STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

ChordSeqAI: Generování akordových sekvencí pomocí hlubokého učení

Petr Ivan
Zlínský kraj

Kroměříž, 2024

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

ChordSeqAI: Generování akordových sekvencí pomocí hlubokého učení

ChordSeqAI: Generating Chord Sequences Using Deep Learning

Autor: Petr Ivan
Škola: Arcibiskupské gymnázium v Kroměříži,
Pilařova 3, 767 01 Kroměříž
Kraj: Zlínský kraj
Konzultant: Bc. Petr Kučera

Kroměříž, 2024

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Kroměříži dne _____

_____

Petr Ivan

Poděkování

Chtěl bych poděkovat Bc. Petru Kučerovi za přínosné konzultace a vedení projektu.

Anotace

Tahle práce představuje nový nástroj řízený umělou inteligencí pro napomáhání při hudební kompozici prostřednictvím generování akordových postupů. Diskutuje o sběru a analýze dat, které odhalují nečekané vzorce v akordových postupech napříč různými hudebními žánry a obdobími. Vyvinuli jsme řadu modelů postavených na hlubokém učení, od jednoduchých rekurentních sítí až po sofistikované architektury Transformer, včetně Transformerů podmíněných a založených na stylu pro lepší ovladatelnost. Lidské hodnocení ukazuje, že v kontextu našich specifických metod zpracování dat jsou akordové sekvence generované pokročilejšími modely prakticky nerozeznatelné od skutečných sekvencí. Modely jsou poté integrovány do uživatelsky přívětivé webové aplikace s otevřeným zdrojovým kódem, která zpřístupňuje tento pokročilý nástroj pro tvorbu hudby širšímu publiku.

Klíčová slova

hluboké učení; PyTorch; akordové postupy; hudební kompozice; webová aplikace

Annotation

This report presents a novel AI-driven tool for aiding musical composition through the generation of chord progressions. Data acquisition and analysis are discussed, uncovering intriguing patterns in chord progressions across diverse musical genres and periods. We developed a range of deep learning models, from basic recurrent networks to sophisticated Transformer architectures, including conditional and style-based Transformers for improved controllability. Human evaluation indicates that, within the context of our specific data processing methods, the chord sequences generated by the more advanced models are practically indistinguishable from real sequences. The models are then integrated into a user-friendly open-source web application, making advanced music composition tools accessible to a broader audience.

Annotation

Deep Learning; PyTorch; Chord Progression; Music Composition; Web Application

Obsah

1 ÚVOD

1.1 Základy hudební teorie

Hudební kompozice je složitě svázána se základy hudební teorie, mezi nimiž hrají zásadní roli akordové postupy. Nejprve je potřeba se ponořit do hudebního kontextu, abychom pochopili složitost generování akordových sekvencí. Soudobá hudba se skládá z několika zásadních prvků, včetně celkové struktury, rytmu, melodie, harmonie, barvy a dynamiky. Harmonie může být konceptualizována jako více melodií vytvářejících polyfonní texturu. Nedávné trendy v hudební kompozici však zaznamenaly posun v této perspektivě, kdy se s harmonií často zachází jako s nezávislou entitou, zatímco melodie, které ji tvoří, přebírají poměrně podřízenou roli vůči historickému pohledu.

Základním stavebním kamenem harmonie se stal akord – skupina tónů znějící společně. Jednoduchou formou, základním kamenem západní hudby, je kvintakord, tvořený základním tónem, tercií a kvintou. Nejběžnější jsou durové, mollové, zmenšené a zvětšené kvintakordy, které se liší v intervalech, ze kterých se skládají. Kromě nich existují mnohé další akordy, vytvořené různými metodami, jako jsou akordy s horními tóny (zejména septakordy, nónové, undecimové a tercdecimové akordy spolu s jejich variacemi), suspendované akordy (sus2 a sus4) nebo lomené akordy (které mají přidaný basový tón).

Zajímavým aspektem akordů je jejich notový zápis. Akordová značka obvykle obsahuje základní tón, kvalitu akordu (udávající, zda je durový, mollový, zmenšený atd.), všechny změněné nebo přidané noty a basový tón, pokud není shodný se základním. Symbolické znázornění akordů není standardizováno; místo toho se používají různé zápisy v závislosti na kontextu. Například durový septakord může být reprezentován jako maj7, M7, $\Delta 7$ nebo dokonce jen ma7 nebo Δ. Tato práce se pokouší o použití jednoznačného akordového zápisu; výšky tónů jsou popsány Vědeckou notací výšky tónu (známou také jako Americká standardní notace výšky tónu).

Akordové postupy jsou tedy sérií akordů hraných v sekvenci. Postupy, v určitých kontextech označované také jako sekvence, nejsou nahodilé; řídí se specifickými vzorci a pravidly, které se vyvíjely v průběhu staletí hudební tradice. Krása akordových postupů spočívá v jejich všestrannosti a expresivitě, kdy každý postup nese svou jedinečnou náladu a charakter. Jedním z klíčových konceptů pro pochopení akordových postupů je myšlenka tonality, která odkazuje na způsob, jakým jsou akordy soustředěny kolem tóniky nebo výchozí tóniny. Tento koncept navazuje na diatonické akordy, které jsou postaveny z tónů jedné tóniny. Tyto akordy jsou obvykle označeny římskými číslicemi, označujícími jejich pozici v tónině (obvykle velké číslice představují durové akordy a malé mollové). Tímto způsobem můžeme popsat některé běžné

akordové postupy, včetně ii-V-I, I-IV-V nebo I-V-vi-IV (viz část 2.2 pro více příkladů).

Dalším důležitým aspektem akordů jsou jejich obraty a vedení hlasů při popisu postupů. Obraty odkazují na varianty akordu s odlišným basem, ale stejnými tóny (běžně konstruované posunutím nejnižších tónů o oktávu nahoru u kvintakordů), zatímco vedení hlasů označuje melodické změny každého hlasu v polyfonním pohledu (obvykle použitím různých obratů akordů, aby se minimalizovala vzdálenost mezi nimi, nebo jejich posunem určitým směrem).

Méně zkušení hudebníci často používají při komponování hudby přístup založený na komponování melodie nebo harmonie jako první, zatímco zkušenější používají hybridnější přístup, který se často zaměřuje na více aspektů hudby současně, protože oddělená tvorba jednotlivých aspektů je omezená (např. přístup založený na skládání melodie jako první často vytváří příliš jednoduché harmonie, zatímco harmonie jako první často postrádá vedení hlasu a melodie mohou být omezené). Skládání harmonie není primitivní úkol, protože standardní metody hledání akordů zapadajících do kontextu vytvářejí velký prostor možností (např. pomocí kvintového kruhu, diatonických akordů nebo vedlejších dominant), ve kterém je obtížné se orientovat. Když se dostaneme ke složitějším akordovým postupům s akordy s přidanými tóny a dalšími změnami, prostor i pravidla se rozšiřují a zkušenost zůstává jediným prostředkem smyslu pro orientaci.

Namísto odvozování obecných pravidel harmonie můžeme vyvinout modely hlubokého učení, které se naučí akordové postupy, aby nás navigovaly. Prostřednictvím tohoto projektu jsme se pokusili pomoci těm, kteří začínají svou cestu v hudební kompozici poskytnutím silného asistenta řízeného umělou inteligencí pro vytváření akordových sekvencí, pomáhajícího prozkoumat prostor možností a zpřístupňujícího metodu založenou na harmonii jako první.

1.2 Hluboké učení v hudbě

Běžné oblasti výzkumu můžeme klasifikovat několika způsoby. Na základě reprezentace použitých dat existují symbolické (např. učení se z MIDI souborů) a subsymbolické (např. použitím spektrogramu zvukového souboru) přístupy. Existují různé cíle, včetně generování melodie [1], harmonie [2, 3, 4], polyfonie [5, 6], doprovodu [7, 8, 9], nebo syntetizování zvukové stopy [10, 11, 12].

Jednoduché Markovovy řetězce se používaly poměrně běžně až do nástupu hlubokého učení. Poté přišly dopředné neuronové sítě, následované (variačními) autoenkodéry, rekurentními neuronovými sítěmi, konvolučními neuronovými sítěmi, generativními adversariálními sítěmi a přístupy zpětnovazebního učení [13]. Nedávné studie také zkoumají použití architektury Transformer [14, 15, 16, 17].

Naše práce se zaměřuje na tvorbu harmonie (bez referenční melodie), která spadá pod symbolickou generaci hudby. Použití hlubokého učení v této úloze není nové, například [18] demonstruje způsob, jak doporučit akordy s přístupem WordToVec na korpusu akordových postupů. [19] představuje způsob, jak modelovat akordové postupy pomocí architektury Transformer, zatímco [4] používá techniku zpětnovazebního učení. Existují i jiné strategie bez použití hlubokého učení, jako jsou ty, které používají evoluční algoritmy [20], umělé imunitní systémy [21], nebo metody používající pravidla [22]. Mnohé z těchto nástrojů nejsou snadno dostupné veřejnosti, jsou omezené počtem různých akordů nebo postrádají ovladatelnost. V této práci jsme se pokusili překonat mnohá omezení těchto technik.

Podrobný přehled hlubokého učení pro generování hudby najdete v [13].

2 DATA

2.1 Sběr dat

Vzhledem k tomu, že výkon jakéhokoli modelu silně koreluje s kvalitou a rozmanitostí dat, na kterých byl trénován, je nezbytné získat vysoce kvalitní datovou sadu. Neexistuje mnoho volně dostupných dostatečně velkých datových sad akordových sekvencí, existuje však pár možností, jako je například datová sada McGill Billboard, která obsahuje 700 položek [23]. Abychom získali zajímavé výsledky, potřebujeme podstatně větší datovou sadu.

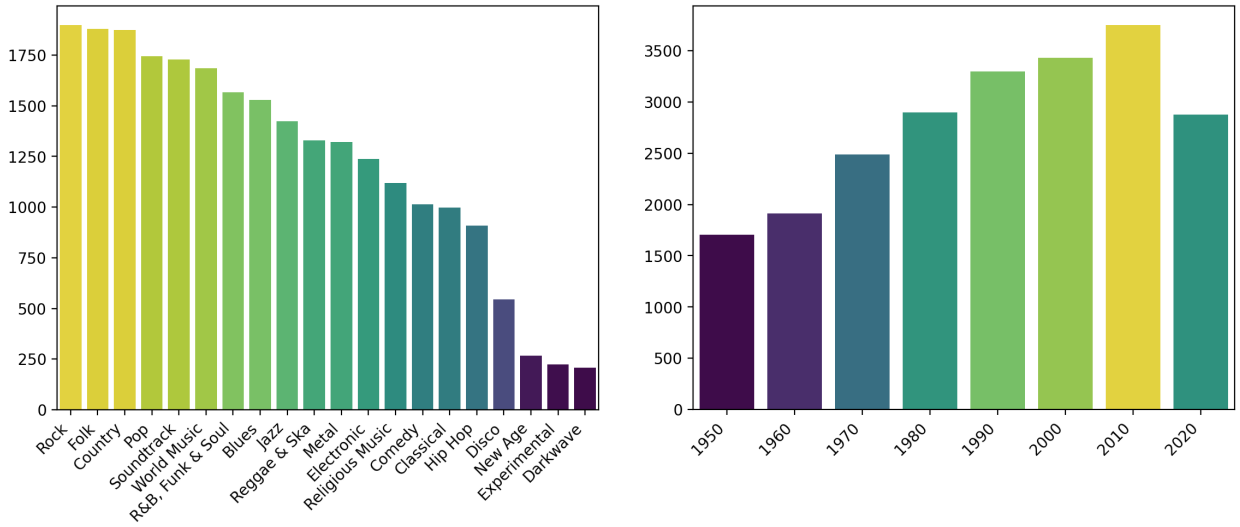
Místo slučování více datových sad jsme se rozhodli získat vlastní z webových stránek obsahujících skladby anotované akordy. Konkrétně Ultimate Guitar byla vybrána pro svou obrovskou sbírku více než milionu skladeb [24]. Kromě akordů umožňuje tato webová stránka získat další údaje o skladbách, jako je jejich žánr, desetiletí nebo styl. I když tyto informace nemůžeme přímo získat ze stránky písně, můžeme skladby filtrovat a nabýt je tímto způsobem. Použité filtry jsou uvedeny v adrese URL, což nám umožňuje snadno je procházet. Tóny každého akordu jsou také odesílány (jako soubor JSON s čísly MIDI not) při otevření vizualizace akordů na klaviatuře v záložce, což se ukázalo jako užitečné při tokenizaci, jak je popsáno v části 2.3.

Webová škrabka byla napsána v Pythonu s knihovnou Playwright [25]. Vzhledem k omezené výpočetní kapacitě bylo pro každý žánr v každém desetiletí seškrábáno pouze prvních 300 položek podle hodnocení, což po vyčištění vytvořilo slušnou sadu 22 367 vzorků. Některé kombinace žánrů a desetiletí obsahovaly méně než 300 položek, zatímco jiné byly mnohem větší a bylo možné je dále proškrabat, aby se soubor dat rozšířil. Tento přístup nám poskytl data, která zahrnují i méně populární žánry, aniž by vyžadovala velký výpočetní výkon. Škrábání probíhalo pomalejším tempem, trvalo několik desítek hodin a nebylo dosaženo omezení rychlosti ze strany serveru. Škrabka, stejně jako veškerý další kód, data a další zdroje, jsou k dispozici na GitHubu (viz část 7).

Při používání uměleckých děl jiných osob pro trénování modelů umělé inteligence vstupují do hry etické aspekty. Obecně platí, že melodie, stejně jako text písně, jsou chráněny, protože jsou považovány za jedinečná díla skladatele nebo textaře, zatímco akordové postupy obvykle nepodléhají autorskému právu. Je to kvůli omezenému prostoru možných postupů, které zapadají do nálady a žánru skladby, takže mnoho skladeb používá ty stejné [26]. Přesto jsme se snažili ujistit se, že modely produkují nové sekvence a ne jen slepě replikují trénovací data.

2.2 Datová analýza

Z nezpracovaných dat byly odstraněny duplikáty a neúplné vzorky. Pokud je skladba spojena s více než jedním žánrem, zobrazí se ve více položkách, které lze sloučit. Tyto žánry ve sloučeném hesle pak byly odděleny svislou čárou, protože mezery, čárky a další běžné oddělovací znaky již byly přítomny v názvech žánrů. Abychom zajistili použitelnost dat, zkonstruovali jsme kompletní mapu akordů z jednotlivých tónů na akordovou značku získanou během škrábání. Vzhledem k tomu, že někdy v datech nebyly přítomny všechny akordy skladby, byla mapa transponována do všech možných tónin s pomocí knihovny music21 [27]. Položky, které obsahovaly akordy, které nebyly přítomny v kompletní mapě akordů, byly odstraněny. Takové skladby měly obecně nižší počet hodnocení a také počet hvězdiček, které uživatelé Ultimate Guitar používají k hodnocení přesnosti použitých akordů. Ověřili jsme, že neexistují žádné nejednoznačné akordové značky (které by byly vícekrát reprezentovány různými notami). Toto zpracování snížilo celkový počet záznamů z 32 792 na 22 367 (většina z nich byly duplikáty). Potenciálně nekvalitní vzorky, které byly označeny buď nízkým počtem hvězdiček, nebo hodnocením, mohly být také odstraněny, ale protože jsme neměli dodatečná opatření, která by zajistila, že se nejedná pouze o nepopulární, ale přesto cenné údaje, byly zachovány. Sloučeny byly i po sobě jdoucí akordy, protože se věnujeme pouze harmonickým změnám.

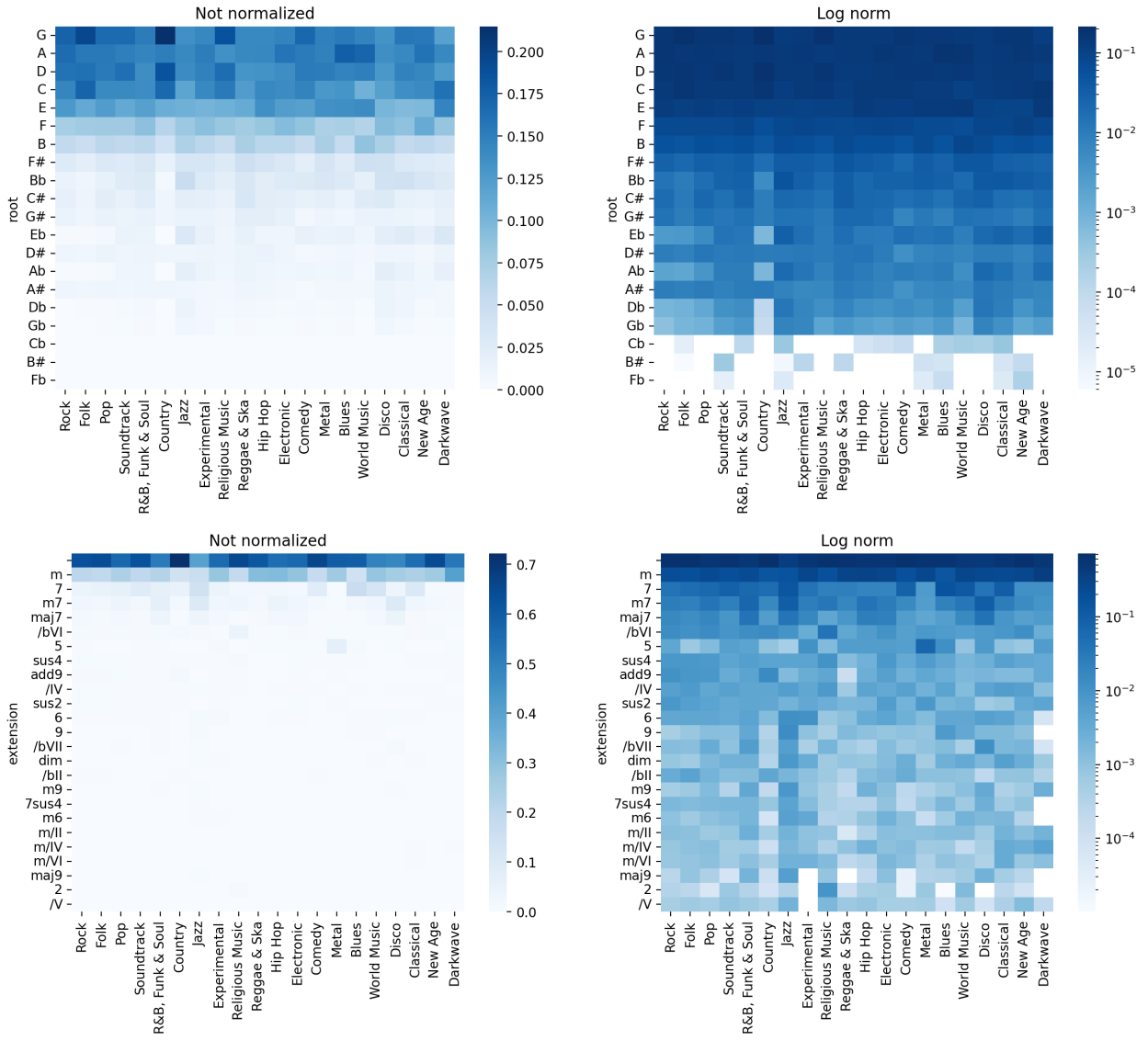


Obrázek 1: Distribuce žánrů a desetiletí v datové sadě.

Můžeme vykreslit rozložení jednotlivých žánrů a desetiletí v datové sadě, jak je znázorněno na obrázku 1. Zdá se, že získáváme vyváženou datovou sadu, přičemž pouze několik žánrů je nedostatečně zastoupeno. Nových písní je více než starých, s výjimkou současného desetiletí (které stále probíhá).

Než jsme se ponořili do akordových postupů, vizualizovali jsme použití jednotlivých akordů

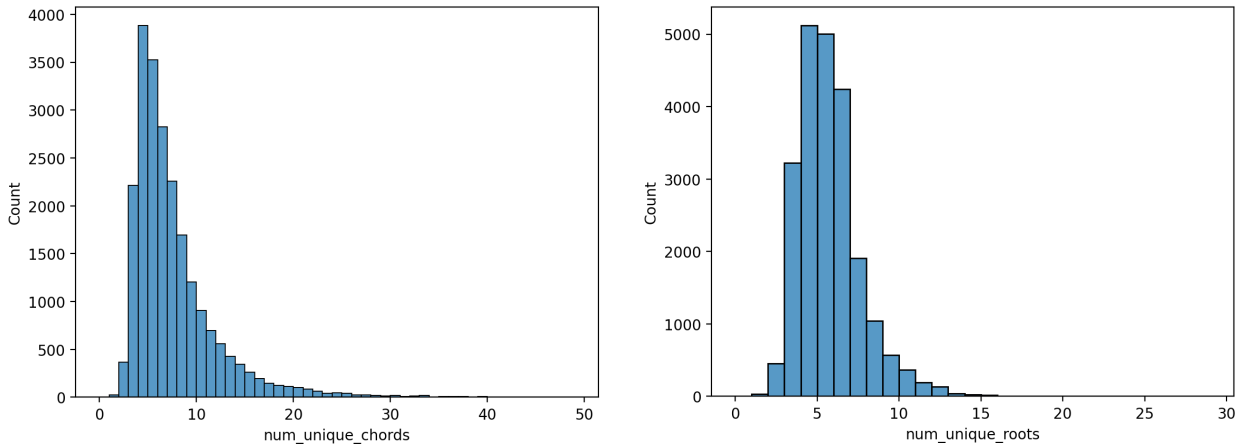
– jejich základní tón (bez sloučení enharmonicky ekvivalentních not) a tvar akordu, který se zde pro jednoduchost nazývá rozšíření (viz obrázek 2). Lomené akordy byly normalizovány tak, aby měly relativní význam (reprezentovaný římskými číslicemi), protože funkce části za lomítkem silně závisí na základním tónu. Pro lepší zobrazení jemnějších detailů byly vytvořeny logaritmicky normalizované grafy. Ukazují nám silný tmavý pruh mezi různými kvalitami a základními tóny akordů používanými v jazzu, zatímco country a darkwave ukazují opak. Z jednotlivých použitých forem lze poukázat na některé zajímavé postřehy, například že metal je jediným žánrem, který ve velké míře používá power akordy (označené 5).



Obrázek 2: Top 25 základních tónů a kvalit akordů napříč žánry.

Poté jsme analyzovali akordové postupy, počínaje složitostí. Jednoduchou metrikou pro složitost harmonie může být počet jedinečných akordů a jejich jedinečných základních tónů,

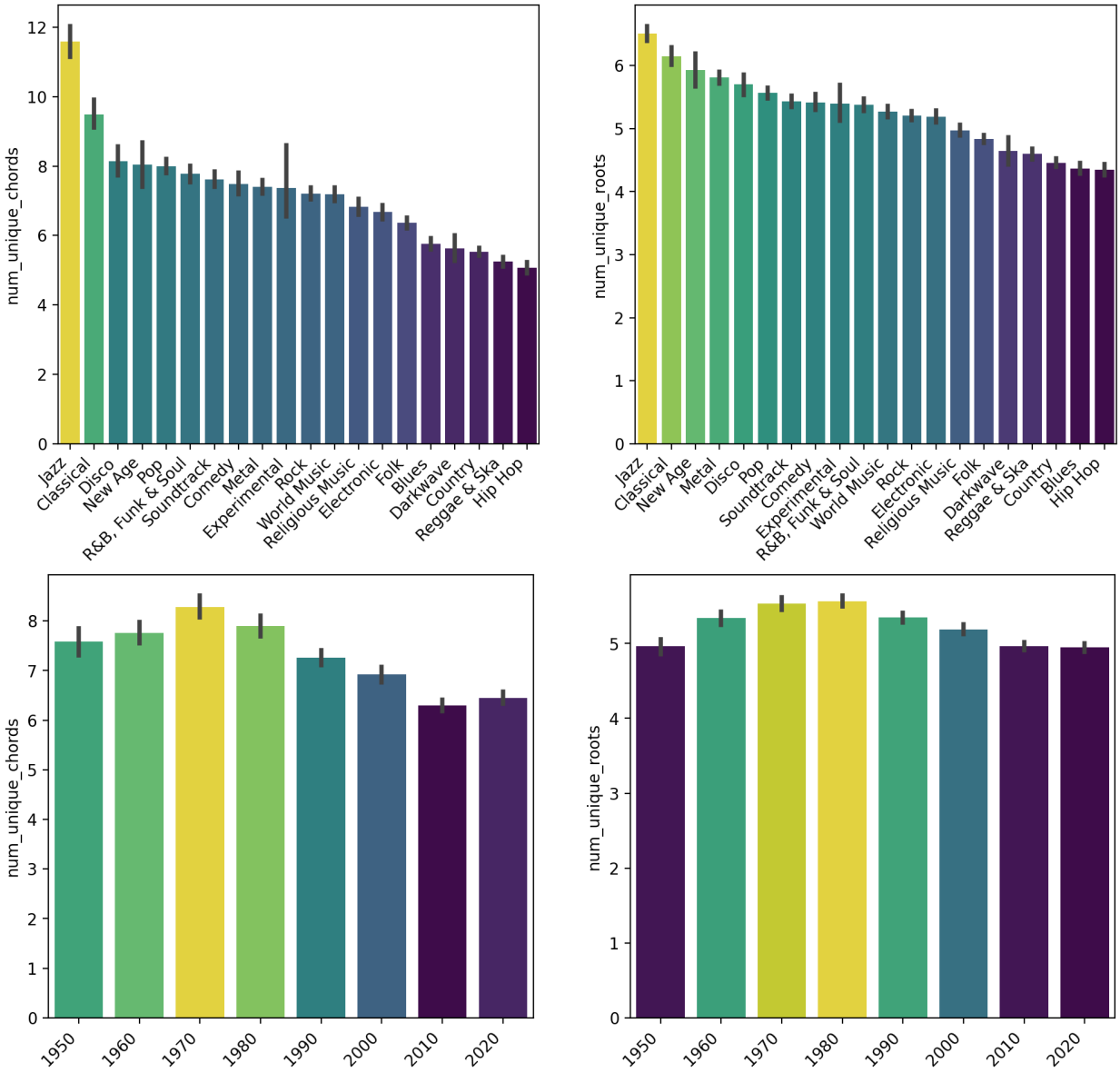
zatímco s enharmonicky ekvivalentními tóny se zachází jako s odlišnými, protože konkrétní notace závisí na kontextu. Celkové rozložení této složitosti je uvedeno na obrázku 3. Podobně můžeme tyto metriky vykreslit napříč různými žánry a desetiletími (obrázek 4). V datech je vidět několik zajímavých postřehů, například jazz dominuje v obou metrikách, zatímco hip hop je nejjednodušší. Experimentální žánr má nejvyšší variabilitu, což naznačuje, že obsahuje jak jednoduché, tak složité formy harmonií. Analýza vůči desetiletím nám ukazuje, že složitost harmonií má tendenci růst přibližně do 70. a 80. let 20. století a poté klesat, s výjimkou současného desetiletí.



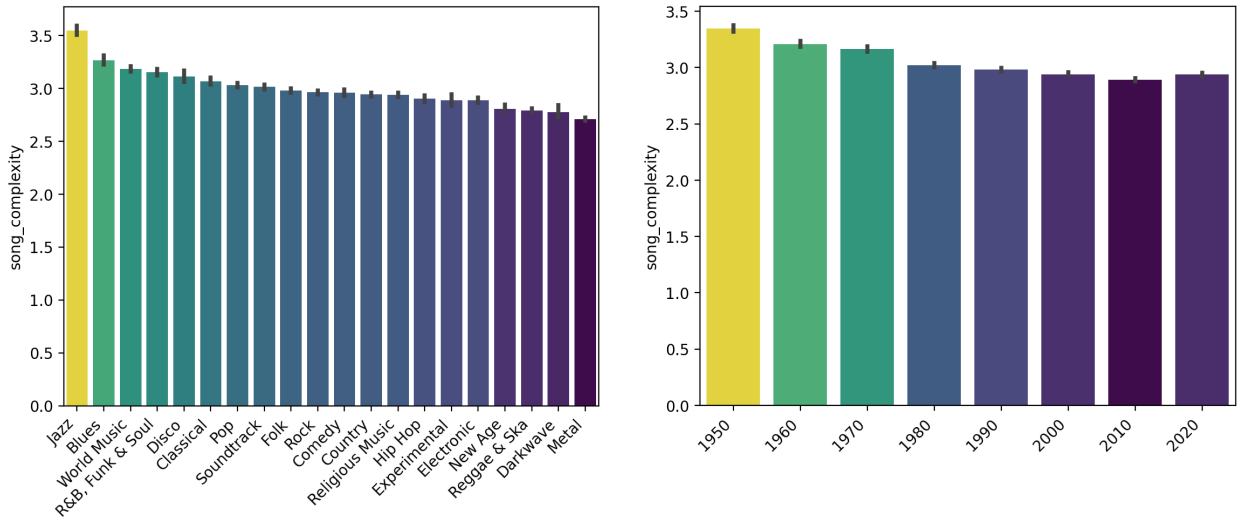
Obrázek 3: Histogramy počtu jedninečných akordů a základních tónů.

Tato technika však nebere v potaz složitost jednotlivých akordů. Vzhledem k tomu, že míry složitosti nejsou striktně definovány, neboť závisí především na lidské interpretaci, pokusili jsme se vytvořit vlastní metriku založenou na disonanci jednotlivých intervalů tvořících akord. Z not tvořících akord byly porovnány všechny možné dvojice a skóre složitosti jednotlivých intervalů bylo sečteno a vyděleno počtem not v akordu, čímž se akordy s více notami staly složitějšími podle notového zápisu zároveň při zachování rovnováhy. Přesná implementace je k dispozici v repozitáři na GitHubu.

Průměrná složitost akordů použitých v písni pak byla získána z písní napříč žánry a desetiletími, jak je znázorněno na obrázku 5. Podle této nové metriky zůstává jazz nejkomplexnějším žánrem, zatímco metal zůstává na druhém konci. To lze vysvětlit z předchozího obrázku 2, kde je zřejmé, že jazz používá řadu různých složitých akordů, zatímco metal se většinou omezuje na použití durových, mollových a power akordů. Zatímco blues je docela jednoduchý vzhledem k počtu jedinečných akordů a jejich základních tónů, tato nová metrika jej překvapivě řadí hned za jazz. Vývoj v průběhu desetiletí nám ukazuje trochu jiný trend než dříve, kdy akordy mají tendenci být v průběhu času méně disonantní, s výjimkou současného desetiletí.



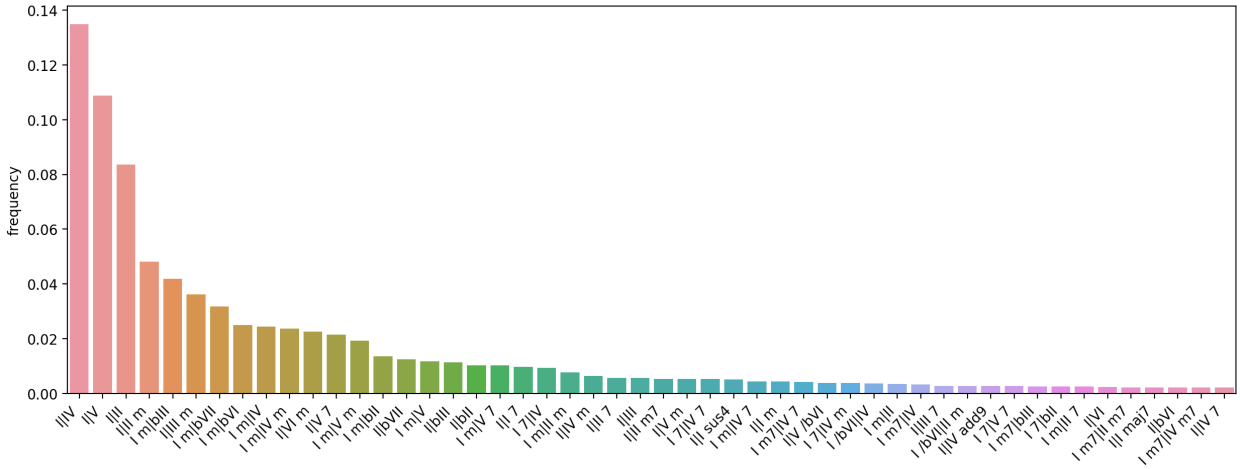
Obrázek 4: Počet jedinečných akordů a základních tónů napříč žánry a desetiletími.



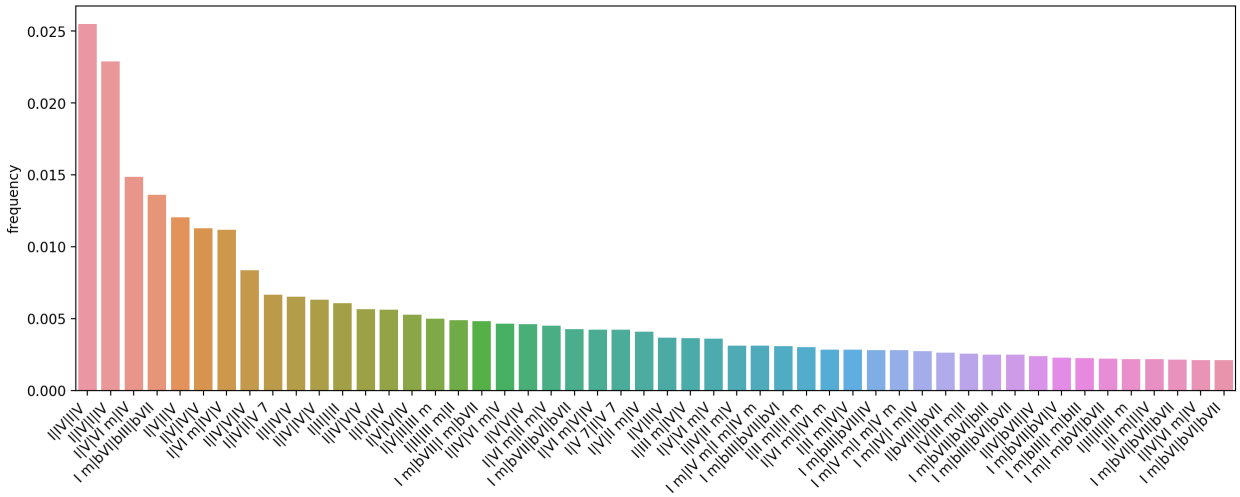
Obrázek 5: Vlastní metrika složitosti napříč žánry a desetiletími.

Pro lepší pochopení akordových postupů byla provedena n-gramová analýza. Předpokládáme tonalitu, proto k popisu akordů používáme římské číslice. Z praktických důvodů zde mollové akordy neoznačujeme malými písmeny, ale připojením „m“ k velké variantě. Vzhledem k tomu, že mnoho písní používá opakované sekvence akordů, standardní metoda by nám poskytla mnoho nadbytečných n-gramů (např. opakováním I IV V vznikne také IV V I a V I IV), proto byl použit jiný přístup. Aby bylo možné považovat všechny tyto různé n-gramy za jeden, byl zaveden pojem cyklických permutací a forma základní cyklické permutace. Pro každý n-gram jsou cyklické permutace tohoto n-gramu všechny n-gramy (stejné délky) vzniklé opakováním tohoto n-gramu. Základní cyklická permutace n-gramu je pak první z cyklických permutací tohoto n-gramu seřazených vzestupně podle pořadí římských číslic; pokud existuje více n-gramů se stejným pořadím římských číslic, použije se lexikografické pořadí pro druhou část akordu. Abychom se zaměřili především na samotné harmonické změny namísto širšího kontextu, je n-gram také normalizován tak, aby nejnižší římská číslice byla vždy I. Všimněte si, že tento proces může produkovat n-gramy začínající na I I, což se může zdát jako chyba, protože po sobě jdoucí akordy byly odstraněny. Tento jev lze vysvětlit uvědoměním, že sekvence jako I IV V I se normalizuje cyklickou permutací na n-gram I I IV V. I když se tento postup může zdát složitý, poskytuje nám snadno interpretovatelné výsledky.

Na obrázcích 6 a 7 je zobrazeno 50 nejběžnějších bigramů a čtyřgramů. Frekvence představuje proporcionální rozdělení n-gramu (normalizovaného a převedeného na základní cyklickou permutaci) v celé datové sadě. Tvar akordu je od římské číslice oddělen mezerou, akordy jsou pak odděleny svislou čárou. Podobné grafy byly vytvořeny také pro čtyři hlavní žánry, jazz a metal (protože naše vlastní metrika je řadí na opačné konce spektra složitosti), lze je nalézt i s kompletní analýzou dat na GitHubu.



Obrázek 6: Top 50 bigramů.

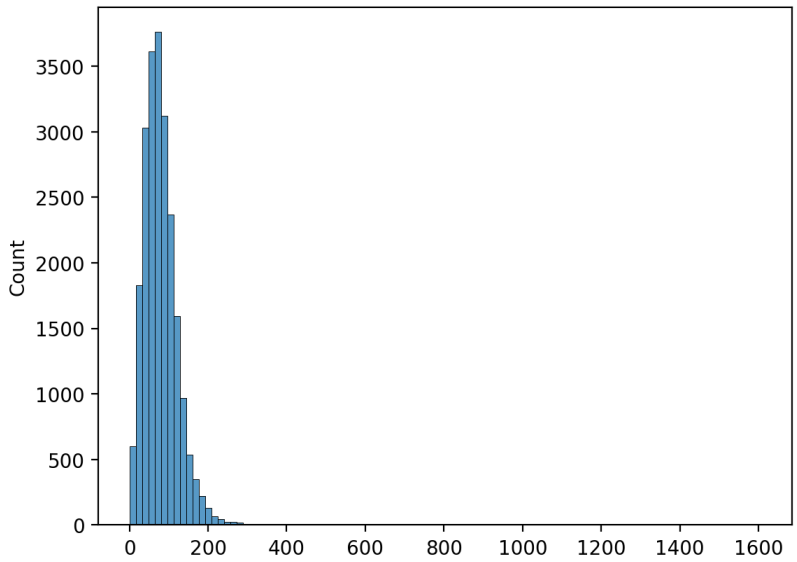


Obrázek 7: Top 50 čtyřgramů.

2.3 Tokenizace dat

Naše datová sada obsahuje velké množství různých akordů, kde existuje mnoho synonym, tvořených nejen enharmonicky ekvivalentními notami, ale také různými použitými notacemi akordů. Knihovna music21 [27] nám umožňuje analyzovat řadu různých akordů, nicméně pro naše účely je tato metoda velmi omezená, protože selhává na významné části dat. Neexistuje široce dostupný veřejně dostupný algoritmus pro analýzu akordových značek, protože jeho vytvoření není triviální. Díky způsobu, kterým byla data seškrábána, máme mapu 4986 akordových značek po jejich transpozici do všech možných tónin. Při sloučení synonym podle not v akordu dostaneme pouhých 3360 tokenů. Toto číslo je však vzhledem k velikosti naší datové sady a dostupných výpočetních prostředků stále příliš velké na to, aby bylo použitelné. Místo toho byla vyvinuta nová metoda pro vytvoření synonym.

V ideálním případě bychom chtěli považovat obraty jednotlivých akordů za stejný akord, protože pouhá změna obratu akordu neznamená podstatnou změnu jeho funkce. Byly by důležité, kdybychom chtěli modelovat i vedení hlasů, ale pro tyto účely by byla vhodnější jiná datová sada (například sestavená z MIDI souborů), necháváme je tedy jako záležitost budoucí práce. Jedním z důležitých aspektů obratů akordů a dalších metod záměny hlasů je, že výšky užitých tónů zůstávají stejné, pokud jsou všechny transponovány do stejné oktávy. Tuto myšlenku lze použít ke konstrukci tokenizace na základě třídy výšky tónu, kde zpracujeme každý akord jako binární vektor dvanácti prvků a každé nové jedinečné reprezentaci přiřadíme token. S tímto přístupem je teoretická horní hranice tokenů $2^{12} = 4096$, ale pro praktické účely použijeme pouze reprezentace, které jsou přítomny v datové sadě, takže nám zůstane pouze 1033 tokenů. V praxi se použijí další dva navíc, a to tokeny označující začátek a konec.



Obrázek 8: Histogram délek sekvencí.

Po tokenizaci dat tímto způsobem byly všechny sekvence s více než 256 tokeny (včetně tokenů začátku a konce sekvence) vyřazeny. Z obrázku 8 vidíme, že to stačí k pokrytí většiny sekvencí. Tento krok je nutný k efektivnímu trénování modelů, protože délky jednotlivých sekvencí uvnitř minidávky musí být stejné (k vyplnění nadbytečných mezer se použije odsazení).

3 VÝVOJ MODELŮ

3.1 Vyhodnocovací metriky

Než se podíváme na architektury používané k modelování dat, představíme několik metrik, které vhodně popisují jejich výkon, protože vyhodnocení generativních modelů není triviální. V části 3.6 se budeme zabývat hodnocením lidmi, protože je to často jediný způsob, jak získat poněkud objektivní měřítko výkonu, které lze porovnat s jinými technikami.

Top-1 přesnost. Poměr výstupů modelu, kdy se předpověď s největší hodnotou shoduje se skutečným tokenem. Snadno interpretovatelná metrika, omezená ignorováním ostatních předpovědí, i když jejich hodnota je také důležitá. Tuto metriku budeme také označovat pouze jako přesnost.

Perplexita. Definováno jako umocnění entropie pravděpodobnostní distribuce; jinými slovy inverzní modelem předpovězená pravděpodobnost referenční posloupnosti tokenů, normalizovaná počtem slov. Intuitivně si ji lze představit jako měřítko, jak "překvapený"je model daty. Přesnost i tuto metriku vypočítáme na původních datových sadách bez použití augmentace. Nižší hodnoty znamenají lepší výkon.

Fréchet Feature Distance. Fréchet Inception Distance (FID) je běžná metrika používaná k popisu výkonu sítí generujících obraz. Porovnává distribuce extrahovaných charakteristických rysů (pomocí modelu InceptionV3, odtud název) reálných a generovaných vzorků. Podobný přístup byl použit pro náš problém, kde byla klasifikační síť Transformer trénována tak, aby předvídala žánr a dekádu vzorku, model bude sloužit jako extraktor rysů. Jeho architektura byla téměř stejná jako u velké varianty našeho generativního Transformeru, popsaného v sekci 3.3, s výjimkou výstupní vrstvy. Hodnoty z poslední vrstvy před výstupní vrstvou byly zprůměrovány napříč dimenzí tokenů, takže nám zbyl 96rozměrný vektor rysů. Fréchet Feature Distance (FFD) byla poté získána porovnáním vektorů rysů testovacích vzorků se stejným počtem generovaných vzorků. Tato metrika je omezená několika způsoby, například že její hodnota má tendenci klesat s počtem použitých vzorků, ale poskytuje lepší představu o distribuci vzorků, které model generuje. Skóre uvádíme pouze na testovací sadě s augmentací, protože chceme, aby vygenerované vzorky byly ze všech možných tónin. Nižší hodnoty znamenají lepší výkon.

Pokusili jsme se také vyvinout několik dalších metrik, například ty, které porovnávaly rozložení n-gramů napříč generovanými a reálnými sekvencemi pomocí kosinové podobnosti a Fréchetovy vzdálenosti, ale většinou byly neúspěšné kvůli nedostatečnému popisu výkonu sítě nebo kvůli paměťovým a výpočetním omezením.

3.2 Recurrent Network

Vzhledem k tomu, že rekurentní architektura je páteří mnoha populárních architektur generujících hudbu, vytvořili jsme jednoduchou hradlovou rekurentní síť, abychom získali referenční výkon pro jiné, sofistikovanější architektury. Hyperparametry jsou zvoleny tak, aby se dosáhlo rovnováhy mezi výkonem a výpočetními prostředky, trénování větší varianty se stejnou celkovou strukturou vedlo k nevýznamným změnám výkonu.

Architektura je strukturována takto:

- **Embedding vrstva**: Mapuje vstup do 96rozměrného prostoru.
  - *Vstup*: Sekvence tokenů
  - *Výstup*: 96rozměrné vektory
- **GRU vrstvy**: Trojvrstvá hradlová rekurentní jednotka (GRU) s 96 dimenzemi.
  - *Vstup*: 96rozměrné vektory z předchozí vrstvy
  - *Výstup*: 96rozměrný GRU výstup
- **Vícevrstvý perceptron (MLP)**: Sekvence afinních transformací a nelineárních aktivačních funkcí.
  - *Vrstvy*:
    1. Lineární vrstva (z 96 do 96 dimenzí)
    2. ReLU aktivační funkce
    3. Lineární vrstva (z 96 dimenzí do velikosti slovníku)
  - *Výstup*: Finální vektor o velikosti slovníku.

  Celkový počet parametrů: 376 683

Datová sada byla rozdělena na trénovací sadu a testovací sadu v rozdělení 80/20 %, aby byla testovací sada reprezentativní pro celkovou distribuci. Během trénování byla použita velikost minidávky 128 vzorků. Aby se uměle zvětšila velikost datové sady, byla použita transpozice k trénování sekvencí v náhodných tóninách. Kromě toho augmentace také činí model implicitně ekviariantním vůči použité tónině. Další obvyklou metodou je transponovat všechny sekvence do stejné tóniny a místo toho trénovat model na takto upravené sadě [13], nicméně tento přístup jsme nepoužili, protože získání správné tóniny je netriviální (nelze jen vybrat tonalitu prvního akordu) a metoda by nebrala v úvahu modulaci (možná by se s modulovanými částmi zacházelo jinak a na takových vzorcích by se dosáhlo horšího výkonu).

Vzhledem k tomu, že jsme trénovali na odsazených datech, byl vyvinut maskovací mechanismus pro ztrátu křížové entropie a výpočet přesnosti. Byl použit optimalizátor Adam s počáteční mírou učení 0.001 ($\beta_1 = 0.9, \beta_2 = 0.999$), s plánovačem rychlosti učení, který ji vynásobí 0.3 každých 10 epoch. Model byl trénován 50 epoch, což umožnilo konvergenci. Embedding vrstva byla trénována z náhodné inicializace, protože použití populárních technik běžně určených pro slova konvergenci jen nevýznamně urychlilo.

Model dosáhl 56.00% přesnosti (Top-1) na trénovací sadě a 56.59% na testovací sadě. Finální perplexita byla 4.927 na trénovací sadě a 4.851 testovací sadě. Na naší FFD metrice získal model skóre 9.290. Tyto metriky naznačují, že nedostatečně modeluje trénovací data (underfitting), ale jak bylo uvedeno výše, zvětšení velikosti modelu neprovedlo významné změny, takže k dosažení lepších výsledků bylo potřeba použít jinou architekturu.

3.3 Transformer

Architektura Transformer zaznamenala v poslední době obrovský úspěch v oblasti jazykového modelování, zejména s nástupem velkých jazykových modelů [28, 29, 30, 31]. Ukázalo se, že je lepší než rekurentní sítě, a to díky tomu, že spoléhá pouze na mechanismus pozornosti, což umožňuje lepší paralelizaci, efektivitu trénování a globální závislosti při dosažení lepšího výkonu [32]. Mechanismus globální pozornosti může intuitivně pomoci v úlohách souvisejících s hudbou, protože umožňuje modelu porozumět složitým vzorcům v dlouhých sekvencích, což je při hudební kompozici zásadní. Jelikož pracujeme s diskrétními sekvencemi tokenů, můžeme pro náš úkol snadno použít architekturu Transformer (konkrétně část dekóder).

Byly vyvinuty tři různé velikosti modelů: TransformerS, TransformerM a TransformerL, které představují malou, střední a velkou velikost respektive. Každý model je založen na standardní architektuře dekóderového Transformeru s různými specifikacemi, jak je podrobně uvedeno v tabulce 1. Maximální délka sekvence je konstantní na 256 tokenech mezi modely, jak je popsáno v části 2.3.

Vlastnost	TransformerS	TransformerM	TransformerL
Embedding rozměr (d_{model})	64	80	96
Počet hlav (n_{heads})	8	10	12
Počet vrstev (n_{layers})	6	12	24
Celkový počet parametrů	433 419	1 100 715	2 883 915

Tabulka 1: Specifikace Transformer modelů.

Bylo použito stejné rozdělení datové sady a trénovací hyperparametry jako u rekurentní

sítě, protože se ukázalo, že jsou dostatečně robustní.

Model	Train perplexita	Test perplexita	Train přesnost	Test přesnost	FFD
TransformerS	3.157	3.123	68.74%	69.05%	3.201
TransformerM	2.682	2.677	73.77%	73.91%	2.814
TransformerL	2.506	2.513	75.98%	76.04%	2.316

Tabulka 2: Výkon Transformer modelů.

Metriky výkonu jsou uvedeny v tabulce 2. Ve srovnání s rekurentní architekturou vidíme výrazné zlepšení, a to i u malé varianty. Výkon se zlepšuje s velikostí modelu, ale zdá se, že s největší variantou začíná trend stagnovat. Vzhledem k tomu, že vidíme prakticky stejný výkon na trénovací i testovací sadě, naznačuje to, že modely se nepřeučují (nedosáhli jsme overfitting) a mohli jsme je dále škálovat, aniž bychom potřebovali více dat.

3.4 Conditional Transformer

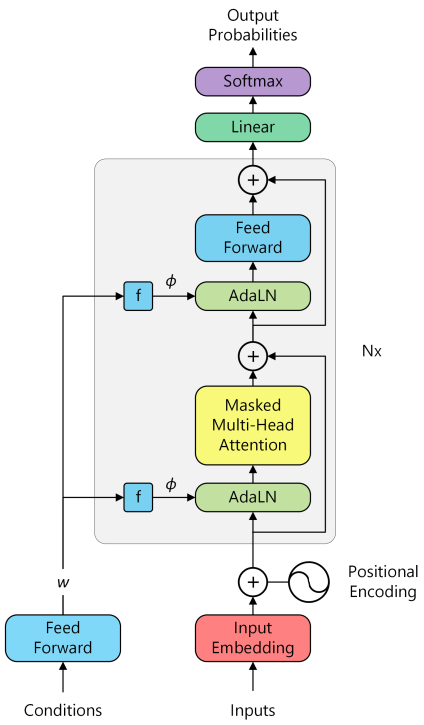
Jedním z důležitých aspektů generativního hlubokého učení je kontrolovatelnost procesu. Některé standardní metody zahrnují podmíněné generování, kdy se kromě sekvenčních dat používá určitá podmínka, například žánr skladby; a přístupy založené na stylu, které generují na základě nepropleteného latentního prostoru, který lze snadno prozkoumat, jako je tomu v případě moderních generativních adversariálních sítí (GAN).

Naivně se lze domnívat, že pouhé přidání dalšího tokenu „stylu“ na začátek sekvence pro architekturu Transformer by jí umožnilo generovat s ohledem na tento styl. Empiricky se však zdá, že tento přístup vede k tomu, že výsledný model pouze ignoruje dodatečné informace o stylu (tuto metodu jsme neúspěšně zkoušeli). Musel být vyvinut sofistikovanější přístup, který by model donutil používat styling. Inspirovali jsme se architekturou StyleGAN [33], která mimo jiné využívá adaptivní normalizaci instancí k řízení procesu generování, a podobnou techniku jsme aplikovali na náš problém.

Jelikož používáme Transformer využívající normalizace vrstev, jednoduše je změníme na adaptivní normalizace vrstev. Nechť $x \in \mathbb{R}^{n \times d_{model}}$ je skrytou reprezentací (kde n označuje délku sekvence) a $w \in \mathbb{R}^{d_{model}}$ je vektor popisující styl sekvence. Poté specializujeme w do $\phi = (\phi_g, \phi_b)$ pomocí naučitelné nelineární transformace (v našem případě používáme ReLU následovanou lineární vrstvou) tak, aby $\phi_g, \phi_b \in \mathbb{R}^{d_{model}}$. Také zopakujeme ϕ_g, ϕ_b podél první dimenze, abychom je dostali do tvaru $n \times d_{model}$. Adaptivní normalizace vrstvy je poté:

$$\text{AdaLN}(x, \phi) = (1 + \phi_g) \odot \text{LayerNorm}(x) + \phi_b, \tag{1}$$

kde LayerNorm je normalizace vrstvy, jak je implementovaná v PyTorch. Výsledkem je afinní transformace normalizované skryté reprezentace x po každém prvku zvlášť.



Obrázek 9: Conditional Transformer.

V případě Conditional Transformer použijeme žánr a dekádu sekvence a reprezentujeme je jako vektor. Vzhledem k tomu, že k sekvenci může být přiřazeno více žánrů, rovnoměrně mezi ně rozdělíme váhu tak, aby součet vah byl roven jedné. Výsledný podmíněný vektor je získán zřetězením za sebe one-hot vektoru žánru a vektoru dekády. Poté použijeme naučitelnou afinní transformaci, abychom ji dostali do požadovaného tvaru a použili ji jako vektor w ve všech adaptivních normalizacích vrstvy.

Tento přístup umožňuje, aby každá vrstva ovlivnila skrytou reprezentaci na základě žánru a dekády nezávisle a jedinečně. $1 + \phi_g$ používame namísto ϕ_g, aby funkce přibližně odpovídala identitě, protože očekáváme, že ϕ_g bude mít očekávanou hodnotu rovnou nule. Transformace po prvcích se používá místo globální, aby se podpořily specifické aspekty skrytého stavu, podobně jako jsou některé mapy rysů zesíleny vůči sobě navzájem v architektuře StyleGAN. Mohl by být vyvinut efektivnější způsob řízení stylingu, ale to necháváme jako předmět budoucí práce.

Stejně jako u architektury Transformer byly vyvinuty tři různé velikosti se stejnými hyperparametry jako dříve. Celkový počet parametrů byl 535 115 pro malou variantu, 1 414 075 pro střední a 3 780 651 pro velkou. Trénování a zpracování dat probíhalo stejným způsobem

jako dříve, s tím rozdílem, že velká varianta byla trénována s velikostí minidávky 96 vzorků kvůli limitaci paměti.

Aby bylo možné FFD vypočítat, musel model generovat sekvence na základě žánrů a desetiletí, takže byly brány z distribuce trénovací sady.

Model	Train perplexita	Test perplexita	Train přesnost	Test přesnost	FFD
ConditionalS	3.548	3.521	64.58%	64.81%	2.998
ConditionalM	2.785	2.774	72.75%	72.99%	2.076
ConditionalL	2.551	2.562	75.14%	75.14%	1.848

Tabulka 3: Výkon Conditional Transformer modelů.

V tabulce 3 vidíme, že zatímco perplexita a přesnost je o něco horší než u sítě Transformer (zejména u menších variant), FFD bylo zlepšeno. To může naznačovat, že způsob, jakým vynucujeme podmínění, je příliš přísný, protože poškozuje predikční schopnosti modelu, zatímco zároveň zlepšuje rozmanitost vzorků.

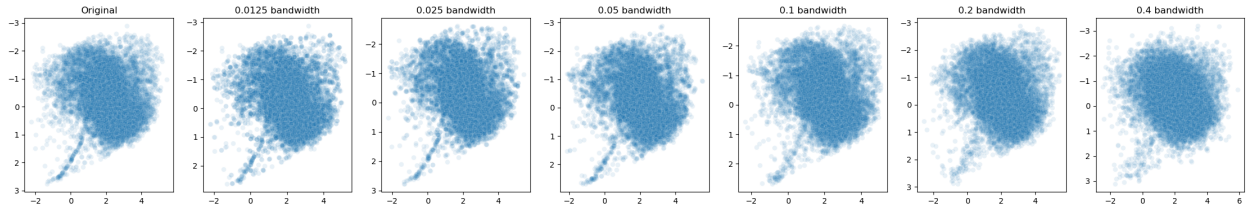
3.5 Style Transformer

Vylepšení FFD pomocí Conditional Transformeru nás motivovalo k tomu, abychom prozkoumali způsoby, jak dále řídit generativní proces vložením více informací o stylu. Byl vyzkoušen adversariální přístup, kdy byla nová sekvence autoregresivně vzorkována a podmíněna náhodným latentním vektorem stylu. Vzhledem k tomu, že vzorkování diskrétních sekvencí je ze své podstaty nediferencovatelné, bylo generování považováno za úlohu zpětnovazebního učení, kde posloupnost předchozích tokenů je stav, rozdělení pravděpodobnosti dalších tokenů je akce a diskriminátor vytváří odměnu. Tato metoda však nebyla účinná, především kvůli výpočetním omezením. Adversariální přístup je navíc často nestabilní a jeho trénování zabere spoustu času i při použití vhodných metod. Podle našich současných znalostí nebyl vyvinut jediný úspěšný stylový Transformer model založený na adversariální metodě pro diskrétní sekvence.

Místo toho, abychom styl získávali přímo nebo přirozeně, vložili jsme výstupy z předtrénovaného extraktoru rysů jako novou podmínku. Není to ideální řešení, neboť může způsobit memorizaci trénovacích dat (overfitting) a ztížit práci s latentním prostorem, ale ostatní potenciální metody ponecháme jako téma budoucí práce. Extraktor funkcí je klasifikátor žánrů a desetiletí založený na Transformeru se 6 vrstvami a stejným d_{model} a n_{heads} jako varianta modelu o stejné velikosti, které odpovídá. Celková struktura Style Transformeru zůstala stejná jako u Conditional Transformeru, pouze se změnila dopředná vrstva, aby se

přizpůsobila novému tvaru stylu, který byl stejný jako d_{model}.

Opět byly vytvořeny a natrénovány tři modely o 541 579, 1 424 715 a 3 796 491 parametrech, stejným způsobem, jaký je popsán výše. Generování sekvencí pro výpočet FFD bylo poměrně složité, protože vektory stylu nemohly být vzorkovány z normálního rozdělení, neboť pocházejí z extraktoru rysů, který vytváří složitý latentní prostor. Místo toho byla distribuce vektorů stylu aproximována pomocí Gaussova kernelového odhadu hustoty (Gaussian KDE) a použité vektory stylu z něj byly vzorkovány. Jedním z klíčových aspektů, pokud jde o KDE, je šířka pásma. Vyšší hodnoty příliš vyhladí rozložení, zatímco nižší způsobí větší shodu s původním rozložením. Jelikož se jedná o úlohu strojového učení, musíme najít rovnováhu, abychom zabránili přeučení a zároveň udrželi distribuci dostatečně blízkou. Na základě vizuální kontroly ve dvou náhodných dimenzích s původní a převzorkovanou sadou bylo vybrána hodnota 0.05 při generování sekvence (viz obrázek 10).



Obrázek 10: Originální rozložení stylu porovnané s převzorkovanými distribucemi.

Tento přístup si vedl o něco lépe než Conditional Transformer co se perplexity týče, zatímco dosáhl o něco horší přesnost (viz tabulka 4). To však neplatilo pro velkou variantu, která z hlediska perplexity a přesnosti dokonce předčila výkon modelu Transformer. FFD bylo výrazně vylepšeno i v případě malé varianty, poráží i velký podmíněný model. Tyto výsledky ukazují, že vložení více informací souvisejících se stylem je užitečné; předpokládáme, že to činí výsledky rozmanitějšími, a proto lépe odpovídají původní distribuci.

Model	Train perplexita	Test perplexita	Train přesnost	Test přesnost	FFD
StyleS	3.495	3.473	62.25%	62.58%	1.728
StyleM	2.581	2.560	72.63%	72.90%	1.139
StyleL	2.252	2.264	76.32%	76.31%	0.824

Tabulka 4: Výkon Style Transformer modelů.

Tato architektura není tak snadno použitelná pro reálný produkt pro generování sekvencí kvůli problémům spojeným se způsobem zpracování stylů. Nicméně, vzhledem k žánru (nebo jakékoli jiné požadované vlastnosti), nová sekvence by mohla být vygenerována pomocí stylu vzorkovaného z Gaussova KDE váženého shodou s touto vlastností (např. podle váhy žánru). Dalším způsobem by bylo podmiňování sekvencí, kde bychom mohli vzít několik referenčních

sekvencí, jejichž styl chceme znovu replikovat, a jako podmínku použít jejich průměrný vektor stylu získaný extraktorem rysů. Problém je však v tom, že náš přístup předvídá pouze na základě jednoho vektoru stylu, a ne na základě celé distribuce stylů; naproti tomu Conditional Transformer používá informace o žánru, které pokrývají větší část prostoru stylu. Stylová interpolace by byla také náročná, neboť rozdělení má složitý tvar (tradičně se používá sférická lineární interpolace na GAN s Gaussovským latentním prostorem, což v našem případě není aplikovatelné). Tato architektura může být také zranitelnější vůči opětovnému vytvoření sekvencí trénovacích dat. Tato omezení ponecháváme jako předmět budoucí práce.

3.6 Lidské hodnocení

Vzhledem k tomu, že popsané metriky nejsou snadno interpretovatelné, rozhodli jsme se použít lidské hodnocení. Náš přístup se snažil změřit, jak obtížné je oklamat lidi, aby uvěřili, že vygenerovaná sekvence je skutečná. Vzhledem k tomu, že nejsou k dispozici žádné aplikace pro průzkum, které by vyhovovaly našim účelům, byla vytvořena jednoduchá webová aplikace. Uživatel byl přivítán vysvětlením projektu a použitých metod zpracování dat. Poté vyplnil sekci o sobě – o své věkové skupině, pohlaví a svých zkušenostech s hudbou. V hlavní části bylo uživateli ukázáno 10 náhodných sekvencí, u kterých měl za úkol uhodnout, zda je každá sekvence skutečná nebo falešná. Měli možnost přehrát sekvenci akordů jako zvukový soubor. Sekvence, které byly uživatelům ukázány, byly náhodně vybrány z 1000 reálných a 1000 generovaných sekvencí (100 pro každý model). Na konci průzkumu měli uživatelé možnost zanechat svou e-mailovou adresu, aby později obdrželi své výsledky o tom, jak úspěšní byli v hádání – abychom zůstali rigorózní, tyto výsledky jim nebyly zaslány před ukončením výzkumu. Aplikace také sledovala čas, který uživatelům trvalo uhodnout, zdali je sekvence reálná.

Průzkum skončil o týden později po distribuci několika různým skupinám. Celkem bylo shromážděno 46 validních odpovědí, což je poměrně malý vzorek, ale i tak se pokusíme z dat vytěžit maximum. 22 respondentů byli muži, 24 ženy, nikdo nezvolil "jiné/raději neodpovím". 5 respondentů bylo ve věkové skupině 10–15 let (horní hranice jsou exkluzivní, např. 15 let a 2 měsíce do této kategorie nespadá), 36 respondentů bylo ve věku 15–20 let, 4 respondenti byli ve skupině 20–30 let a jeden uživatel vybral 50+. Věkové rozdělení není neočekávané, protože tento průzkum byl distribuován převážně studentům středních škol. 14 respondentů mělo 0–1 rok hudební praxe (tj. hraní na nástroj, skládání hudby nebo pracovní zkušenosti), 2 1–3 roky, 5 respondentů 3–5 let, 13 5–10 let, 10 10–20 let a 2 více než 20 let. Průzkum byl v průměru dokončen za 3 minuty a 18 sekund se směrodatnou odchylkou 2 minuty a 33 sekund.

Konečná průměrná přesnost odpovědí respondentů ve všech sekvencích (reálných i generovaných) byla 50.22 %, což není o nic lepší než náhodné hádání. Překvapivě nebyla zjištěna významná korelace mezi časem potřebným k dokončení a přesností respondenta, zkušenosti měly dokonce mírně negativní korelaci s přesností. Věk respondentů měl podobnou tendenci jako zkušenosti. I když tyto efekty mohou s větším počtem respondentů vymizet, je zajímavé je sledovat. Obě pohlaví si vedla přibližně stejně dobře.

Vzhledem k malému počtu odpovědí si můžeme udělat jen hrubou představu o skutečném výkonu. Všimněte si, že i když si někdo může myslet, že místo použití mnoha reálných sekvencí bychom je mohli jednoduše přesměrovat do modelů, destabilizovalo by to očekávané procento reálných a generovaných vzorků. Termín vnímaná reálnost používáme k popisu procenta odpovědí, které byly považovány za reálné. Vzhledem k tomu, že pracujeme s binárními daty, byla použita standardní chyba pro aproximaci spolehlivosti našich zjištění. 95% konfidenční hladinu lze získat vynásobením standardní chyby zhruba 1.96. Obecně byly reálné sekvence považovány za reálnější než falešné, nicméně generované sekvence měly přibližně stejnou hodnotu vnímané reálnosti. Nejnižší skóre získala rekurentní síť. Porovnání různých sítí transformátorů je obtížné kvůli velké nejistotě přibližně 20 % pro 95% konfidenční hladinu.

Model	Hodnocených ukázek	Vnímaná reálnost	Standardní chyba
Reálné (referenční)	233	58%	3%
RecurrentNet	21	38%	11%
TransformerS	23	70%	10%
TransformerM	30	53%	9%
TransformerL	23	61%	10%
ConditionalS	17	65%	12%
ConditionalM	18	61%	11%
ConditionalL	24	54%	10%
StyleS	21	57%	11%
StyleM	23	57%	10%
StyleL	27	63%	9%

Tabulka 5: Výsledky hodnocení lidmi.

Závěrem lze říci, že vzhledem k našim metodám zpracování dat a dalším omezením se zdá, že modely Transformer a jejich varianty generovaly sekvence prakticky nerozeznatelné od reálných, i když by měl být proveden další výzkum s větším počtem účastníků, aby se tato zjištění potvrdila.

3.7 Přehled výkonu modelů

Model	Počet parametrů	Perplexita	Top-1 přesnost	FFD	Vnímaná reálnost
RecurrentNet	376 683	4.851	56.59%	9.290	38 ± 21%
TransformerS	433 419	3.123	69.05%	3.201	70 ± 19%
TransformerM	1 100 715	2.677	73.91%	2.814	53 ± 18%
TransformerL	2 883 915	2.513	76.04%	2.316	61 ± 20%
ConditionalS	535 115	3.521	64.81%	2.998	65 ± 23%
ConditionalM	1 414 075	2.774	72.99%	2.076	61 ± 23%
ConditionalL	3 780 651	2.562	75.14%	1.848	54 ± 20%
StyleS	541 579	3.473	62.58%	1.728	57 ± 21%
StyleM	1 424 715	2.560	72.90%	1.139	57 ± 20%
StyleL	3 796 491	2.264	76.31%	0.824	63 ± 18%

Tabulka 6: Srovnání modelů.

Počet parametrů a výkon modelů je uveden v tabulce 6. Perplexita a přesnost (Top-1) jsou vypočteny na testovací sadě, intervaly spolehlivosti vnímané reálnosti označují 95% konfidenční hladinu.

3.8 Ukázky generovaných vzorků

Níže jsou uvedeny tři ručně nevybírané akordové sekvence generované každým modelem. Byly zkráceny na 24 akordů, aby se vešly do rozumného prostoru. Různé další vygenerované sekvence najdete v repozitáři na GitHubu v tokenizované reprezentaci.

Recurrent Net

- D G D A D G D A D G A D G D G D G D G D G D G D...
- F# G# A#m F# C# G# A#m F# G G# A#m F# G# Fm A#m F# D#m G# C# F# G# C# F# G#...
- D#m7 Badd9 G#m D#m7 C#m7 D#m7 G#m D#m7 Emaj7 D#m7 C#m7 D#m7 Emaj9 D#m7 G#m7 C#m7 Amaj9 D#m7 Badd9 F#m Emaj7 D#m7 Badd9 D#m7...

Transformer S

- B C#m E C#m A C#m E B F#7 B C#m E C#m B G#m C#m E G#m C#m E G#m
- A#m7 D#7 G#maj7 G# A#m7 D#7 G#maj7 A#m D#7 G#maj7 A#m D#7 G#maj7 A#m D#7 G#maj7 A#m D#7 G#maj7 A#m D#7 G#maj7 A#m D#7...
- G C D G Em C D G C G Em C D G C G Em C D G Em Bm G D...

Transformer M

- D# F A# A#maj7 Cm7 F A# A#maj7 Cm7 F A# Gm7 D# A# A#maj7 Cm7 F A# F A# A#maj7 Cm7 F A#...
- C A# C A Dm A7 Gm7 C#aug A7 Dm A# A7 Dm7 A7 Dm C A# C Dm A# C Dm A# A7...
- D#m C#7 F# D#m C#7 F# B F# D#m B F# C# B C# F# D#m C#7 F# B F# C# B F# D#m...

Transformer L

- Bmaj7 D7 Am7 D7 G A#7 Am7 D7 Gmaj7 Em Am7 D7 Gmaj7 Am7 G7 D7 Am7 D7 Gmaj7 C9 Bmaj7 G# Am7 D7...
- Gm Dm Gm Dm Gm Dm Gm Dm Gm Dm Gm Dm Gm Dm Gm Dm Gm
- C# D# G# D# G# D# G# D# G# D#7 G# C# D# G# D# G# D# G# D#7 G# C# D# G# D#...

Conditional Transformer S

- B C#m A B C#m A B C#m A B A B A B C#m A B C#m A B E A B C#m...
- Dm Am A# F Dm Am A# C F Dm Am Gm A# F Dm Am A# F Dm Am A# F Dm Am...
- A A7 D7 G A7 D7 G A7 D7 G A7 D7 G A7 D7 G A7 D7 G A7 D7 G A7 D7...

Conditional Transformer M

- G A D D7 G Gm6 A D G D Em7 A D A D Bm G A D
- C# F# B F# C# F# A#m B F# C# F# G#m B F# C# F# G#m B F# C# F# A#m B F#...
- F#m G#m F#m G#m C#m E F#m G#m F#m G#m C#m E F#m G#m C#m E F#m G#m C#m E F#m G#m C#m E...

Conditional Transformer L

- B E A B E A B E A B E A B E A B E A B E A B E A...
- A#m D#7 D# C# G# A#m D#7 G# A#m D#7 Cm7 D#7 C# G# A#m D#7 G# A#m D#7 G# Fm A#m D#7 Cm7...
- G# C# G# C# G# C# G# C# G# C# G# C# G# C# G# C# G# C# G# C# G# C# G# C#...

Style Transformer S

- G D Bm D# C G Em C Dm G A F D# C#m D Gm C F D# F D# A# A D#...
- G# A#m7 G# D# G# D# A#m7 G# D# G# C# A#m7 D# G# C# D# G# D# C# A#m7 D# G# D# C#...
- A# Dm A# Gm Cm A# D# A# Gm Cm A# A7 G# D# A# Gm Cm A# D# A# A7 D#maj7 D#m D#...

Style Transformer M

- C# G# E F# D# F# G# C# G# E F# C# G# E F# C# G# G#m E F# D#m E F# C#...
- A# Gm Dm C C7 F Dm A# F Dm A# Gm Dm C Dm A# Gm Dm C Dm A# Gm A5 Dm...
- B C# D#m B C# D#m B C# D#m B C# D#m B C# D#m B C# F# C# F# C# F# C# F#...

Style Transformer L

- D G Bm D G Bm D G Bm D Cmaj7 Bm D G D Em Bm D A D Gmaj7 G D Em...
- G#m7 C#m6 A9 F#7 C#m7 F#7 C#m6 Bmaj7 G#m7 C#m6 A9 G#7 C#m7 F#7 C#m6 Bmaj7 G#m7 C#m6 A9 G#7 C#m7 F#7 C#m6 Bmaj7...
- Em A Em D Em A Em Bm C#m F#m Am D Em Bm C#m F#m Am D Em Bm C#m F#m C#m D...

4 WEBOVÁ APLIKACE

4.1 Úvod

Modely hlubokého učení mohou být jen tak užitečné, jak užitečný je kontext, ve kterém se používají, a proto jsme se rozhodli vytvořit webovou aplikaci zaměřenou na naše modely, aby byl výsledek naší práce použitelný pro veřejnost. Hlavní myšlenkou bylo pomoci hudebníkům, zejména těm, kteří začínají svou cestu, skládat krásné akordové postupy tím, že jim model navrhne další akord v kontextu předchozích akordů. Uživatel by viděl jejich sekvenci a dostal návrhy, co dělat dál, což by umožnilo efektivní průzkum prostoru možností.

Podobné produkty již existují, ale obvykle jsou placené, nepraktické na používání nebo jinak omezené. Vytvoření open-source, volně použitelného nástroje poháněného umělou inteligencí by proto mohlo komukoli umožnit jednoduše skládat akordové postupy. Aplikace by také mohla sloužit jako způsob, jak se naučit část hudební teorie, protože akordové postupy a jejich zápis jsou poměrně složité na pochopení.

4.2 Web stack

Next.js 14 [34] byl vybrán jako framework pro vývoj webu. TypeScript byl použit namísto JavaScriptu a Tailwind CSS [35] byl užit jako knihovna pro stylování. Zustand [36] sloužil jako knihovna pro správu stavu, ONNX runtime [37] byl použit ke spuštění modelů umělé inteligence, Tone.js [38] sloužil pro přehrávání zvuku složených akordových postupů.

4.3 Uživatelské rozhraní

Obecná struktura. Uživatelské rozhraní se skládá z několika komponent, včetně časové osy, návrhů, výběru modelu/stylu, nabídky transpozice/importu/exportu a rozbalovací nabídky variant akordů. Klávesové zkratky jsou k dispozici pro většinu funkcí aplikace. Když umístíte kurzor na libovolnou interaktivní část komponenty, zobrazí se, co se stane po kliknutí, a také zkratka pro použití. Stav aplikace, včetně postupu akordů, vybraného modelu a stylu, taktového předznamenání, tempa a výchozích variant, se automaticky uloží lokálně v prohlížeči, takže uživatel o data nepřichází, dokud manuálně nesmaže data webu. Ponoříme se do každé z komponent do hloubky a půjdeme v přirozeném pořadí, jak může uživatel aplikaci prozkoumat.

Časová osa. Srdcem aplikace je časová osa, komponenta, ve které je sestaven akordový postup. Ovládací prvky jsou podobné jako u video editorů, prostřední tlačítko myši lze použít

Obrázek 11: Webová aplikace.

k přiblížení/oddálení a tažením tohoto tlačítka se pohled posune.

Nový akord lze přidat kliknutím na ikonu plus nad časovou osou nebo klávesovou zkratkou `A`. Všechny nově přidané akordy jsou ve výchozím nastavení prázdné (označené otazníkem), lze je v návrzích změnit na jiný akord. Kliknutím na libovolný akord na časové ose jej vyberete, což je signalizováno světlejší barvou. Kliknutím na již vybraný akord výběr zrušíte, případně můžete použít klávesovou zkratku `Esc`. Šipky na klávesnici lze použít k navigaci ve výběru akordů. Pokaždé, když je vybrán akord (buď kliknutím na něj, nebo pomocí kláves se šipkami), je přehrán, aby bylo snazší pochopit, co uživatel skládá. Přidáním nového akordu se buď připojí na konec sekvence, nebo pokud je nějaký akord vybrán, nový akord se vloží hned za vybraný. Po zpracování může obsahovat sekvence maximálně 255 akordů (po sloučení po sobě jdoucích identických akordů a ignorování prázdných akordů).

Vybraný akord lze smazat kliknutím na ikonu koše (zkratkou `Del`). Dobu trvání akordu lze změnit úpravou jeho velikosti – tažením pravého okraje akordu (najeďte myší na pravý okraj akordu, klikněte a pak myš přetáhněte); okraj se přichytí k době. Všechny změny, které jsou v sekvenci provedeny, včetně výběru a zrušení výběru akordů, se zaznamenávají (až 64 kroků do minulosti), můžete je vrátit zpět/obnovit pomocí ikon nalevo od koše nebo klávesovými zkratkami `Ctrl + Z`/`Ctrl + Y`.

Vlevo od ovládacích prvků pro přidání/odstranění/zpět/obnovit jsou ovládací prvky přehrávání. Prostřední ikona spustí nebo pozastaví přehrávání (`Mezerník`). Při přehrávání sekvence je aktuální čas vizualizován modrou pohybující se přehrávací hlavou. Polohu přehrávací hlavy lze změnit kliknutím nebo přetažením na časových dílcích (horní a dolní část

časové osy kolem akordů), a to jak při přehrávání, tak i mimo něj. Jakmile přehrávací hlava dosáhne konce sekvence, přehrávání skončí a hlava přeskočí zpět na začátek sekvence. Ikona metronomu vlevo zapíná a vypíná metronom (`M`). Tempo přehrávání v úderech za minutu lze změnit přes ikonu nastavení vpravo.

Taktové předznamenání (počet dob na takt) postupu akordů lze přizpůsobit kliknutím na předznamenání v levé části časové osy. Když se předznamenání změní, doby trvání akordů zůstanou konstantní. Předznamenání je také vizualizováno dílky v horní a dolní části časové osy, kde o něco větší dílek označuje hranici dvou taktů.

Návrhy. Když je vybrán libovolný akord, jsou k dispozici návrhy pro daný akord na základě jemu předcházejících. Seznam lze posunout dolů a zobrazit další návrhy. Kliknutím na libovolný návrh se vybraný akord nahradí návrhem a zazní nový akord. Navrhované akordy jsou zbarveny od fialové po černou barvu a seřazeny podle pravděpodobnosti předpovězené modelem. Úpadek zbarvení lze ovládat posuvníkem úpadku v pravém horním rohu. Logaritmické zbarvení se používá místo lineárního, aby byly méně pravděpodobné akordy stále viditelné, což poskytuje přirozenější způsob, jak přemýšlet o předpovědích modelu.

Konkrétní akordy lze také hledat z vyhledávacího pole vlevo nahoře; pokud nenajdete požadovaný akord, zkuste povolit *Include variants*, protože daný akord může být variantou jiného základního akordu (např. Am7 a C6 jsou varianty stejného akordu, protože jsou vzájemně svými obraty). Akordy můžete také filtrovat podle použitých tónů pod ikonou klavíru, která otevře virtuální klaviaturu, na které můžete zadávat noty kliknutím na jednotlivé klávesy. Při použití libovolného vyhledávacího dotazu se zobrazí ikona pro vymazání filtrů (dotazy se po kliknutí vymažou).

Načtení návrhů může chvíli trvat, protože model běží přímo v prohlížeči pomocí modulu ONNX runtime. Jakmile je návrh vytvořen, je uložen do mezipaměti pro pozdější použití (do mezipaměti je uloženo až 32 předpovědí, poté jsou první z nich odstraněny, aby se uvolnila paměť).

Výběr modelu a stylu. Z nabídky vlevo nahoře si uživatelé mohou vybrat model, který chtějí použít pro návrhy, kliknutím a výběrem jiné varianty z rozbalovací nabídky. Větší model může vytvářet lepší návrhy za cenu delší doby předpovědi. Je-li vybrán podmíněný model, lze kliknutím na pravou část této komponenty otevřít nabídku stylu (resp. podmínky). Ve výběru stylu jsou zahrnuty dvě karty, žánr a desetiletí. Kliknutím na libovolný prvek stylu z rozbalovací nabídky se změní jeho stav (použít/nepoužít). Lze vybrat více žánrů a desetiletí. Kromě toho lze zadat relativní váhu každého ze stylů. Doporučujeme používat malé celočíselné hodnoty, aby se o nich snadno přemýšlelo. V zákulisí se váhy normalizují tak, aby se v obou případech sečetly do jedné. Vzhledem k tomu, že použitý styl mění předpovědi

modelů, návrhy se aktualizují při každé změně. Žánry jsou seřazeny podle jejich výskytu v datové sadě. Modely Style Transformer nebyly přidány kvůli obtížím, které jsou s nimi spojeny (viz poslední odstavec části 3.5).

Transpozice/Import/Export. Sekvenci akordů lze snadno transponovat pod levou ikonou tohoto menu. Kladné celočíselné hodnoty transponují o daný počet půltónů nahoru, zatímco záporné transponují dolů. Sekvenci lze exportovat do formátu .chseq nebo .mid pod pravou ikonou, což odpovídá vlastnímu formátu (preferovanému pro ukládání/načítání sekvencí, je to jen JSON na pozadí) a MIDI (pro použití v jiném hudebním produkčním softwaru). Importy se provádějí pod prostřední ikonou, opět můžete importovat soubor .chseq nebo .mid. Ruční úprava souboru .chseq může způsobit poškozený stav aplikace po importu, což lze opravit vymazáním dat webu. Importované MIDI soubory by měly být jednostopé, pouze s akordy ve zhuštěném formátu (bez rozkladů, vybrnkávaných akordů a dalších variant), bez dalších hlasů (jako jsou perkuse a melodie). Pokud akord není rozpoznán nebo v MIDI souboru nejsou žádné noty (odpovídající pomlce), bude nahrazen neznámým tokenem (označeným otazníkem).

Varianty akordů. Vzhledem k použitým metodám zpracování dat jsou obraty akordů a dalších hlasů mapovány na stejný token. Aby byla aplikace intuitivnější, může uživatel zvolit variantu libovolného akordu v časové ose a v návrzích. Toto menu lze otevřít kliknutím pravého tlačítka myši na libovolný akord. Vizualizace not, ze kterých se akord skládá, je zobrazena na virtuální klaviatuře a pod ní jsou další varianty akordů. Po kliknutí na kteroukoli jinou variantu se znovu zobrazí a přehraje. Když je toto menu otevřeno z časové osy, nově vybraná varianta může být aplikována buď jednou (pouze na daný akord), nebo na všechny (nahrazením všech stejných akordů touto variantou). Když je otevřen z návrhů, lze jej použít jednou (nahrazení vybraného akordu touto variantou) nebo nastavit jako výchozí (což z něj činí preferovanou variantu v návrzích). Tuto nabídku můžete zavřít pomocí ikony křížku (alternativně `Esc`).

I když může být lákavé použít varianty akordů ke skládání s hlasovým vedením, není to doporučený přístup, protože varianty byly přidány pouze pro intuitivnější vyhledávání akordů a umožnění použití více možných notací. Místo toho by tato aplikace měla být používána pouze pro základní akordové postupy a hlasy by měly být realizovány v jiném hudebním produkčním softwaru (po exportu do formátu MIDI).

4.4 Další aspekty

Modely. Vzhledem k tomu, že modely nejsou příliš výpočetně náročné, byly nasazeny přímo na koncové zařízení. Modely byly převedeny do formátu ONNX a spouštějí se na místě

v prohlížeči pomocí webového modulu ONNX runtime. Aby se data, která model obdrží, zpracovala stejným způsobem, na kterém byla natrénována, identické po sobě jdoucí a neznámé akordy se ignorují. Doba trvání a varianta akordu nemají vliv na předpovědi. Zajištění funkce ONNX v prohlížeči bylo problematické kvůli potřebě umístit balíček webového sestavení do konkrétní statické cesty, což vyžaduje různá nastavení cesty modulu copy webpack pro vývoj a nasazení s Next.js 13+.

Progresivní webová aplikace. Tuto aplikaci lze také nainstalovat na počítač jako PWA.

Ikony. Ikony v aplikaci pocházejí z Font Awesome Icons V6 [39] a Tabler Icons [40]. Ikona transpozice je tvořena dvěma ikonami. Logo aplikace bylo vytvořeno pomocí Inkscape [41] a mělo by představovat hybrid zvukové vlny a digitálního mozku.

Zvuky. Zvuk piana použitý pro přehrávání, stejně jako tikající zvuk metronomu, jsou ze zvukové knihovny Tone.js [42].

Nasazení. Tato aplikace nebyla nasazena online jako součást této práce, nicméně hostovaná verze již existuje. URL lze nalézt v části 7.

Podpora. Tato aplikace je v současné době podporována pouze na počítačích. Byla vytvořena v prohlížeči založeném na Chromiu, takže tato aplikace může být v jiných prohlížečích nestabilní.

5 BUDOUCÍ PRÁCE

Přestože tento projekt přinesl užitečný nástroj, mnohé by se dalo zlepšit. V této části popíšeme některé přístupy, které bychom mohli v budoucnu vyzkoušet.

Jedním z největších omezení této práce je nezohlednění délky akordů. Nyní, když existuje slušná mapa pro získání symbolu akordů pouze na základě tónů, by bylo možné použít jinou datovou sadu. Pokud bychom měli velkou datovou sadu souborů MIDI, bylo by možné extrahovat jejich akordové postupy a zpracovat je tak, aby bylo možné získat akordy a jejich délku. Model by pak mohl být natrénován tak, aby přijímal trvání akordu podobným způsobem jako poziční kódování, predikce trvání akordu by mohla být provedena přidáním dalších tokenů, jednoho pro každou délku každého akordu (jeden výstup pro délku by nestačil, protože do každého kontextu se hodí různé akordy s různým trváním). V konečném produktu by pak uživatel mohl procházet akordy s délkou, případně pravděpodobnosti akordů nezávisle na době trvání sloučením všech možných délek (zkonstruovaných součtem pravděpodobností stejných akordů lišící se délkou), což by fungovalo podobně jako v naší aplikaci, jen by se při předpovědi zohledňovaly doby trvání akordů. Jiný přístup by mohl zahrnout délku akordu jako jiný token, takže by model osciloval mezi předvídáním akordu a tokenu délky.

Dalším aspektem, který není dostatečně zohledněn, jsou různé varianty akordů. Výkonnější nástroj vytrénovaný na větším souboru dat by mohl předvídat pravděpodobnosti pro každou variantu nezávisle. Uživatel takové aplikace by je poté mohl sbalit a zahrnout pod jeden token (stejně jako v naší aplikaci, jen by pod variantami byly vidět pravděpodobnosti) nebo je vidět odděleně. Takový nástroj by teoreticky mohl smysluplně vytvářet hlasové vedení, na rozdíl od naší práce, která se zaměřuje pouze na použité tónové třídy.

Pokud by se podařilo získat dostatečně velký soubor dat s popisy skladeb, mohl by se pak náš podmíněný model použít spolu s (případně předtrénovaným) jazykovým Transformerem, který by umožnil generovat sekvence na základě textových popisů.

Použití stylu je stále problematické a mohla by být vyvinuta generativní metoda učení na základě stylu pro diskrétní sekvence s Transformery. Takový průlom by mohl znamenat revoluci v oblasti generování symbolické hudby, protože tyto problémy by pak bylo možné řešit podobně jako generování obrázků.

Práce na tomto projektu stále pokračuje a zaměřuje se především na zlepšení uživatelského pohodlí a vytvoření výukových materiálů pro zájemce používající tento nástroj. V současné době existuje několik uživatelů, kteří poskytují cennou zpětnou vazbu.

6 ZÁVĚR

V tomto projektu jsme představili ChordSeqAI, nástroj využívající hluboké učení pro generování akordových sekvencí. Přistoupili jsme ke komplexnímu přístupu, počínaje sběrem a analýzou dat, přes vývoj a trénování různých modelů, až po finální použití ve webové aplikaci.

Zkoumání a analýza velké sady akordových postupů odhalily zajímavé vzorce v akordových postupech napříč různými žánry a desetiletími. Naše modely hlubokého učení sahají od základních rekurentních sítí až po složitější architektury Transformerů. Zavedení Conditional a Style Transformerů dále obohatilo možnosti našeho nástroje a umožnilo kontrolovatelnější generování akordových postupů, které jsou k nerozeznání od skutečných. Vývoj webové aplikace byl zásadním krokem k zpřístupnění našeho nástroje širšímu publiku. Integrací našich modelů do intuitivního uživatelského rozhraní jsme otevřeli možnost amatérským i profesionálním hudebníkům prozkoumávat a experimentovat s akordovými postupy.

Závěrem lze říci, že ChordSeqAI představuje pokus zpřístupnit spojení umělé inteligence a hudební kompozice komukoliv. Zjednodušením a obohacením kompozice harmonie, kde je umělec zodpovědný za generativní proces, si ChordSeqAI klade za cíl být spíše podpůrným nástrojem než náhradou, obohacovat kreativitu a nabízet nové perspektivy v hudební kompozici. Je to krok směrem k demokratizaci hudební tvorby, kde hluboké učení v kombinaci s uměním dláždí cestu neprobádaným územím v kreativitě.

7 DALŠÍ ZDROJE

Kód použitý pro tento projekt je k dispozici na GitHubu pod licencí MIT, s výjimkou webové aplikace pro lidské hodnocení, protože byla vytvořena jako rychlá a neupravená alternativa k jiným průzkumným službám.

- Data, modely, vyhodnocování: https://github.com/StudentTraineeCenter/chord-seq-ai
- Kód webové aplikace: https://github.com/PetrIvan/chord-seq-ai-app
- Webová aplikace: https://chordseqai.com

8 POUŽITÁ LITERATURA

1. DAI, Shuqi; JIN, Zeyu; GOMES, Celso; DANNENBERG, Roger B. Controllable deep melody generation via hierarchical music structure representation. *CoRR*. 2021, roč. abs/2109.00663. Dostupné z arXiv: 2109.00663.
2. GAROUFIS, Christos; ZLATINTSI, Athanasia; MARAGOS, Petros. An LSTM-Based Dynamic Chord Progression Generation System for Interactive Music Performance. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, s. 4502–4506. Dostupné z DOI: 10.1109/ICASSP40776.2020.9053992.
3. DALMAZZO, David; DÉGUERNEL, Ken; STURM, Bob L. T. The Chordinator: Chord progression modeling and generation using transformers. In: *International Society for Music Information Retrieval Conference*. 2023. hal-04289026.
4. SHUKLA, Shipra; BANKA, Haider. An Automatic Chord Progression Generator Based On Reinforcement Learning. In: *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2018, s. 55–59. Dostupné z DOI: 10.1109/ICACCI.2018.8554901.
5. LEE, Sang-gil; HWANG, Uiwon; MIN, Seonwoo; YOON, Sungroh. A SeqGAN for Polyphonic Music Generation. *CoRR*. 2017, roč. abs/1710.11418. Dostupné z arXiv: 1710.11418.
6. DONG, Hao-Wen; YANG, Yi-Hsuan. Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation. *CoRR*. 2018, roč. abs/1804.09399. Dostupné z arXiv: 1804.09399.
7. REN, Yi et al. PopMAG: Pop Music Accompaniment Generation. In: *Proceedings of the 28th ACM International Conference on Multimedia*. Seattle, WA, USA: Association for Computing Machinery, 2020, s. 1198–1206. MM '20. ISBN 9781450379885. Dostupné z DOI: 10.1145/3394171.3413721.
8. SIMON, Ian; MORRIS, Dan; BASU, Sumit. MySong: Automatic Accompaniment Generation for Vocal Melodies. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Florence, Italy: Association for Computing Machinery, 2008, s. 725–734. CHI '08. ISBN 9781605580111. Dostupné z DOI: 10.1145/1357054.1357169.
9. JIANG, Nan; JIN, Sheng; DUAN, Zhiyao; ZHANG, Changshui. RL-Duet: Online Music Accompaniment Generation Using Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020, roč. 34, č. 01, s. 710–718. Dostupné z DOI: 10.1609/aaai.v34i01.5413.
10. MUBERT-INC. *Mubert* [https://mubert.com/, https://github.com/MubertAI/Mubert-Text-to-Music]. 2022.
11. FORSGREN, S.; MARTIROS, H. *Riffusion - Stable diffusion for real-time music generation* [https://riffusion.com/about]. 2022.

12. AGOSTINELLI, Andrea et al. *MusicLM: Generating Music From Text*. 2023. Dostupné z arXiv: 2301.11325 [cs.SD].

13. BRIOT, Jean-Pierre; HADJERES, Gaëtan; PACHET, François. Deep Learning Techniques for Music Generation - A Survey. *CoRR*. 2017, roč. abs/1709.01620. Dostupné z arXiv: 1709.01620.

14. JIN, Cong et al. A transformer generative adversarial network for multi-track music generation. *CAAI Transactions on Intelligence Technology*. 2022, roč. 7, č. 3, s. 369–380. Dostupné z DOI: https://doi.org/10.1049/cit2.12065.

15. DHARIWAL, Prafulla et al. *Jukebox: A Generative Model for Music*. 2020. Dostupné z arXiv: 2005.00341 [eess.AS].

16. VERMA, Prateek; CHAFE, Chris. A Generative Model for Raw Audio Using Transformer Architectures. In: *2021 24th International Conference on Digital Audio Effects (DAFx)*. 2021, s. 230–237. Dostupné z DOI: 10.23919/DAFx51585.2021.9768298.

17. MUHAMED, Aashiq et al. Symbolic Music Generation with Transformer-GANs. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021, roč. 35, č. 1, s. 408–417. Dostupné z DOI: 10.1609/aaai.v35i1.16117.

18. HUANG, Cheng-Zhi Anna; DUVENAUD, David; GAJOS, Krzysztof Z. ChordRipple: Recommending Chords to Help Novice Composers Go Beyond the Ordinary. In: *Proceedings of the 21st International Conference on Intelligent User Interfaces*. Sonoma, California, USA: ACM, 2016, s. 241–250. IUI '16. ISBN 978-1-4503-4137-0. Dostupné z DOI: 10.1145/2856767.2856792.

19. DALMAZZO, David; DÉGUERNEL, Ken; STURM, Bob L. T. *The Chordinator: Chord progression modeling and generation using transformers*. 2023. Dostupné také z: https://hal.science/hal-04289026. Late Breaking Demo.

20. OTANI, Noriko; SHIRAKAWA, Shoko; NUMAO, Masayuki. Symbiotic Evolution to Generate Chord Progression Consisting of Four Parts for a Music Composition System. In: PHAM, Duc-Nghia; PARK, Seong-Bae (ed.). *PRICAI 2014: Trends in Artificial Intelligence*. Cham: Springer International Publishing, 2014, s. 849–855. ISBN 978-3-319-13560-1.

21. NAVARRO-CÁCERES, María; CAETANO, Marcelo; BERNARDES, Gilberto; DE CASTRO, Leandro Nunes. ChordAIS: An assistive system for the generation of chord progressions with an artificial immune system. *Swarm and Evolutionary Computation*. 2019, roč. 50, s. 100543. ISSN 2210-6502. Dostupné z DOI: https://doi.org/10.1016/j.swevo.2019.05.012.

22. WHORLEY, Raymond P.; CONKLIN, Darrell. Music Generation from Statistical Models of Harmony. *Journal of New Music Research*. 2016, roč. 45, č. 2, s. 160–183. Dostupné z DOI: 10.1080/09298215.2016.1173708.

23. BURGOYNE, John Ashley; WILD, Jonathan; FUJINAGA, Ichiro. An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In: KLAPURI, Anssi; LEIDER, Colby (ed.). *Proceedings of the 12th International Society for Music Information Retrieval Conference*. Miami, FL, 2011, s. 633–638.

24. ULTIMATE GUITAR. *Ultimate Guitar: Chords, Tabs & Lyrics*. [B.r.]. Dostupné také z: https://www.ultimate-guitar.com. Accessed: September 2023.

25. CORPORATION, Microsoft. *Playwright*. 2023. Ver. 1.37.0. Dostupné také z: https://playwright.dev/.

26. EASY SONG HELP CENTER. *What Parts of Music Can Be Copyrighted? (...and What Can't?)* [https://support.easysong.com/hc/en-us/articles/1500009595681-What-Parts-of-Music-Can-Be-Copyrighted-and-What-Can-t-]. 2023. Accessed: 2023-12-19.

27. CUTHBERT, Michael Scott Asato. *music21*. 2023. Ver. 8.3.0. Dostupné také z: https://web.mit.edu/music21/.

28. BROWN, Tom B. et al. *Language Models are Few-Shot Learners*. 2020. Dostupné z arXiv: 2005.14165 [cs.CL].

29. CHOWDHERY, Aakanksha et al. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*. 2023, roč. 24, č. 240, s. 1–113. Dostupné také z: http://jmlr.org/papers/v24/22-1144.html.

30. TOUVRON, Hugo et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. Dostupné z arXiv: 2307.09288 [cs.CL].

31. WORKSHOP, BigScience et al. *BLOOM: A 176B-Parameter Open-Access Multilingual Language Model*. 2023. Dostupné z arXiv: 2211.05100 [cs.CL].

32. VASWANI, Ashish et al. Attention is All you Need. In: GUYON, I. et al. (ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, sv. 30. Dostupné také z: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

33. KARRAS, Tero; LAINE, Samuli; AILA, Timo. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2019. Dostupné z arXiv: 1812.04948 [cs.NE].

34. VERCEL INC. *Next.js*. 2023. Ver. 14.0.4. Dostupné také z: https://nextjs.org/.

35. WATHAN, Adam; REININK, Jonathan; HEMPHILL, David; SCHOGER, Steve. *Tailwind CSS*. [B.r.].

36. POIMANDRES. *Zustand*. 2023. Ver. 4.4.7. Dostupné také z: https://github.com/pmndrs/zustand.

37. MICROSOFT CORPORATION. *ONNX Runtime*. 2023. Ver. 1.16.3. Dostupné také z: https://onnxruntime.ai/.

38. YOTAM MANN. *Tone.js*. 2023. Ver. 14.7.77. Dostupné také z: https://tonejs.github.io/.

39. FONTICONS, Inc. *Font Awesome Icons V6*. 2023. Dostupné také z: https://fontawesome.com.

40. CODECALM.NET. *Tabler Icons*. 2023. Ver. 2.44.0. Dostupné také z: https://tabler.io/icons. Accessed: 2023-12-27.

41. THE INKSCAPE TEAM. *Inkscape*. 2022. Ver. 1.2.2. Dostupné také z: `https://inkscape.org`.

42. TONEJS. *Audio files used in Tone.js examples*. 2023. Dostupné také z: `https://github.com/Tonejs/audio`.

Seznam obrázků

Seznam tabulek