
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**

# فصل اول
# آموزش مدل و معماری

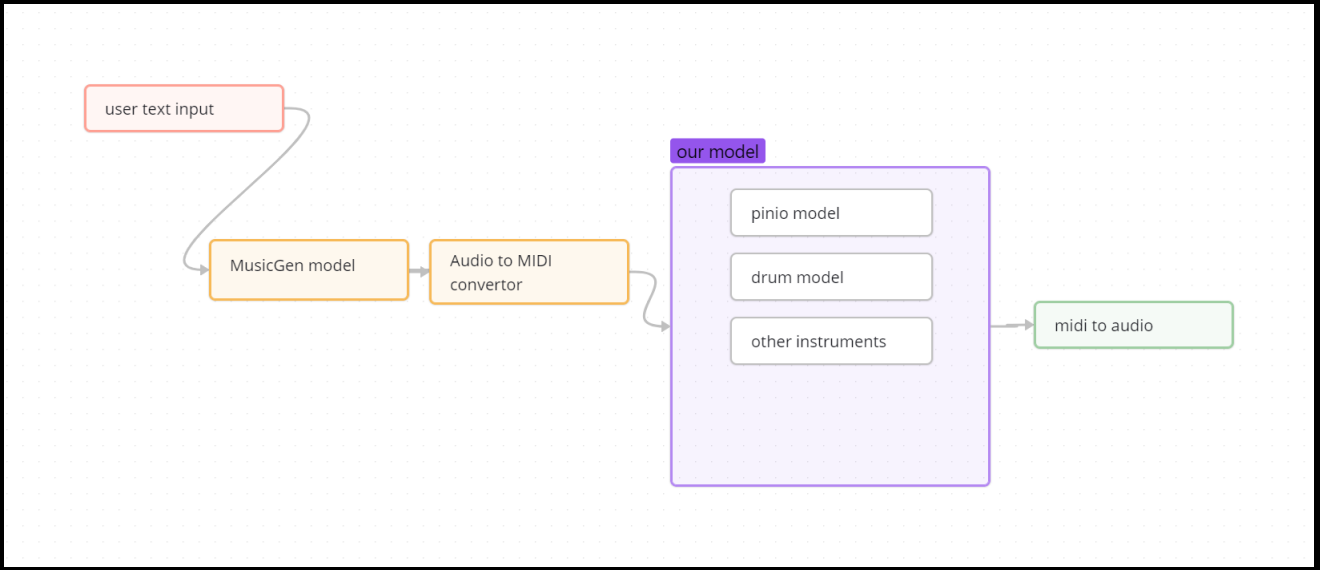
## ۱-۱ معماری کلی پروژه

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 ۱-۱ - ساختار pipeline**

علاوه بر آموزش مدل‌ها، ما یک pipeline توسعه داده‌ایم که تجربه کاربری را بهبود می‌بخشد. این خط لوله همانطور که در ۱-۱ 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 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.

## ۱-۲ دیتاسیت ها

### ۱-۲-۱ 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 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.

### ۱-۲-۲ 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 ۱-۱ - 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 ۱-۱ نشان داده شده است.

## ۱-۳ تبدیل MIDI به متن

## ۱-۴ 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 ۱-۱.** برای مثال `<start> 26:2 26:0 t8 26:2 26:0 t5 <end>` می تواند خروجی الگوریتم ۱-۱ باشد.

### ۱-۴-۱ 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:

#### ۱-۴-۱-۱ 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.

#### ۱-۴-۱-۲ 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 ۱-۱** 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.

#### ۱-۴-۱-۳ 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 [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.

### ۱-۵ آموزش مدل

#### ۱-۵-۱ 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 $6 \times 10^{-4}$ و $6 \times 10^{-5}$ است.

#### ۱-۵-۲ نحویه آموزش مدل

الگوریتم ۱-۲ برای مدیریت و بهینه‌سازی فرآیند آموزش مدل با استفاده از PyTorch Lightning [؟] 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 می دهد.

**الگوریتم ۱-۲** 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')
```

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) [؟] 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.

### ۱-۵-۲-۱ کتابخانه‌ها استفاده شده

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

[1] warm 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.

## ۱-۶ ارزیابی عملکرد مدل

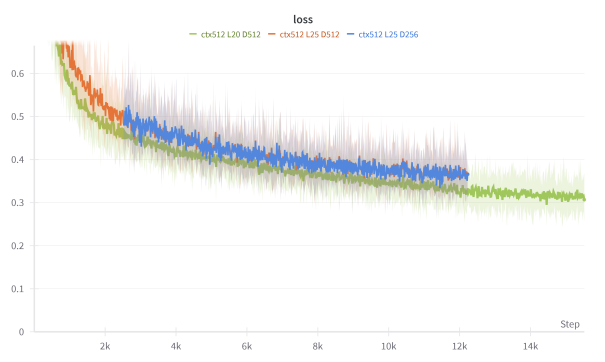
### ۱-۶-۱ ارزیابی مدل پیانو

با توجه به شکل ۱-۲ 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.

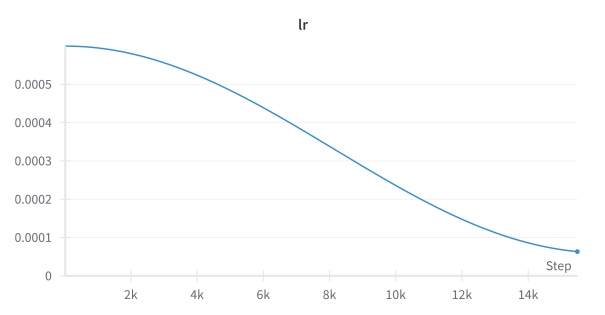
### ۱-۶-۲ ارزیابی مدل درام

مطابق با نمودار ۱-۳ 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 داده های دارم خیلی بالا نیست و ادامه بیش از این احتمالا باعث overfit می شود.

[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.

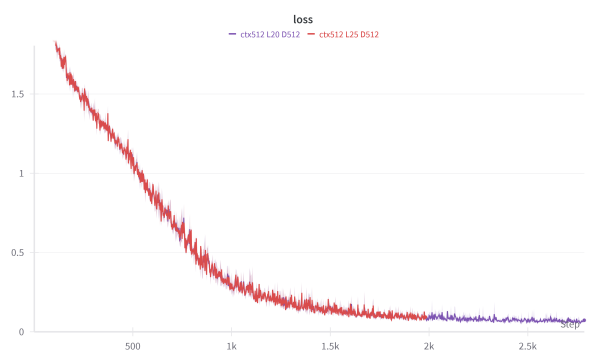



(الف) تغییر مقدار تابع خطا

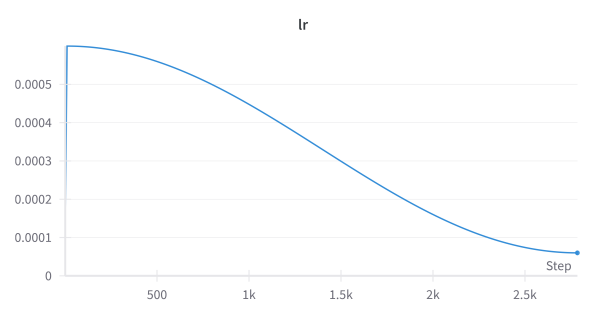



(ب) تغییر نرخ یادگیری

**شکل ۱-۲ - نمودار های پیشرفت یادگیری مدل پیانو**




(الف) تغییر مقدار تابع خطا




(ب) تغییر نرخ یادگیری

**شکل ۱-۳ - نمودار های پیشرفت یادگیری مدل درام**

### ۱-۶-۳ 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 می‌کشند. اکثر این میعار ها در مقاله A Comprehensive Survey for Evaluation Methodologies of AI-Generated Music [؟] معرفی شده اند. و ما آن ها را با کتابخانه music21 [؟] پیاده سازی کردیم.

#### ۱-۶-۳-۱ 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
- $\mu$ میانگین طول نت ها است

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

**الگوریتم ۱-۳** 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

[۱] کد پاینون این الگوریم 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)

### ۱-۶-۳-۲ 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 ۱-۴ نشان دهنده نحویه انجام این کار برای یک فایل MIDI است. [۱]

**الگوریتم ۱-۴** 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

### ۱-۶-۳-۳ 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 ۱-۵ نشان دهنده نحویه انجام این کار برای یک فایل MIDI است.[۱]

**الگوریتم ۱-۵** 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

### ۱-۶-۳-۴ هماهنگی هارمونیک (Harmonic Coherence Metric)

هماهنگی هارمونیک یک متد برای اندازه گیری هماهنگی هارمونیک یک ملودی است. این متد از نسبت هارمونی سازگار (کنسانه) و غیرسازگار (دیسونانسه) در یک ملودی برای محاسبه هماهنگی هارمونیک استفاده می‌کند.

فرمول:

هماهنگی هارمونیک = (نسبت کنسانه + نسبت دیشنانسه)

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

به عبارت دیگر، هرچه نسبت کنسانه در یک ملودی بیشتر باشد، هماهنگی هارمونیک آن بیشتر است. الگوریتم ۱-۶ نشان دهنده نحویه انجام این کار برای یک فایل MIDI است.[۱]

**الگوریتم ۱-۶** 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 ۱-۲ - نتایج این معیار ها برای مدل پیانو**

| متد | مقدار |
|---|---|
| هماهنگی ریتم | ۳۴.۰ |
| هماهنگی هارمونیک (کنسانه) | ۸۷.۰ |
| هماهنگی هارمونیک (دیسونانسه) | ۱۳.۰ |
| تغییرات کلید | ۲.۰ |

### ۱-۶-۳-۵ نتایج این معیار ها برای مدل پیانو

تحیل جدول ؟؟ 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 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.

# منابع و مآخذ


[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. 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.

[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