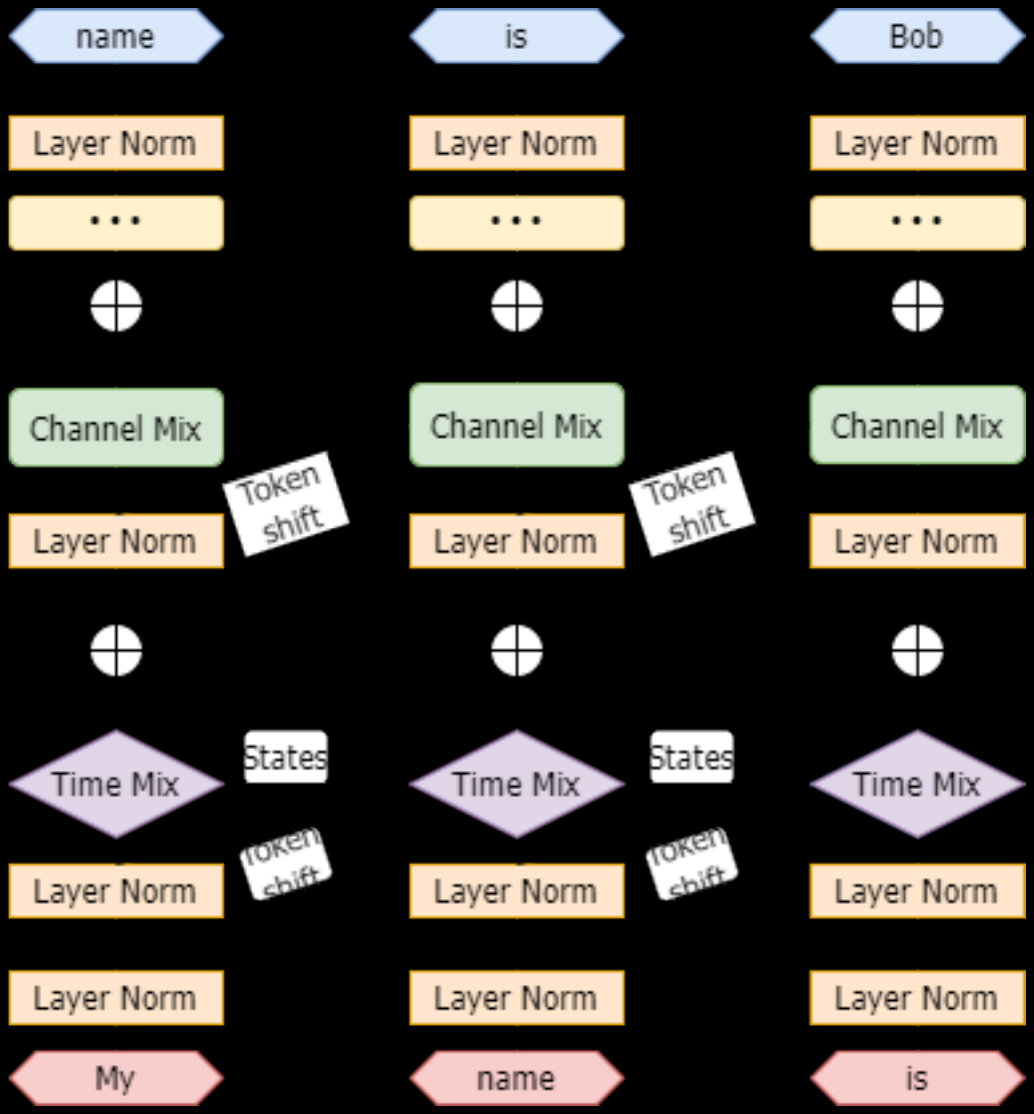
---

[1] (LayerNorm)

[2] (logits)

[3] (cross-entropy loss)

[4] Musical Instrument Digital Interface



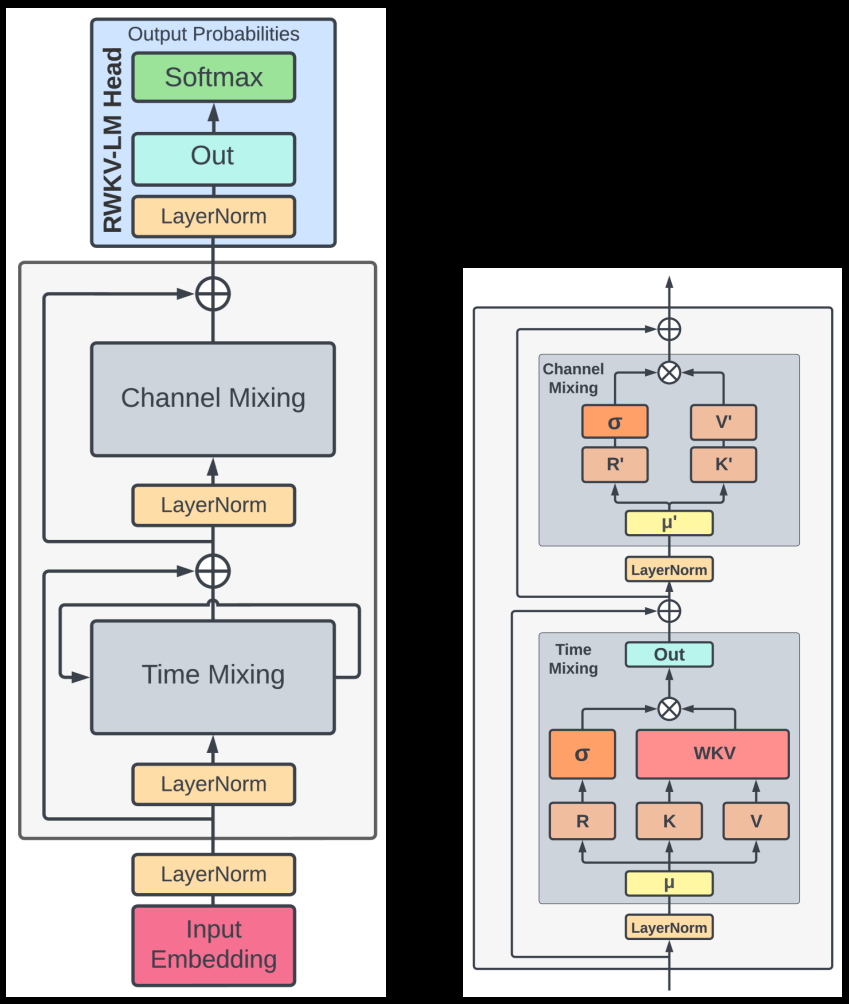
شکل ۲-۱ - معماری RWKV برای مدل های زبان

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

[1] Status Messages
[2] Data Messages



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

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

۳. **زمان‌بندی** [1]: 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 [2] 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

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

[1] Timing
[2] Note On

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.

۲-۲ روشهای پیشین

۲-۲-۱ استفاده از معماری VAE

پروژه jacbz/Lofi [؟] 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 [2] مزایای متعددی نسبت به VAE [3] 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.

---

[1] Track
[2] Lo-Fi
[3] Variational Autoencoder

۲-۲-۱-۱ 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.

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 [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 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

[1] Long Short-Term Memory

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

فصل سوم
آموزش مدل و معماری

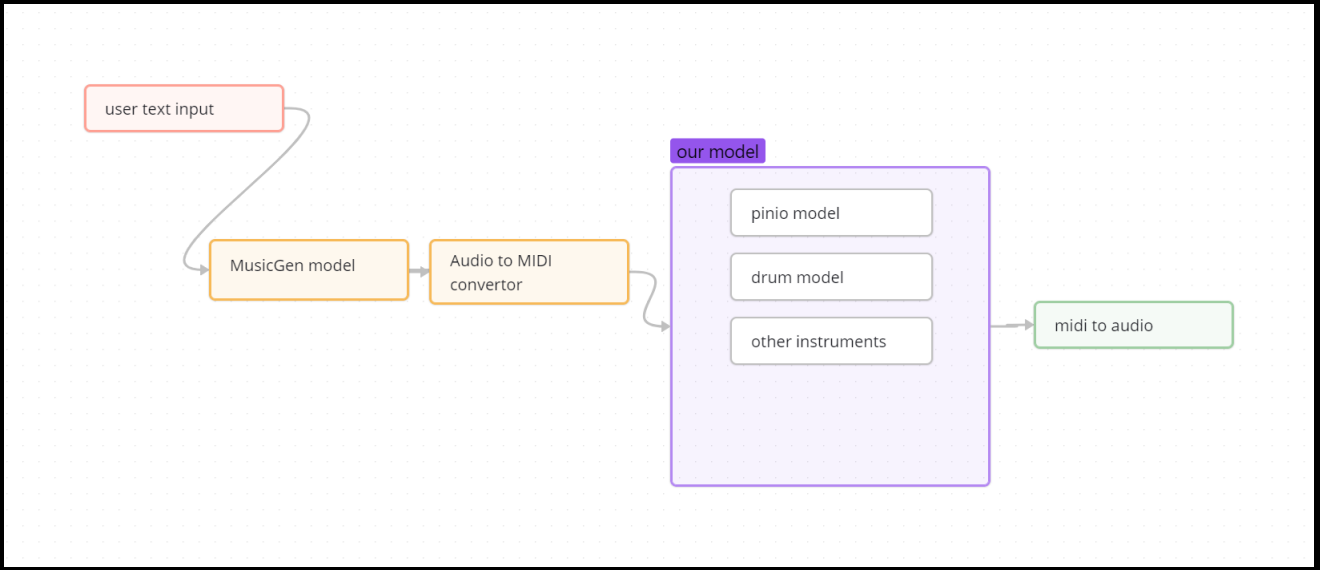
۳-۱ معماری کلی پروژه

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) [؟] 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

۳-۲-۱ مدل پیانو

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

۳-۲-۲ مدل دارم

مجموعه داده گسترده EGMD [؟]

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

۳-۴ 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.

- **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.

مثال ۳-۱. برای مثال `<start> 26:2 t8 26:0 26:2 t5 26:0 <end>` می تواند 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 [؟] استفاده کردیم. در اینجا توضیح دقیقی از این فرآیند آورده شده است:

۳-۴-۱-۱ 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.

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

الگوریتم ۳-۱ توکن کردن فایل های MIDI

```
function mix_volume(velocity, volume, expression)
    return velocity × (volume / 127.0) × (expression / 127.0)
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
```

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 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 [؟] 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

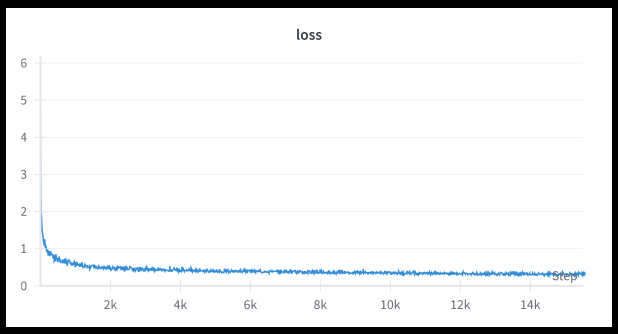
۳-۵-۱ پارامترهای آموزش مدل

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

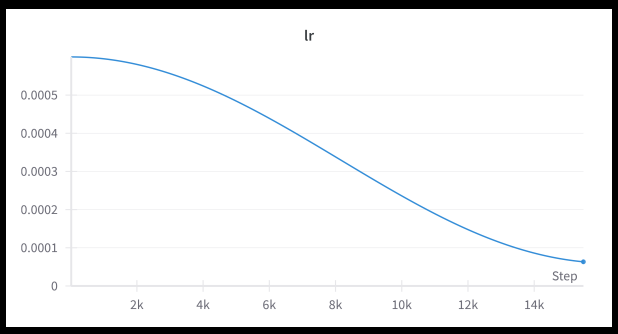
ما از معماری `rwkv-0.6` [؟] استفاده کردیم. همچنین امکان استفاده از مدل دارد. مدل ما شامل ۲۰ لایه و embedding برابر با ۵۱۲ است. Context Length [۱] مدل برابر با ۵۱۲ است. مقدار نرخ یادگیری اولیه و نهایی برابر با 6×10^{-4} و 6×10^{-5} است.

۳-۵-۱-۱ پیشرفت آموزش مدل

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

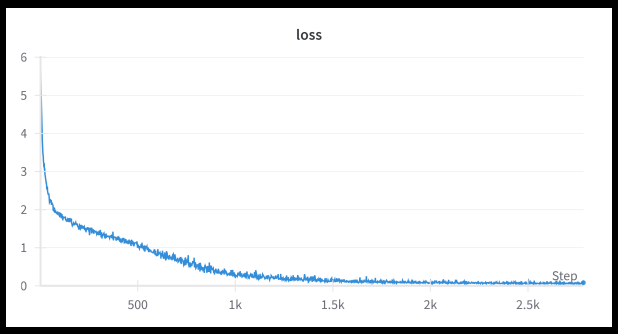


(الف) تغییر مقدار تابع خطا

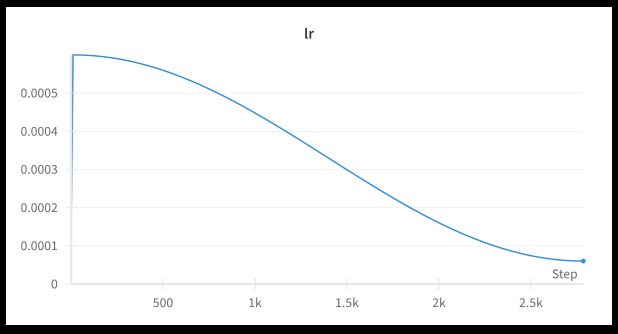


(ب) تغییر نرخ یادگیری

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