VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

AUTOMATICKÉ GENEROVÁNÍ HARMONIE

AUTOMATIC HARMONY GENERATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. MARTIN BOBČÍK**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. ZDENĚK VAŠÍČEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Bobčík Martin, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Informační systémy a databáze
Název: **Automatické generování harmonie**
Automatic Harmony Generation
Kategorie: Umělá inteligence
Zadání:

1. Seznamte se s problematikou generování harmonie na základě znalosti melodie, principem a využitím neuronových sítí v této úloze.
2. Navrhněte systém, který bude umožňovat automatické generování harmonie pro melodii zadanou v některém ze standardních formátů (např. MIDI).
3. Zpracujte studii na témata uvedená v předchozích dvou bodech zadání.
4. Implementujte navržený systém ve vhodném programovacím jazyce (např. Python) s využitím vhodného frameworku pro neuronové sítě (např. TensorFlow, Caffe, apod).
5. Za pomoci vhodné datové sady vyhodnoťte vlastnosti navrženého řešení.
6. Diskutujte dosažené výsledky a možnosti dalšího rozšíření.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Vašíček Zdeněk, doc. Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 30. července 2021
Datum schválení: 29. června 2021

Abstrakt

Cílem této diplomové práce je studium problematiky generování harmonie na základě znalosti melodie a navržení systému, který tuto činnost smysluplně automatizuje. V práci je popsán základ hudební nauky pro toto téma a předchozí a jiné přístupy k této problematice. Dále je popsáno strojové učení, neuronové sítě a rekurentní neuronové sítě. Je nastíněn návrh systému, postup jeho zprovoznění a použití. Se systémem byly provedeny čtyři experimenty. Harmonizace krátkých melodií uspokojivé nebyly. Nicméně harmonizace delších melodií vykazovaly obecně lepší výsledky. Jako možný důvod se jeví relativně malá použitá neuronová síť v systému.

Abstract

Goal of this master thesis is to study harmonization based on knowledge of given melody and to design a system which will meaningfully automate this activity. In the work there is covered basics of music theory needed for this topic and previous other approaches to this problematic. There is also covered machine learning, neural networks and recurrent neural networks. In the end, there is outlined design of the system, how to make it work and how to use it. Four experiments were executed with the system. Harmonization of the short melodies were unpleasant. Harmonization of longer melodies were overall more successful though. A possible cause might be relatively small used neural network of the system.

Klíčová slova

hudba, melodie, harmonie, harmonizace, strojové učení, hluboké učení, neuronové sítě, rekurentní neuronové sítě, automatická harmonizace, Google Magenta

Keywords

music, melody, harmony, harmonization, machine learning, deep learning, Neural Networks, Recurrent Neural Networks, automatic harmonization, Google Magenta

Citace

BOBČÍK, Martin. *Automatické generování harmonie*. Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Zdeněk Vašíček, Ph.D.

Automatické generování harmonie

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doc. Ing. Zdeňka Vašíčka Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....

Martin Bobčík

1. srpna 2021

Obsah

Kapitola 1

Úvod

V dávných dobách byla hudba pouze monofonní, což znamená, že se skládala ze samostatných melodií bez doprovodu. Až v průběhu středověku byla vynalezena harmonie. Tím rozumíme skupinu tónů, které zní všechny ve stejnou dobu. Použitím harmonie získáme bohatší a barevnější hudbu. Rozšiřování melodií o akordy, souzvuky, je umělecký proces, a po muzikantovi či skladateli vyžaduje znalost přechodů akordů a harmonie tónů. Nabytí těchto znalostí je jedna věc, avšak jejich uvedení do praxe může být pro laika nebo hudebníka bez dostatečných zkušeností velmi obtížné.

Tématem této práce je vytvořit nástroj, který k známé melodii doplní vhodný harmonický doprovod. Tento nástroj pak může být užitečný právě pro hudební začátečníky, kteří si nejsou svými schopnostmi jistí. Dále může být využit v procesu skládání plnohodnotné hudby počítačem. V takovém případě by navrhovaný systém následoval za generátorem melodie a jeho výsledek doplňoval. Skládání hudby, tedy i harmonie, je umění a vyžaduje hudební cit, proto podobné systémy bývají zatíženy určitou chybou, popřípadě může být jejich výtvor poněkud nudný oproti profesionálnímu složení.

Cílem práce je seznámit se s problematikou automatické harmonizace na základě znalosti melodie, s principem a využitím strojového učení v této úloze, a to zejména s neuronovými sítěmi. Dalším bodem je navržení systému, který automatické generování harmonie umožní. Poté implementovat daný systém s pomocí vhodného frameworku. Implementovaný systém je potřeba otestovat na vhodné datové sadě. Ta obsahuje přepis rozličných písniček ze serveru hooktheory.com do formátu MIDI. Na závěr budou diskutovány výsledky testování a možné pokračování této práce.

1.1 Struktura práce

Tato práce je členěna následovně. V první kapitole této práce se nachází přehled hudební teorie. Jsou představeny základní kameny hudby – tóny, jejich soustavy a alterace. Následuje část věnovaná notopisu, tedy přesnému zápisu hudby speciálními znaky.

Následuje teoretická rešerše strojového učení, tedy disciplíny, na níž stojí správnost automatizace. Nejprve je představeno strojové učení jako takové, potom je popis zaměřen na ty oblasti, které jsou v této práci využity. Postupně jsou rozebrány neuronové sítě a neurony, rekurentní neuronové sítě a jejich LSTM architektura.

Čtvrtá kapitola obsahuje popis předchozích přístupů jiných autorů k této problematice. Mezi nimi lze nalézt například model založený na shodě šablon, jenž porovnává části vstupní melodie s těmi, které zná. Další přístup je založen na skrytých Markovových modelech, které

odhadují nejpravděpodobnější posloupnost akordů k dané melodii. Na neuronových sítích jsou pak založeny modely firmy Google, které jsou představeny jako poslední v této kapitole. Jeden z těchto je použit v této práci.

Pátá kapitola popisuje použitou datovou sadu. Nejprve je prozkoumán jeden z formátů pro ukládání hudby. Je představeno, jak použití tohoto formátu, tak jeho vnitřní struktura. Dále se kapitola věnuje poskytnuté databázi. Ta obsahuje hudbu v XML souborech. Tyto jsou prozkoumány, a následně je stanoven postup jejich konverze na vhodnější formát.

Šestá kapitola se věnuje návrhu systému, který umožňuje automatické generování harmonie na základě vstupní melodie. Popisuje použitý model a také to, jak budou vypadat vstupní a výstupní data.

V další kapitole je tento systém představen více z blízka. V první části je popsán postup instalace, poté příprava dat pro použití v procesu trénování modelu. Tento proces je popsán záhy. Poslední část dává návod, jak použít tento systém pro generování nových skladeb.

Osmá kapitola obsahuje jednotlivé experimenty, které budou s modelem prováděny. Nejprve má model za úkol harmonizovat jednu z krátkých skladeb, které se nacházela v trénovací sadě. Druhá melodie je podobná té první, avšak v trénovací sadě nebyla. Poslední skladbou, se kterou bude experimentováno, je skladba čtyřikrát delší než předchozí dvě.

V poslední kapitole před závěrem jsou diskutovány dosažené výsledky. Jsou představeny některé problémy použitého systému společně s několika návrhy, jak by šly řešit. V druhé části jsou nastíněny možné rozšíření a pokračování této práce.

Kapitola 2

Hudební teorie

Hudební dílo tvoří několik částí, přičemž různé hudební žánry kladou důraz na jiný prvek. Prvním z nich je melodie, uspořádaná sekvence jednotlivých tónů tak, aby šla zahrát, nebo zazpívat. Melodie vyjadřuje hlavní hudební myšlenku skladby. Přidáním více souznějících tónů dostáváme harmonii – vertikální složku hudby. Rytmus dodává hudbě dynamiku střídáním dlouhých a krátkých, přízvučných a nepřízvučných tónů.

Nejen pro harmonizaci melodie je potřeba znát hudební nauku. V této kapitole budou nastíněny základy a pojmy hudební nauky, tedy co to jsou tóny, noty, jejich vlastnosti a stupnice. Dále bude rozdělena melodie a harmonie a popsány akordy a tóniny. Text této kapitoly čerpá z [36], [7] a [22].

2.1 Tóny

Chvěním těles, které rozechvívají okolní vzduch, vzniká zvuk, tedy to, co slyšíme. Nepravidelným chvěním vznikají hluky. Naopak zvuky vznikající pravidelným kmitáním tělesa nazýváme tóny. V hudbě jsou především využívány právě tóny. Ty mají čtyři základní vlastnosti.

Podle různé doby chvění pružného tělesa rozlišujeme délku tónu. Další vlastností je síla, která je dána vzdáleností krajních bodů, mezi nimiž těleso kmitá. Tato vzdálenost se také označuje rozkmit nebo amplituda. Velikost amplitudy je přímo úměrná síle, hlasitosti tónu. Podle původu rozlišujeme barvu, někdy také témbr tónu. Barva závisí na počtu znějících alikvotních tónů, které se rozezní současně s hlavním hraným tónem. To je závislé na materiálu chvějícího se tělesa. Pro představu, lze rozeznat, zda zní klavír, housle, trumpeta či mužský nebo ženský hlas, navzdory tomu, že všechny mají stejnou výšku. Rozmanitou barvitost mají především velké orchestry. Výšku tónu určujeme podle frekvence chvění čili počtu kmitů za vteřinu. Čím vyšší frekvence, tím vyšší tón a čím nižší frekvence, tím hlubší tón.

2.2 Soustava a jména tónů

Tónová soustava je přehledné uspořádání všech tónů užívaných v hudbě. Z vlastností tónů, vyjmenovaných výše, tónová soustava bere v úvahu pouze výšky tónu. Současnou tónovou soustavu tvoří sedm základních tónů, které byly v minulosti označeny písmeny obyčejné abecedy: *a, b, c, d, e, f, g.* Později se jako počátek hudební abecedy určilo písmeno *c* a písmeno *b* se nahradilo písmenem *h.* Tím vzniká nynější hudební abeceda *c, d, e, f, g, a, h.*

C_2	D_2	E_2	F_2	G_2	A_2	H_2	Subkontra oktáva
C_1	D_1	E_1	F_1	G_1	A_1	H_1	Kontra oktáva
C	D	E	F	G	A	H	Velká oktáva
c	d	e	f	g	a	h	Malá oktáva
c^1	d^1	e^1	f^1	g^1	a^1	h^1	Jednočárková oktáva
c^2	d^2	e^2	f^2	g^2	a^2	h^2	Dvoučárková oktáva
c^3	d^3	e^3	f^3	g^3	a^3	h^3	Tříčárková oktáva
c^4	d^4	e^4	f^4	g^4	a^4	h^4	Čtyřčárková oktáva

Tabulka 2.1: Značení tónů a názvy jednotlivých oktáv.

Těchto sedm tónů se pravidelně opakuje ve vyšších a nižších polohách. Při postupu od výchozího c k nejbližšímu následujícímu, respektive předchozímu, c nacházíme osm stupňů (včetně obou c). Vzdálenost mezi dvěma tóny stejného jména se proto nazývá oktáva (z latinského octo - osm). Tónová soustava má oktáv devět. Jelikož se základní tóny opakují stejnými jmény v různé výši, jsou tyto oktávy dále pojmenovány. Díky tomu má každý jednotlivý tón nejen svůj vlastní název, ale také ustálené označení (viz tabulka 2.1). Například tón e v dvoučárkové oktávě se nazývá dvoučárkové e a značí se buď e^2, nebo e".

Kromě označení tónů písmeny se lze také setkat se solmizačními slabikami do, re, mi, fa, sol, la, si, které v 10. století zavedl italský mnich Guido z Arezza.

2.3 Alterace

Každý ze základních tónů tónové soustavy můžeme jednou nebo dvakrát snížit nebo zvýšit. Takto zvýšené a snížené tóny se souhrnně nazývají odvozené nebo alternované. Alterace je pak shrnující název pro zvyšování a snižování tónů. Zvýšení tónu značíme příponou -is v jeho názvu. Naopak snížení označujeme příponou -es. Existují ovšem různé výjimky. Například snížením tónů e a a získáme es, respektive as. Pro snížené h se, místo hes, používá název b.

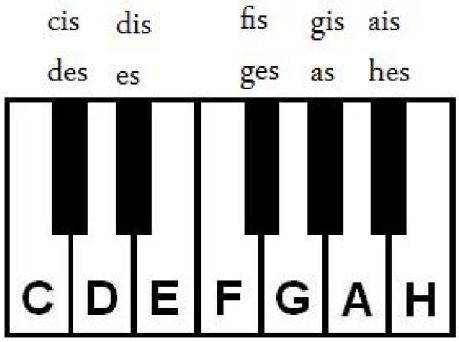
Alterace posunuje tón vždy právě o půltón, což je nejmenší vzdálenost mezi dvěma tóny užívaná v naší hudbě. Celý tón je pak tvořen dvěma půltóny. V rozmezí oktávy se nachází dva půltóny (e-f, h-c), a pět celých tónů. Dohromady oktáva sestává z dvanácti půltónů. Přehledné uspořádání tónů a půltónů v oktávě lze vidět na klaviaturách klávesových nástrojů (viz obrázek 2.1). Základní tóny leží na bílých klávesách, jejich alterace pak na černých.

Lze si všimnout tónů, které mají stejnou výšku, ale různá jména. Těmto tónům říkáme enharmonické. Enharmonická záměna je pak nahrazení tónu tónem stejné výšky, ale jiného jména.

2.4 Rytmus

V melodii hudebních skladeb se málokdy vyskytují tóny a noty pouze jediného druhu. Zpravidla se střídají dlouhé a krátké tóny, přízvučné a nepřízvučné. Kolem přízvučného (silněji hraného) tónu se vyskytuje skupina delších a kratších tónů. Tyto tvoří rytmický útvar a opakování rytmických útvarů udává rytmus.

Rytmus je patrný především v hudbě určené pro tanec. Mezi oblíbené tance s nápadnými rytmy patří například česká polka, valčík a moderní tango. Některé tance a rytmy jsou spíše regionálního charakteru, mezi ně řadíme polské tance polonéza a mazurka, maďarský



Obrázek 2.1: Přehledné uspořádání tónů a půltónů v oktávě.

čardáš, či španělské bolero. Složitými rytmy pak vynikají písně moravské a slovenské. Rytmus je podle některých [1] nejpodstatnější složkou hudby. To je patrné především v hudbě národů na nízkém stupni kultury, kde rytmy určené bicími nástroji, doplněné tleskáním a podupáváním, zaujímají přední postavení tamní hudby.

2.5 Stupnice a tónina

Stupnice je postupná řada tónů v rozsahu jedné oktávy. Při poslechu je zřejmé, že jednotlivé stupně mezi sebou nemají konstantní vzdálenost. Některé dvojice dělí půltón čili malá sekunda, jiné větší krok (celý tón), velká sekunda. Ta je rovna vzdálenosti dvou půltónů. Vzdálenosti jednotlivých stupňů jsou dány pravidly. Stupnice je možné rozdělit podle vzdáleností do několika kategorií.

1. diatonické
   (a) staré (církevní)
   (b) moderní (durové, mollové)
2. chromatické
   (a) chromatické
   (b) alterované
3. exotické
   (a) cikánská
   (b) pentatonická (čínská)
   (c) celotónová

V diatonických stupnicích jsou od sebe stupně vzdáleny jak o celé tóny, tak o půltóny. V chromatických pak pouze o půltóny. Exotické stupnice se od předchozích liší zvláštním uspořádáním stupňů.

Stupnici lze vytvořit od libovolného tónu. V praxi se však používají jen ty, v nichž se uplatňují odvozené tóny nanejvýš dvakrát zvýšené nebo snížené. Výchozí tón stupnice je

[1] „Na počátku byl rytmus“ – Hans Bülow

jejím základním tónem a podle něj se stupnice jmenuje. Například stupnice G dur začíná tónem *g*, fis moll tónem *fis*. Rozdělení dur nebo moll se určuje podle vzdáleností mezi jednotlivými stupni. Durové stupnice označujeme velkým písmenem, mollové malým. Jednotlivé stupně stupnice se značí římskými číslicemi. Mimo to má každý stupeň svůj funkční název:

I tónika

II supertónika (super = nad) nebo střídavá dominanta

III vrchní medianta

IV subdominanta (spodní dominanta)

V dominanta (dominující, vládnoucí tón)

VI spodní medianta

VII citlivý tón - tíhne k půltónovému pokračování k tónice

Tóniny a stupnice mají stejné základní tóny a od toho také jména. Opět, například pokud je tónina G dur, pak jejím základním tónem, tónikou je tón *g*. Tónina je volné pořadí tónů stupnice tvořící hudbu. Není nutné, aby byly použity všechny tóny stupnice, ale jejich uspořádání musí dát vyniknout tónice. Tónika se určuje sluchem pomocí tonálního cítění. Velmi často melodie tónikou začíná nebo se k ní vrací. V závěru melodie bývá tónika uplatňována nejčastěji. V takovém případě působí melodie přesvědčivě uzavřena a posluchač necítí potřebu dalšího pokračování.

2.5.1 Durové stupnice

Základní durovou stupnicí je C dur (c, d, e, f, g, a, h, c). Durové, neboli tvrdé stupnice jsou charakteristické III. stupněm, který s I. stupněm tvoří interval velké tercie. Jinými slovy, první a třetí tón stupnice jsou vzdáleny dva celé tóny. Vzdálenost mezi jednotlivými stupni durové stupnice jsou dva celé tóny, půltón, tři celé tóny, půltón.

$$1 - 1 - 1/2 - 1 - 1 - 1 - 1/2$$

Půltóny jsou tedy mezi III.–IV. a VII.–VIII. stupněm. Pro stupnice začínající od jiného základního tónu než *c* je potřeba pomocí posuvek (viz část 2.6.5:Posuvky) upravit vzdálenosti mezi jednotlivými stupni tak, aby byly intervaly uspořádány jako ve stupnici C dur. Takové stupnice se jmenují transponované, je jich dohromady 14. Sedm s křížky a sedm s béčky.

Při tvoření transponovaných durových stupnic s křížky se začíná od základní stupnice C dur (bez křížků i béček). Následující stupnice (s n křížky) začíná od pátého stupně té předchozí (s $n-1$ křížky). Od tohoto tónu se opíše předchozí stupnice a předposlední VII. stupeň se zvýší jedním křížkem. Nová stupnice má tedy o jeden křížek navíc. Takto mohou vzniknout v pořadí následující stupnice: G dur (jeden křížek), D dur, A dur, E dur, H dur, Fis dur a Cis dur (sedm křížků). Křížky v předznamenání stupnic jsou v tomto pořadí: fis, cis, gis, dis, ais, eis, his. Tento způsob tvoření stupnic se nazývá *postup v kvintovém kruhu.*

Durové stupnice s béčky se vytvářejí podobně. Opět se začíná od C dur, ale nyní se nová, následující stupnice (s n béčky) opíše od čtvrtého stupně předchozí (s $n-1$ béčky) a v nové stupnici se sníží jedním béčkem čtvrtý tón. Tímto způsobem vznikají stupnice:

F dur (jedno bé), B dur, Es dur, As dur, Des dur, Ges dur, Ces dur (sedm bé). Béčka v předznamenání jsou v tomto pořadí: b, es, as, des, ges, ces, fes. Způsob tvoření stupnic s béčky se nazývá *postup v kvartovém kruhu.*

2.5.2 Mollové stupnice

Každá durová stupnice má svou paralelní mollovou, měkkou stupnici. Ty mají při poslechu poněkud smutnější, tesklivější ráz. Mollové stupnice se tvoří od VI. stupně durové stupnice. Intervaly mezi stupni zůstávají nezměněny. Například od stupnice C dur je odvozena základní mollová stupnice *a* moll. Vzdálenosti mezi stupni jsou $1 - 1/2 - 1 - 1 - 1/2 - 1 - 1$. Stejně jako u durových stupnic lze odvozovat jiné, následující stupnice s křížky a béčky. Následující stupnice s křížkem začíná na V. stupni a zvyšuje se II. stupeň. Takto vzniknou stupnice e moll (jeden křížek), h moll, fis moll, cis moll, gis moll, dis moll a ais moll (sedm křížků). Nová stupnice s bé začíná na IV. stupni a sníží se VI. stupeň. Postupně vzniknou stupnice d moll (jedno bé), g moll, c moll, f moll, b moll, es moll, as moll (sedm bé).

2.5.3 Akordy

Akord je souzvuk alespoň tří různých tónů. Pro tóny akordů platí určitá pravidla. Pokud současně zní jiné tóny, pak spolu neladí. Vztahem jednotlivých souzvuků se zabývá harmonie. Při souznění všech tónů akordu najednou se jedná o harmonický akord. Pokud jsou tóny zahrány po sobě, pak jde o melodický rozklad akordu. Akordy lze třídit podle různých hledisek.

- Podle počtu tónů: trojzvuky, čtyřzvuky, pětizvuky, vícezvuky
- Podle stavby: složené z tercií, z kvart
- Podle zvukového působení: konsonantní, disonantní
- Podle tvaru: základní, obraty akordu

Tercie je interval dvou tónů a dělí se na malou a velkou tercii. Velká tercie je interval mezi prvním a třetím tónem durové stupnice a má tedy čtyři půltóny. Malá tercie je také interval mezi prvním a třetím tónem, nyní ale mollové stupnice. Má tedy pouze tři půltóny.

Nejčastěji používané akordy jsou ty složené právě z tercií. Pro vznik akordu je potřeba vedle sebe položit alespoň dvě tercie, které dohromady tvoří kvintakord, kde interval mezi prvním a posledním tónem tvoří kvintu. Přidáním další tercie vzniká septakord, ze čtyř za sebou pak nónový akord. Změnou pořadí tónů akordu (například posunutím základního tónu o oktávu) vznikne obrat akordu.

Kvintakordy se tedy tvoří z durových nebo mollových stupnic. Jakost tercií v akordu dává jeho druh. Durový akord se skládá ze spodní velké tercie a z horní malé. Mollový ze spodní malé a horní velké. Dále je možné použití dvou stejně velkých tercií. Tak vzniká zvětšený nebo zmenšený kvintakord složený ze dvou velkých, respektive malých tercií. Pro akordy se vžila stejná pravidla jako pro stupnice, respektive tóniny. Durové se zapisují velkým písmenem, mollové malým. Tak jako stupnice, také durové akordy znějí vesele. Mollové pak smutně, snížené až stísněně a zvětšené ostře, jasně. Příkladem může být akord C dur složený z tónů *c-e-g* nebo a moll z tónů *a-c-e*.

Kvintakordy dur a moll jsou konsonantní. Svým zněním plně uspokojují posluchače a nevyžadují dalšího postupu (rozvedení hudby). Naopak zvětšené nebo zmenšené jsou disonantní a je zvykem je dále rozvést (pokračovat) jiným, konsonantním akordem.

2.5.4 Funkce akordů

Harmonie se, kromě stavby akordů, zabývá také jejich uplatněním v hudbě. Základním pojmem je harmonická funkce akordu. Tedy význam akordu ve skladbě ve vztahu k její hlavní tónině. Harmonie úzce souvisí s tóninou. V durových i mollových tóninách existují tři základní harmonické funkce: tónika, dominanta, subdominanta. Tónika je kvintakord postavený na I. stupni, dominanta na V. a subdominanta na IV. stupni. Za povšimnutí stojí, že mají stejné názvy jako některé stupně (viz část 2.5). Na každém stupni durové, popřípadě mollové stupnice je totiž možné postavit kvintakord. Každá stupnice je tak vybavena sedmi kvintakordy (osmý je stejný jako první). Akordy na vedlejších stupních (II., III., VI., VII.) vyjadřují také tyto harmonické funkce, ale jiným způsobem. Pomocí nich je možné akordy hlavních stupňů zastupovat a tímto způsobem obohatit harmonický doprovod písně.

Hlavní akordy mají stejný tónorod jako tónina. V durové tónině jsou durové, v přirozené mollové jsou mollové. Avšak někdy se v mollové tónině využívá harmonická mollová stupnice, která má zvýšený VII. stupeň. U takových stupnic je pak dominanta durová.

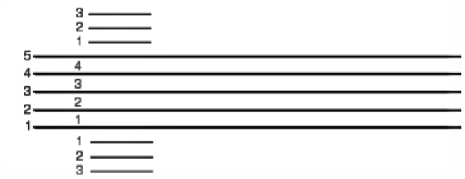
Funkčnost akordů vyjadřuje snahu akordu po pohybu, jeho pohybovou energii. Funkční značky (tónika, dominanta, subdominanta, ...) vyjadřují smysl a celkový význam akordu. Například, pokud po přehrání stupnice C dur zazní její tónický kvintakord, pak má posluchač dojem klidu. Ovšem při zaznění dominantního kvintakordu je cítit snaha po dalším postupu. Člověk má dojem, jako by tóny dominanty chtěly stoupat vzhůru. Pro uklidnění je po dominantě potřeba zahrát tóniku. Ta je akordem klidu, harmonickým centrem tóniny. Ve většině písní také slouží jako přesvědčivý závěr. Naopak dominanta, protože obsahuje citlivý tón stoupající, je akordem pohybu a snaží se stoupat k tónice. Subdominanta je také akordem pohybu a má snahu je od tóniky odklonit k dominantě. Pokud je subdominanta mollová, pak také obsahuje citlivý tón, nyní ale klesající. Tyto tři hlavní kvintakordy velmi často stačí k harmonizaci jednoduchých a krátkých skladeb. Harmonizace moderních či více uměleckých skladeb využívá mimo jiné také kvartové akordy, nepravidelné souzvuky, akordy zhuštěné sekundami (interval mezi I. a II. stupněm) nebo kombinace tonálně uvolněných akordů, případně až atonálních. Chvilková disharmonie skladby také nemusí být na škodu, jedná-li se o umělecký záměr skladatele.

2.6 Notopis

Alfabetické znaky nejsou pro značení tónů dostatečné, protože popisují pouze výšku tónu. Pro hru na hudební nástroj nebo skládání hudby je potřeba specifikovat také zbývající vlastnosti. Po staletí trvajícím vývoji se ustálil a používá se notopis.

2.6.1 Notová osnova

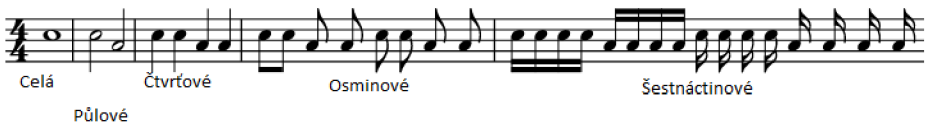
Základem jsou noty a notová osnova, kterou tvoří pět linek a čtyři mezery. Noty se zapisují na linky i do mezer. Takto lze zapsat pouze devět not. Notovou osnovu lze však rozšířit pomocnými linkami nad, popřípadě pod notovou osnovou. Pomocné linky se zapisují pouze tak dlouhé, jak je nezbytně nutné. Pro snadnou orientaci v osnově se v praxi počítají linky a mezery směrem nahoru (s výjimkou pomocných linek pod osnovou). Linky a mezery se počítají zvlášť. První mezera je mezi první a druhou linkou.



Obrázek 2.2: Číslování notové osnovy.

2.6.2 Noty

Samotné noty jsou písemné značky pro tóny. Tvar noty vyjadřuje délku noty a pozice v notové osnově její výšku. Nota se skládá z několika částí. Hlavička určuje pozici noty a může být buď vyplněná, nebo nevyplněná . Nožka opět buď může být, anebo nemusí. Pokud nota nožku má, pak je, u not na třetí, prostřední lince a vyšších, psána směrem dolů a naopak u nižších not směrem nahoru. I toto pravidlo má ovšem nějaké výjimky. Poslední částí noty je praporec, těch může mít nota nula až čtyři. Pokud je za sebou víc not s praporci, lze je spojit a praporce nahradit trámci. V hudbě se používají tyto základní noty:

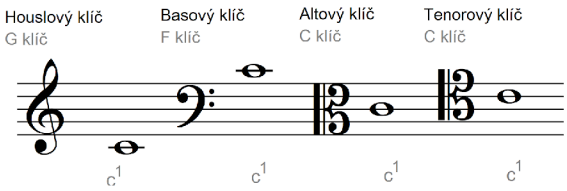


Obrázek 2.3: Noty ve čtyř čtvrťovém taktu.

Podle názvů not je zřejmé, že nota celá má délku dvou půlových not, čtyř čtvrtinových, osmi osminových a tak dále. Nejbližší nižší nota má délku poloviční a nejbližší vyšší dvojnásobnou.

2.6.3 Klíče

Jména (výšky) not v osnově jsou určeny takzvaným klíčem, který se píše vždy na začátek každého řádku. Klíčem se označuje pozice a jméno jedné noty a od ní se pojmenují ostatní v pořadí, přičemž pojmenování se řídí hudební abecedou. Tvary klíčů se vyvinuly z písmen not, jejichž linku označují. V současné době se používají čtyři klíče (viz obrázek 2.4), a to G-klíč (houslový), F-klíč (basový) a dva C-klíče (altový a tenorový).



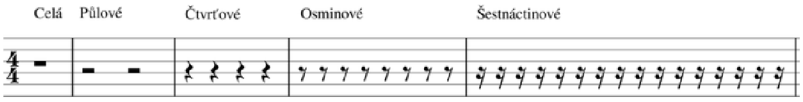
Obrázek 2.4: Hudební klíče s příslušnou pozicí noty c^1.

Houslový klíč označuje notu g^1 na druhé lince. Basový určuje malé f na čtvrté lince, altový c^1 na třetí a tenorový také c^1 na čtvrté lince. Basový klíč je nejvhodnější pro notaci hlubokých tónů a houslový k notaci vysokých. Tyto dva klíče se používají nejčastěji. Pro

orientaci v těchto klíčích je dobré vědět, že tón c^1 se v houslovém klíči nachází na první spodní pomocné lince a v basovém se nachází na první horní pomocné lince.

2.6.4 Pomlky

Kromě značení tónů je v hudbě potřeba zaznačit také odmlky, tedy místa, kdy má být ticho. Ticho se značí zvláštními značkami, pomlkami v notovém zápise a mají stejné hodnoty jako noty. Délka pomlky je opět, jako u not, dána tvarem (viz obrázek 2.5). Celá pomlka značí pomlku na jakýkoliv celý takt.



Obrázek 2.5: Pomlky ve čtyř čtvrťovém taktu.

2.6.5 Posuvky

Samotné noty značí pouze základní tóny. Aby nota značila zvýšený tón, je potřeba před ni (na její linku nebo mezeru) napsat křížek (♯). Je-li před notou béčko (♭), pak nota značí tón snížený. Posuvky lze psát buď přímo před notu, kterou chceme alterovat, nebo na začátek každého řádku, pak předznamenává tóninu. Platnost posuvky před notou je do konce taktu. Je možné ukončit účinnost dříve, odrážkou (♮). Zde ovšem platí, že pokud byla posuvka na začátku řádku, odrážka platí do konce taktu.

2.6.6 Takty

Noty a pomlky se řadí do krátkých úseků, ze kterých se skládá celá hudební skladba. Tyto úseky se nazývají takty. V notové osnově jsou od sebe odděleny tenkými svislými čarami. Pokud je skladba pro více nástrojů a je notována na více řádků, pak protíná taktová čára všechny řádky. Na konci skladby je taktová čára doplněna ještě jednou silnější čarou. Takty se dělí na doby – stejně dlouhé časové úseky. Podle počtu dob jsou takty dvoudobé až třídobé. Doby mohou být půlové, osminové, nejčastěji však čtvrťové. První doba je vždy přízvučná, hraje se silněji. Ostatní doby jsou hrány bez přízvuku, pokud nejsou opatřeny zvláštní značkou (>) nad notou. Takto popsané jsou takty jednoduché. Spojením dvou jednoduchých taktů vzniká takt složený. Složené takty mají dva přízvuky. Silnější na začátku taktu a o něco slabší tam, kde by začínal druhý jednoduchý takt.

Součet hodnot not v každém jednom taktu se musí rovnat počtu dob taktu. Čili například v jednom dvoučtvrťovém taktu může být jedna nota půlová (má dvě doby), dvě noty čtvrťové, jedna čtvrťová a dvě osminové, a tak dále. To, jaký má skladba takt, zjistíme na začátku skladby, kde je zapsán zlomek. Čitatel uvádí počet dob a jmenovatel hodnotu počítané doby. V průběhu skladby může dojít ke změně taktu. Tato změna je značena taktovým označením hned za taktovou čáru, kde změna začíná. Čtyřčtvrťový takt se většinou nazývá celý a místo zlomku je značen velkým písmenem C

Kapitola 3

Strojové učení

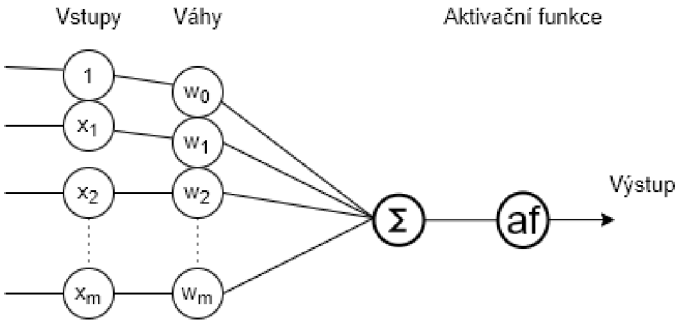
Umělá inteligence dokázala relativně rychle řešit problémy, které jsou pro člověka intelektuálně náročné, ale jsou popsány formálními pravidly. Existuje však celá řada problémů a úloh, k jehož řešení člověk přistupuje intuitivně či kreativně. V případě intuitivního přístupu lze hovořit např. o rozpoznávání objektů na scéně, nebo rozpoznání řečníků v nahrávce. Společným rysem těchto úloh je nemožnost formalizace. Jedinou aktuálně používanou technikou řešení takových problémů je tzv. učení s učitelem založené na existenci datové sady, která slouží k vytvoření znalostí o daném problému. Strojové učení umožňuje počítači naučit se ze zkušeností a porozumět světu v souvislostech. Získáváním zkušeností je odbourán proces formálního zápisu veškerých vědomostí, které jsou k řešení problému potřeba [13]. Harmonizace melodie je někde na pomyslné půli cesty mezi těmito extrémy, tedy formalizovanou hudební naukou, a intuicí skladatele. Text této kapitoly čerpá z [13], [28], [5], [26], [28], [27] a [24].

Strojové učení je ve své podstatě forma aplikované statistiky, s důrazem na odhad složitých funkcí počítačem. Většinu algoritmů lze rozdělit do dvou kategorií – učení s učitelem (supervised) a učení bez učitele (unsupervised). Při učení s učitelem se algoritmus snaží nějakému vstupu přiřadit daný výstup na základě trénovací sady příkladů vstupů a výstupů. Trénovací výstupy může být obtížné získat automaticky, a proto musí být dodány člověkem, učitelem. Mezi algoritmy učící se s učitelem patří například Support Vector Machines, k-nejbližších sousedů nebo rozhodovací stromy.

Algoritmy bez učitele se učí pouze na základě vlastností objektů a už nemají reflexi, zda se naučili správně nebo ne. Učení bez učitele se zabývá extrakcí informací z rozdělení, u kterého nejsou potřeba člověkem označená trénovací data. Nejčastěji se jedná o odhad hustoty, odšumění dat s nějakým rozdělením, hledání střední hodnoty všech dat nebo shlukování dat do skupin podobných objektů. Hledáme tedy nejlepší reprezentaci dat. Tato reprezentace data zjednodušuje, ale tak, aby bylo zachováno co nejvíc informací. Mezi představitele tohoto přístupu patří například shlukování k-nejbližších sousedů, analýza hlavních komponent nebo lineární regrese.

Neurony

Základním stavebním prvkem umělých neuronových sítí je neuron, který je inspirován biologickým neuronem, buňkou mozku. Ty zachycují synapse, neboli signály pomocí dendritů, a šíří je do jádra buňky. Tam díky vnějším signálům vzniká potenciál a pokud je dostatečně velký, neuron jej pošle dál svým axonem. Axon je na konci rozvětven a připojen na dendrity dalších neuronů.



Obrázek 3.1: Schéma neuronu.

V matematickém modelu neuronu (viz obrázek 3.1) se signály x z předchozích neuronů násobí s váhami w. Váhy jsou trénovatelné hodnoty, které řídí velikost potenciálu jednotlivých předchozích neuronů na neuron následující. Každému vstupu neuronu je přiřazena váha, která jej buď posílí, nebo utlumí. Stejně jako v biologickém neuronu vedou všechny dendrity (hodnoty xw) do těla neuronu, kde jsou sečteny. Neuron vyšle signál dál pouze pokud vnitřní potenciál překoná jistý práh. Ten je ve formě záporné váhy w_0, která do neuronu vstupuje nezměněna. Nakonec v modelu existuje aktivační funkce, kterou prochází suma všech váhovaných vstupů a určuje, jestli a jak moc může signál pokračovat skrz síť. Tato funkce by měla být nelineární. Používá se například funkce Sigmoid nebo hyperbolický tangens (zkracováno jako *tanh*). Definice obou aktivačních funkcí je uvedena v rovnici 3.1 a 3.2 a jejich průběh je pro ilustraci znázorněn taktéž na obrázku 3.2. případně rovnice níže. Z tvarů křivek lze identifikovat několik oblastí. Pokud je vnitřní potenciál nízký (záporný), výstup neuronu s aktivační funkcí sigmoidy se bude blížit nule. Naopak, je-li potenciál dostatečně vysoký, výstupem neuronu bude 1. Neuron s aktivační funkcí tanh se chová obdobně, ale nízké hodnoty potenciálu se na výstupu blíží hodnotě -1.

$$Sigmoid(x) = \frac{e^x}{e^x + 1} \tag{3.1}$$

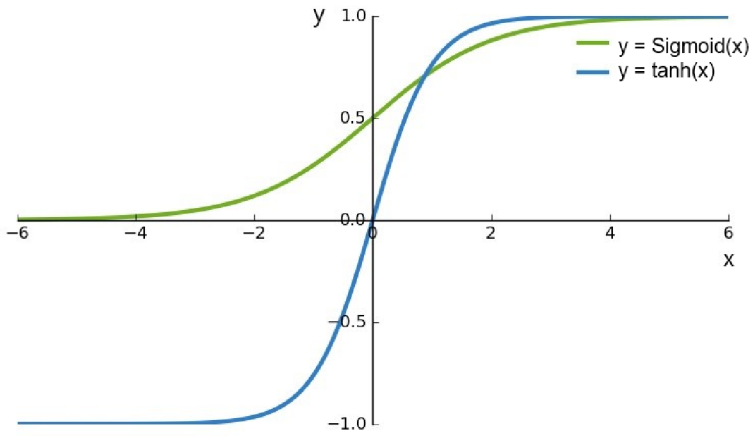
$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.2}$$

Mimo tyto základní aktivační funkce, které jsou historicky nejstarší, však existují také další varianty, které jsou typickou součástí tzv. moderních neuronových sítí [1]. Tyto funkce snižují výpočetní náročnost odstraněním několika operací. Nejběžnější alternativou Sigmoid je např. *Rectified Linear Unit* (dále jen ReLU). Definice funkce ReLU je následující:

$$ReLU(x) = max(0, x) \tag{3.3}$$

Při kladných vstupních hodnotách propaguje hodnotu na výstup, zatímco místo záporných hodnot propaguje nulu. To s sebou přináší problém při učení, kdy při záporných hodnotách je gradient nulový a učení stagnuje. Tento problém lze řešit upravenou funkcí *Leaky ReLU*, která místo nuly při záporném vstupu propaguje zápornou hodnotu blízkou nule. Gradient tedy není vždy nulový a neuron tak nepřestane reagovat.

Aktivační funkce obvykle používaná u klasifikačních problémů se nazývá softmax. Úkolem je přiřadit vzorek do jedné z konečného počtu definovaných tříd. Tato funkce se tak používá až v poslední vrstvě, kde se stará o to, aby se suma výstupních hodnot neuronů



Obrázek 3.2: Aktivační funkce sigmoid a hyperbolický tangens.

rovnala jedné. Výstupní hodnoty tak představují pravděpodobnosti příslušnosti prvku k jednotlivým klasifikačním třídám. Následuje definice funkce softmax:

$$softmax(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}} \tag{3.4}$$

,kde $j \in (1, N)$ a N je délka výstupního vektoru. Na výstupní hodnotu je aplikována exponenciální funkce a výsledek je normalizován sumou exponenciál všech prvků vektoru.

3.1 Perceptron

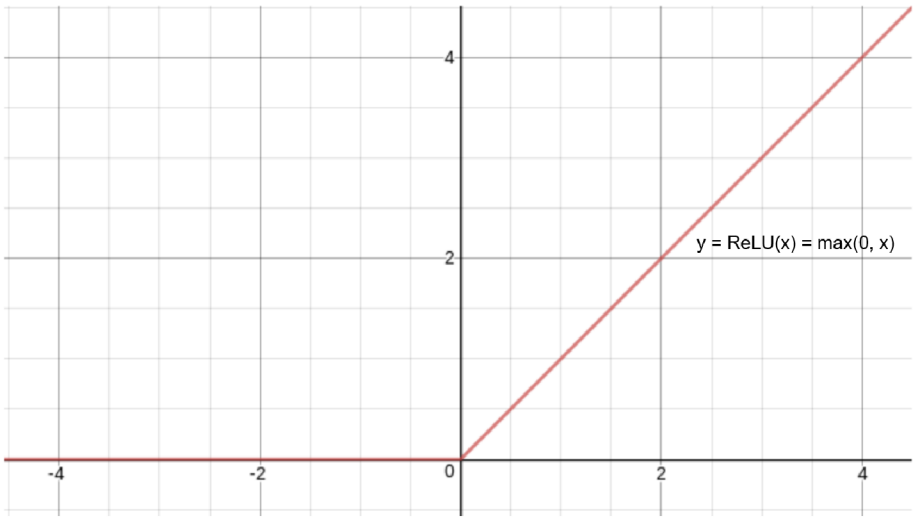
Nejjednodušší neuronovou sítí je perceptron. Tento model je složen z jediného matematického modelu neuronu. Perceptron dokáže klasifikovat pouze dvě lineárně separovatelné kategorie. Vstupem je, stejně jako u jediného neuronu, vektor vlastností x násobený vektorem vah $\mathbf{w}$ a s přičteným prahem w_0. Výsledkem je rovnice nadroviny

$$w_0 + \sum_{i=1}^{n} x_i w_i = y \tag{3.5}$$

,která dělí n-dimenzionální prostor na dva poloprostory. Je-li výsledný potenciál dostatečně velký, bod leží v jednom poloprostoru, naopak je-li potenciál příliš malý, pak leží v druhém poloprostoru. Kategorie bodu je dána jeho příslušností k jednomu, nebo druhému poloprostoru. Učení spočívá ve změně vah, tedy ve změně pozice nadroviny tak, aby každý poloprostor nejlépe odpovídal pozici kategorie, kterou vyjadřuje.

3.2 Neuronové sítě

Neuronové sítě jsou výpočetní model používaný v umělé inteligenci. Existuje několik variant. Základním přístupem jsou dopředné neuronové sítě (Feedforward Neural Networks). Dále rozlišujeme konvoluční sítě[23] a rekurentní sítě[21]. Vzory jsou jakékoliv data, čísla, obrázky, zvuk, text a další. Ty musí být převedeny na vektory čísel, které jsou následně



Obrázek 3.3: Aktivační funkce Rectified Linear Unit.

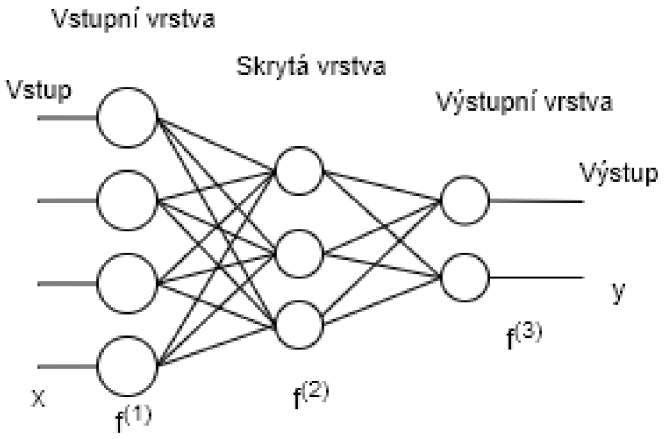
použity jako vstup. Před uvedením sítě do provozu je potřeba ji nejprve jednorázově natrénovat na datech z datové sady. Ty se nazývají trénovací a jsou podobná těm, která bude naučená síť zpracovávat. Učení sítě spočívá v úpravách hodnot vah mezi jednotlivými neurony. Jednotlivé váhy postupně konvergují ke své optimální hodnotě. Tato fáze bývá výpočetně i časově velmi náročná, kvůli velkému množství výpočtů prováděných nad každým prvkem. Po dokončení učení je síť připravena k použití. To už zpravidla nepředstavuje takovou výpočetní zátěž a použití sítě je velmi rychlé.

Neuronové sítě mají většinou několik propojených vrstev. Neurony jedné vrstvy jsou spojeny pouze s neurony z jiné vrstvy. Neurony v jedné vrstvě mezi sebou nesdílí žádné vazby. První vrstva se také nazývá vstupní, zatímco poslední je výstupní. Vrstvy mezi vstupní a výstupní jsou takzvaně skryté. Počet vrstev udává hloubku modelu, odtud také hluboké učení, a velikost vektoru jeho šířku. Každý neuron sítě řeší část celého problému a výsledek je tvořen kompozicí všech. Každý neuron sítě, stejně jako perceptron, dělí prostor na dva poloprostory. Jedna vrstva, obsahující několik neuronů, rozdělí prostor na několik různých poloprostorů, což jej umožňuje dobře zmapovat. Větším množstvím neuronů se zvyšuje počet různých rozdělení prostoru a tedy i rozlišovací schopnost sítě. Současně se však také zvyšují výpočetní nároky na učení sítě. Vstupem následující vrstvy už nejsou původní příznaky, ale příslušnost prvku k jednotlivým oblastem. Celý problém je transformován do nového prostoru a druhá vrstva rozdělení zpřesňuje.

Úkolem neuronové sítě je aproximovat neznámou funkci $y = f'(x)$ funkcí $y = f(x; \theta)$, kde x a y značí vstup, respektive výstup. K hodnotě θ dojde neuronová síť tak, aby výsledný model nejlépe aproximoval původní funkci. Samotná funkce f se skládá z několika zřetězených funkcí $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, kde $f^{(n)}$ označuje vrstvy sítě, jejichž vstupem je vektor z předchozí vrstvy a výstupem je opět vektor.

3.3 Učení neuronové sítě

Jedním z nejpoužívanějších algoritmů k učení neuronových sítí je back-propagation, tedy zpětné šíření chyby. Tento algoritmus patří do třídy gradientních metod a zajišťuje změnu vah neuronů v jednotlivých vrstvách sítě. Algoritmus začíná pracovat u výstupní vrstvy



Obrázek 3.4: Příklad neuronové sítě se čtyřmi vstupy a dvěma výstupy tvořené třemi vrstvami.

a postupně šíří chybu dalšími vrstvami směrem ke vstupní. Postupně upravuje jednotlivé váhy neuronů, dokud nedojde k lokálnímu nebo, v lepším případě, globálnímu minimu chyby. V algoritmu se pracuje také s parametrem míry učení (anglicky learning rate), který určuje rychlost učení. Tento parametr nabývá hodnot z intervalu $< 0, 1 >$. Při vyšších hodnotách je váhy modifikují výrazněji, ale mohou uváznout v lokálním minimu, z něhož už se nebudou moci dostat. Proto některé algoritmy míru učení postupně snižují v průběhu tréningu.

Algoritmus back-propagation klade podmínku na využitou aktivační funkci každého neuronu. Ta musí být diferencovatelná, tedy musí existovat její derivace. Tato vlastnost je zásadní pro gradientní sestup (anglicky gradient descent), což je optimalizační metoda pro hledání minima nějaké funkce. Gradient je vektor největšího růstu funkce. Pro hledání minima tedy algoritmus postupuje opačným směrem, proti směru gradientu. Jeho velikost se dále používá pro určení kroku každé iterace. Délka kroku je přímo úměrná velikosti gradientu a také již zmiňované míře učení. Následuje vztah pro postupnou změnu vah:

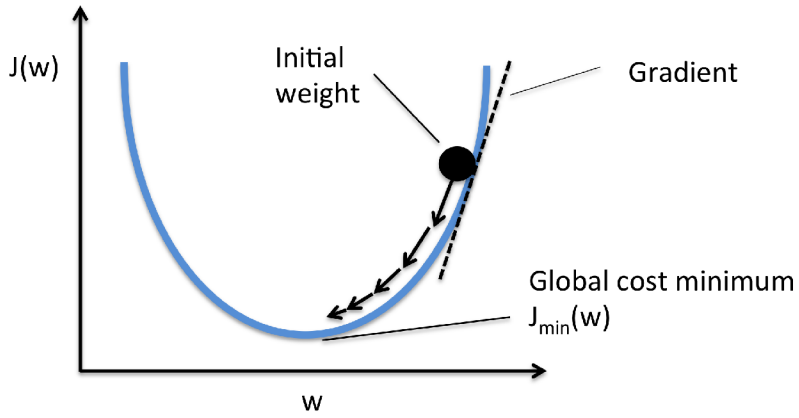
$$w_{i+1} = w_i - \eta \nabla G(w_i) \tag{3.6}$$

Kde w je hodnota váhy, η je hodnota míry učení, ∇G je hodnota gradientu a i je aktuální krok. Před začátkem algoritmu jsou všechny váhy v sítě inicializovány na náhodnou hodnotu z rozmezí $< 0, 1 >$. Iterativní postup algoritmu ke globálnímu minimu ilustruje obrázek 3.5. Algoritmus provede výpočet nad trénovací datovou sadou s počátečními nastavenými váhami a vypočítá chybu. Následně s pomocí gradientního sestupu zjistí jak se mají váhy změnit pro dosažení optimálnějšího řešení. Algoritmus se opakuje, dokud nenarazí na lokální, případně globální minimum chybové funkce v závislosti na váhách v neuronové síti [31, 13].

3.3.1 Chybové funkce

Abychom mohli provést aktualizaci vah v průběhu učící fáze, používáme chybové funkce. Ty slouží k určení přesnosti sítě a při trénování také k učení sítě zpětným šířením chyby přes všechny neurony.

Jednou z chybových funkcí je střední kvadratická chyba, neboli MSE z anglického *Mean squared error*. Tato funkce odečte hodnotu na výstupu sítě od očekávané hodnoty. Aby



Obrázek 3.5: Gradientní sestup. [31]

chyba nebyla záporná, rozdíl hodnot je umocněn na druhou. Tímto způsobem se uměle zvyšuje rozdíl chyby mezi korektně a chybně klasifikovaným vzorkem, což zrychluje konvergenci trénování sítě. Nevýhodou naopak je, že jediná odlehlá predikce může drasticky podhodnotit výkon modelu. Případně mnoho malých chyb (menších než 1) může výkon přecenit. Definice funkce čtvercové chyby je následující:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(d_i - y_i)^2 \tag{3.7}$$

,kde N je počet prvků z trénovací sady, d_i je očekávaná hodnota a y_i je výstup sítě.

Jinou, hojně využívanou chybovou funkcí je křížová entropie (anglicky Cross-Entropy), která se používá při klasifikaci do tříd. Předpokladem k použití jsou dva a více neuronů ve výstupní vrstvě. Tato funkce se často používá zároveň s aktivační funkcí softmax, která normalizuje výstupní hodnoty poslední vrstvy do pravděpodobností náležitosti k jednotlivým třídám. Následuje definice funkce křížové entropie (CE) pro počet výstupních neuronů $C > 2$:

$$CE = -\sum_{i}^{C} d_i log(y_i) \tag{3.8}$$

,kde opět d_i je očekávaná hodnota a y_i je výstup sítě.

Pro binární klasifikátor, tedy $C = 2$, lze funkci zjednodušit následovně:

$$CE = -d_i log(y_i) + (1 - d_i) log(1 - y_i) \tag{3.9}$$

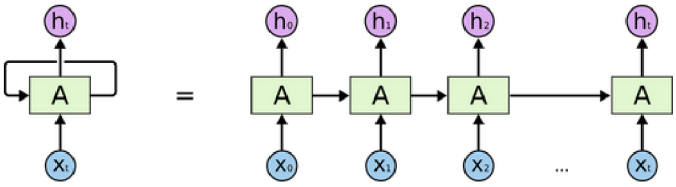
Stejně jako u MSE platí, že lépe naučená síť má nižší hodnoty křížové entropie. Dokonale naučená neuronová síť má křížovou entropii rovnu nule.

3.4 Rekurentní neuronové sítě

Základní struktura neuronové sítě může být však pro některé úlohy omezující neboť předpokládá, že na výstupu neuronová síť produkuje sekvence na sobě nezávislých vektorů. Rekurentní sítě umožňují mít takovou sekvenci jako vstup, jako výstup nebo jako obojí. Principem RNN jsou zpětnovazební smyčky, kdy výstupy jedné vrstvy jsou propojeny se

svými vstupy. Systém si tak při řešení problému uchovává informaci o předešlých prvcích vstupních dat, která může ovlivnit následující výsledky[21]. Takový přístup může být výhodný například v úloze harmonizace melodie, kde jsou typické závislosti akordů na předchozím průběhu skladby. Použití RNN je však spojeno s řadou komplikací. Například znatelně zvětšují prohledávaný prostor řešení, což zvyšuje požadavky na výpočetní čas při učení.

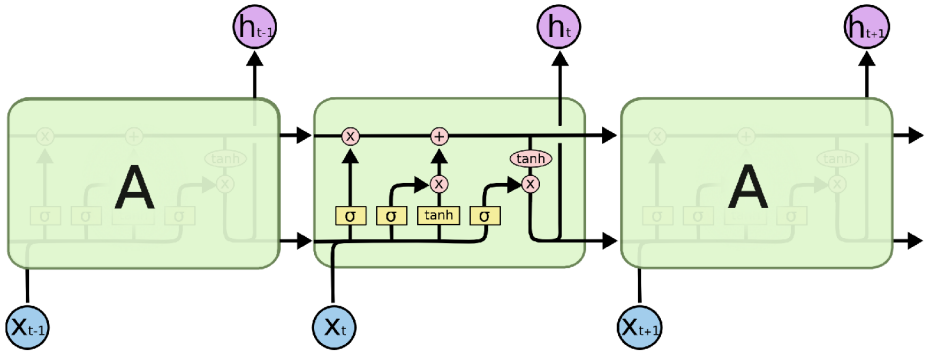
Přenášení předchozích znalostí z důvodu ovlivnění následujících prvků je problémové. Velmi totiž závisí na vzdálenosti mezi prvky, které se mohou ovlivnit. Pokud jsou od sebe prvky až příliš vzdáleny, rekurentní síť se nedokáže naučit spojitost mezi nimi [29].



Obrázek 3.6: Schéma rekurentní neuronové sítě. [29]

3.4.1 Long Short-Term Memory

Long Short-Term Memory (dále jen LSTM) jsou speciálním druhem rekurentních neuronových sítí. Jsou schopny se naučit i dlouhodobější závislosti v datech. S touto vlastností byly primárně navrženy. Princip činnosti LSTM bude vysvětlen podrobněji, neboť jsou základem použitého modelu v této práci. Následující text čerpá z [29].

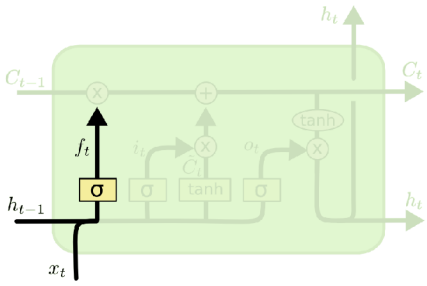


Obrázek 3.7: Schéma opakujícího se LSTM modulu. [29]

Narozdíl od RNN mají LSTM čtyři vrstvy v jednom opakujícím se modulu. Schéma LSTM modulu je na obrázku 3.7. Charakteristickým rysem je přítomnost vektoru který je znázorněn v horní částí diagramu. Ten vede stav buňky (anglicky cell state). Informaci ve stavu lze modifikovat malými, lineárními změnami. Každá buňka může informaci přidat, odebrat, nebo nechat beze změny.

V průběhu inference se musí LSTM nejprve rozhodnout, kterou informaci chce zapomenout, tedy vyřadit ze stavu buňky. Toto rozhodnutí provádí takzvaná "forget gate layer", tedy vrstva s aktivační funkcí sigmoidy. Tato brána vezme v úvahu aktuální vstup a výstup předchozí buňky a převede je na hodnotu z intervalu $< 0, 1 >$, přičemž 1 znamená ponechat informaci a 0 znamená kompletně zapomenout. Práce forget gate layer a její funkce jsou

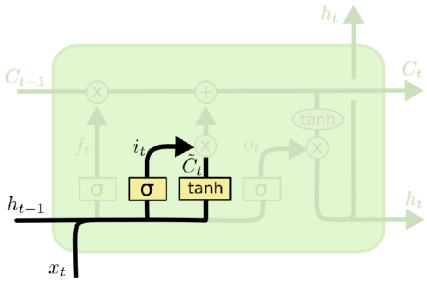
ilustrovány na obrázku 3.8, kde W a b jsou váhy a práh (bias), h_{t-1} je předchozí výsledek, x_t je aktuální vstup a $[h_{t-1}, x_t]$ je konkatenace prvků.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Obrázek 3.8: Schéma a funkce forget gate layer. [29]

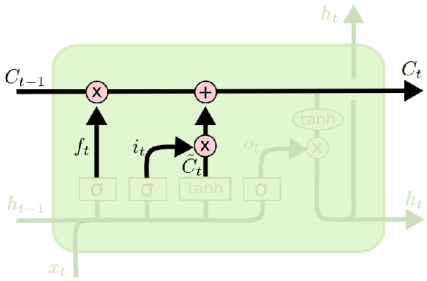
Následuje rozhodnutí, jakou informaci ve stavu uchovat. Tento proces se skládá ze dvou částí. První se nazývá input gate layer, opět vrstva se aktivační funkcí Sigmoid. Tato brána rozhoduje o tom, které hodnoty budou aktualizovány. Druhou částí je vrstva s aktivační funkcí tanh, jenž sestavuje vektor nových kandidátních hodnot určených pro přidání do stavu. Činnost input gate layer opět ilustruje následující obrázek 3.9.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Obrázek 3.9: Schéma a funkce input gate layer. [29]

Po výpočtu změn v předchozích dvou krocích je potřeba aktualizovat starý stav buňky. Nejprve je násoben vektorem f_t, tedy tím, co má být zapomenuto. Následně je přičteno $i_t\tilde{C}_t$, tedy nové kandidátní informace násobené hodnotou, která určuje, jak moc tyto informace ovlivní stav.

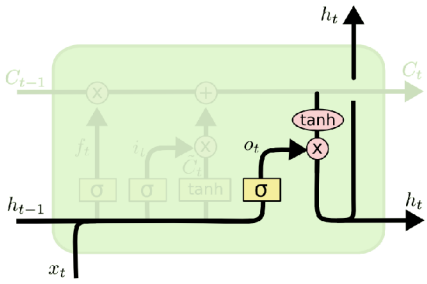


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Obrázek 3.10: Schéma a funkce aktualizace stavu buňky. [29]

Zbývajícím krokem je určit výstupní data buňky v aktuálním kroku. Dá se říci, že tento krok je podobný obrácení předchozího kroku. Nejprve projde aktuální vstup vrstvou sigmoid, která rozhodne, co bude na výstupu. Následně projde stav buňky vrstvou s aktivační

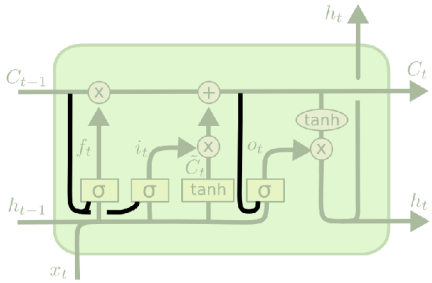
funkcí tanh a výsledek se vynásobí s výstupem sigmoid vrstvy. Práce je patrná na obrázku 3.11



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

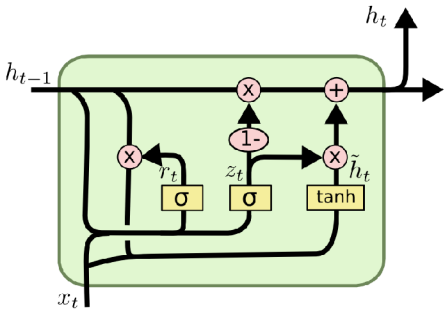
Obrázek 3.11: Schéma a funkce výpočtu výstupních hodnot LSTM buňky. [29]

Kromě výše popsané implementace vycházející z [19] však existuje i řada variant [6, 11]. Jedna z variant například dovoluje forget gate a input gate podívat se na stav buňky ještě předtím, než rozhodnou o jeho modifikaci[11]. Viz obrázek 3.12.



Obrázek 3.12: Schéma varianty LSTM buňky. [29]

Další variantou vycházející z architektury LSTM je Gated Reccurent Unit (dále jen GRU) uvedená v [6]. Nejvýznamnější je kombinace forget gate a input gate do jediné update gate layer. Dále také kombinuje stav buňky s výstupní hodnotou. Strukturu buňky GRU reprezentuje obrázek 3.13. Výsledkem těchto modifikací je buňka, která zjednodušuje jak výpočet, tak trénování. Díky tomu začíná být více populární a je často používána.



Obrázek 3.13: Schéma varianty Gated Reccurent Unit. [29]

Kapitola 4

Přehled přístupů automatické harmonizace

Pojem automatická harmonizace melodie označuje disciplínu, která se zabývá tvorbou modelu, který dokáže vygenerovat harmonický doprovod k dané melodii. Samotná harmonizace je složitý úkol, protože tu samou melodii lze harmonizovat mnoha různými způsoby. Správná harmonie závisí na subjektivním pocitu, hudebním žánru a dalších faktorech. V hudbě se ustálilo mnoho pouček a zákonů o tom, které akordy k sobě patří a jak na sebe navazují. Často určují jemné nuance, či jsou podmíněny kulturním vlivem, což je pro stroj velmi obtížné zachytit [35].

V této kapitole bude představeno několik přístupů k automatické harmonizaci.

4.1 Model založený na shodě šablon

Tento model nejdříve rozdělí všechny melodie z trénovací sady po polovinách taktů a pro každý, takto vytvořený segment vytvoří profil výšky tónů (dále jen PCP[1]). [35] PCP je dvanácti-dimenzionální binární vektor, kde každá složka reprezentuje přítomnost každého z dvanácti možných půltónů v daném půl-taktu. [9] PCP nového segmentu se porovná s těmi z trénovací sady a spočítá se podobnostní skóre. Nový segment je pak označen takovým akordem, který měl jemu nejpodobnější segment v trénovací sadě – s nejvyšším skóre. Jestliže má nejvyšší skóre několik segmentů, je výsledný akord vybrán náhodně s rovnoměrným rozložením pravděpodobnosti. Označení akordu je opět ve formě PCP, a to tak, že všechny tóny náležící akordu jsou nastaveny do 1.

Nevýhodou tohoto modelu je, že určuje jednotlivé segmenty nezávisle bez toho, aby bral v úvahu sousední akordy nebo postup akordů ve skladbě [35].

4.2 Skryté Markovovy modely

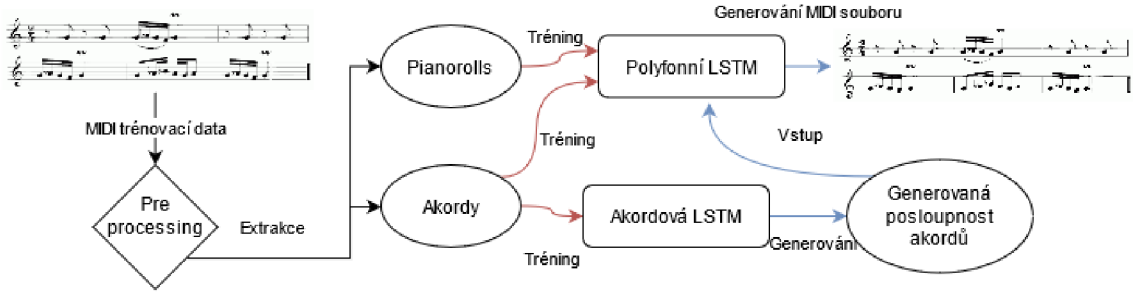
Jednou z nejpoužívanějších metod generování akordů a harmonizace melodie, před současným rozšířením hlubokého učení, byly skryté Markovovy modely (dále HMM [2]). HMM jsou pravděpodobnostní nástroj, s jehož pomocí lze modelovat sekvence se skrytými proměnnými. Označení akordů je považováno právě za skrytou proměnnou a HMM se snaží

[1] z anglického Pitch Class Profile
[2] z anglického Hidden Markov Model

odhadnout nejpravděpodobnější sekvenci akordů k daným notám melodie. Na rozdíl od předchozího přístupu tento model bere v úvahu předcházející akordy. [35]

4.3 JamBot

JamBot [4] je jeden z novějších přístupů generování akordů a polyfonní hudby založený na LSTM sítích. Pracuje ve dvou krocích. V prvním kroce akordová LSTM vygeneruje postup, vývoj akordů ve skladbě. Tato série akordů je následně vstupem druhé LSTM sítě, která na základě vstupu vygeneruje polyfonní hudbu (viz obrázek 4.1). Tento model tedy nejprve vygeneruje harmonii a k ní poté melodii.



Obrázek 4.1: Architektura modelu JamBot. [4]

K reprezentaci dat pro akordovou LSTM je využita technika ze zpracování přirozeného jazyka (dále jen NLP z anglického Natural Language Precessing). Každý jedinečný akord je nahrazen číselným identifikátorem (id). K překladu mezi akordy a identifikátory pak slouží slovník dvojic (id, akord). V korpusu akordů je pouze N nejčastěji použitých akordů v trénovací sadě. Akordy, které se do korpusu nevlezly (vyskytují se příliš řídce), získají označení "neznámý akord". V práci byla zvolena velikost korpusu 50 akordů. Před vstupem do LSTM sítě byly identifikátory zakódovány na *jedna z N* reprezentaci [2]. Vstupní vektor má velikost korpusu. Všechny záznamy vektoru jsou nastaveny na 0, kromě toho s indexem rovným identifikátoru akordu, který je nastaven na 1.

Akordová LSTM má dvě vrstvy. První je takzvaná embeddingová vrstva, která opět využívá techniku používanou v NLP. Tou jsou word embeddings, které mapují slova (v tomto případě akordy) z korpusu do vektorů reálných čísel. Word embeddings nejsou pevně dané a musí být natrénovány. Vektorový prostor pak zachycuje vztahy mezi jednotlivými slovy a sémanticky podobná slova jsou v tomto prostoru blízko sebe. V této práci byly místo slov použity akordy. Za embeddingovou vrstvou následuje LSTM vrstva, která předpovídá akordy, které následují po aktuálním. K predikci nové posloupnosti potřebuje základ libovolné délky. Další akord je získán vzorkováním výstupního pravděpodobnostního vektoru s parametrem temperature. Tento akord je potom dán zpět na vstup této vrstvy a na jeho základě je generován další a tak dále. Parametr temperature ovlivňuje různorodost generované posloupnosti. Nulová temperature znamená, že pro daný základ budou všechny následující akordy stejné.

Vstupní data pro polyfonní LSTM jsou ve tvaru tzv. pianoroll (více o pianoroll viz níže). Každý takt je rozdělen na osm částí a každá část je reprezentována vektorem s velikostí rovné počtu tónů. Pokud v dané části tón zní, pak je příslušný prvek vektoru nastaven na

[2] One-hot-encoding

1. Jedná se o podobný princip jako v části 4.1. Při generování melodie má vstup polyfonní LSTM několik částí. První je již zmiňovaný pianoroll vektor aktuálního kroku. Kromě aktuálně znějících tónů je na vstupu také embedding aktuálního akordu. Díky tomu je možné naučit se, které tóny zní při kterém akordu. V hudbě melodie většinou "postupuje"k dalšímu akordu. Kvůli tomu je další částí vstupu akord následující aktuálnímu kroku. Generované písničky jsou tak více strukturované. Poslední částí vstupu je binární počítadlo, které počítá od 0 do 7 v každém taktu. Toto umožňuje LSTM orientovat se uvnitř taktu a ví, kolik kroků zbývá do změny akordu (akordy se mění pouze na začátku taktu, viz výše). Při generování nové písně je na vstupu opět nejprve základ skládající se z pianoroll vektoru a příslušných akordů. Noty jsou vzorkovány individuálně. Pokud je nota vybrána, pravděpodobnost výběru ostatních not ovlivněna není [4].

4.4 Google Magenta

Magenta je open-source výzkumný projekt založený Google Brain týmem. Jedná se o soubor nástrojů a modelů hlubokého i posilovaného učení. Zaměřuje se především na využití strojového učení jako nástroje v kreativním procesu. Projekt je distribuován pro Python a JavaScript. Magenta.js je pouze API pro používání předtrénovaných modelů v prohlížečích. Magenta pro Python je celá knihovna, která zahrnuje kromě předtrénovaných modelů také nástroje pro manipulaci dat, trénování vlastních modelů a generování nových výtvorů na jejich základě [14]. Vytvořené modely se zabývají generováním a modifikací hudebních a obrazových uměleckých dat. Obrazové modely jsou například image_stylization[3], který vytvoří nový obraz na základě obsahu jednoho a stylu druhého[8], nebo sketch-rnn[4], tedy rekurentní neuronovou síť, která se pokouší domalovat člověkem vytvořený náčrt, popřípadě načrtnout obrázek dle zadané třídy[18].

Z hudebních modelů Magenta obsahuje Coconet, který doplňuje zadanou melodii o kontrapunkt, tedy o protihlas více melodií[22, 20]. Rozdílem oproti klasickým algoritmům doplňující kontrapunkt je oproštění od chronologického postupu od začátku skladby ke konci. Naopak noty jsou generovány v jakémkoliv pořadí tak, že model opakovaně přepisuje a maže svou vlastní práci. Při trénování model přijímá čtyřhlasou harmonii s náhodně odstraněnými notami a snaží se tento vstup rekonstruovat. Na nižší úrovni je vstup tří dimenzionální pole, kde jedna z dimenzí vymezuje čtyřhlas, další číslo MIDI noty a poslední čas[5]. Tento objekt je vstupem konvoluční neuronové sítě. Výstupem je objekt o stejné velikosti, nyní obsahující pravděpodobnosti not ve smazaných částech [20]. Tento natrénovaný model lze vyzkoušet jako Google Doodle na https://www.google.com/doodles/celebrating-johann-sebastian-bach

Jedním z dalších Magenta modelů, které se zabývá hudbou je Polyphony RNN. Ten aplikuje jazykové modely ke generování polyfonní hudby za použití LSTM neuronových sítí a je založen na projektu BachBot, který je popsán v následující citaci [25]. Na rozdíl od modelu Melody RNN, který také využívá jazykových modelů, Polyphony RNN dokáže pracovat s více souběžnými notami. Tento model tedy pracuje také s vícehlasem. Vícehlasý vstup je zde modelován jako jeden proud notových událostí se speciálními řídícími symboly. Skladba je rozdělena na stejné časové úseky (stepy) a každý z úseků obsahuje seznam aktivních not. Noty jsou v každém úseku seřazeny podle výšky (v MIDI formátu) a je popsán jejich vztah v závislosti k předchozímu stavu tónu [16].

[3] https://github.com/magenta/magenta/tree/master/magenta/models/image_stylization
[4] https://github.com/magenta/magenta/tree/master/magenta/models/sketch_rnn
[5] V literatuře se pole s dimenzemi (MIDI nota, čas) nazývají pianoroll

Pro generování nové skladby potřebuje model tzv. primer, tedy pár not, použitých jako počátek skladby. Primer lze vložit ve třemi způsoby, a to jako počáteční akord, počáteční melodie, nebo MIDI soubor. Tento vstup lze vložit do modelu před generováním, tedy ve vygenerované sekvenci vstup nebude, ale bude jím ovlivněn. Toto se hodí například pro ustanovení tóniny generované skladby. Dále je možné použít vstup jako součást generované sekvence, což je vhodné při harmonizaci již existující melodie [16].

Kapitola 5

Trénovací sada

Základním předpokladem pro aplikaci neuronových sítí je existence vhodné trénovací sady, která obsahuje anotovaná data. V našem případě je nutné generovat harmonii na základě znalosti melodie. Jednou z možností, jak k tomuto problému přistoupit je využit model neuronové sítě Magenta. Pro tento model jsou potřeba trénovací data, která mají na vstupu izolované noty melodie a výstupem jsou izolované akordy

Vstupní data budou v rámci práce získána tak, že se využijí hudební soubory z platformy youtube a anotace, která je dostupná v rámci projektu hooktheory.

Pro sjednocení trénování i inference, bude pro vstup použit standardní formát pro popis zvukových dat, tzv. MIDI.

5.1 Formát MIDI

MIDI (Musical Instrument Digital Interface) je volně přístupný standard, který specifikuje hardware i software pro digitální komunikaci hudebních nástrojů, sekvencerů, počítačů, mixérů a dokonce i jevištní techniky, jako jsou reflektory a lasery [34].

Standardní MIDI soubor (zkráceně SFM) je formát binárních souborů určených pro uložení a přenos hudebních dat mezi zařízeními. Soubory v tomto formátu lze rozeznat pomocí *.mid* přípony. Oproti ostatním formátům pro uložení zvuku SFM neukládají digitalizovaný zvuk, ale parametry použitých nástrojů, informace o tempu, kanálech, jednotlivé noty uložené jako kombinace (čas, hodnota, rychlost) a další MIDI události. Data z těchto souborů jsou načteny do nějakého přehrávače a výsledný zvuk je vytvořen připojeným sound-enginem [34, 12].

Data v SFM jsou uložena v blocích. Každý z těchto bloků začíná čtyřbajtovým textovým označením a čtyřbajtovou délkou datového bloku udávající počet bajtů. Ihned po délce začíná datová část o délce zadané v hlavičce. Za datovou částí začíná buď další blok, nebo konec souboru [34, 2].

Soubory MIDI mají dva druhy bloků. Hlavičkový blok poskytuje informace vztahující se k celému souboru. Bloky stopy obsahují proudy MIDI dat až šestnácti MIDI kanálů. MIDI soubor vždy začíná hlavičkovým blokem a je následován jedním nebo více bloky stop[2].

MThd <délka hlavičky>

<data hlavičky>

MTrk <délka stopy>

<data stopy>

MTrk <délka stopy>

<data stopy>

...

5.1.1 Hlavička

Blokem hlavičky začíná každý MIDI soubor a obsahuje jeho základní popis. Je uvozen typem bloku, čtyřmi ASCII znaky "MThd"a 32bitovou reprezentací čísla 6 (délka bloku). Začátek SMF je tedy pevně daný a takový soubor lze na první pohled rozeznat začínající sekvencí *0x4D 54 68 64 00 00 00 06*. Datovou část tvoří tři 16bitové slova. První z nich specifikuje formát, celkovou organizaci souboru. Tím je myšleno uspořádání uložených stop. Jsou zde tři možnosti:

0. soubor obsahuje jedinou více-kanálovou stopu
1. několik souběžných stop
2. několik nezávislých stop

Formát 0 je nejrozšířenější a je podporován i nejjednoduššími programy a hardware [2]. Formát 1 obsahuje jednu nebo více vertikálně synchronních stop. Jinými slovy, při přehrávání takového souboru začnou všechny stopy ve stejný čas a mohou reprezentovat různé části písničky. V posledním, asynchronním formátu nemusí stopy nutně začínat ve stejnou dobu [12]. Tento formát je nejméně používaný [34].

Dalším slovem hlavičky je počet stop. Ten je u nultého formátu vždy roven jedné. Poslední část definuje kódování času, které může být zadáno dvěma způsoby. Ty jsou dány prvním bitem slova. Pokud je bit nulový, pak zbývající bity reprezentují počet "tiků"generátoru hodin, které se vlezou do jedné čtvrťové noty. Pokud je bit nastaven na 1, pak je čas vyjádřen jako rozdělení vteřin podle SMPTE standardu a MIDI Time Codu [2]. Filmovým průmyslem vytvořený SMPTE, mimo jiné, definuje čtyři různé snímkovací frekvence. Je podle něj možné rozdělit vteřinu na 24, 25, 29 nebo 30 snímků. Pro hudební účely je však potřeba ještě jemnější rozlišení. Každý snímek je tedy dále možné rozdělit na "sub-snímky"[12]. Bity 14 až 8 časového kódování udávají počet SMPTE snímků za vteřinu a zbývající část je rozlišení jednoho snímku. Pro představu, při časování událostí na milisekundy je potřeba nastavit 25 SMPTE snímků a rozlišení 40, neboť $40 * 25 = 1000$ "sub-snímků"za vteřinu, stejně jako milisekund [2].

5.1.2 Stopy

Každý MIDI soubor obsahuje nejméně jeden blok stopy. Ten zahrnuje parametry použitých nástrojů, jednotlivé noty skladby, ale také různé textové informace jako název skladby nebo její text. Za identifikátorem bloku "MTrk"a jeho délkou následuje posloupnost MIDI událostí ve tvaru časový přírůstek, stavový bajt, datové bajty [2].

Časový přírůstek (deltatime) značí počet "tiků"od předchozí události. Tato hodnota má proměnnou délku. Její formát umožňuje velkým číslům využít tolik bajtů, kolik potřebují bez toho, aby malá čísla plýtvala místem a plnila jej nulami. Hodnoty jsou kódovány do 7-bitových bajtů a osmý, nejvýznamnější bit je nastaven na jedničku. Pokud je bajt poslední

Oktáva	Čísla not											
	C	**C#**	**D**	**D#**	**E**	**F**	**F#**	**G**	**G#**	**A**	**A#**	**B**
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	222	123	124	125	126	127				

Tabulka 5.1: Čísla všech MIDI not po oktávách. [2]

v posloupnosti, pak na nulu. Celý deltatime by však ideálně neměl přesáhnout 4 bajty [12]. První událost stopy nebo dvě současné události mají časový přírůstek roven nule [2].

Za časovým údajem následuje MIDI zpráva tvořená stavovým bajtem a až dvěma datovými bajty. Stavový bajt má vždy nejvýznamnější bit jedničku, zatímco datové mají nulu. Pokud je stav vynechán a za deltatime následují datové bajty, pak je lze poznat právě nulovým nejvyšším bitem. Existuje několik druhů stavů a je možné je rozdělit na kanálové zprávy (Channel Messages) a systémové zprávy (System Messages). Kanálové zprávy se vztahují k specifickému kanálu. Jeho číslo je zapsáno ve spodním půlbajtu stavu zprávy. Tedy například *0x91* představuje změnu stavu na kanále jedna. V proudu je možné rozeznat dva druhy kanálových zpráv. Channel Voice Message přenáší hudební produkci a tvoří většinu provozu MIDI proudu. To, jak na tyto zprávy budou přijímající nástroje reagovat, je dáno Channel Mode zprávami. Systémové zprávy jsou buď určeny všem přijímačům v systému (System Common Message), použity pro synchronizaci hodin MIDI komponent (System Real Time Message), nebo použity pro přenos dat mezi komponenty jednoho výrobce (System Exclusive Message) [2].

Aktivace a vypuštění jedné noty je v MIDI považováno za dvě rozdílné události. Jsou to kanálové zprávy *Note on* a *Note off*, z nichž první rozezní daný tón určitou silou a druhý ji utiší. Přijetí *Note On* lze rozeznat stavovým bajtem *0x9k*, kde *k* určuje na kolikátém kanále bude nota znít. Následuje datový bajt s číslem rozeznívané noty. Čísla not jsou odvozena od středního, jednočárkového c, kterému je přiřazena hodnota 60 (viz tabulka 5.1). Z toho vyplývá, že subkontra oktáva je nultá a čtyřčárková je sedmá.

MIDI bylo původně určené pro klávesové nástroje, druhý datový bajt je tedy pojmenován rychlost (zmáčknutí klapky), ale ve výsledku určuje to, jak hlasitě (s jakou amplitudou) bude tón znít oproti ostatním.

Pro ztlumení znící noty slouží zpráva *Note Off* označená bajtem *0x8k*. Stejně jako při rozeznění je potřeba vědět, o jakou notu se jedná. A také v tomto případě je ve zprávě obsažena rychlost (puštění klapky), která je ale ve většině případech ignorována [34].

5.2 Databáze skladeb

Data k projektu byla získána ve formě SQLite [1] databáze, se kterou jde jednoduše manipulovat pomocí jazyka Python. Tato databáze obsahuje záznamy o písničkách ve formátu XML. Velikost databáze je 32.7MB (34 334 720B) Databáze se skládá z několika tabulek, z nichž jedna obsahuje komprimovaný XML soubor. Uvnitř každého souboru se nachází jedna skladba, kterou je nutné extrahovat do MIDI souboru.

Kořenový uzel XML souboru z databáze obsahuje dva potomky – meta a sections. Meta zahrnuje základní informace popisující část písničky. Jedná se o jméno autora písničky, jednoznačné jméno části (například Hey Jude Chorus), počet dob v taktu, počet dob za minutu (BPM[2]), opět tónina, mód části, odkaz na Youtube a časy začátku a konce dané části v Youtube videu.

Sections obsahuje uzel pojmenovaný podle názvu části písničky (Intro, Verse, Chorus...). Části písniček jsou rozděleny na segmenty. Uvnitř segmentů jsou kategorie, které dále obsahují samotné noty (uzel notes) a akordy (uzel chords). Každá nota má několik vlastností. První čtyři se týkají času. Pro orientaci uvnitř taktu slouží *start_beat*, jenž značí, v kolikáté době taktu nota začíná. Tento čas je počítán od jedničky. Takže například nota začínající znít v první době taktu má *start_beat* nastaven na 1. Takt, ve kterém je nota aktivní, označuje *start_measure*. Obě tyto vlastnosti dává dohromady vlastnost *start_beat_abs*, která udává absolutní počet dob od začátku písničky po začátek noty. Délka noty v dobách je zapsána v *note_length*. Následují dvě vlastnosti týkající se výšky tónu. *Scale_degree* nabývá hodnot 1 až 7 podle použitého stupně tóniny, popřípadě *rest*, pokud je místo noty použita pomlka. Oktávu noty udává *octave* jako počet oktáv, které je nutno přičíst k oktávě s notou C60 (viz tabulka 5.1) Například *octave* noty C60 je nula, noty C48 minus jedna a noty C72 plus jedna. Poslední xml vlastností noty je redundantní *isRest*, pravdivostní hodnota s jedničkou, pokud se jedná o pomlku.

Akordy jsou v xml popsány podobně jako noty. Vlastnosti *start_beat*, *start_measure*, *start_beat_abs* a *isRest* mají stejný význam. Délku trvání akordu značí *chord_duration* stejně jako u not v počtu dob. Zbývající vlastnosti popisují akord jako takový. První tón akordu, a tedy i jeho název, je ukryt v uzlu se zkratkou *sd*, nejspíš s anglického scale degree. Stejně jako u not má tato vlastnost rozsah 1 až 7 podle stupně tóniny popřípadě *rest*. Vlastnost a zkratka *fb* je z anglického figural bass, tedy číslovaný bas. Číslo ve vlastnosti značí, na kterém stupni se nachází základní tón akordu[22]. Zkratka *sec* mění základní tóninu, od které se počítá stupeň akordu (*sd*). Akord, ve kterém je vyměněna tercie za kvartu, případně sekundu, se jmenuje suspended akord. Pokud se jedná o tento akord, pak vlastnost *sus* obsahuje řetězec *sus4* v případě kvarty, *sus2* v případě sekundy, nebo *sus42* v případě obou[32]. Akord lze také vyměnit za akord z jiného modu. Toto "vypůjčení"je zaznamenáno v uzlu *borrowed*. Složka *alternate* značí akordy alterované, tedy akordy s jedním nebo více tóny alterovanými [33].

Kromě těchto popsaných tabulek obsahuje databáze také další, které popisují žánr písničky, nebo její metriky zavedené a vypočítané webem hooktheory.com. Tato data ale nejsou vzata v úvahu, neboť s nimi nelze počítat ve standardních MIDI souborech.

[1] https://sqlite.org/index.html
[2] z anglického Beats per minute

5.2.1 Převedení databáze na MIDI soubory

Pro účely práce je nutné jednotlivé záznamy databáze konvertovat na množinu MIDI souborů. Proto byl vytvořen skript *dbToMIDI.py* v jazyce Python, který sekvenčně dotazuje jednotlivé části písniček v databázi a pro každou vytvoří samostatný MIDI soubor. Tyto soubory uloží do složky a pojmenuje ve formátu <id písničky>___part<id části>.mid. Cestu k databázi a jméno výstupní složky je nutno zadat jako vstupní parametry složky v tomto pořadí. Pro práci s databází skript využívá knihovnu *sqlite3*[3]. Data jako noty a akordy jsou v xml souboru, který je uložený v jednom ze sloupců. Soubor xml je dekomprimován metodou *decompress()* z knihovny *zlib*[4], Pro extrakci těchto dat je využita knihovna *lxml.etree*[5]. Manipulace s MIDI soubory je prováděna pomocí knihovny Mido[6]. Každý záznam části písničky v databázi má, mimo jiné, xml uzel s notami a uzel s akordy. Skript postupně projde nejprve všechny noty a poté akordy a na základě poskytnutých dat vypočítá výšky tónů, čas začátku a čas konce. S pomocí výsledných parametrů jsou vytvořeny *mido.Message* objekty, které abstrahují MIDI zprávy. Tyto objekty jsou poté vloženy do dvou objektů *mido.MidiTrack*. Jeden slouží pro ukládání melodických not a druhý pro harmonické akordy. Z těchto stop je vytvořen *mido.MidiFile* objekt, který zaštiťuje práci s MIDI souborem. Některé záznamy písniček obsahují pouze akordy nebo pouze noty, tedy melodii bez harmonie. Soubor je na tento neduh otestován a pokud obsahuje obě složky, pak je uložen. Ty, kterým jedna z částí chybí, uloženy nejsou. Z celkového počtu 13633 je tímto způsobem odstraněno zhruba 1500 částí písniček. Celková velikost všech souborů je 9.57MB (10 042700B)

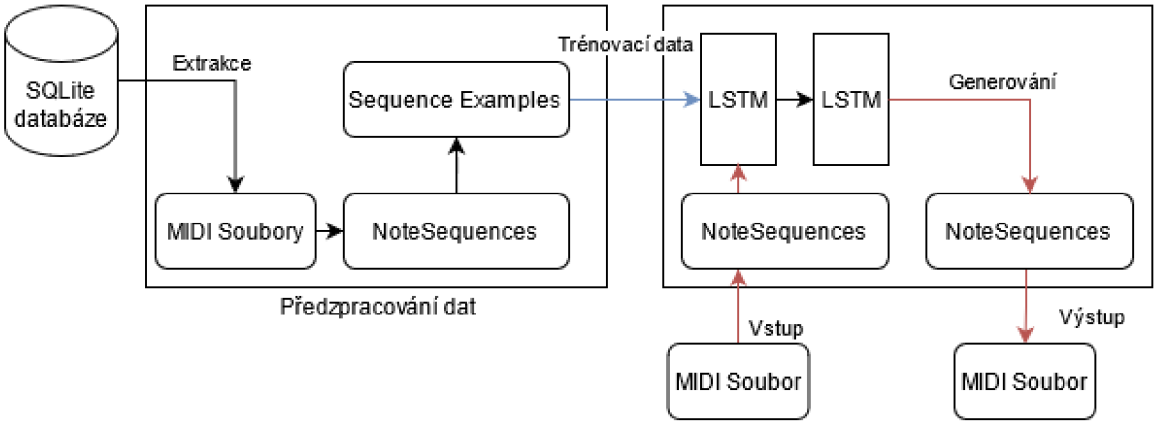
[3] https://docs.python.org/3/library/sqlite3.html
[4] https://docs.python.org/3/library/zlib.html
[5] https://lxml.de/
[6] https://github.com/mido/mido

Kapitola 6

Návrh systému

Cílem práce je vytvořit systém, který bude schopen automaticky vygenerovat harmonii k již existující melodii, což je úloha obtížnější, než prosté generování harmonie, neboť jednotlivé akordy nelze umisťovat svévolně. Vstupem systému jsou postupně jednotlivé noty melodie Jak bylo napsáno, harmonie je souzvuk více zvuků najednou. Výstupem jsou tóny melodie doplněné o tóny harmonie.

Architektura systému je uvedena na obrázku 6.1. Systém se skládá z několika částí, které na sebe navazují. Vstupem je melodie ve formátu MIDI, která je konvertována do posloupnosti not ve formátu NoteSequence. Výstupem jsou opět noty ve stejném formátu, který je konvertován do formátu MIDI. Pro potřeby učení jsou použity anotovaná data ve formátu SequenceExample.



Obrázek 6.1: Diagram architektury navrženého systému.

6.1 Vstup

Vstupem celého systému je MIDI soubor, obsahující melodii, pro kterou je cílem vygenerovat harmonický doprovod. Soubory MIDI jsou získány z databáze. Postup je popsán v kapitole 5.2. Formát MIDI však je příliš komplikovaný, aby byl zpracováván přímo neuronovou sítí. Z tohoto důvodu jsou vstupní data transformována na posloupnost not. Tato posloupnost je uložena pomocí formátu Protocol buffer [17], který umožňuje nezávislost na platformě a jazyce a který je nativně podporován frameworkem TensorFlow. Protocol buffers je mecha-

nismus firmy Google pro serializaci strukturovaných dat. Veškeré protocol buffers zprávy jsou uložené v jednom .tfrecord souboru. Definici zpráv určují takzvané .proto soubory. V projektu Magenta se pro serializaci not používá jednotná definice zvaná NoteSequence [15].

Pro trénování je vhodné samostatné serializované noty dále převést na tzv. SequenceExamples [16], což je TensorFlow definice protocol bufferů určená k práci se sekvenčními daty. Definice obsahuje dvě části – context a feature_lists. Kontext obsahuje nesekvenční vlastnosti (features) jako například tempo a feature_lists obsahuje sekvenční vlastnosti, jednotlivé noty. Výhodou SequenceExamples je možnost distribuce trénování, kdy je možné rozdělit jeden .tfrecord soubor na více. Další výhodou je znovupoužitelnost modelu, který SequenceExamples využívá. Nevýhodou je velikost výsledných .tfrecord souborů s touto definicí oproti původním datům [3]. Dle dokumentace [3] může použití tohoto formátu způsobit desetinásobný nárůst velikosti dat. Původní data však v článku uvedena nebyla a je tedy nemožné specifikovat důvod malého nárůstu. V našem v případě byl zaznamenán 9410 násobný nárůst velikosti, neboť z 12103 MIDI souborů zabírajících 10MB celkem bylo vygenerován soubor s příponou .tfrecord zabírajícíc celkem 94GB dat (viz výpočet podle měření 6.1 níže).

$$tf/m = 94454904591B/10038169B = 9.409575e^3 \tag{6.1}$$

kde tf je velikost všech výsledných .tfrecord souborů a m je velikost všech vstupních MIDI souborů.

6.2 Model

Z modelů představených v kapitole 4 byl vybrán Polyphony RNN z projektu Magenta. V porovnání s ostatními přístupy je nejnovější a využívá framework TensorFlow [35, 16]. Magenta poskytuje model[1] již natrénovaný na chorálech, které složil Johann Sebastian Bach[2].

Kromě toho lze model natrénovat s pomocí vlastních dat, kdy je také možné nastavit hyperparametry sítě jako například počet a velikost RNN vrstev [16].

Melodie bude představovat v tomto případě tzv. primer modelu. Vstupní soubor je transformován na NoteSequence strukturu přesně, jak bylo uvedeno v předchozí sekci. Úkolem systému je doplnění této melodie o složku harmonie. Melodie proto bude použita jako součást výstupu.

6.3 Výstup

Výstupem samotného modelu je protocol buffer kódující obsah struktury NoteSequence, stejně jako na vstupu. Tato NoteSequence je interně převedena na výsledný MIDI soubor, který obsahuje původní melodii(primer), která je následována její harmonizovanou podobou [16].

[1] dostupný z http://download.magenta.tensorflow.org/models/polyphony_rnn.mag

[2] dostupné z https://web.archive.org/web/20150503021418/http://www.jsbchorales.net/xml.shtml, popřípadě z https://github.com/cuthbertLab/music21/tree/master/music21/corpus/bach

Kapitola 7

Trénování systému

V této kapitole je představen postup zprovoznění navrženého systému a jeho trénování. Všechny kroky byly automatizovány pomocí skriptu v jazyce BASH. Vstupní data a postup jejich vytvoření je popsán v kapitole 5.2. Využitý systém je představen v části 4.4 a podrobnější návrh potom v 6. Postup harmonizace melodie modelem a provedené experimenty jsou popsány v kapitole následující.

7.1 Instalace

Před začátkem trénování je nutné nainstalovat Magentu. Jako testovací prostředí byl zvolen operační systém Ubuntu 20.04 LTS (Focal Fossa) nainstalovaný na počítači s procesorem Intel Core i7-4720HQ s frekvencí 2.6GHz a s 8GB operační paměti.

Pro instalaci knihovny na použitý operační systém je poskytnut instalační skript.

```
URL=https://raw.githubusercontent.com/tensorflow/ \
magenta/master/magenta/tools/magenta-install.sh
curl $(URL) > /tmp/magenta-install.sh
bash /tmp/magenta-install.sh
```

Instalační skript nainstaluje Python distribuci Anaconda[1] a s pomocí systému pro správu balíků *conda* vytvoří nové virtuální prostředí *magenta*. Do tohoto prostředí dále nainstaluje potřebné balíky. Mimo jiné například TensorFlow, magenta, scipy a numpy. Pro prohlížení průběhu trénování je také nainstalován vizualizační nástroj TensorBoard. Po dokončení instalace je potřeba restartovat okno s terminálem, aby se načetly nové, změněné proměnné prostředí. Nové, virtuální prostředí, které obsahuje korektní instalaci knihovny Magenta, se aktivuje příkazem

```
source activate magenta
```

Nyní jsou knihovna Magenta a všechny její závislosti připraveny k použití v Pythonu a v Jupyter sešitech. Mimo to jsou Magenta skripty zavedeny v proměnné *PATH*. [16]

Pro automatizaci celého procesu instalace byly vytvořeny skripty *prerequisities.sh* a *install.sh*. První nainstaluje potřebné prerekvizity a vytvoří virtuální prostředí. Druhý skript do virtuálního prostředí nainstaluje veškeré vyžadované python balíčky. Z důvodu instalace prerekvizit skript vyžaduje administrátorská práva. Pokud počítač neobsahuje systém pro

[1] https://www.anaconda.com/

správu balíků Anaconda, skript *prerequisities.sh* se jej pokusí nainstalovat sám. V takovém případě je nutné odsouhlasit autorská práva a jiné požadavky této instalace. Po instalaci prerekvizit prvním skriptem je vhodné restartovat okno s terminálem a teprve spustit skript *install.sh*.

7.2 Příprava dat

Dalším krokem je převedení sady trénovacích MIDI souborů na protocol buffer soubor s definicí NoteSequence. K tomuto účelu existuje skript *convert_dir_to_note_sequences*, který sekvenčně načte složku obsahující MIDI (případně MusicXML, nebo ABC) soubory, zkonvertuje je na NoteSequence a výsledek uloží do .tfrecord souboru. Jeho chování lze ovlivnit argumenty. Vstupní složka a výsledný soubor jsou jediné dva povinné parametry. Parametry pro nastavení jsou *input_dir*, respektive *output_file*. Pro rekurzivní procházení adresářů je nutno nastavit parametr *recursive*. [16] Použití pak vypadá následovně:

```
convert_dir_to_note_sequences \
    --input_dir=./midi \
    --output_file=./tmp/notesequences.tfrecord \
    --recursive
```

Konverze souborů ze složky *./midi* netrvá dlouho a výsledný soubor *./tmp/notesequences.tfrecord* má velikost 29.7MB (31244793B)

Nyní je nutné vytvořit SequenceExample soubory, s jejichž pomocí bude model trénován a vyhodnocován. Každá SequenceExample zpráva obsahuje sekvenci vstupů a sekvenci příslušných označení, která reprezentuje polyfonní, harmonizovanou sekvenci. Toho lze dosáhnout voláním příkazu *polyphony_rnn_create_dataset*. Pomocí povinných parametrů *input* a *output_dir* je nastaven vstupní soubor a složka pro uložení výsledků. Parametrem *eval_ratio* se nastavuje, kolik procent dat bude použito jako datová sada určená k vyhodnocení modelu a kolik jako datová sada k trénování modelu. [16] Příkaz se správnými argumenty je

```
polyphony_rnn_create_dataset \
    --input=./tmp/notesequences.tfrecord \
    --output_dir=./tmp/polyphony_rnn/sequence_examples \
    --eval_ratio=0.10
```

Generování trénovacích dat trvá zhruba čtyři až šest hodin. Ve složce ./tmp/polyphony_rnn/sequence_examples budou po dokončení vygenerovány dva soubory. Jeden s trénovacími daty, *training_poly_tracks.tfrecord*, který obsahuje 90 % dat, a druhý, *eval_poly_tracks.tfrecord* s daty pro vyhodnocení modelu. Tento obsahuje zbývajících 10 % dat. Velikost obou souborů dohromady je $8.39GB + 79.5GB = 87.9GB$ (94 454 904 591B).

Celý proces automatizuje skript *prepareData.sh*, který má minimálně jeden a maximálně dva argumenty. První argument specifikuje cestu ke složce s MIDI soubory. Tento argument je povinný. Pokud je toto jediný argument, počítá se s tím, že složka MIDI soubory už obsahuje. Skript také automatizuje konverzi databáze, která je popsaná v kapitole 5.2. V tomto případě je opět nutné jako první argument zadat složku do které budou ukládány výsledky a jako druhý argument cestu k souboru se SQLite databází. Tímto způsobem bude převedena databáze na SequenceExample soubor, který je možné použít pro trénování sítě.

Skript lze také použít pouze na soubor s NoteSequence prvky. V tom případě se jako první argument musí zadat přepínač *-s*, a jako druhý argument cesta k .tfrecord souboru obsahující NoteSequence zprávy.

Ve všech případech se jedná o časově náročnou operaci, která může trvat až 8 hodin.

7.3 Trénování

Nyní je vše připraveno k trénování modelu. Příslušný příkaz lze ovlivnit několika parametry. S pomocí *run_dir* je určena cesta k adresáři, který je určen k ukládání průběžných výsledků trénování (checkpoints) a k ukládání souhrných dat pro TensorBoard. Uvnitř tohoto adresáře jsou dva podadresáře, do nichž jsou uložena zvlášť data z trénování a z vyhodnocení modelu. Souhrnná data se ukládají každý desátý krok. Pro změnu kroku existuje parametr *summary_frequency*. Průběžné výsledky trénování nejsou ukládány všechny, nýbrž pouze posledních deset. Toto číslo lze opět změnit parametrem *num_checkpoints*. Při jeho nastavení na nulu jsou uloženy všechny a žádný se nemaže. Cesta k trénovacím, případně vyhodnocovacím datům, je dána parametrem *sequence_example_file*. Nepovinný parametr *num_training_steps* určuje, kolik trénovacích cyklů model prodělá, než se trénování ukončí. Pokud tento není zadán, trénování pokračuje nepřetržitě, dokud není manuálně ukončeno. Vyhodnocení modelu se aktivuje parametrem *eval*. Při jeho nastavení nejsou aktualizovány váhy modelu. Posledním důležitým, ale nepovinným parametrem je *hparams*. Tímto lze změnit základní nastavení hyperparametrů modelu. Jedná se o seznam dvojic klíč-hodnota oddělených čárkami, kde klíč je jméno měněného hyperparametru a hodnota je nová hodnota, na kterou bude tento změněn. Takto zadané hyperparametry jsou sloučeny s původními. Jména hyperparametrů jsou batch_size a rnn_layer_sizes. Prvním se nastavuje počet SequenceExample zpráv, trénovacích dat, které jsou použity na jeden průchod sítí, po kterém se provede aktualizace vah. Základní nastavení je 128 zpráv. Druhým se nastavuje velikost sítě. Jedná se o pole čísel, kde každý prvek pole značí jednu RNN vrstvu, a hodnota prvku udává počet neuronů dané vrstvy. Výchozí hodnoty vytvoří síť se třemi vrstvami, každá o 256 neuronech. [16] Pro zkrácení doby učení a snížení paměťové náročnosti budou použity jiné, nižší hyperparametry. Příkaz pro trénování Polyphony RNN na vlastní datové sadě tak bude vypadat následovně:

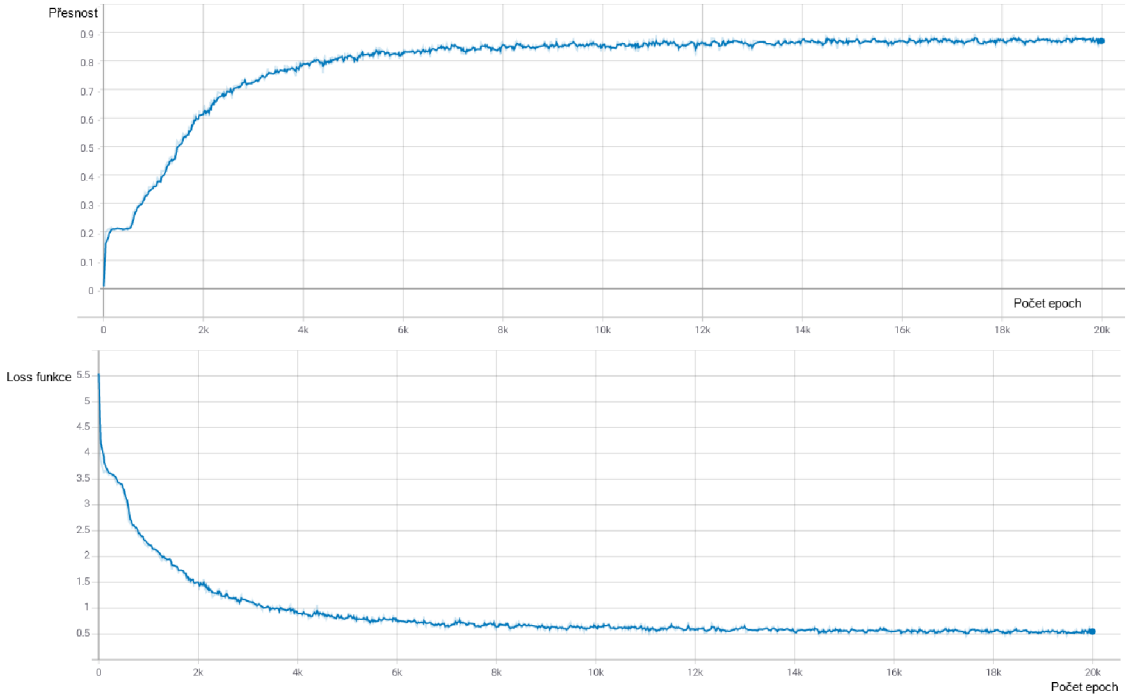
```
polyphony_rnn_train \
    --run_dir=./tmp/polyphony_rnn/logdir/run1 \
    --sequence_example_file=./tmp/polyphony_rnn/sequence_examples\
          /training_poly_tracks.tfrecord \
    --hparams="batch_size=64,rnn_layer_sizes=[64,64]" \
    --num_training_steps=20000
```

S tímto nastavení trvá trénování na testovacím počítači přibližně 25 hodin. Skript průběžně zapisuje statistiky o právě probíhajícím cyklu, včetně průměrného počtu cyklů za vteřinu. Pomocí této hodnoty lze odhadnout dobu trvání trénování při jiných hyperparametrech, například těch výchozích. Výchozí hodnoty popisují mnohem větší model o třech RNN vrstvách s 256 neurony v každé. Ovšem ani po uplynutí dvaceti hodin od začátku trénování tohoto většího modelu, se nedokončil jediný trénovací cyklus. Odhadem by tak trénování na tomto stroji trvalo doslova desítky let (viz odhad 7.1 níže, kde t je odhadnutý čas jednoho cyklu a x je odhad doby trvání trénování). Trénování větších modelů je tak podmíněno použitím počítačů s větší výpočetní silou.

$$x = t * num\_training\_steps = 20h \times 20000 \approx 45\text{let} \tag{7.1}$$

Volitelně je možné paralelně spustit stejný skript s přidaným parametrem *eval*, což spustí vyhodnocovací úlohu. V tomto případě bude po každém trénovacím cyklu proveden vyhodnocovací cyklus. Vyhodnocení se provede vždy po dokončení cyklu učení a není tedy možné tento příkaz spustit po dokončení trénování. [16] Průběh trénovacího a vyhodnocovacího procesu lze sledovat na adrese http://localhost:6006 po spuštění TensorBoard nástroje příkazem

```
tensorboard --logdir=./tmp/polyphony_rnn/logdir
```



Obrázek 7.1: Horní graf zobrazuje přesnost modelu v závislosti na čase trénování. Spodní pak ztrátu modelu.

Na obrázku 7.1 lze vidět ztráta a přesnost modelu v průběhu učení. Horizontální osy zobrazují počet cyklů učení (epoch). Je vidět, že naměřená ztráta se postupem učení snižuje, zatímco přesnost modelu se zvyšuje

Po dokončení učení modelu je možné s ním experimentovat a nechat jej harmonizovat melodie uložené v MIDI souborech. Případně je možné používat model, který nedokončil všechny cykly určené v parametrech. V tom případě se použijí váhy z posledního dokončeného cyklu.

Naučený model ve formě posledních n checkpointů lze transformovat na více kompaktní, zabalený soubor. Ten, kromě posledních checkpointů, obsahuje také graf a některá metadata modelu. Příkaz *polyphony_rnn_generate* s příznakem *save_generator_bundle* načte model ze složky uvedené v parametru *run_dir*. Pro správnou funkci je nutné dodat stejný popis hyperparametrů v parametru *hparams*, jaký byl použit při trénování. Cesta k výslednému .mag souboru je určena v *bundle_file*. Jedná se opět o zabalený protocol buffer, nyní však

s definicí GeneratorBundle. Zabalení modelu do jednoho souboru je vhodné při sdílení předtrénovaného modelu. [16]

```
polyphony_rnn_generate \
    --run_dir=./tmp/polyphony_rnn/logdir/run1 \
    --hparams="batch_size=64,rnn_layer_sizes=[64,64]" \
    --bundle_file=./tmp/polyphony_rnn.mag \
    --save_generator_bundle
```

Proces tréningu automatizují dva skripty, *train.sh* a *eval.sh* Oba vyžadují dva argumenty. Prvním z nich je cesta k adresáři pro ukládání průběžných výsledků a druhý je cesta k souboru datové sady sloužící pro tréning, nebo evaluaci. Po dokončení tréningu je model automaticky zabalen jak bylo popsáno výše.

7.4 Použití modelu

Generovat polyfonní a harmonizované skladby s použitým modelem lze téměř od samého začátku trénování. Kromě toho lze použít také model zabalený v .mag souboru. Není tedy nutné trénovat vlastní model, ale je možné využít již předtrénovaný a exportovaný do tohoto formátu. Chování generátoru upravuje několik parametrů. Výběr modelu specifikují parametry *run_dir* a *bundle_file*, kdy první určuje cestu ke složce s trénovaným modelem (přesněji s uloženými checkpointy) a druhý určuje cestu k .mag souboru s již natrénovaným modelem. Při zadání obou má prioritu druhý popisovaný parametr. Při použití parametru *run_dir* je nutné uvést také parametr *hparams* tak, jak byl použit při trénování (některé z uvedených hyperparametrů budou ignorovány, například batch_size). Složku pro uložení vygenerovaných MIDI souborů vymezuje parametr *output_dir*. Počet nových souborů pak parametr *num_outputs*. Každý generovaný soubor je rozdílný, liší se provedením harmonizace. Je vhodné výsledky poslechnout a zjistit jejich kvalitu.

Délka výsledné skladby je dána parametrem *num_steps* Jednotka délky kroku je v tomto případě šestnáctinová nota, jinými slovy – do vygenerované melodie se vleze právě *num_steps* šestnáctinových not. Délka jednoho taktu je šestnáct kroků. Problém tohoto modelu je, že do generované skladby nejdříve vloží melodii samotnou, jako primer popsaný v kapitole 6. Je tedy nutné délku původní melodie násobit dvěma, aby se do výsledného souboru dostala jak původní melodie, tak její rozšířená část. [16]

Základ generátoru (primer) lze specifikovat třemi způsoby. Jelikož je model původně určený pro generování polyfonních skladeb, je možné začít pouze jedním akordem. Ten se vkládá parametrem *primer_pitches*, který obsahuje řetězec reprezentující Python pole. Jednotlivé prvky tohoto pole jsou MIDI výšky tónů, z nichž je tento akord složen. Ve vygenerované skladbě trvá základ, vložený tímto způsobem, první dobu (délka čtvrťové noty). Například celý parametr určující akord C dur vypadá následovně:

```
--primer_pitches="[67, 64, 60]"
```

Další možností je vložení počáteční melodie. Ta je vložena ve formě Python pole, stejně jako v předchozím případě. Noty ve formě MIDI výšek jsou doplněny o dvě řídící hodnoty. Žádnou událost značí číslo -2 a deaktivace znějící noty (note-off) je reprezentována číslem -1. Parametr pro tuto možnost je *primer_melody*.

```
--primer_melody="[60, -2, 60, -2, 67, -2, 67, -2 ]"
```

Poslední možností je načtení základní stopy z MIDI souboru, jehož cesta je v parametru *primer_midi*.

```
--primer_midi=./primer.mid
```

Při použití více než jednoho parametru pro vložení základu mají parametry následující prioritu:

1. primer_pitches
2. primer_melody
3. primer_midi

Další parametry upravují použití základu generátorem. Pokud je argument *condition_on_primer* nastaven na true, pak je primer načten modelem ještě před začátkem generování nové sekvence. Tento argument se používá dohromady s *primer_pitches* ke stanovení tóniny generované stopy jejím akordem. Nastavením parametru *inject_primer_during_generation* na true bude poskytnutý základ použit jako součást generované skladby. Toto chování se hodí právě pokud je úkolem harmonizace existující melodie. V takovém případě by se neměl používat předchozí parametr *condition_on_primer*, protože model nejdříve uvidí monofonní melodii a má generovat polyfonní. [16]

Posledním pro experimenty zajímavým parametrem je *temperature*, který ovlivňuje náhodnost generovaných stop. Číslo větší než jedna znamená více náhodný výsledek, naopak číslo menší než jedna vede k menší náhodnosti. Výchozí hodnota je pochopitelně 1.0.

Celý příkaz pro vygenerování harmonie k melodii uložené v MIDI souboru vypadá následovně:

```
polyphony_rnn_generate \
    --bundle_file=./tmp/polyphony_rnn.mag \
    --output_dir=./generated \
    --num_outputs=10 \
    --num_steps=268 \
    --primer_midi=./melody.mid \
    --condition_on_primer=false \
    --inject_primer_during_generation=true
```

Harmonizaci skladby lze provést také skriptem *harmonize.sh*, který už obsahuje všechny potřebné parametry. Je ale potřeba v parametrech specifikovat cestu k *bundle_file* souboru, který obsahuje natrénovaný model. Dále cestu k MIDI souboru určenému k harmonizace, a nakonec počet taktů, které tento soubor obsahuje.

Kapitola 8

Experimenty

Tato kapitola popisuje možnosti natrénovaného modelu a jsou představeny provedené experimenty. Model jako takový je popsán v kapitolách 4.4 a 6, postup jeho zprovoznění, trénování a použití pak v kapitole 7. Použitá trénovací sada je popsána v kapitole 5.2. Model, na kterém byly experimenty prováděny, byl natrénován s hyperparametry, které byly představeny v minulé kapitole.

Jak bylo uvedeno v kapitole 7, natrénování modelu s těmito hyperparametry trvalo 25 hodin. Parametry použité při generování, pokud není stanoveno jinak, jsou:

- condition_on_primer = false
- inject_primer_during_generation = true
- bundle_file = ./tmp/polyphony_rnn.mag
- num_outputs = 10
- num_steps = 64
- temperature = 1.0

Výsledné MIDI soubory byly analyzovány s pomocí open-source sequenceru a MIDI editoru Sekaiju 6.1[1]. Obrázky not byly generovány programem MuseScore[2], který je vydáván pod licencí GNU GPL.

8.1 Způsob vyhodnocení výsledků

Měření výkonu v této oblasti není jednoduché. Z důvodu širokého spektra cílů výzkumů v oblasti generování hudby je možné setkat se v existující literatuře s různými přístupy k vyhodnocování výstupů harmonizačních algoritmů. Jedná se například o porovnávání vygenerované sekvence akordů s původními akordy z anotované skladby, dále porovnávání jednotlivých generovaných akordů nebo vyhodnocení subjektivním poslechem. [30] Posledním způsobem byl hodnocen například projekt JamBot uvedený v části 4.3 (v článku [4]).

Pro vyhodnocení poslechem je možné využít, mimo jiné, placených crowdsourcingových služeb jako například Amazon MTurk. Využít masy lidí pro rozličné účely (Crowdsourcing), které zahrnují i poslech generované hudby, však lze bezplatně. Například skupina, která

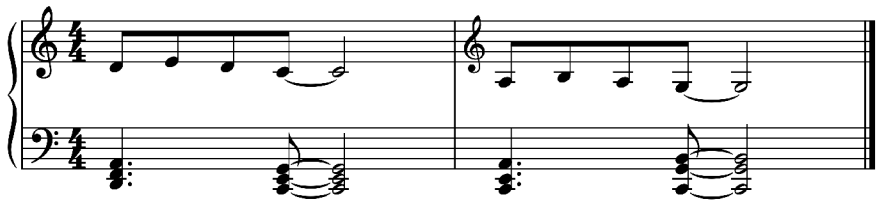
[1] dostupný z openmidiproject.osdn.jp/Sekaiju_en.html
[2] dostupný z musescore.org/en

stojí za projektem BachBot, vyvinula testovací prostředí svých nahrávek na webu a ten šířila virálně skrz sociální sítě. [25] Některé články dokonce neposkytují žádné vyhodnocení své harmonizace. [30] Vzhledem k povaze navrženého systému budou výsledky harmonizace manuálně hodnoceny subjektivním poslechem. Na základě empirických zkušeností s hudbou bude určeno, zda je skladba harmonizována správně. Předpokladem je schopnost rozlišit úplnou disharmonii od harmonie.

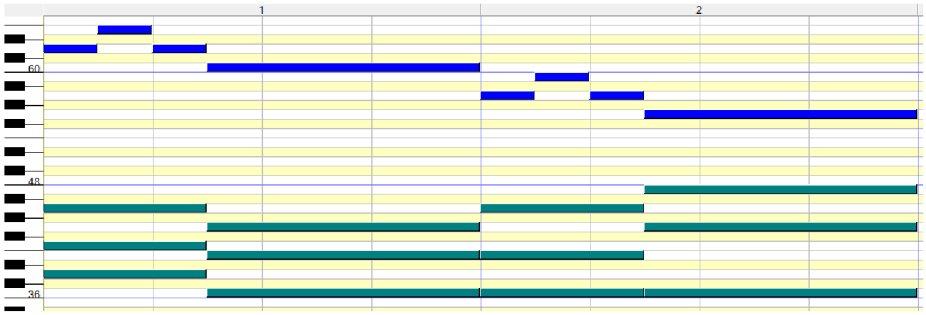
Dalším sledovaným parametrem je čas, který systém potřebuje k harmonizaci jedné melodie.

8.2 Harmonizace krátké známé melodie

Prvním experimentem nechť je harmonizace skladby o pouhých dvou taktech, která se nacházela v trénovací datové sadě (viz noty na obrázku 8.1, případně pianoroll MIDI souboru na obrázku 8.2). Modelu známá melodie by měla vyústit ve harmonii velmi podobnou té původní. Naproti tomu je melodie velmi krátká. Pro extrakci samotné melodie z MIDI souboru obsahující tuto skladbu byl vytvořen Python skript *./harmonizer/melodyFromMidi.py*, který jako první parametr přijímá cestu k tomuto souboru. Z původního souboru *./midi/590914___part590914.mid* tímto způsobem vytvořen soubor *./melodyOnly/590914___part590914_Short.mid.melodyOnly.mid*.



Obrázek 8.1: Notový zápis použité skladby. Horní osnova je melodie, spodní harmonie.



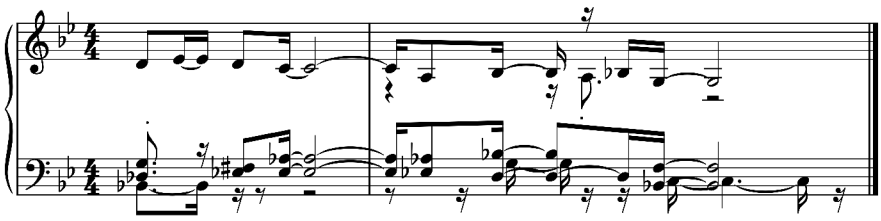
Obrázek 8.2: Pianoroll použité skladby. Modré obdélníky jsou melodie, zelené harmonie. Vertikální osa jsou MIDI výšky tónů, zatímco horizontální je čas. Čísla nahoře značí první a druhý takt skladby.

Harmonizace byla provedena čtyřikrát s různou hodnotou parametru temperature. Nejprve byly generovány soubory s výchozí hodnotou (1.0), poté s nižší (0.7) a nakonec dvakrát

s vyššími hodnotami (1.3, respektive 2.0). Při každém běhu bylo vytvořeno 10 souborů, které byly ručně analyzovány a poslechnuty.

Vygenerování deseti souborů trvá 8 vteřin, z nichž první čtyři probíhá inicializace systému a samotné generování trvá 4 vteřin. Průměrně tak vygenerování jedné skladby trvá 0.2 vteřiny, plus 5 fixních vteřin kvůli inicializaci. Při uvážení délky harmonizované skladby (dva takty) lze předpokládat, že harmonizace jednoho taktu trvá 0.1 vteřiny. Při zvážení délky přehrávání jednoho taktu, a sice 2 vteřiny, je takový výsledek příznivý.

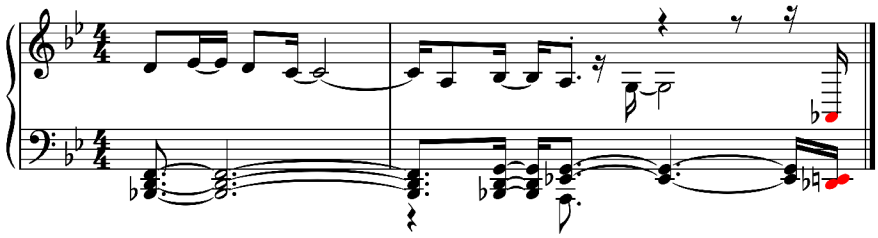
Skladby generované s nízkou hodnotou temperature vykazovaly, jak předpovídala dokumentace, vyšší náhodnost vytvořených tónů. Model většinou používá mnoho krátkých akordů, ne vždy ale začínají ve správný čas, akordy nedoplňují melodii a celkově je poslech těchto skladeb ne příliš příjemný. Existují ovšem výjimky, viz notový zápis na obrázku 8.3, kde doplněné noty původní melodii rozvádí přechody mezi jednotlivými akordy.



Obrázek 8.3: Notový zápis skladby vygenerované s nižšími než výchozími hodnotami.

Harmonizace prováděná s výchozí hodnotou temperature byly obecně příjemnější k poslechu než předchozí přístup. Akordy začínaly zároveň s tóny původní melodie a byly delší. U některých výsledků se však vyskytoval jistý nešvar, totiž tóny akordů trvající velmi dlouho, někdy i déle než jeden takt. Poslech tak ovlivňuje i nastavený hudební nástroj při syntéze MIDI souboru, neboť některé se rozezní pouze na začátku MIDI události, ale brzy dozní a po zbytek události je tón neslyšitelný. Pro správný poslech je tak nutné nastavit MIDI nástroj, který zní celou dobu (například *16-Varhany*, nebo *40-Housle*). Toto chování lze připsat použité datové sadě, kdy se na serveru hooktheory vkládají většinou pouze harmonizační funkce akordů, které trvají i více taktů a nemění se tak často.

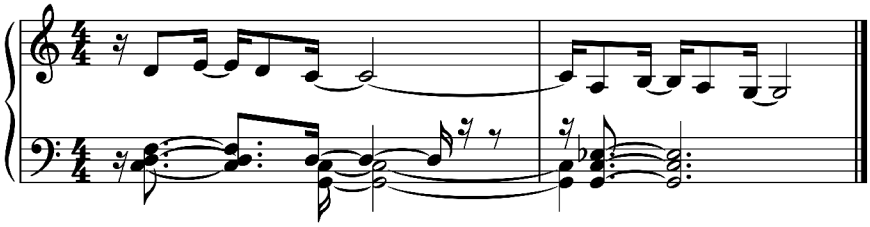
V některých případech také model těsně před koncem skladby vygeneruje změnu akordu, který má tendenci v melodii pokračovat (viz obrázek 8.4).



Obrázek 8.4: Notový zápis vygenerované skladby s výchozím parametrem temperature. Červené noty znázorňují náhlou změnu na konci skladby.

Dva běhy harmonizace s parametrem temperature vyšším než původním dále prodlužovaly použité akordy. Harmonie byla také více předvídatelná a tolik se neměnily použité

akordy. Pokusy o pokračování skladby se objevovaly spíše zřídka. Mezi výsledky procesů s hodnotami parametru, které byly vyšší než jedna, nebyl pozorován zásadní rozdíl. Zdařilá harmonizace s tímto nastavením je na obrázku 8.5.



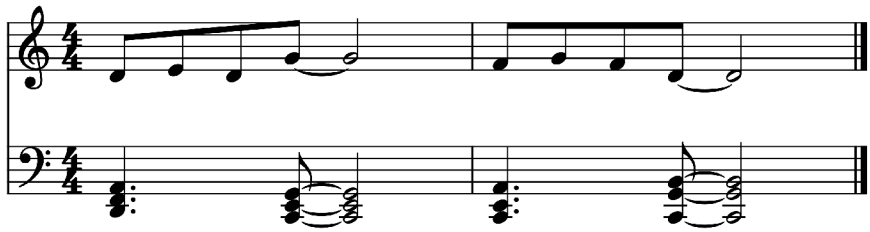
Obrázek 8.5: Notový zápis vygenerované skladby s vyšším parametrem temperature. Tóny harmonie trvají několik dob.

Za povšimnutí stojí fakt, že ani jediná z vygenerovaných harmonizací se úplně nepodobá té původní z trénovací datové sady. Dále si je patrné, že model používá spíše nestandardní a nelibozvučné souzvuky a standardní durové, nebo mollové akordy volí spíše zřídka. Automatická harmonizace s pomocí tohoto modelu na krátkých skladbách je tak spíše nefunkční, ikdyž se mu v několika málo případech podařilo vygenerovat harmonii (nebo alespoň její část) správnou a poslouchatelnou. Důvodem může být velikost modelu (dvě RNN vrstvy o 64 neuronech každá), oproti autory zamýšlené výchozí velikosti (tři RNN vrstvy o 256 neuronech každá). Nedokáže se tak naučit složité znaky používané v harmonii.

8.3 Harmonizace krátké neznámé melodie

Dalším experimentem je harmonizace krátké melodie, podobné té z prvního experimentu (viz obrázek notového zápisu na obrázku 8.6, MIDI soubor *./melodyOnly/MyHarmonyShort.mid*). Tato melodie byla vytvořena ručně a nenacházela se v trénovací sadě. Stejně jako při předchozím experimentu je harmonizace provedena v několika běžích, vždy s jiným parametrem temperature. Hodnoty parametrů jsou zvoleny stejně, tedy 0.7, 1.0, 1.3 a 2.0.

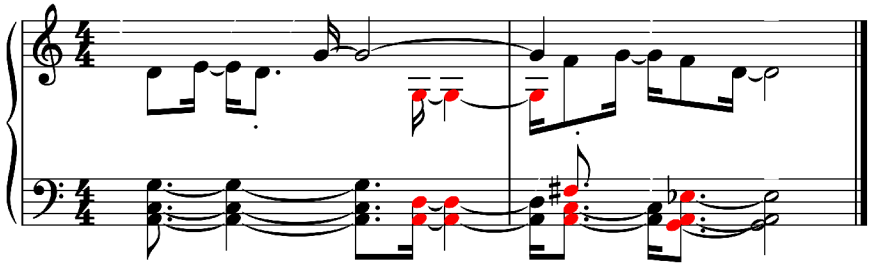
Čas potřebný k harmonizaci skladby se oproti předchozímu experimentu nezměnil. Harmonizace deseti skladeb opět trvala 8 vteřin se stejným rozdělením času mezi inicializaci systému a generování harmonie.



Obrázek 8.6: Notový zápis modelu neznámé melodie, s příslušnou harmonií.

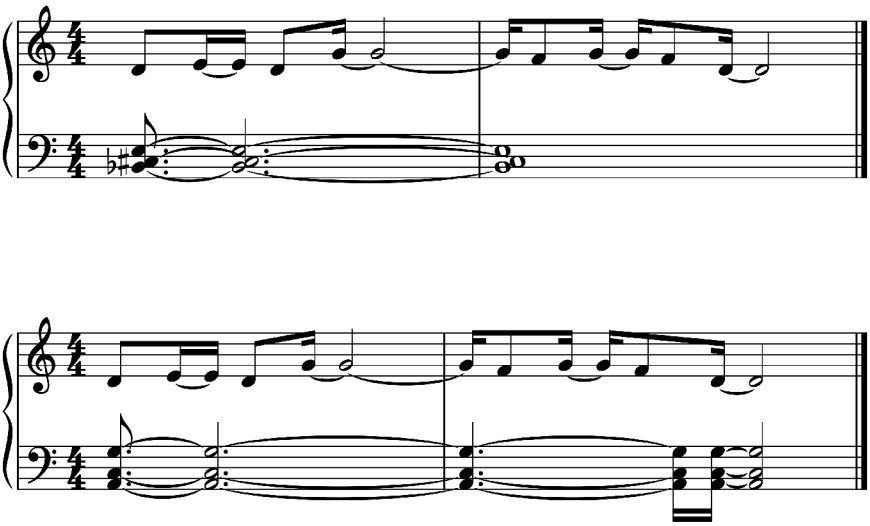
Skladby generované s použitím nízké hodnoty parametru temperature, oproti předchozímu experimentu, obsahují delší akordy. Ty se ale také často mění. Většinou ve špatnou

dobu, v průběhu jiné noty. Skladby tak zní velmi rušivě a disonantně. To je dobře vidět na obrázku 8.7, kde červené noty značí náhlé změny v harmonii. Poslech těchto skladeb, stejně jako u předchozího experimentu (při použití stejné hodnoty temperature), není příliš příjemný.



Obrázek 8.7: Notový zápis harmonizace neznámé skladby s nízkou hodnotou temperature. Červené noty jsou špatně umístěné.

Výchozí hodnoty parametru temperature přináší delší akordy. Na rozdíl od předchozí hodnoty se zde vyskytuje hodně skladeb, jenž mají jediný akord roztažený po celou dobu trvání melodie. Mezi jednotlivými skladbami se však mění právě tento použitý akord. Tímto se mění nálada a vyznění melodie. Toto chování je demonstrováno na obrázku 8.8.

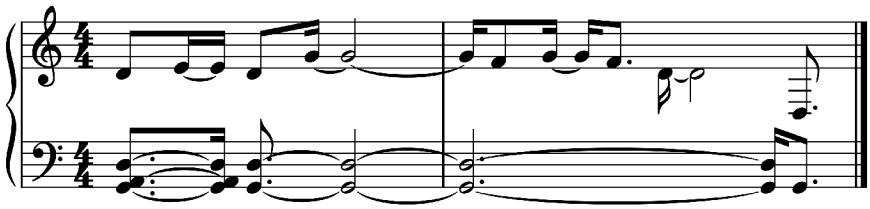


Obrázek 8.8: Notový zápis harmonizací neznámé skladby s výchozí hodnotou temperature. Za povšimnutí stojí použití jednoho akordu pro celou melodii, který je rozdílný pro obě skladby (horní a dolní).

Generování harmonie s parametrem temperature vyšším se opět vyznačuje dlouhými akordy. Rozdílem oproti nižší hodnotě je jejich opakování ve skladbě. Tedy stejný akord se rozezní několikrát po sobě. V některých případech je jeden z tónů akordu vyměněn. Na

obrázku 8.9 je také poznat, že model není úplně obeznámen s tím, že akord by měl být souzvuk tří tónů, a ne pouze dvou.

Nejvyšší použitá hodnota se vyznačuje nejlépe zvládnutou harmonií. Povedené skladby obsahují dva až tři akordy, které s poskytnutou melodií zní. Navíc jsou zavedeny ve správný čas a výsledný souzvuk dojem ze skladby nepoškozuje, jako tomu bylo u předchozích pokusů.



Obrázek 8.9: Notový zápis harmonizace neznámé skladby s vyšší hodnotou temperature.

Systém se při harmonizaci těchto skladeb dopouští velmi podobných chyb, jako v předchozím experimentu. Příčiny tohoto chování lze spatřovat opět v malém modelu. Pokud si není schopen zapamatovat krátkou melodii z trénovací sady, je pravděpodobné, že neznámou melodii harmonizovat lépe nebude.

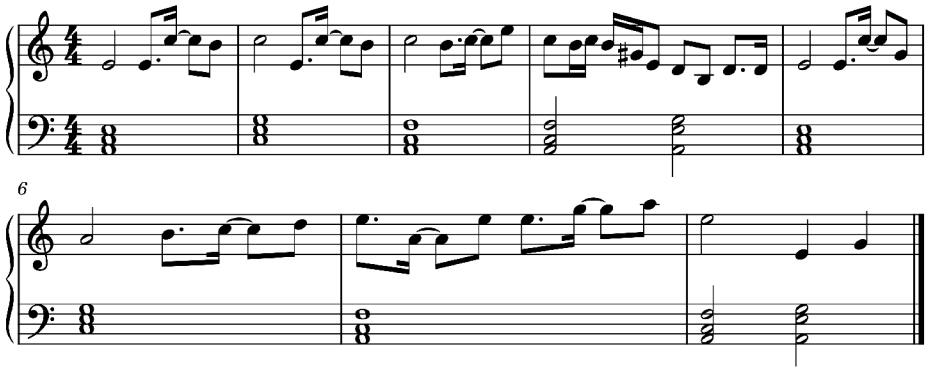
8.4 Harmonizace dlouhé melodie

V tomto experimentu má model za úkol harmonizovat delší skladbu. Oproti předchozím skladbám je tato dlouhá osm taktů, je tedy čtyřikrát delší (viz obrázek notové osnovy 8.10). Kvůli této skutečnosti je potřeba změnit parametr *num_steps* z původní hodnoty 64 na 256. Parametr temperature bude nabývat stejných hodnot jako v předchozích případech. Cílem je zjistit, jak se model chová při harmonizaci delší skladby. Jestli model špatně zvládá harmonizovat jenom krátké skladby, nebo mu větší délka skladby pomůže.

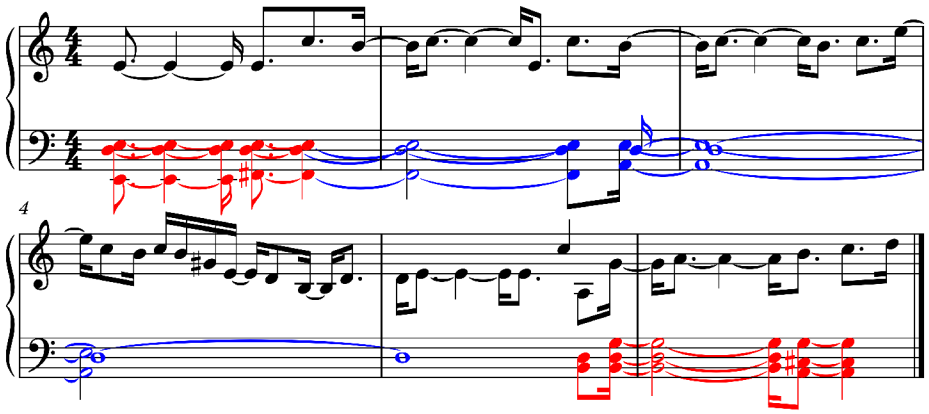
Generování deseti harmonizovaných souborů na základě delší melodie zabralo systému 16 vteřin. Inicializace opět trvala 4 vteřiny, čímž je potvrzena její fixní doba. Zbývajících 12 vteřin systém generoval jednotlivé soubory. Při deseti souborech to je průměrně 1.2 vteřiny na jeden soubor. Harmonizace každého taktu melodie tak v průměru zabere 0.15 vteřiny, což je pomalejší o 0.05 sekund oproti melodiím s dvěma takty.

U vygenerovaných skladeb v tomto experimentu lze pozorovat velmi podobné chování jako u předchozích. Celé skladbě většinou dominuje jediný akord, který zní po celou dobu skladby, čímž trpí syntéza MIDI souboru (viz výše). Kvalita výsledků se relativně zlepšuje s vyšším použitým parametrem temperature. To znamená, že pokud se akordy mění, je to ve správný čas. Ve výsledku však lze říci, že delší skladby, oproti těm krátkým, jsou harmonizovány lépe. Důvodem je, že v celé době trvání má systém více prostoru se projevit a vygenerovat části, které s původní melodií ladí.

Jako příklad za všechny z tohoto experimentu, nechť poslouží část notové osnovy na obrázku 8.11. Na tomto výtvoru je jak povedená část (vyznačená červeně), tak dva stejné, nepřiměřeně dlouhé akordy (vyznačené modře). Je také vidět, že poslední modrá část už není celý akord, ale samotný tón.



Obrázek 8.10: Původní notový zápis dlouhé skladby.



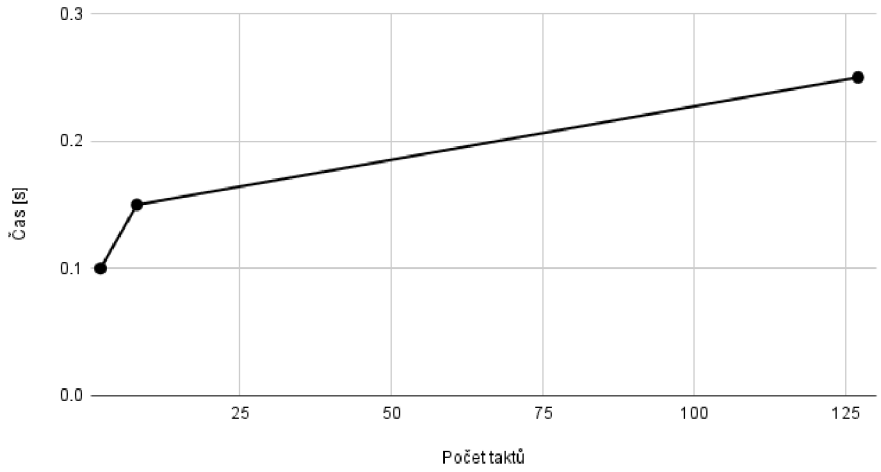
Obrázek 8.11: Část not povedené harmonizace delší melodie.

8.5 Harmonizace velmi dlouhé melodie

Posledním experimentem s modelem je harmonizace velmi dlouhé skladby. Uvažovaná skladba má délku 127 taktů, a tak je potřeba změnit paramet *num_steps* z původní hodnoty 64 na $32 * 127 = 4064$ kroků. Stejně jako u předchozích experimentů budou skladby generovány s parametrem temperature o hodnotách 0.7, 1.0, 1.3 a 2.0. Cílem je ověřit, že model vykazuje lepší výsledky při harmonizaci delších skladeb. Dále je cílem zjistit, o kolik se zvýší doba harmonizace jednoho taktu, s ohledem na předchozí experiment, kdy se doba zvýšila o 0.05 sekundy při použití čtyřikrát delší skladby.

Vygenerování deseti skladeb trvalo 5:24 minut. Po odečtení doby inicializace zbývá na generování 5:20 minut, tedy 320 vteřin. Průměrná doba harmonizace jedné skladby činí 32 sekund. Průměr každého taktu je tedy 0.25s. V porovnání s délkou harmonizace předchozích experimentů se jedná o nepatrné zvýšení o 0.15, respektive 0.1 vteřiny. Srovnání lze pozorovat na grafu na obrázku 8.12.

Z harmonizačního hlediska jsou na tom výsledky lépe než v předchozích experimentech. Ve skladbě nevadí dlouhé akordy (za předpokladu použití správného syntetizátoru), jelikož je model včas změní na jiný. Dlouhé akordy se, oproti předchozím experimentům, vyskytují spíše ve skladbách generovaných s nízkou (0.7) a výchozí (1.0) hodnotou temperature. Har-



Obrázek 8.12: Graf založený na datech ze všech experimentů, zobrazující vztah mezi délkou skladby, a dobou trvání harmonizace jednoho taktu.

monizace skladeb s tímto nastavením lze považovat za dobře provedené. Nicméně i v nich lze najít disharmonie.

Naopak při generování s vyššími hodnotami temperature (1.3 a 2.0) se akordy a jejich tóny střídají až příliš rychle. Harmonizace je s těmito hodnotami spíše náhodná.

Potvrdila se tedy domněnka z předchozího experimentu, že model vykazuje lepší výsledky při harmonizaci delších skladeb.

Kapitola 9

Diskuse

Tato kapitola se zaměřuje na diskusi výsledků experimentů, které byly provedeny na modelu Polyphony RNN. Ten byl natrénován na vlastní datové sadě. Model jako takový je popsán v kapitole 6, provedené experimenty pak v 8. Dále jsou navrženy možnosti dalšího pokračování, případně rozšíření této práce.

Některé harmonie, které generoval popisovaný model, jsou spíše neuspokojivé. Jak již bylo popisováno v experimentech, pro model bylo obtížné harmonizovat i známé melodie, které by měl znát z trénovací sady. Ve vygenerovaných skladbách byly patrné dva směry. Prvním je použití mnoha akordů. Ty ale byly málokdy zavedeny do skladby v pravý čas, a tak celkový dojem z poslechu spíše kazily. Správná harmonie těchto akordů byla ojedinělá. Toto bylo pozorováno především u krátkých skladeb s nízkou hodnotou parametru temperature. Druhým jevem byl jeden akord, který byl nevhodně dlouhý, případně se opakoval hned po sobě.

Složení akordů z menšího počtu tónů byl také jeden z častých jevů. Případně akord začal celý a postupně přestaly znít některé jeho tóny. Přes všechny tyto problémy se v každém experimentu objevilo několik skladeb, které měly alespoň z části správnou nebo zajímavě provedenou harmonii. V každém experimentu (40 skladeb) bylo možné najít 4 až 6 skladeb, které měly zdařilou harmonizaci. To znamená úspěch v deseti až patnácti procentech případů. Obecně lze říci, že krátké skladby, které byly generovány s vyšší hodnotou parametru temperature, byly lépe zpracované ve smyslu správné harmonizace. Harmonizace dlouhých skladeb s původní hodnotou temperature byly převážně harmonizovány správně i přes částečné disonance.

Co se týče trvání generování harmonie byly výsledky velmi uspokojivé. Doba generování jednoho taktu se i u delších skladeb pohybovala okolo 0.15 vteřiny. V porovnání s dobou znění jednoho taktu je doba generování téměř zanedbatelná.

Tyto skutečnosti naznačují, že i když je model pro použití v harmonizaci malý, přece jenom se něco naučil a byl schopen poskytnout dobré výsledky. Z toho lze usoudit, že použití většího modelu by mohlo přinést výsledky lepší, a ve více případech. Jak ale bylo řečeno v kapitole výše (viz 7), trénování takového modelu je podmíněno použitím větší výpočetní síly, případně paralelizací tohoto procesu.

Trénování většího modelu je tedy jednou z možností pokračování této práce. Dalším rozšířením může být použití a trénování modelu na jiné datové sadě, neboť většina výzkumů používá pouze korpus chorálů Johanna Sebastiana Bacha. Bach však nebyl jediným skladatelem. Jako vhodný základ by tak mohla sloužit hudba jiných skladatelů.

Možným důsledkem špatných výsledků této práce může být nevhodná konverze písniček z XML souborů v dodané databázi na MIDI soubory. Toto by mohlo být eliminováno přímo použitím jiných dat již ve formátu MIDI.

Zajímavým pokračováním této práce by bylo vytvoření systému, který by melodii harmonizoval ve stylu nějakého žánru. Případně již harmonizovanou skladbu v nějakém stylu převedl na jiný styl. Podobně, jako se provádí u obrázků [1]. Tato práce by však vyžadovala použití datové sady, která by všechny žánry dostatečně pokrývala a všechny skladby by byly anotované. Dále by mohl být problém s licencemi skladeb případně celé takové datové sady.

Pro automatizaci harmonizace by bylo možné využít také architektury neuronových sítí jiné než LSTM. Z nich jsou pro tento problém vhodné například generativní kontradiktorní sítě, více známe pod svým anglickým překladem Generative adversarial networks, nebo pod zkratkou GAN. Případně auto-regresivní modely, které při generování sekvencí berou v potaz i předchozí vygenerovanou sekvenci.

[1] viz neural style transfer v citaci [10]

Kapitola 10

Závěr

Cílem této práce bylo seznámit se s harmonizací známé melodie a s možností automatizace této činnosti za pomoci neuronových sítí. Následně navrhnout systém, který bude umožňovat automatickou harmonizaci souborů ve formátu MIDI. Navržený systém dále implementovat a otestovat na vhodné datové sadě. Posledním bodem zadání je rozbor dosažených výsledků a nastínění možných rozšíření této práce.

Základům hudby a strojového učení byla věnována druhá, respektive třetí kapitola. Čtvrtá kapitola představila formát MIDI a podobu datové sady použité v této práci. Ta byla poskytnuta ve formě databáze XML souborů a bylo ji nutno převést na MIDI soubory písniček, jak požadovalo zadání. Postup konverze souborů byl také součástí této kapitoly. Další kapitola popisovala různé přístupy k automatické harmonizaci. Některé poznatky z těchto prací byly vybrány a použity v návrhu systému v kapitole následující. Jako harmonizační systém byla vybrána již existující neuronová síť, jejíž použití je popsáno v kapitole sedmé. Po jejím natrénování na popsané datové sadě byly provedeny experimenty, jimž se věnuje další kapitola. Zkoumanými vlastnostmi byla doba potřebná k harmonizaci skladby a správnost harmonizace, která byla manuálně zjištěna subjektivním poslechem. Následuje diskuse dosažených výsledků během testování, návrhy dalšího pokračování této práce a závěr.

Ne všechny výsledky natrénovaného systému byly uspokojivé a většinou s původní melodií nezněly a kazily její dojem. Nicméně mezi vygenerovanými skladbami se našly světlé výjimky, které se daly považovat za správnou a zajímavou harmonizaci. Obecně lze říci, že model při harmonizaci dlouhých skladeb vykazoval lepší výsledky.

Použitý model byl však, oproti doporučeným hodnotám hyperparametrů, malý. Lepší výsledky by tak mohlo přinést použití většího modelu, k jehož natrénování je však potřeba větší výpočetní síly, než bylo použito v této práci.

Literatura

[1] Agarap, A. F. Deep Learning using Rectified Linear Units (ReLU). *CoRR*. 2018, abs/1803.08375. Dostupné z: http://arxiv.org/abs/1803.08375.

[2] Back, D. *Standard MIDI-File Format Spec. 1.1, updated* [online]. 1999. Dostupné z: http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html.

[3] Britz, D. *RNNs in Tensorflow, a Practical Guide and Undocumented Features* [online]. Srpen 2016. Dostupné z: http://www.wildml.com/2016/08/rnns-in-tensorflow-a-practical-guide-and-undocumented-features/.

[4] Brunner, G., Wang, Y., Wattenhofer, R. a Wiesendanger, J. JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs. In: *29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017*. IEEE Computer Society, 2017, s. 519–526. DOI: 10.1109/ICTAI.2017.00085. Dostupné z: https://doi.org/10.1109/ICTAI.2017.00085.

[5] Chalupník, V. *Biologické algoritmy (4) - Neuronové sítě* [online]. 2012. Dostupné z: https://www.root.cz/clanky/biologicke-algoritmy-4-neuronove-site/.

[6] Cho, K., Merriënboer, B. van, Gulcehre, C., Bahdanau, D., Bougares, F. et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, říjen 2014, s. 1724–1734. DOI: 10.3115/v1/D14-1179. Dostupné z: https://aclanthology.org/D14-1179.

[7] Cmíral, A. a. P. J. *Základní pojmy hudební*. 9. vyd. Supraphon, 1974. Dostupné z: https://dnnt.mzk.cz/uuid/uuid:cb736220-216c-11e6-8d01-005056827e52.

[8] Dumoulin, V., Shlens, J. a Kudlur, M. *A Learned Representation For Artistic Style*. 2017.

[9] Fujishima, T. Realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music. In: Stanford University. *Proceedings of the 1999 International Computer Music Conference, ICMC 1999, Beijing, China, October 22-27, 1999*. Michigan Publishing, 1999, s. 464–467. ISSN 2223-3881.

[10] Gatys, L. A., Ecker, A. S. a Bethge, M. *A Neural Algorithm of Artistic Style*. 2015.

[11] Gers, F. A., Schmidhuber, J. a Cummins, F. Learning to Forget: Continual Prediction with LSTM. *Neural Computation*. Říjen 2000, sv. 12, č. 10, s. 2451–2471. DOI: 10.1162/089976600300015015. ISSN 0899-7667. Dostupné z: https://doi.org/10.1162/089976600300015015.

[12] Glatt, J. *About MIDI files* [online]. 2004. Dostupné z: http://midi.teragonaudio.com.

[13] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. 1. vyd. MIT Press, 2016. ISBN 9780262035613. http://www.deeplearningbook.org.

[14] Google AI. *Magenta* [online]. 2021. Dostupné z: https://magenta.tensorflow.org/.

[15] Google AI. *Making music with Magenta: Colaboratory* [online]. 2021. Dostupné z: https://colab.research.google.com/notebooks/magenta/hello_magenta/hello_magenta.ipynb#scrollTo=oLSgiA6Uktpm.

[16] Google AI. *Polyphony RNN: magenta/magenta/models/polyphony_rnn at master* [online]. 2021. Dostupné z: https://github.com/magenta/magenta/tree/master/magenta/models/polyphony_rnn#polyphony-rnn.

[17] Google Inc.. *Protocol Buffers* [online]. 2021. Dostupné z: https://developers.google.com/protocol-buffers.

[18] Ha, D. a Eck, D. *A Neural Representation of Sketch Drawings*. 2017.

[19] Hochreiter, S. a Schmidhuber, J. Long Short-Term Memory. *Neural Computation*. 1997, sv. 9, č. 8, s. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. Dostupné z: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[20] Huang, C.-Z. A., Cooijmans, T., Roberts, A., Courville, A. C. a Eck, D. Counterpoint by Convolution. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference*. Suzhou, China: ISMIR, říjen 2017, s. 211–218. DOI: 10.5281/zenodo.1416370. ISBN 978-981-11-5179-8. Dostupné z: https://doi.org/10.5281/zenodo.1416370.

[21] Karpathy, A. *The Unreasonable Effectiveness of Recurrent Neural Networks* [online]. Květen 2015. Dostupné z: http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[22] Kofroň, J. *Učebnice harmonie*. 1. vyd. Bärenreiter Praha, 2015. ISBN 978-80-86385-16-7.

[23] Krizhevsky, A., Sutskever, I. a Hinton, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. 2012, s. 1097–1105.

[24] Kvasnička, V., Beňušková Ľubica, Pospíchal, J., Farkaš, I., Tiňo, P. et al. *Úvod do teorie neuronových sítí*. 1. vyd. Iris, 1997. ISBN 80-88778-30-1.

[25] Liang, F. T., Gotham, M., Johnson, M. a Shotton, J. Automatic Stylistic Composition of Bach Chorales with Deep LSTM. In: Cunningham, S. J., Duan, Z., Hu, X. a Turnbull, D., ed. *ISMIR*. 2017, s. 449–456. ISBN 978-981-11-5179-8. Dostupné z: http://dblp.uni-trier.de/db/conf/ismir/ismir2017.html#LiangG0S17.

[26] Mehrotra, K., Mohan, C. a Preface, S. *Elements of Artificial Neural Nets*. MIT Press, leden 1997. ISBN 0-262-13328-8.

[27] Nicholson, C. *A Beginner's Guide to Multilayer Perceptrons (MLP)* [online]. 2020. Dostupné z: https://wiki.pathmind.com/multilayer-perceptron.

[28] Nicholson, C. *A Beginner's Guide to Neural Networks and Deep Learning* [online]. 2020. Dostupné z: https://wiki.pathmind.com/neural-network.

[29] Olah, C. *Understanding LSTM Networks* [online]. 2015. Dostupné z: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[30] Raczynski, S., Fukayama, S. a Vincent, E. Melody Harmonization With Interpolated Probabilistic Models. *Journal of New Music Research*. Říjen 2013, sv. 42, s. 223–235. DOI: 10.1080/09298215.2013.822000.

[31] Raschka, S. *Gradient Descent and Stochastic Gradient Descent* [online]. 2020. Dostupné z: https://rasbt.github.io/mlxtend/user_guide/general_concepts/gradient-optimization/.

[32] Symplifying Theory. *How to use Suspended Chords* [online]. 2021. Dostupné z: https://www.simplifyingtheory.com/how-to-use-suspended-chords/.

[33] Szkandera, J. *Alterované akordy: Pojednání o hudební harmonii* [online]. 2021. Dostupné z: https://www.harm.cz/maturita/klasika/Alterovan%C3%A9_akordy.

[34] (TMA), T. M. A. *Getting Started with MIDI Tutorials* [online]. 2021. Dostupné z: https://www.midi.org/categories-new/midi-tutorials.

[35] Yin Cheng, Y., Wen Yi, H., Fukayama, S., Kitahara, T., Genchel, B. et al. Automatic Melody Harmonization with Triad Chords: A Comparative Study. *ArXiv.org*. 1. vyd. Ithaca: Cornell University Library, arXiv.org. 2020, č. 1. ISSN 2331-8422. Dostupné z: http://search.proquest.com/docview/2334888248/.

[36] Zenkl, L. *ABC hudební nauky*. 3. vyd. Supraphon, 2003. ISBN 80-86385-21-3.