An odd Music Generator

داستان پروژه:

امیرحسین که به تازگی به نواختن موسیقی علاقه‌مند شده، تصمیم به ثبت‌نام در کلاس پیانو میگیرد. او پس از تلاش فراوان و در کردن اسم و رسمی، از طرف ارکستر سمفونیک تهران دعوت به تک‌نوازی قطعه های *یوهان سباستیان باخ* در اجرای تابستان این گروه می‌شود. اما از آنجا که او شدیدا درگیر پایان‌نامه ارشد است، از شما کمک خواسته تا با استفاده از دانش یادگیری عمیق و شبکه‌های عصبی که در طول ترم بدست آورده‌اید، سیستمی هوشمند طراحی کنید تا به‌جای او در ارکستر قطعه‌های پیانو را *باخانه* (همچون یوهان سباستیان باخ) بنوازد.

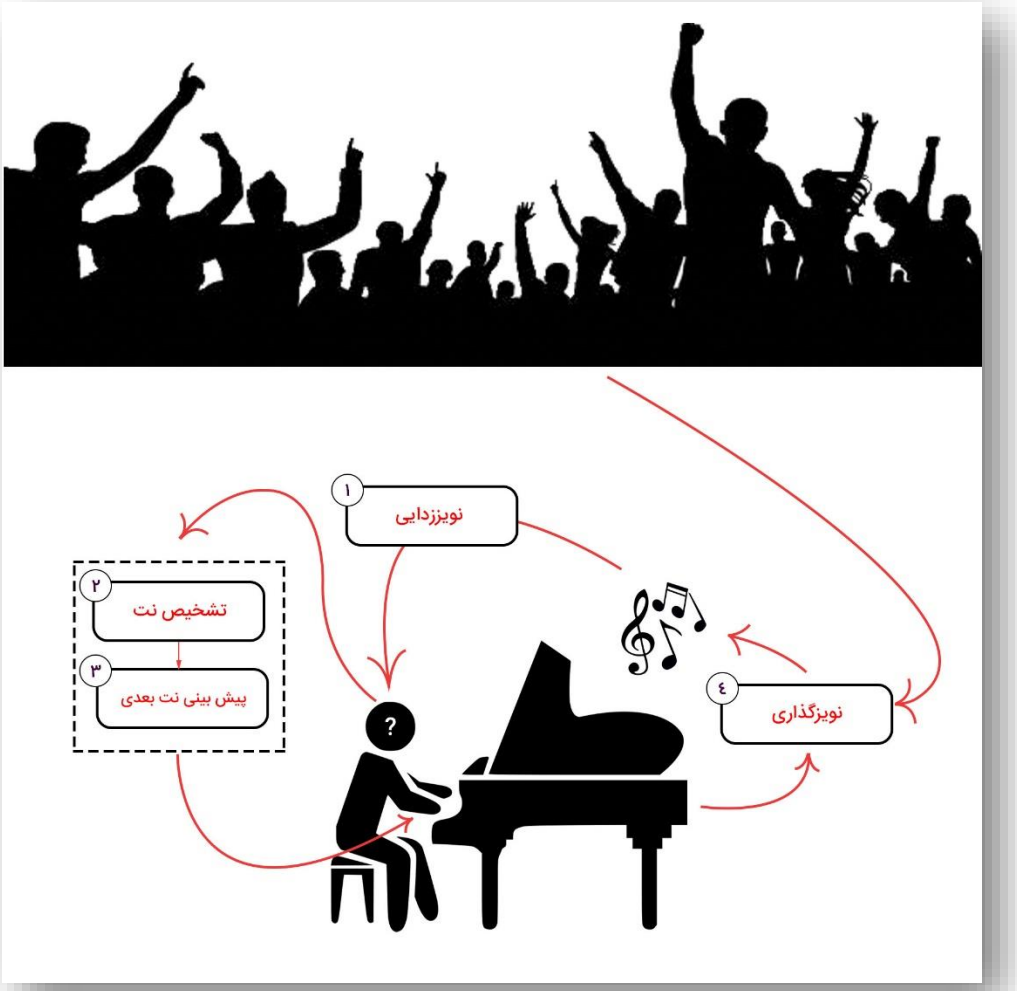
هدف پروژه:

هدف از این پروژه، آشنایی دانشجو با خودرمزگذاران نویززدا (Denoising Autoencoders)، تکنیک‌های داده‌افزایی (Data Augmentation)، کار با فایل های صوتی، مدل‌های رشته-به-رشته (Seq2seq)، و سیستم‌های هوشمند می باشد.

شرح پروژه:

این پروژه شامل چرخه‌ای متشکل از ٤ بخش اصلی است که شما با تکمیل کردن هرکدام از آن‌ها می‌توانید سرآخر با اتصال درست آن‌ها به یکدیگر، سیستمی هوشمند طراحی کرده باشید که توانایی نواختن موسیقی دارد. به صورت کلی،

این ۴ بخش عبارتند از: (۱) نویززدایی ، (۲) تشخیص نت، (۳) پیش‌بینی نت بعدی، و (۴) نویزگذاری. منطق کلی این چرخه به شکل روبروست:



۱. **نویززدایی:** در طول اجرا، نوازنده باید تمام حواسش به صدای موسیقی بوده و نباید اجازه دهد تا سروصدا و آوای تشویق جمع (!) تمرکز او را ازبین ببرد.

۲. **تشخیص نت:** در حین اجرا، نوازنده باید در ذهن خود سلسله نت‌هایی را که اجرا کرده را به خاطر سپرده تا از بداند در اجرای یک قطعه تا به هر لحظه چه نت‌هایی را زده است.

۳. **پیش‌بینی نت بعدی:** در حین اجرا، نوازنده باید در ذهن خود با توجه به سلسله نت‌هایی که تا به لحظه اجرا کرده، نت بعدی را پیش‌بینی کند تا آن‌را به صدا درآورد.

۴. **نویزگذاری:** طبیعتا از آنجایی که جمعیت حاضر در سالن از فرط لذت در پوست خود نمی‌گنجند، در حین اجرای موسیقی کلاسیک با دست و سوت سیستم تولید موسیقی شما را تشویق می‌کنند.

چرخه سیستم شما هم شامل این ۴ بخش اصلی بوده که به نوعی قصد دارد در هر مرحله اجرای زنده را شبیه‌سازی کند. بدیهی است که سیستم هوشمند شما به خودی خود توانایی نویززدایی، تشخیص نت، و پیش‌بینی نت بعدی را نداشته و این وظیفه شما است که شبکه‌های عصبی مختلفی را با این اهداف آموزش دهید. بدین منظور، شما نیاز به دیتاهای متفاوتی دارید که تمامی آن‌ها در یک فایل زیپ از آدرس (shorturl.at/oxBQU) قابل دریافت است. این فایل زیپ شامل سه فولدر به نام های artifacts، piano_pieces، و piano_triads می‌باشد.

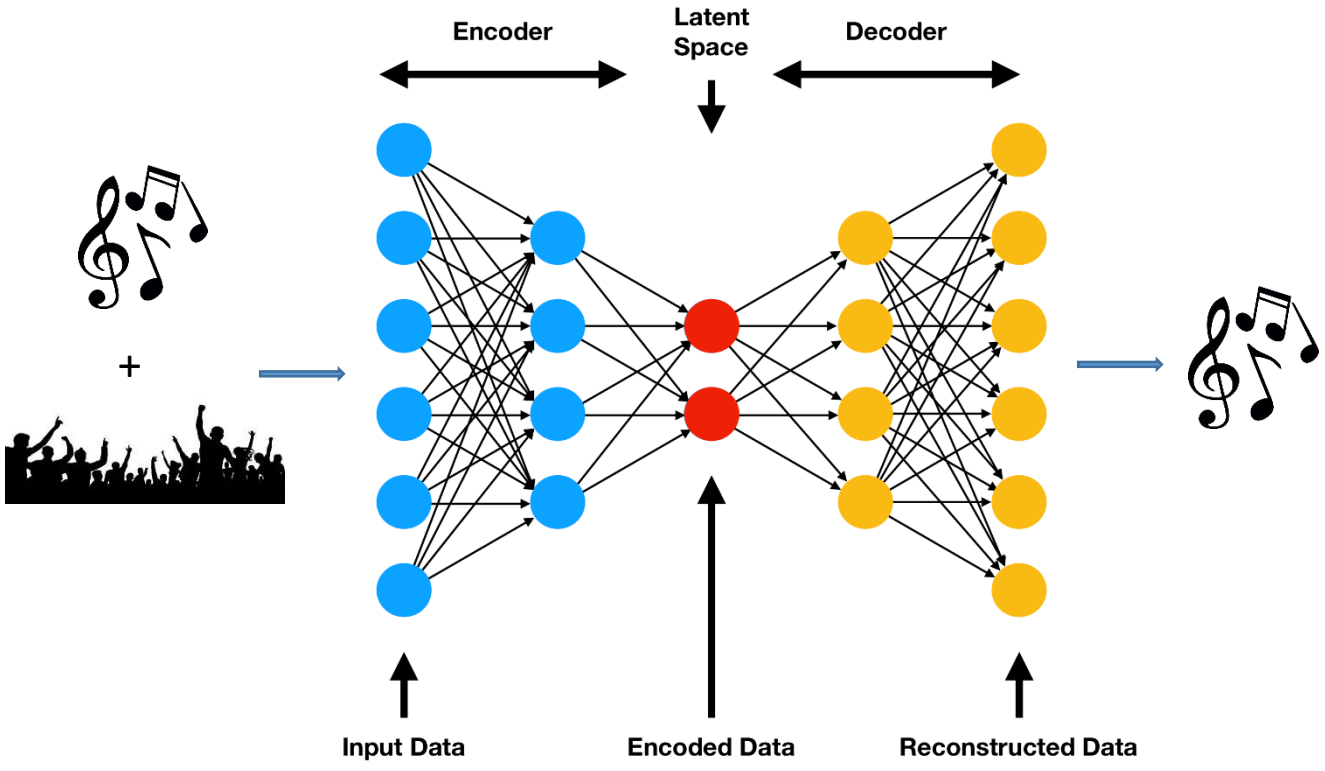
- فولدر artifacts شامل چندین فایل صوتی نویز است که در مرحله نویزگذاری استفاده می‌شود.
- فولدر piano_triads شامل چندین فایل صوتی ۳ ثانیه‌ای از نت‌های (برای درک راحتتر، آن‌ها را نت صدا می‌کنیم) پیانو بوده که اسم هر فایل صوتی نشان‌دهنده اسم نت‌ای است که فایل صوتی آن را به صدا درمیاورد.
- فولدر piano_pieces شامل ۴۰۰ فایل متنی آهنگ های پیانو کلاسیک است. در هرکدام از این فایل های متنی، نت‌های هر آهنگ با ' / ' از هم جدا شده است.

```
notes_list = select_random_notes(n=3)
initial_piece = append_notes(notes_list)
noisy_piece = add_random_noise(initial_piece)
final_piece = None
piece_limit, counter = 30, 0
while True:
    denoised_piece = DAE(noisy_piece)
    sub_notes = break_down_piece(denoised_piece)
    sub_notes_classes = [NID(note) for note in sub_notes]
    sub_notes_vec = note2vec(sub_notes_classes)
    next_note = S2S(sub_notes_vec)
    notes_list = sub_notes_classes + [next_note]
    new_piece = append_notes(notes_list)
    if next_note == '<END>' or counter == piece_limit:
        final_piece = new_piece
        break
    else:
        noisy_piece = add_random_noise(new_piece)
        counter += 1
```

تیکه کد بالا، نمونه‌ای از این چرخه را نشان می‌دهد که سیستم شما بایستی به منظور تولید یک قطعه موسیقی آن را طی کند. در ابتدا، برنامه شما بایستی به کمک تابعی، اسامی n نت تصادفی را از بین تمامی نت‌های موسیقی که در فولدر piano_triads موجود است، به عنوان n نت آغازین قطعه موسیقی درنظر بگیرد (خط ۱). برای مثال برنامه شما سه نت تصادفی A_dim_2_0، B_min_3_1، و E_dim_5_1 را انتخاب می‌کند. سپس، اسامی این نت ها به تابع append_notes فرستاده می‌شود و به عنوان خروجی، یک فایل صوتی به عنوان قطعه آغازین (initial_piece) برگردانده می‌شود. بدیهی بوده که طول این فایل صوتی که شامل n نت آغازین ۳ ثانیه‌ای بوده، ۳ * n ثانیه است که در طی ۳ ثانیه اول نت A_dim_2_0، در طی ۳ ثانیه دوم نت B_min_3_1، و در طی ۳ ثانیه سوم نت E_dim_5_1 به صدا درمیاید. حال، به کمک تابع add_random_noise، فایل صوتی آغازین را نویزی میکنیم. شما در این تابع،

یکی از نویزهای موجود در فولدر artifacts را بر روی فایل صوتی که به عنوان ورودی به تابع داده می‌شود میاندازید. خروجی این تابع در اصل همان قطعه آغازین بوده که حال نویزی است. تا به اینجای کار شما مراحل اولیه‌ای که نیاز بوده تا قبل از شروع چرخه انجام شود را پیاده‌سازی کرده‌اید. حال با وارد شدن به حلقه در خط ۵، به توشیح هریک از ۴ بخش اصلی می‌پردازیم. لازم به ذکر است که برای قطعه‌های تولیدی، محدودیت ماکسیمم ۳۰ نت را درنظر می‌گیریم.

همانطور که در خط ششم ذکر شده، از یک مدل Denoising Autoencoder (DAE) به منظور نویززدایی استفاده می‌کنیم:



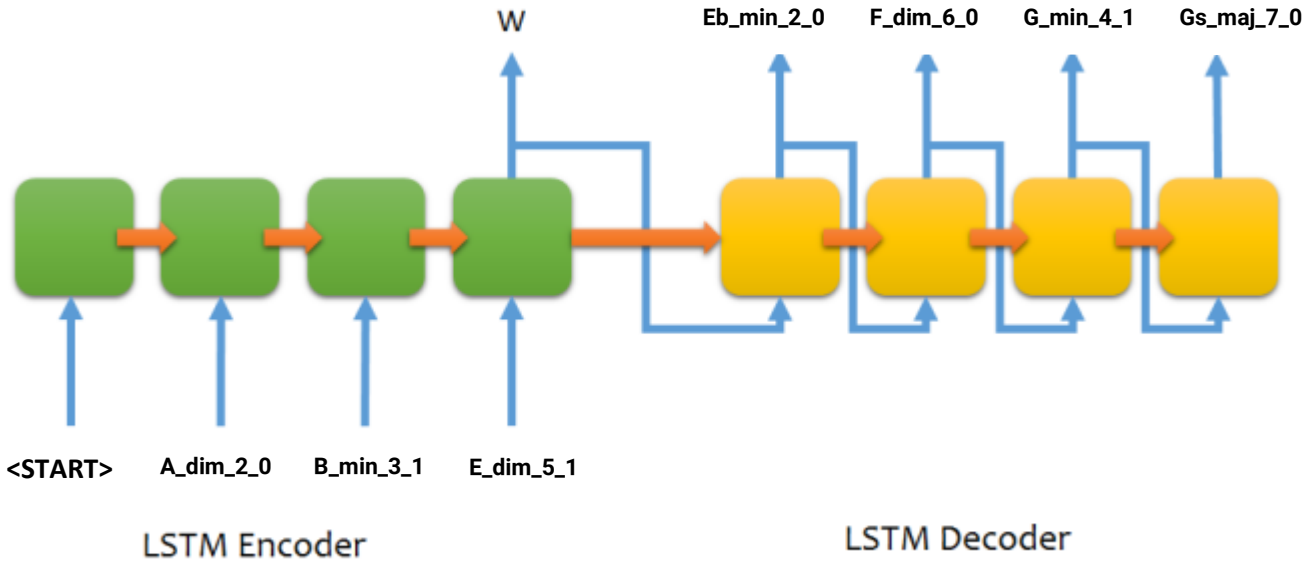
این مدل، بایستی چنان آموزش داده‌شود تا با دریافت فایل صوتی همراه نویز (noisy_piece)، فایل صوتی عاری از نویز (denoised_piece) را تولید کند.

پس از اتمام مرحله نویززدایی، نوبت به تشخیص نت میرسد. در این مرحله قصد داریم لیست نت هایی که تا به این مرحله در denoised_piece نواخته شده را تشخیص دهیم. بدیهی است که می‌توان لیستی از اسامی نت ها در حافظه ذخیره کرد، ولی هدف ما در این بخش استفاده از مدلی جدید بدین منظور است. لذا، همانطور که در خط هفتم قابل مشاهده است، از تابعی با نام break_down_piece استفاده می‌کنیم تا فایل صوتی denoised_piece را به تکه‌های ۳ ثانیه‌ای تشکیل دهنده‌اش تقسیم کنیم. حال، به کمک مدل Note Identification که بایستی قبل از ورود به چرخه آموزش دیده باشد، اسم نت‌ای که هر کدام از این فایل‌های صوتی ۳ ثانیه‌ای به صدا درمیاوردند را تشخیص می‌دهیم

و اسامی نت‌ها را در sub_notes_classes ذخیره می‌کنیم (همانطور که احتمالا متوجه شده‌اید، محتویات این لیست در اولین پیمایش حلقه برابر با محتویات لیست notes_list است). مدل Note Identification صرفا یک طبقه‌بند (Classifier) است که می‌تواند با دریافت یک فایل صوتی ۳ ثانیه‌ای، اسم نت‌ای که در این فایل به صدا درآمده را تشخیص دهد (تشخیص نت).

تا به اینجای کار، سیستم هوشمند شما می‌تواند درمیان شلوغی، صدای نت ها را تمییز دهد و نوع آن‌ها را شناسایی کند. حال، بایستی با توجه به توالی نت‌هایی که تا به لحظه نواخته، نت بعدی که می‌خواهد *باخانه* بنوازد را پیش‌بینی کند. بدین منظور، شما بایستی به کمک داده‌هایی که در اختیارتان قرار داده شده، یک مدل Sequence-to-sequence (S2S) را آموزش داده تا بتواند احتمال شرطی زیر را در ازای هر نت ممکن محاسبه کند:

$$\underset{w_{t+1},\ldots,w_{t+T}}{\operatorname{argmax}} \prod_{j=1}^{T} P(w_{t+j}|w_1,\ldots,w_{t+j-1})$$



در رابطه بالا، T برابر است با تعداد تمام نت‌های موجود و توالی $w_1, \ldots, w_{t+j-1}$ به عنوان یک توالی نت شناخته می‌شود. طبق رابطه بالا، این وظیفه شبکه S2S شما بوده تا بتواند احتمال شرطی نت بعدی را با توجه به توالی نت داده شده حساب کند. نتیجتا، خروجی خط دهم (next_note)، برابر با آن نت‌ای بوده که احتمال شرطی محاسبه شده آن ماکسیمم باشد. نکته قابل توجه این است که (برای مثال) توالی [A_dim_2_0, B_min_3_1, E_dim_5_1] نمی-تواند مستقیما به عنوان ورودی به شبکه عصبی داده‌شود و ما نیاز داریم تا این توالی را به صورت عددی که برای شبکه قابل فهم باشد درآوریم. لذا، قبل از اعمال S2S، در خط نهم تابع note2vec را فراخوانی می‌کنیم.

پس از اتمام مرحله پیش‌بینی نت بعدی، نام نت جدید را به لیست نت ها اضافه می‌کنیم و فایل صوتی جدید را (new_piece)، که حال ۳ ثانیه طولانی‌تر از فایل اولیه است را به کمک تابع append_notes تشکیل می‌دهیم. منتها، از آنجایی که شبکه‌های S2S نیازمند دو نشانگر آغاز (<START>) و پایان (<END>) هستند، شبکه می‌تواند به صلاح

خودش یادگرفته تا در زمان مناسب و زودتر از محدودیت ۳۰ نت، قطعه را به پایان برساند. بدین منظور پس از پیش‌بینی next_note، چک می‌کنیم که ۱) آیا محدودیت ۳۰ نت به پایان رسیده و ۲) آیا این متغیر حاوی نشانگر پایان است یا خیر. در صورت درست بودن هرکدام از این دو شرط، متغیر new_piece حاوی فایل صوتی نهایی سیستم است. در غیر این صورت، مرحله نویزگذاری اعمال شده که در طی آن باری دیگر یک نویز تصادفی بر قطعه جدید (new_piece) اضافه می‌شود.

پس از آموزش مدل‌های ذکر شده و پیاده سازی چرخه، سیستم شما بایستی در مرحله تست (inference)، با لود کردن مدل‌های اموزش دیده، با هربار اجرا، قطعه‌ای جدید تولید کند که طبیعتا طول آن ۳۰ نت یا کمتر است.

نکات تکمیلی:

- قطعه تولید شده نهایی **باید** قابلیت پخش شدن را داشته باشد. دقت کنید که در حین تحویل، کد نهایی بایستی اجرا شده تا در حضور هم از اجرای زنده قطعات موسیقی که سیستم هوشمندتان تولید می‌کند لذت ببریم.
- محدودیتی در انتخاب هایپرپارامترها و ساختار دقیق شبکه‌ها وجود ندارد. هرچند دانشجو موظف است تا با انتخاب مقادیر درست و آموزش بهتر مدل، کیفیت قطعه موسیقی پایانی را افزایش دهد.
- محدودیتی در انتخاب نرخ نمونه‌برداری (sampling rate) وجود ندارد.
- در آموزش شبکه‌های عصبی، نحوه استفاده **درست** از داده‌هایی که در اختیار دانشجو قرار داده شده دارای اهمیت بالایی بوده و صحت کار مدل‌ها را تعیین می‌کند.
- از آنجایی که نت‌ها متنوع بوده و به طبع تعداد کلاس‌های قابل پیش‌بینی در آموزش مدل Note Identification بالاست، دانشجو موظف است به مقدار نیاز از تکنیک های داده‌افزایی (Data Augmentation) استفاده کند.
- بخش قابل توجهی از نمره دهی این پروژه برمبنای کیفیت قطعه‌های موسیقی خروجی است.
- دانشجو موظف است به منظور اثبات عملکرد مدل Note Identification، لیستی از نت های اولیه (notes_list) و تولیدشده (next_note ها) را در لیست NOTES_GROUND_TRUTH ذخیره کند و میزان شباهت اعضای آن‌را با اعضای sub_notes_classes مقایسه کند.
- مراحل اصلی چرخه بایستی حتما به همین ترتیب ذکر شده انجام شود. دانشجو مختار است به غیر از ترتیب چرخه، جزییات کدها را بنا به خواسته‌اش تغییر دهد.
- گزارش پروژه بایستی کامل بوده و دقیق به سوالات و بخش‌های مربوطه پاسخ داده‌باشد.
- پروژه دانشجو بایستی نمودارهای تغییر Loss و Accuracy هر دو مجموعه آموزش و تست را داشته باشد.
- پیاده سازی به صورت گروهی (**حداکثر ۲ نفره**) است و **هیچ محدودیتی** برای زبان برنامه نویسی و فریمووورک یادگیری عمیق مورد استفاده وجود ندارد.
- بحث و بررسی میان دانشجویان آزاد است اما هر دانشجو موظف است پروژه را به تنهایی انجام دهد و در هنگام تحویل حضوری، به تمام جزئیات کد کاملاً مسلط باشد. با موارد تقلب و کپی کردن، طبق تشخیص دوستان حل تمرین، برخورد جدی خواهد شد.
- توجه کنید که کدهای شما باید خوانا و دارای کامنت گذاری مناسب باشد.
- زمان بندی و چگونگی تحویل حضوری پروژه، متعاقباً اعلام خواهد شد.