VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POČÍTAČEM KOMPONOVANÝ HUDEBNÍ DOPROVOD

ALGORITHMIC ACCOMPANIMENT COMPOSITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAKUB VINŠ**
AUTHOR

VEDOUCÍ PRÁCE **MARTIN KOLÁŘ, M.Sc.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Vinš Jakub**
Program: Informační technologie
Název: **Počítačem komponovaný hudební doprovod**
Algorithmic Accompaniment Composition
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s hudební teorií a s počítačem generovanou hudební tvorbou
2. Vytvořte kolekci metod automatického generování hudby, rozdělené podle metodologie
3. Seznamte se s dopřednými neuronovými sítěmi
4. Natrénujte neuronovou síť generující *piano roll matrix* notaci pro doprovod, podmíněnou *piano roll matrix* hlavního hlasu
5. *Piano roll matrix* notaci konvertujte zpět do audio signálů
6. Natrénujte pro sérii nástrojů, a vytvořte audio ukázky výstupu

Literatura:

- McLean, A. and Dean, R.T. eds., 2018. *The Oxford handbook of algorithmic music*. Oxford University Press. (dostupná u vedoucího)
- https://en.wikipedia.org/wiki/Algorithmic_composition
- https://sites.google.com/site/sd16spring/home/project-toolbox/algorithmic-music-composition

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Kolář Martin, M.Sc.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 14. května 2020
Datum schválení: 6. listopadu 2019

Abstrakt

Práca sa zaoberá počítačom generovanou hudbou, so zameraním na dotvorenie doprovodu do vstupnej, existujúcej piesne vo formáte MIDI pomocou neurónových sietí. V úvodnej kapitole sú predstavené existujúce prístupy pre generovanie hudby. Ďalej sú opísané problémy a ich riešenia spojené s konverziou MIDI súborov do matíc adekvátnych pre vstup do neurónovej siete a ich spätná transformácia. Následne sú predstavené, vytvorené, optimalizované a vyhodnotené modely pre generovanie saxofónového a klavírneho doprovodu pomocou doprednej a rekurentnej neurónovej siete. Pre praktické vyskúšanie modelu je na záver vygenerovaný doprovod do vlastnej piesne.

Abstract

This thesis deals with problems of computer music, especially with generating accompaniment to an existing song in MIDI format by means of artificial neural networks. Existing methods of algorithmic music composition are presented in the beginning. Followed by problems and their solutions connected with the conversion of MIDI files to matrices, which are suitable as an input for neural network and their inverse transformation. Subsequently are proposed, created, optimized and evaluated models which generate saxophone and piano accompaniment by means of feedforward and recurrent neural network. At the end model generates accompaniment to my own song as a form of a test.

Kľúčové slová

počítačová hudba, harmonizácia, MIDI, piano roll, dopredná neurónová sieť, rekurentná neurónová sieť, LSTM, saxofón, klavír

Keywords

computer music, harmonization, MIDI, piano roll, feedforward neural network, recurrent neural network, LSTM, saxophone, piano

Citácia

VINŠ, Jakub. *Počítačem komponovaný hudební doprovod*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Martin Kolář, M.Sc.

Počítačem komponovaný hudební doprovod

Prehlásenie

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Martina Koláře, M.Sc. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jakub Vinš
28. mája 2020

Poďakovanie

Rád by som poďakoval môjmu vedúcemu práce pánovi Martinovi Kolářovi, M.Sc. za čas a odborné rady, ktoré mi boli poskytnuté pri tvorbe tejto práce.

Obsah

Kapitola 1

Úvod

Hudba je neodmysliteľnou súčasťou života pre mnohých či už ako spôsob relaxácie, komunikácie, alebo realizácie ľudskej kreativity. Je to akási univerzálna reč ľudstva, jazyk pozostávajúci z tónovej abecedy kde nástroje sú prízvuky nesúce schopnosť formovať autorove vnemy. Je však schopný počítač, bezvedomá skrinka, produkovať niečo takto delikátne, v našich očiach čisto ľudské? S príchodom informačných technológií sa objavili aj prvé pokusy zodpovedať túto otázku.

Snaha využiť matematické modely, natvrdo určené syntaktické pravidlá v podobe gramatík alebo rôzne evolučné metódy za účelom napodobenia ľudských hudobných kompozícií započali algoritmické komponovanie. Zaujímavosť výsledkov stúpla v posledných rokoch spolu s rastúcim záujmom o hlboké neurónové siete. Modely generujúce viac nástrojové skladby, pri ktorých by ste pravdepodobne nezistili, že sa jedná o výtvor bezvedomej skrinky ako napríklad Aiva[1].

V mojej práci sa nebudem zaoberať skladaním nových skladieb ale naopak potenciálnym vylepšovaním už existujúcich, prostredníctvom generovania hudobného doprovodu – tzv. harmonizácie pomocou neurónových sietí. Táto téma bola rozoberaná v niekoľkých prácach ako napríklad Harmonet[2] alebo Bass Harmonization[7]. Moja práca sa od nich líši, okrem iného, formátom skladieb. Do piesní vo formáte MIDI obsahujúcich gitaru a basu generujem prostredníctvom neurónovej siete saxofónovú alebo klavírnu stopu. Modely sú trénované na vybratých skladbách z archívu obsahujúceho viac ako 130 000 MIDI súborov.

V 2. kapitole sa venujem kolekcii už existujúcich metód na generovanie hudby a neurónovým sieťam, ktoré budem následne v práci využívať. V 3. kapitole predstavím bližšie MIDI formát, problémy a riešenia spojené s transformáciou MIDI piesní do vhodného tvaru pre vstup do neurónovej siete a jeho spätného prevedenia do skladby vo formáte MIDI. V kapitole štvrtej vytvorím model, dataset a zahájim trénovanie doprednej neurónovej siete generujúcej saxofónovú stopu do existujúcej skladby. Predstavím jej výsledky a aplikujem potenciálne zlepšenia. V 5. kapitole využijem na generovanie saxofónovej stopy rekurentnú neurónovú sieť obsahujúcu LSTM vrstvu. V 6. kapitole vytvorím opäť model doprednej neurónovej siete, avšak tentokrát generujúci klavírnu stopu a vyhodnotím jeho výsledky. Záverečná 7. kapitola slúži na vyhodnotenie všetkých použitých prístupov a navrhnutie možných zlepšení.

[1] https://www.aiva.ai/

Kapitola 2

Prehľad metód automatického generovania hudby

Táto kapitola je založená na snahe automatizovať komponovanie hudby. Cieľom tejto snahy je model, ktorý samostatne komponuje určitý hudobný žáner v podobe skladby, ktorá preberá typické črty a postupy už existujúcich piesní. Validita takéhoto modelu je overiteľná neschopnosťou testovacieho subjektu rozlíšiť, či je pieseň komponovaná modelom alebo umelcom. Tieto modely sú usporiadateľné do viacerých kategórií [3] na základe ich štruktúry a spôsobu spracovania dát .

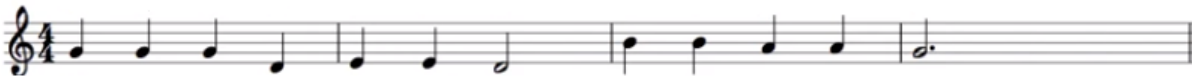
2.1 Matematické modely

Sú založené na matematických rovniciach a náhodných udalostiach. Najbežnejší spôsob je využitie stochastických procesov. Skladateľ kontroluje proces kompozície len z malej časti a to napríklad určovaním pravdepodobnosti jednotlivých nôt. Málokedy je vyžadované, aby mala každá nota rovnakú pravdepodobnosť. Na zaistenie frekventovanejšieho používania vybraných tónov, sa používa napríklad Gaussovo rozloženie, kde sa maximum nachádza v strede a minimá po okrajoch. Najznámejším príkladom stochastických algoritmov sú Markovove reťazce [10].

2.1.1 Markovove reťazce

Markovove reťazce sú diskrétne systémy, ktorých pravdepodobnosť prechodu do nasledujúceho stavu záleží na určitom počte predchádzajúcich stavov. Inak povedané si tento proces pamätá predchádzajúce udalosti, ktoré ovplyvňujú jeho budúcnosť. Nultý rád Markovových reťazcov neberie do úvahy žiadneho predchodcu, dá sa považovať za skutočné pravdepodobnostné rozloženie. Výsledok prvého rádu Markovových reťazcov je ovplyvnený jedným predchádzajúcim výstupom, druhého rádu dvomi výstupmi atď [3]. Druhý a vyšší rád je považovaný za hudobne zaujímavý.

Hudobná skladba sa dá reprezentovať Markovovým reťazcom ako postupnosť rôznych stavov. Stav môže vyjadrovať napríklad silu úderu, frekvenciu noty alebo jej dĺžku. Pre každý jedinečný stav sa vytvorí vektor prechodových funkcií $P(A, B)$ reprezentujúci pravdepodobnosť prechodu zo stavu A do stavu B. Spojením všetkých týchto vektorov vznikne prechodová matica obsahujúca pravdepodobnosť prechodu z aktuálneho stavu do nasledujúceho. Dá sa považovať za stochastický model slúžiaci na generovanie nových skladieb.



Obr. 2.1: Začiatok piesne *Old MacDonald had a farm.*

Stav systému je v maticiach reprezentovaný frekvenciou tónu (D,E,G,A,B)

P(A,B)	D	E	G	A	B
D	0	1/2	0	0	1/2
E	1/2	1/2	0	0	0
G	1/3	0	2/3	0	0
A	0	0	1/2	1/2	0
B	0	0	0	1/2	1/2

Tabuľka 2.1: Prechodová matica prvého rádu.

P(A,B)	D	E	G	A	B
D E	0	1	0	0	0
D B	0	0	0	0	1
E D	0	0	0	0	1
E E	1	0	0	0	0
G D	0	1	0	0	0
G G	1/2	0	1/2	0	0
A A	0	0	1	0	0
B A	0	0	0	1	0
B B	0	0	0	1	0

Tabuľka 2.2: Prechodová matica druhého rádu.

2.2 Translačné modely

Prístup, ktorý je založený na preklade informácií z existujúceho nehudobného média na nový zvuk. Preklad môže prebiehať na základe pravidiel alebo stochasticky. Napríklad preklad obrázku na zvuk, kde horizontálna čiara reprezentuje konštantný tón a šikmá čiara stupnicu [10]. Ďalšie využitie je pri preklade textu na hudbu. Význam slov je prekladaný do durových (šťastných) alebo mólových (smutných) akordov.

2.3 Gramatiky

Gramatiku môžme chápať ako súbor syntaktických pravidiel a hudbu ako jazyk s určitou gramatikou. Pravidlá určujú rytmus, harmónie a využitie jednotlivých nôt. Sú základom pre vygenerovanie nových skladieb.[10]

2.4 Evolučné metódy

Sú založené na princípe genetických algoritmov. Proces začína výberom skladby či už vygenerovanej automaticky, alebo zloženej ľudským skladateľom. Následne sa vyvíjajú prostredníctvom mutácie a prirodzeného výberu rôzne hudobnejšie alternatívy. Kľúčové je zvolenie

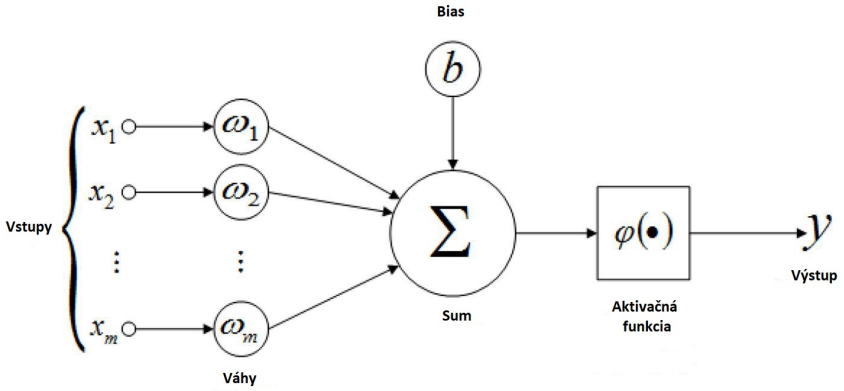
funkcie pre ohodnotenie kvality, ktorej účel je rozhodnutie, či kandidát prežije do nasledujúcej generácie. [11]

2.5 Celulárne automaty

Celulárne automaty sú spojené systémy zložené z automatov – buniek, ktoré sú na začiatku charakterizované počiatočným stavom, ktorý sa mení na základe evolučného pravidla [3]. Hodnoty stavov ďalšej generácie sa určia na základe funkcie, ktorej argumenty sú tvorené hodnotami stavov susedov bunky. V hudobnej sfére automat modeluje napríklad výšku, farbu alebo amplitúdu tónu.

2.6 Umelá neurónová sieť

Keďže je moja práca založená na neurónových sieťach, tak je vhodné, aby som priblížil ich spôsob fungovania, rôzne typy a využitie.



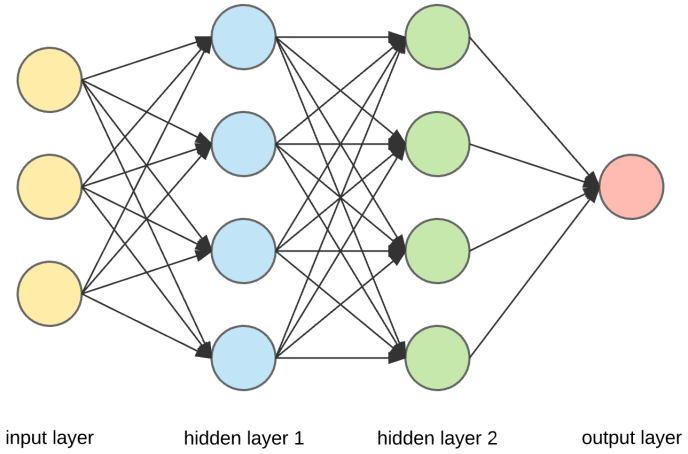
Obr. 2.2: Model umelého neurónu (prevzaté[1]).

Základnou stavebnou jednotkou, ako už názov napovedá je umelý neurón (ďalej už len neurón) inšpirovaný biologickým svetom. Každý neurón má jeden alebo viac vstupov X, ktoré sú individuálne vážené pomocou váh W, čím je váha vyššia, tým je vstup dôležitejší. Neurón obsahuje tiež nastaviteľnú hodnotu bias, ktorá implicitne ovplyvňuje aktiváciu neurónu. Suma vstupov a biasu prechádza ďalej aktivačnou funkciou, ktorá prináša nelineárnosť do celého procesu a rozhoduje o tom, či sa neurón aktivuje alebo nie. Výstup aktivačnej funkcie predstavuje výstup neurónu Y, ktorý sa prenáša na vstup ďalšieho neurónu. V mojej práci budem používať primárne aktivačnú funkciu logistického sigmoidu, ktorá mapuje vstup na hodnotu v rozmedzí 0 až 1. Zatiaľ čo vstupov môže mať neurón neobmedzené množstvo, výstup má len jeden.

Neuróny sa ďalej skladajú do vrstiev a tvoria neurónovú sieť. Sieť musí obsahovať minimálne dve vrstvy – vstupnú a výstupnú. Výpočtová sila takejto siete je ale mizivá, preto sa medzi nimi obvykle objavujú aj skryté neurónové vrstvy viz. 2.3.

[1] https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e

Neuróny môžu byť plne prepojené, kedy je každý neurón jednej vrstvy spojený s každým neurónom tej ďalšej alebo len určitá skupina je spojená s určitým neurónom ďalšej vrstvy. Existujú rôzne typy neurónových sietí s odlišnými topológiami a učiacimi sa algoritmami. Rozhodol som sa bližšie priblížiť tie, ktoré v mojej práci aktívne používam.



Obr. 2.3: Príklad neurónovej siete (prevzaté[1]).

2.6.1 Dopredná neurónová sieť

Dopredná neurónová sieť, tiež nazývaná ako viacvrstvový perceptron je považovaná za najzákladnejší druh neurónovej siete. Výstup neurónu je prenášaný na vstup všetkých neurónov nasledujúcej vrstvy. Nazýva sa doprednou, pretože neobsahuje žiadnu spätnú väzbu, ktorou by sa výstup neurónu dostal na jeho vlastný vstup, inak povedané neobsahuje žiadne rekurentné slučky.

2.6.2 Rekurentná neurónová sieť (RNN)

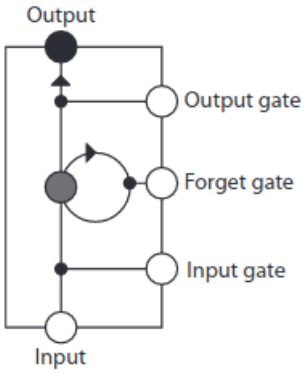
Rekurentná neurónová sieť obsahuje narozdiel od doprednej aj spätné prepojenia medzi neurónami, prípadne celými vrstvami. Tieto prepojenia sú podstatné pri predikciách súvisiacich s časovým kontextom, sieť je pomocou nich schopná do výstupu zahrnúť aj predchádzajúce výpočty. V hudbe sa to dá vnímať ako schopnosť reagovať na to čo už sieť zahrala, registrovať akýsi kontext skladby, ktorý je veľmi vítaná vlastnosť. Avšak doba počas ktorej si dokážu rekurentné neurónové siete zapamätať tento kontext je obmedzená problémom s názvom **Vanishing gradient problem** [1]. Jedná sa o situáciu, pri ktorej sa znižuje gradient chybovej funkcie, až dosiahne hranicu, kedy sa prestanú efektívne aktualizovať hodnoty váh, čo má za následok aj stratu kontextu.

2.6.3 Long Short-Term Memory (LSTM)

Jedná sa o typ rekurentnej neurónovej siete, ktorá je podľa viacerých zdrojov ([1], [8]) schopná riešiť problém miznúceho gradientu. Na obrázku 2.4 môžme vidieť model LSTM

[1] https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

bloku, ktorý je schopný uchovať svoj vnútorný stav na dlhú dobu. Stabilita takéhoto bloku je dosiahnutá pomocou troch brán nachádzajúcich sa okolo Bunky v jadre (šedej farby). Vstupná brána (Input gate) kontroluje tok dát smerujúcich do Bunky, čo jej efektívne umožnuje ignorovať vstup vo vhodnú chvíľu. Zabúdajúca brána (Forget gate) umožňuje vyprázdniť obsah Bunky v adekvátny moment. Výstupná brána (Output gate) umožňuje Bunke schovať svoj obsah pred inými jednotkami v sieti. Napríklad blok, ktorý dosahuje zlé výsledky v určitom kontexte sa môže naučiť odpojiť od nasledujúcich jednotiek v sieti pomocou Výstupnej brány. Do akej miery je stabilita tohto bloku pravdivá a do akej miery sa bude výstup môjho modelu podobať výroku Mozera, ktorý zhodnotil hudbu vygenerovanú jeho RNN modelom ako *„music only its mother could love"* [1] , sa budem snažiť dokázať v piatej kapitole tejto práce.



Obr. 2.4: Model LSTM bloku (prevzaté[1]).

Kapitola 3

Transformácia vstupných a výstupných dát

V tejto časti rozoberiem veľmi podstatnú časť mojej práce, prípravu dát pre neurónovú sieť. Priblížim formát MIDI, ktorý túto prácu umožnil a transformáciu piesní tohto formátu na matice.

3.1 MIDI formát

MIDI (Musical Instrument Digital Interface) formát je využívaný okrem komunikácie medzi elektronickými zariadeniami (napríklad počítač a keyboard) aj na ukladanie MIDI dát o piesňach na disk. Hlavný rozdiel v porovnaní s inými formátmi na ukladanie zvuku je, že MIDI formát nezaznamenáva audio signály ale informácie sú uložené pomocou správ. Na obrázku 3.1 je vidieť 2 správy a ich reprezentáciu v knižnici jazyka Python *pretty_midi*. Prvá správa mení program na danom kanály (v tomto prípade ide o kanál 1), program reprezentuje jeden zo 128 možných nástrojov, hodnota 25 predstavuje akustickú gitaru. MIDI súbor môže obsahovať až 16 kanálov, každý z nich predstavuje jeden nástroj. Druhá správa vyjadruje zaznenie MIDI noty po dobu 4.1 sekundy s hodnotou 60 (tón C4) a dôrazom o veľkosti 72.

program_change channel=1 program=25 time=0
note_on channel=1 note=60 velocity=72 time=4.1

Obr. 3.1: Príklad MIDI správ.

Uchovanie dát pomocou správ miesto audio signálov výrazne zníži veľkosť MIDI súborov a zjednoduší prácu s nimi, vďaka čomu sa stal MIDI formát ideálnym kandidátom pre túto prácu.

3.2 Pretty_midi

Prácu s midi súbormi v Pythone mi výrazne uľahčila knižnica pretty_midi. Slúži na vytváranie, manipuláciu a analýzu MIDI súborov [6]. MIDI dáta reprezentuje ako hierarchiu tried. Na vrchu sa nachádza *PrettyMIDI trieda*, ktorá obsahuje globálne informácie ako

napríklad zmeny tempa, slúži na načítanie MIDI súboru a jeho spätný zápis. Tiež obsahuje zoznam objektov *Instrument*, ktorý mi prišiel veľmi nápomocný pri filtrovaní súborov. Každý *Instrument* je špecifikovaný číslom programu (napríklad 25 predstavujúca akustickú gitaru v sekcii vyššie) a tiež boolean hodnotou, ktorá značí, či sa jedná o bicie. Objekt *Instrument* obsahuje tiež tri zoznamy, pre objekty *Note*, *PitchBend* a *ControlChange*. Trieda *Note* slúži na uchovanie MIDI nôt, ich tón, dôraz, čas začiatku a konca. Podobne *PitchBend* trieda obsahuje atribúty pre ohyb tónu a *ControlChange* trieda nesie čas a hodnotu zmeny kontroly ako napríklad hĺbku vibrata.

3.3 Filtrovanie MIDI súborov

Na populárnom internetovom fóre [4] som našiel balíček obsahujúci viac ako 130 000 MIDI súborov. Tieto súbory obsahujú rôznych hudobníkov, žánre, nástroje, a keďže v hlavnej časti bakalárskej práci generujem saxofónovú stopu do piesní obsahujúcich gitaru a basgitaru, tak bolo treba vyfiltrovať piesne, ktoré tieto nástroje obsahujú. K tomuto mi pomohla už spomínaná trieda Instrument, ktorá obsahuje informácie o nástrojoch nachádzajúcich sa v danom MIDI súbore.

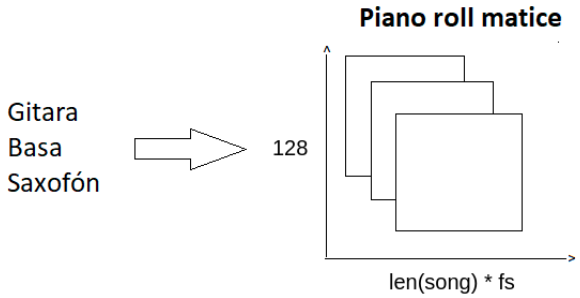
V jazyku Python som si vytvoril jednoduchý skript *instrument_recognizer*, ktorý preiteroval všetky MIDI súbory a o každom vypísal štatistiky do textového súboru ako vidieť na obrázku 3.2. *Guitar, Piano, Bass, Sax* označujú koľko krát sa nástroj v piesni nachádza. *Others* nesie informáciu o počte iných nehľadaných nástrojoch. *Drums* môže nadobúdať hodnotu 0 alebo 1 a značí, či pieseň obsahuje bicie alebo nie, *saxP* predstavuje percentuálne zastúpenie saxofónu v piesni a *Len* dĺžku piesne v sekundách. Tým som získal ľahko načítateľné a spracovateľné informácie o MIDI súboroch. Následne som vybral všetky, ktoré obsahujú aspoň jednu gitaru, basu, saxofón, je v nich menej ako 5 iných nástrojov a zastúpenie saxofónu je vyššie ako 20 %. Výsledkom tohto filtrovania je niečo okolo 4500 midi súborov.

Guitar	Piano	Bass	Sax	Others	Drums	saxP	Len	Name
0	0	0	1	4	0	17.18%	149.41	Midis/Archive/Classical_mfiles.co.uk_MIDIRip/at-peace.mid
4	5	2	0	2	0	0.0%	194.04	Midis/Archive/Classical_mfiles.co.uk_MIDIRip/myValentine.mid
0	0	1	0	2	0	0.0%	35.23	Midis/Archive/Classical_mfiles.co.uk_MIDIRip/Light-Scot.mid

Obr. 3.2: Výstup skriptu instrument_recognizer.

3.4 Piano roll matica

Prvým krokom v úprave MIDI súborov do podoby, ktorú bude neurónová sieť schopná spracovať je vytvorenie piano roll matíc. Piano roll maticu môžme vnímať ako 2D numpy pole. Počet hodnôt na x-ovej osi tohto pola je vzorkovacia frekvencia fs vynásobená dĺžkou piesne v sekundách. Počet hodnôt na y-ovej osi je 128, takéto pole je znázornené na obrázku 3.3. Vzorkovaciu frekvenciu môžme chápať ako počet kúskov na ktoré je rozdelená každá sekunda v tomto prípade je to 96. Každá vzorka môže nadobudnúť jednu zo 128 hodnôt, ktoré reprezentujú možné dosiahnuteľné tóny na y-ovej osi. Pre každý z požadovaných nástrojov je vytvorená jedna piano roll matica pomocou funkcie get_piano_roll() z už spomínanej knižnice pretty_midi.



Obr. 3.3: Piano roll matice.

3.5 Úprava matice pre neurónovú sieť

Piano roll je už blízko finálnej podobe vstupu neurónovej siete, avšak stále obsahuje pár nedostatkov. Výstup funkcie get_piano_roll reprezentuje každú zahranú notu v jednej vzorke číslom od 0 po 127. Toto číslo predstavuje hlasitosť noty, čo je v našom prípade zanedbateľná informácia, ktorá by zbytočne komplikovala učenie, to isté platí aj pre ohyb noty. Danú funkciu preto zjednoduším, aby produkovala 1 v prípade že nota znie a 0 pokiaľ nie. Ďalším nedostatkom je príliš široký rozsah tónov, zo 128 sa polovica vyskytuje veľmi zriedkavo preto som sa rozhodol 44 vrchných a 20 spodných tónov odstrániť. Po odstránení mi ostalo 64 hodnôt na y-ovej osi. Každý zo 4500 MIDI piesní som upravil na takéto matice a skomprimoval do numpy .npz súborov.

3.6 Konverzia piano rollu na MIDI formát

Posledný krok v transformácii MIDI formátu je jeho spätné nadobudnutie z piano roll matice. Bohužiaľ knižnica *pretty_midi* nám takúto možnosť priamo neponúka, avšak na githube[1] je v neoficiálnych príkladoch zdieľaná funkcia od niekoho kto mal rovnaký problém a ponúkol svoje riešenie, ktoré transformuje piano roll maticu na *pretty_midi* objekt. Funkcia je postavená nad *pretty_midi* knižnicou a priamo využíva jej triedy. V jednoduchosti popísané funkcia prechádza všetky zmeny tónov v matici a transformuje ich do *Note* objektov. Tieto noty su ďalej priradené *Instrument* objektu. Funkcia avšak berie do ohľadu aj hlasitosť noty, čo sa mi podarilo ale rýchlo eliminovať, všetkým notám dávam rovnakú hlasitosť o hodnote 100. Ďalší problém nastal v tom, že funkcia vracia už hotový *pretty_midi* objekt avšak iba s jedným nástrojom. Riešenie je jej úprava do podoby, kedy vracia *Instrument* objekt, ktorý je ďalej pridávaný manuálne do *pretty_midi* objektu reprezentujúceho finálnu skladbu. Posledný z problémov je stanovenie tempa piesne, na to má však knižnica *pretty_midi* funkciu *estimate_tempo()*, ktorá dokáže z piano rollov odhadnúť tempo a vykazuje primerane adekvátne výsledky.

[1] https://github.com/craffel/pretty-midi/blob/master/examples/reverse_pianoroll.py

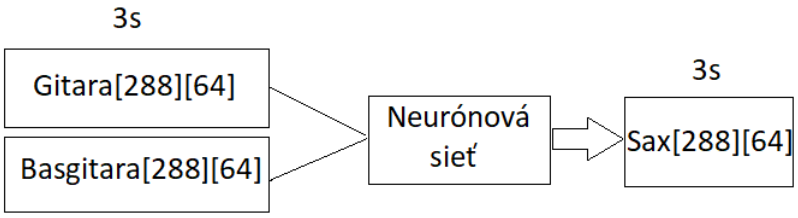
Kapitola 4

Generovanie saxofónu v podobne piano roll matice doprednou neurónovou sieťou

V tejto a nasledujúcej kapitole sa zameriam na hlavnú časť mojej bakalárskej práce, na samotné experimentovanie s neurónovými sieťami. Rozoberiem trénovanie pomocou doprednych a rekurentných neurónových sietí a porovnám ich výstupy ako subjektívne, tak aj na základe ich podobnosti s originálom. Doprednú neurónovú sieť som sa rozhodol použiť skrz jej jednoduchosť a možnosť generovania nasledujúcej sekvencie, nie len jednotlivých úsekov. *Pri každej referencii vygenerovanej piesne priložím názov danej skladby. Piesne sa nachádzajú v prílohovej časti tejto práce a sú rozdelené do súborov podla modelu, pomocou ktorého boli vytvorené.*

4.1 Prvotný prístup

Keďže s neurónovými sieťami som v tomto bode nemal žiadne skúsenosti, rozhodol som sa využiť veľmi jednoduchý prístup. Sieť ktorá na vstup dostane 3 sekundy gitarovej a basovej stopy v podobe dvoch matíc o veľkosti 288 na 64 a vyprodukuje 3 sekundy saxofónovej stopy viz. obrázok 4.1.



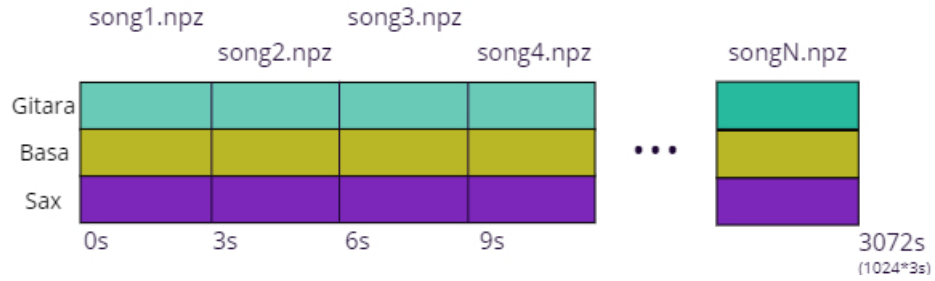
Obr. 4.1: 1. verzia siete.

V predchádzajúcej kapitole som popísal transformáciu MIDI súborov a ich komprimáciu na formát .npz. Tieto súbory treba upraviť do vhodného formátu pre náš prvý model neurónovej siete. 4000 piesní som rozdelil do 259 datasetov, zvyšných 500 som si ponechal na testovanie. Dataset obsahuje tri numpy listy, ktoré si môžme predstaviť ako 3 matice, jednu pre každý nástroj (gitara,basa,saxofón) obsahujúcu 1024 troj sekundových úsekov,

čiže 3 matice o veľkosti 1024x288x64. Každý numpy list obsahuje minimálne desať rôznych piesní, začína úsekom z prvej piesne, následne z druhej, tretej, až príde po desiatu, po ktorej nasleduje znova ďalší úsek z prvej. Nikdy nenasleduje dvakrát po sebe úsek z rovnakého npz súboru viz. obrázok 4.2. Pokiaľ je pieseň kratšia ako jej možné zastúpenie v datasete (cca 5 minút), tak sa automaticky berie ďalšia a pokračuje sa v cykle s ňou. Po 1024 iteráciách sú tri numpy listy uložené do datasetu pomocou numpy funkcie *savez_compressed()*, z ktorého sú naďalej prístupne následovne:

```
dataset = np.load(dataset.npz)
guitar = dataset['guitar']
bass = dataset['bass']
sax = dataset['sax']
```

Novovytvorené datasety som následne skontroloval v snahe odstrániť potenciálne nekonzistentné dáta a v potrebe overiť, že obsahujú iba hodnoty 1 a 0. Testy prebehli úspešne, čím som mohol pristúpiť k samotnej tvorbe modelu.



Obr. 4.2: Dataset pre 1. verziu siete.

4.1.1 Tvorba prvotného modelu

Pre prácu s modelmi neurónových sietí využívam knižnicu **Keras**[1] (verzia 2.3.0) napísanú v programovacom jazyku Python. Knižnica Keras je schopná využívať viacerých backendov [9], v práci využívam ako backend knižnicu *Tensorflow*[2] (verzia 2.2.0).

Keras poskytuje dva API prístupy pre vytvorenie modelu:

- Sekvenčné (Sequential) API - Je to najjednoduchší prístup ako vytvoriť model v Kerase, dovoľuje užívateľovi skladať jednu vrstvu za druhou [9]. Problém sekvenčného API je, že neumožňuje modelu vlastniť viac vstupov alebo výstupov, čo je v tejto práci zásadný problém, keďže na vstupe dostane model dve matice.
- Funkčné (Functional) API - Umožňuje vytváranie komplexných modelov obsahujúcich viacero vstupov a výstupov, čo je pre túto prácu ideálna voľba.

Model dostane na vstupe dve matice o veľkosti [batch_size, 288, 64], *batch_size* reprezentuje počet vzoriek, po ktorých bude neurónová sieť načítavať dataset. Matice je následne treba zlúčiť do jednej k čomu nám poslúži funkcia *concatenate()* pod knižnicou *keras.layers*, jej výstupom je matica o veľkosti [batch_size, 288, 128]. Jej veľkosť je avšak pre výstup nevyhovujúca, čo je ľahko napraviteľné pridaním jednej *dense* vrstvy o veľkosti 64, získavame maticu s tvarom [batch_size, 288, 64]. Na výstupe modelu sa nachádza *dense* vrstva

[1] https://keras.io
[2] https://www.tensorflow.org/

obsahujúca aktivačnú funkciu *logistický sigmoid*. Tá pretvára vstup na hodnotu v intervale 0 až 1, ktorý je ideálny pre náš prípad, čím je výstup bližšie k 1, tým si je sieť istejšia, že v danom čase zaznie vybraný tón. Ukážku kódu vytvorenia modelu môžme vidieť na obrázku 4.3. Vhodné by bolo ešte spomenúť metódu compile(), pomocou ktorej nastavujem pred trénovaním stratovú funkciu, optimalizátor a hodnoty, ktoré chcem mať prí výstupe učenia dostupné, v tomto prípade *accuracy*. Presnosť (accuracy) slúži na posúdenie výsledkov modelu.

```
def create_model():
  guitar_input = layers.Input(shape=(288,64), name='guitar_input')
  bass_input = layers.Input(shape=(288,64), name='bass_input')
  x = layers.concatenate([guitar_input, bass_input])
  x = layers.Dense(64)(x)
  output = layers.Dense(64, activation='sigmoid')(x)
  model = Model([guitar_input,bass_input],output)
  model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
  )
  return model
```

Obr. 4.3: Výpis vytvorenia modelu 1. verzie siete

Keďže ale výstup neurónovej siete bude zriedkavo nadobúdať hodnotu 1, je potrebné si vytvoriť hranicu od ktorej sa bude tón považovať za zahraný. K tomuto problému sa dá pristupovať dvoma spôsobmi.

Prvý prístup (ďalej nazývaný ako *sólový prístup*) je vybrať v každej vzorke iba jednu notu a to tú obsahujúci maximálnu hodnotu, ktorá musí byť vyššia ako zvolený prah. Pokiaľ neobsahuje vzorka hodnotu vyššiu ako prah, tak v jej čase nezahrá žiaden tón. Hodnotu prahu som zvolil na základe subjektívneho hľadiska na 0,7, plánujem s ňou však ešte experimentovať v nadchádzajúcich častiach tejto práce. Takýto spôsob vyzerá v programe nasledovne:

```
tresh = 0.7 # hodnota prahu (treshold)
pred = model.predict([guitar_pt, bass_pt])[0]
if solo:
    for i in range(288):
        max = tresh
        if pred[i].max() > tresh:
            max = np.amax(pred[i])
        pred[i] = np.where(pred[i] == max, 1, 0)
```

Funkcia model.predict() slúži na získanie výstupu z natrénovanej siete. V tomto prípade dostáva na vstupe *guitar_pt* a *bass_pt*, ktoré reprezentujú gitarovú a basovú stopu. Jej výstup je uložený do premennej *pred* a to v podobe matice o veľkosti [288,64], reprezentujúcu 3 sekundy vygenerovanej saxofónovej stopy. Boolean *solo* je nastavený na hodnotu *True* v prípade, že sa jedná o sólový prístup generovania. Cyklus for preiteruje všetkých 288 vzoriek a v prípade, že sa vo vzorke nachádza nota s hodnotou vyššou ako 0,7, tak nastaví nové

maximum. Následne nastaví notu obsahujúcu maximum na hodnotu 1 a zvyšné na 0.

Druhý prístup (ďalej nazývaný ako *prahový prístup*) nastaví všetky noty s hodnotou vyššou ako prah 0,7 na 1 a zvyšné na 0. Použitie prahu vyzerá v kóde následovne:

```
sax = np.where(sax > 0.7, 1, 0)
```

Sólový prístup je adekvátnejší z dôvodu, že saxofón je sólový nástroj avšak prahový prístup znie bohatšie a výsledky mi prišli zaujímavejšie. Na trénovanie používam online nástroj s názvom *Google Colab*, ktorý slúži ako pomôcka pre strojové učenie. Užívateľ môže spúšťať svoje Python programy vo forme jupyter notebookov na grafických kartách a pamätiach RAM poskytnutých spoločnosťou Google. Free verzia sa rýchlosťou vyrovná môjmu stolovému počítaču a dovoľuje mi mať spustených viacero inštancií, čiže je to pre mňa ideálny nástroj.

4.1.2 Trénovanie a vyhodnotenie sólového prístupu

Na to, aby som bol schopný vyhodnotiť výsledky modelu, budem potrebovať funkciu, ktorá je schopná porovnať originálnu saxofónovú stopu s vygenerovanou stopou. Rozhodol som sa k tomu pristupovať dvoma spôsobmi. Prvým z nich je už spomínaná hodnota *Accuracy*, ktorú ponúka keras pri vyhodnotení modelu pomocou *keras.Model.evaluate()*. Avšak tento prístup nespĺňa úplne moje požiadavky, pretože pracuje s výsledkami aktivačnej funkcie, čo sú hodnoty v rozmedzí 0 a 1, zatiaľ čo ja po vyhodnotení manuálne premieňam reálny výsledok na hodnoty 0 a 1. Preto som sa rozhodol aj pre 2. prístup, v ktorom spočítam ekvivalentné výskyty hodnoty 1 medzi originálnou a vygenerovanou saxofónovou stopou po úprave, vynásobím ich dvomi a vydelím ich súčtom výskytov hodnoty 1 v oboch stopách. Úsek kódu v ktorom je podobnosť počítaná vyzerá nasledovne:

```
total = np.count_nonzero(orig_sax == 1) + np.count_nonzero(gen_sax == 1)
accuracy = orig_sax[gen_sax == 1].sum()*2/total
```

Orig_sax predstavuje originálnu saxofónovú stopu a *gen_sax* novo vygenerovanú. Tento prístup mi umožní spočítať podobnosť po úprave pomocou prahu. Ďalšia výhoda tohto prístupu je zameranie sa len na hrané noty, resp. nehodnotenie generovaných núl, ktorých je prevážna väčšina.

Na prehranie piesní používam program s názvom *MuseScore*[1], v ktorom sú MIDI stopy transformované do notového zápisu a je umožnené ich prehranie. Vďaka tomu dokážem rýchlejšie zbadať, kde sa v piesni deje niečo zaujímave a naopak kde len program opakuje rovnaké noty alebo kopíruje basovú a gitarovú stopu. Po cca 16 minútach sa modelu podarilo prejsť všetky datasety – prešla tzv. *epocha*. Po náhodnom vygenerovaní dvadsiatich testovacích piesní som došiel na pár subjektívnych poznatkov. Model sa snaží generovať harmónie ku hraným gitarovým alebo basovým tónom, čím z nich vytvára akordy. Niektoré tóny znejú kratšie ako by mali, vytvárajúc nekonzistentný, sekaný dojem. Na moje prekvapenie väčšina tónov príjemne ladí, samozrejme sa nájdu aj tóny, ktoré režú ušné bubienky ale sú v podstatnej menšine. Z výsledku celkovo veľmi príjemne prekvapený.

Následne som vygeneroval 400 testovacích piesní a skontroloval ich podobnosť s originálom. Výsledky som spracoval do jednoduchej tabuľky 4.1.

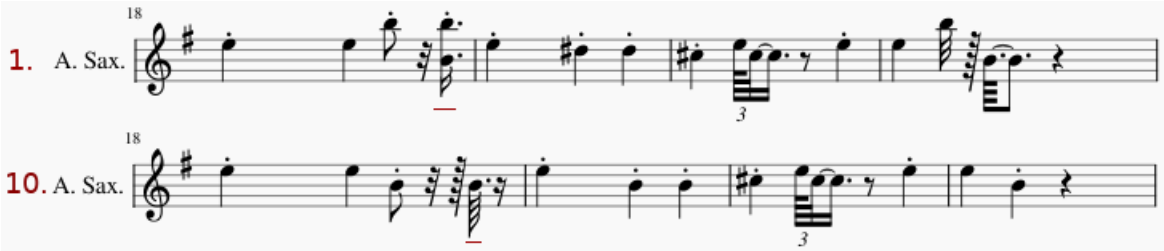
[1] https://musescore.com

N	Keras Funkcia	Moja Funkcia
50	5.1%	5.82%
100	6.1%	6.84%
200	6.1%	6.85%
400	7.03%	7.91%

Tabuľka 4.1: Výsledky po 1. epoche.

Ako môžme vidieť keras funkcia považuje originál za menej podobný vygenerovanej skladbe ako moja funkcia. Tento výsledok som predpokladal, z dôvodu upravenia podľa prahu, avšak vo väčšom merítku ako skutočne nastal. Tiež môžme pozorovať, ako veľmi záleží na samotných piesňach. U prvých 200 sa podobnosť pohybuje okolo siedmych percent, zatiaľ čo po nasledujúcich 200 piesňach sa táto hodnota zvýši takmer o 1/5.

Ako ďalšiu hranicu som sa rozhodol zvoliť 10 epoch. Podľa môjho subjektívneho hľadiska je to zmena k lepšiemu, aj keď je menej patrná ako som očakával. Hlavná časť vygenerovanej stopy je viac-menej rovnaká, avšak sieť už negeneruje niektoré nemilé noty, ktoré som si všimol pri piesňach vygenerovaných po 1. epoche.



Obr. 4.4: Porovnanie 1. a 10. epochy.

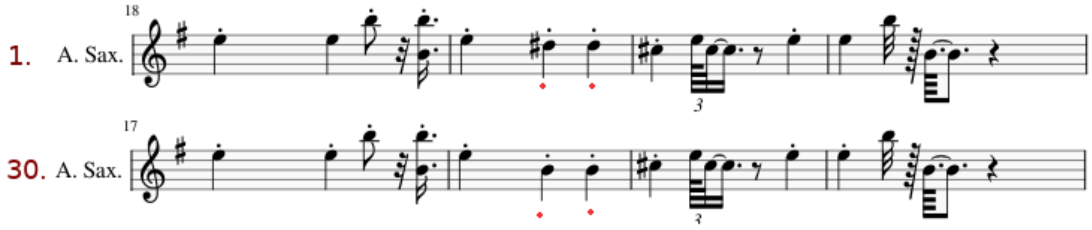
Na obrázku 4.4 vidíme dva úseky saxofónových stôp vygenerovaných do piesne *Bad Moon Rising* od kapely Creedence Clearwater Revival. Vrchná stopa zodpovedá 1. epoche (**0,7T_1.epoch_BadMoonRising.mid**) a spodná 10. epoche (**0,7T_10.epoch_Bad MoonRising.mid**). Na oboch som podčiarkol 4. notový výskyt, ktorý v prvom prípade obsahuje dokonca až 2 noty, to je spôsobené tým, že mali obe rovnakú hodnotu väčšiu ako 0,7, v druhom prípade už je model deterministickejší a rozhodol sa práve pre nižšiu notu. Celkovo je na takomto krátkom úseku až 6 odlišností, obe varianty sú však relatívne príjemne na počutie.

Výsledky porovnania s originálom pomocou dvoch funkcií som opäť transformoval do tabuľky 4.2. Podobnosť s originálom na základe Keras funkcie je v tomto prípade dokonca nižšia ako po jednej epoche. Moja funkcia vykazuje zlepšenie o cca 0.5 %.

N	Keras Funkcia	Moja Funkcia
50	4.99%	6.06%
100	5.97%	7.09%
200	6.04%	7.20%
400	6.97%	8.30%

Tabuľka 4.2: Výsledky po 10. epoche.

Ako poslednú hranicu som sa rozhodol zvoliť 30 a 100 epoch. Na moje počudovanie sa výsledky priblížili viacej prvej epoche ako desiatej. Na obrázku 4.5, môžme vidieť porovnanie prvej a tridsiatej epochy (**0,7T_30.epoch_BadMoonRising.mid**), opäť na na úseku zo skladby *Bad Moon Rising*. Výsledky sú takmer totožné až na dva prípady, ktoré som označil červenými bodkami, kedy sa model rozhodol generovať tón o tri poltóny vyšší než v prvej epoche. Úsek tejto piesne zo stej epochy bol v tomto prípade identický s tridsiatou.



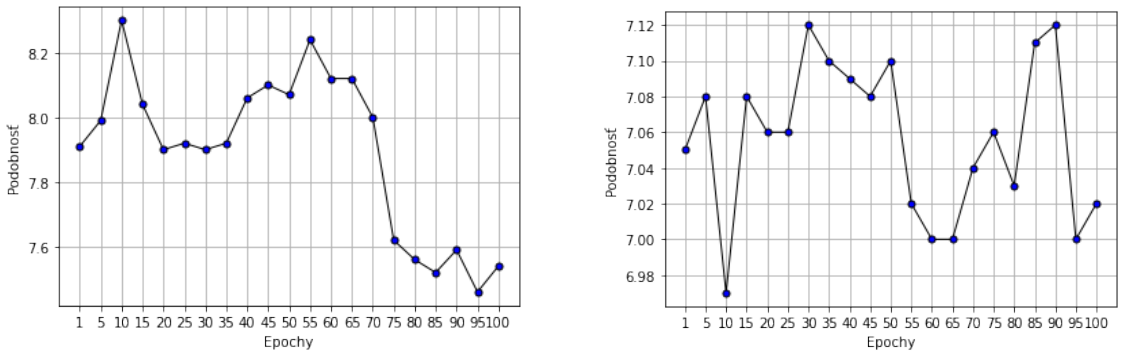
Obr. 4.5: Porovnanie 1. a 30. epochy.

Podobnosť po tridsiatich a sto epochách som opäť raz spracoval do tabuľky 4.3. Po 30 epochách môžme vidieť jemné zvýšenie zhody u Keras funkcie a naopak mierne zníženie u mojej funkcie. Na moje počudovanie po sto epochách sú obe hodnoty podobnosti nižšie ako po tridsiatich epochách.

	30 epoch		**100 epoch**	
N	***Keras Funkcia***	***Moja Funkcia***	***Keras Funkcia***	***Moja Funkcia***
50	5.29%	6.03%	5.19%	5.83%
100	6.18%	6.98%	5.94%	6.45%
200	6.22%	6.86%	6.11%	6.60%
400	7.12%	7.90%	7.05%	7.54%

Tabuľka 4.3: Výsledky po 30. a 100. epoche

V tomto momente sa rozplynula moja naivná predstava o tom, že po 100 epochách bude model najpresnejší a rozhodol som sa nájsť po koľkých epochách sa bude v tomto stave nachádzať. Z dôvodu časovej náročnosti som vzal každý piaty model a vypočítal ich podobnosť s originálom pomocou mojej funkcie a Keras funkcie. Výsledky som vygeneroval do grafov na obrázku 4.6 pomocou numpy funkcií *plot()* a *show()*.

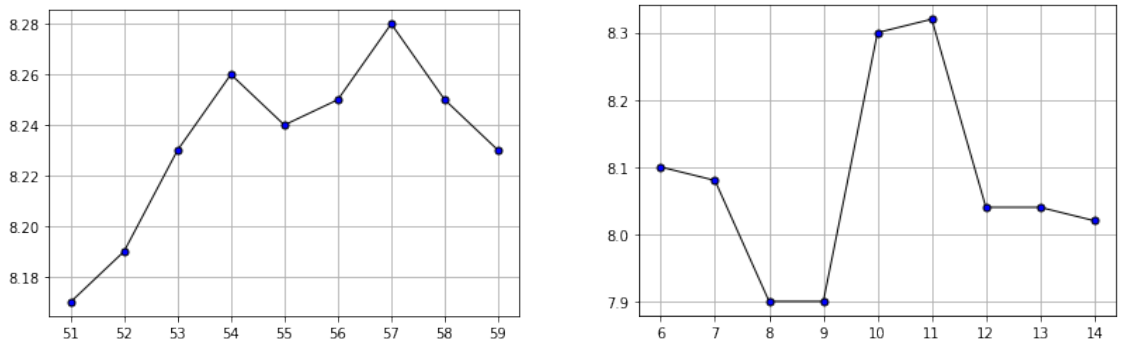


(a) Výsledky podobnosti mojej funkcie

(b) Výsledky Keras funkcie

Obr. 4.6: Porovnanie podobností

Dve potenciálne maximá pri podobnostiach vypočítaných mojou funkciou sa nachádzajú v okolí 10. a 55. epochy, preto som sa pre kompletnosť rozhodol zmerať ešte podobnosť epoch v ich blízkosti.



(a) 51 až 59 epocha sólového prístupu

(b) 6 až 14 epocha sólového prístupu

Obr. 4.7: Porovnanie podobností

Na obrázku 4.7, môžme vidieť dva grafy. Vľavo podobnosť na výsledkoch modelu od 51. až po 59. epochu. Najvyššiu hodnotu dosahuje model po 57. epoche a to 8,28%. Po pravej strane je zobrazená podobnosť modelu medzi epochami 6 až 14. Práve na tomto obrázku sa nachádza dosiahnuté maximum pri prahu o hodnote 0,7 a to po 11. epoche o hodnote 8,33%. Z výsledku som prekvapený, keďže som očakával maximálnu podobnosť až v neskorších epochách.

Porovnanie prahov

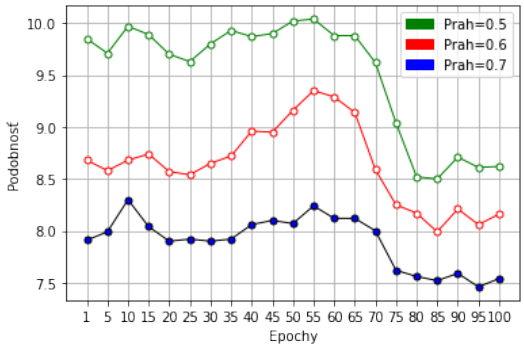
Poslednú modifikáciu, ktorú nad týmto modelom vykonám je zmena prahu od ktorého budem notu považovať za zahranú. Pri začiatku tejto kapitoly som zmienil jej hodnotu 0,7, ktorá bola zvolená subjektívne. Začal som generovať piesne s nižším prahom a merať ich podobnosť pomocou mojej funkcie. Vygenerované piesne s hodnotou prahu nižšou ako 0,7 boli podobnejšie originálu. Piesne som generoval s čím ďalej tým nižším prahom až kým podobnosť prestala rásť.

	0.7	**0.6**	**0.5**	**0.45**	**0.4**	**0.35**	**0.3**	**0.25**	**0.2**	**0.15**
Max %	8.3	9.35	10.12	10.15	10.19	10.26	10.28	10.27	10.28	10.19
Max epocha	10.	55.	55.	30.	5.	50.	30.	30.	30.	70.
AP %	7.90	8.64	9.51	9.91	10.06	10.14	10.16	10.18	10.16	9.83

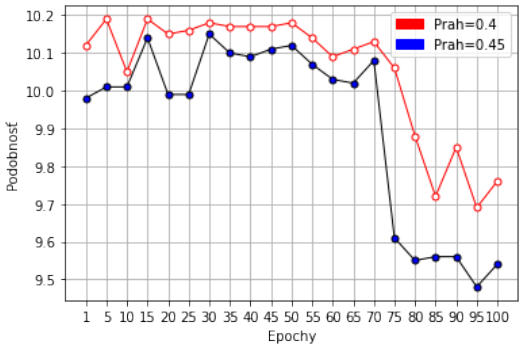
Tabuľka 4.4: Porovnanie výsledkov jednotlivých epoch

Výsledky som spracoval do tabuľky 4.4, na ktorej môžme vidieť maximálnu dosiahnutú podobnosť medzi jednotlivými epochami s konkrétnou hodnotou prahu. Tiež môžme vidieť epochu, po ktorej je toto maximum dosiahnuté a aritmetický priemer medzi všetkými vybratými epochami, ktoré boli zvolené rovnaké ako na obrázku 4.6, čiže každá piata. Maximálna podobnosť bola dosiahnutá niekde v blízkosti prahu o hodnote 0,3 a to cca. 10,28%. Najvyššiu hodnotu aritmetického priemeru podobnosti naprieč zvolenými epochami dosiahli piesne vygenerované s prahom v blízkosti hodnoty 0,25 a to niečo okolo 10,18%. Bol som

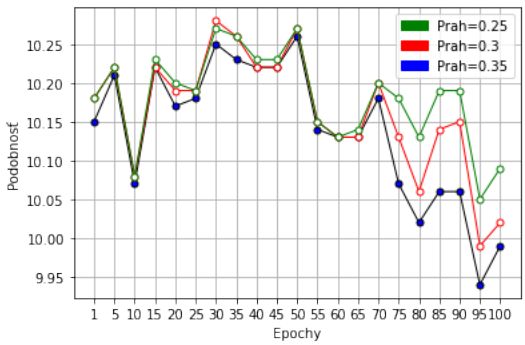
prekvapený že podobnosť rástla až po prah o hodnote 0,25 odkiaľ začala pomaly klesať. Výsledky som pre lepšiu vizualizáciu spracoval aj do grafov 4.8.



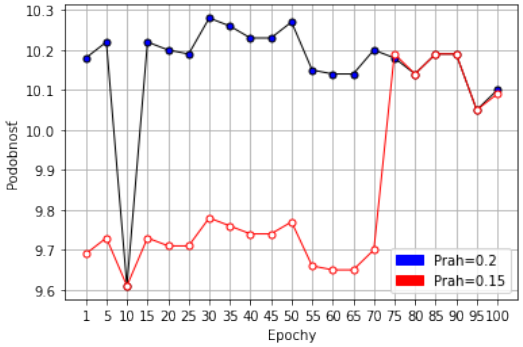
(a) 0.7,0.6 a 0.5



(b) 0.45 a 0.4



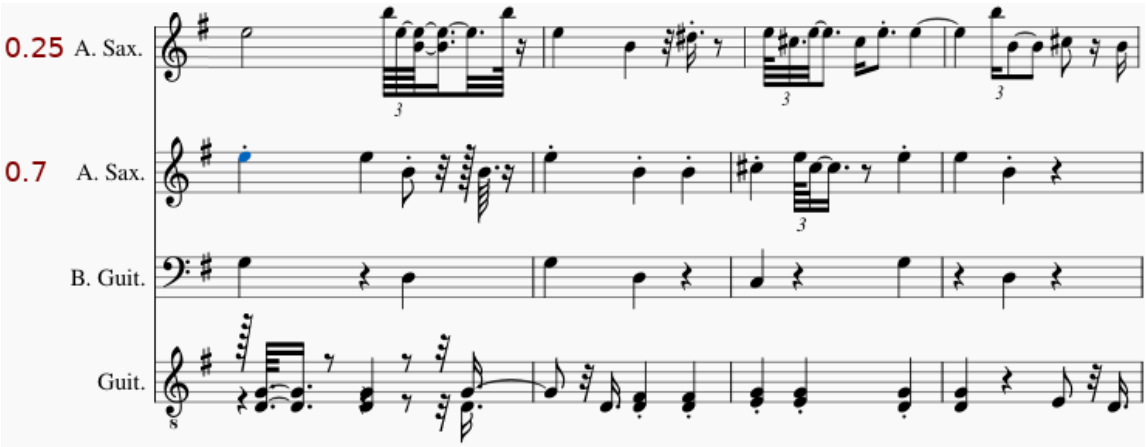
(c) 0.35 až 0.25



(d) 0.2 a 0.15

Obr. 4.8: Porovnanie prahov

Na záver sólového prístupu rozoberiem posledné porovnanie, opäť na skladbe *Bad Moon Rising*. Notové zápisy jednotlivých stôp môžme vidieť na obrázku 4.9.



Obr. 4.9: Porovnanie prahov 0,25 a 0,7 na piesni Bad Moon Rising.

Vyššia stopa (**0,25T_30.epoch_BadMoonRising.mid**) bola vygenerovaná z kombinácie maximálnej podobnosti a to s prahom o hodnote 0,25 a modelu po 30. epoche. Druhá stopa (**0,7T_10.epoch_BadMoonRising.mid**) tu už raz zobrazená bola a jedná sa o 10. epochu a maximum z nej, teda prah o hodnote 0,7. Na obrázku je okamžite povšimnuteľné, že vyššia stopa obsahuje viac tónov, čo notový zápis natiahlo a bol som nútený ho v grafickom editore zmenšiť z dôvodu jednoduchšieho porovnania so spodnou stopou avšak všetky noty ostali zachované, len sú menšie. Hneď na začiatku môžme vidieť zhluk nôt, tieto noty sú zahrané veľmi rýchlo po sebe a pôsobia useknuté. Práve táto skutočnosť sa mi javí ako hlavný problém nízkeho prahu. Síce nízke hodnoty dosahujú vyššiu podobnosť s originálom ale veľa nôt znie nekompletne. Tento problém by sa dal vyriešiť napísaním skriptu, ktorý by zarovnával dĺžku nôt avšak rozhodol som sa to zatiaľ neriešiť.

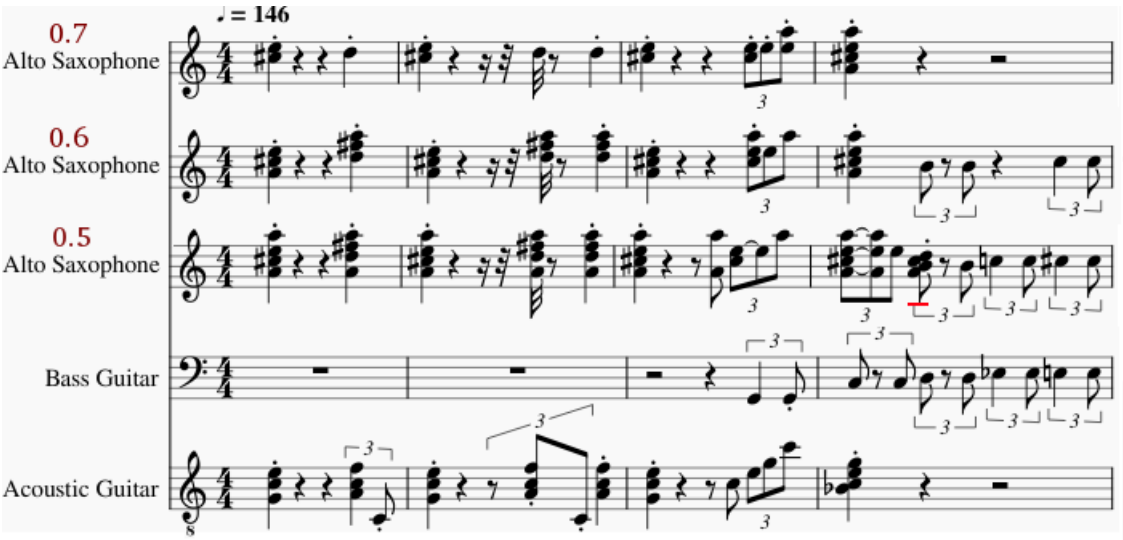
4.1.3 Vyhodnotenie prahového prístupu

Na začiatku tejto kapitoly som predstavil dva prístupy, ktorými budem reprezentovať výsledky modelu. Sólový som už zmienil a druhým z nich je prahový prístup, považujúc za zahrané noty všetky, ktoré majú hodnotu vyššiu ako zvolený prah. Podľa môjho subjektívneho názoru je tento prístup hudobne zaujímavejší a bohatší. Bohužiaľ na zmeranie presnosti mojou funkciou je nepoužiteľný, keďže generuje v jednej vzorke niekedy až 6 tónov, čo je však po druhej stránke pôvod jeho bohatšieho zvuku. Výsledky keras funkcie zo sólového prístupu sú identické s prahovým, keďže podobnosť je počítaná pred úpravou výstupu modelu na 0 a 1, čo nám umožňuje ich znova využiť. Keďže rozdiely medzi jednotlivými epochami sú pri tomto prístupe relatívne malé, zameriam sa primárne na experimentovanie s prahom od ktorého považujem tón za zahraný alebo nie. Nasledujúce výsledky sú vytvorené modelom z 30. epochy, z dôvodu najvyššej podobnosti s originálom, podľa Keras funkcie.

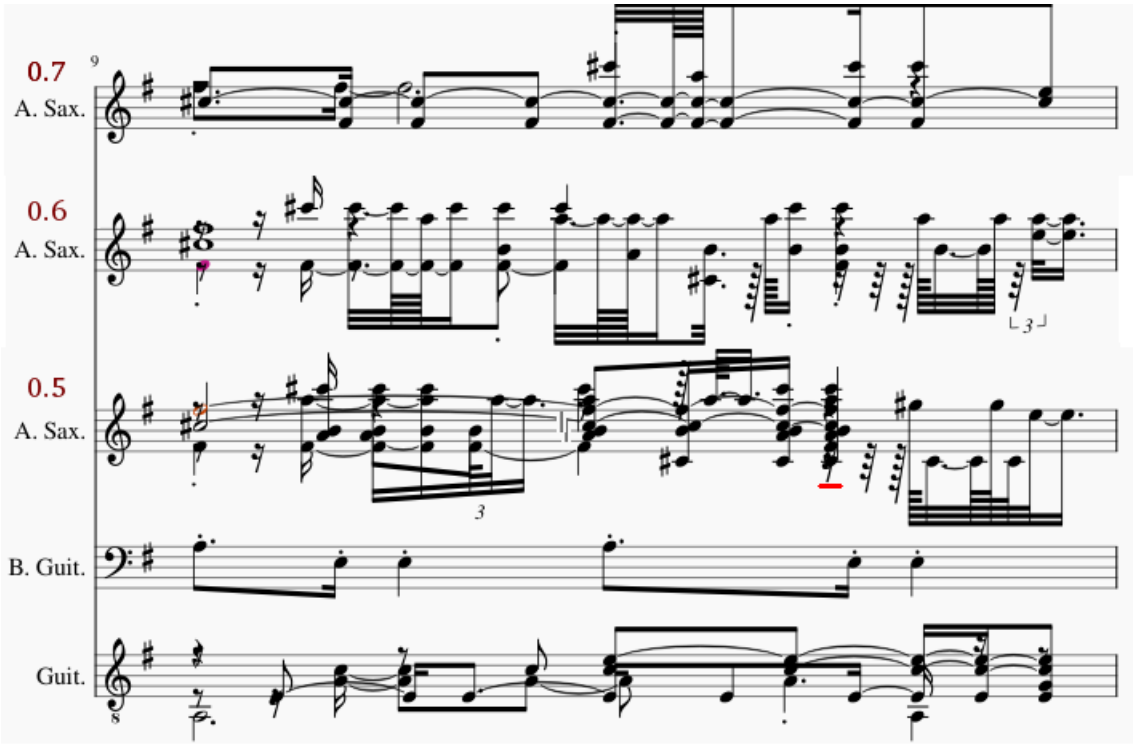
Jeden z príjemne znejúcich vygenerovaných doprovodov môžme vidieť na obrázku 4.10, jedná sa o pieseň *Everybody's trying to be my baby* od kapely *The Beatles*. Na obrázku môžme vidieť tri saxofónové stopy (**xT_BeatlesEverybodysTryingToBeMyBaby.mid**, kde **x** odpovedá hodnote prahu). Prvá z nich obsahuje noty, ktorých hodnota bola vyššia ako 0,7, druhá stopa má prah nastavený na 0,6 a tretia na 0,5. Ako sa dalo očakávať, čím je prahová hodnota nižšia tým viac nôt stopa obsahuje. Model generuje nepriamo celé akordy, ktoré buď kopírujú tóny z gitarovej stopy alebo pridávajú ladiace harmónie. Tiež môžme pozorovať, že 2. stopa generuje narozdiel od prvej aj doprovod ku base v podobe štyroch nôt na konci, ktoré sú identické basovým. Tretia stopa hrá dokonca viac tónov ako gitarová, všetky však príjemne ladia. Zaujímavý úkaz nastal pri tretej stope, na obrázku 4.10 je zvýraznený červeným podčiarknutím, kedy model vygeneroval na základe 2 basových tónov až štvortónový akord.

Nie vždy je zníženie prahu ideálnym vylepšením vygenerovanej saxofónovej stopy. Na obrázku 4.11 môžme vidieť jeden z týchto prípadov. Jedná sa o úsek z nemeckej skladby *Uber den Wolken* od umelca *Reinharda Meya*. Opäť sú zobrazené tri verzie saxofónovej stopy, líšiace sa v prahu (**xT_Uber-Den-Wolken.mid**, kde **x** odpovedá hodnote prahu), podľa ktorého boli noty generované. Prvá verzia prahu s hodnotou 0,7, vygenerovala príjemný doprovod, gitarovú stopu obohatila o pridané výšky. Druhá verzia je stále relatívne príjemne počúvateľná aj keď sa už stáva jemne presýtenou. Posledná verzia s prahom 0,5 presiahla svoj limit presýtenosti, hoci všetky tóny harmonicky ladia, stal sa z nej notový guláš. Pri

konci tohto úseku môžme badať podčiarknutý akord, hudobné monštrum, pozostávajúce až z deviatich tónov.



Obr. 4.10: Everybody's trying to be my baby, porovnanie prahov

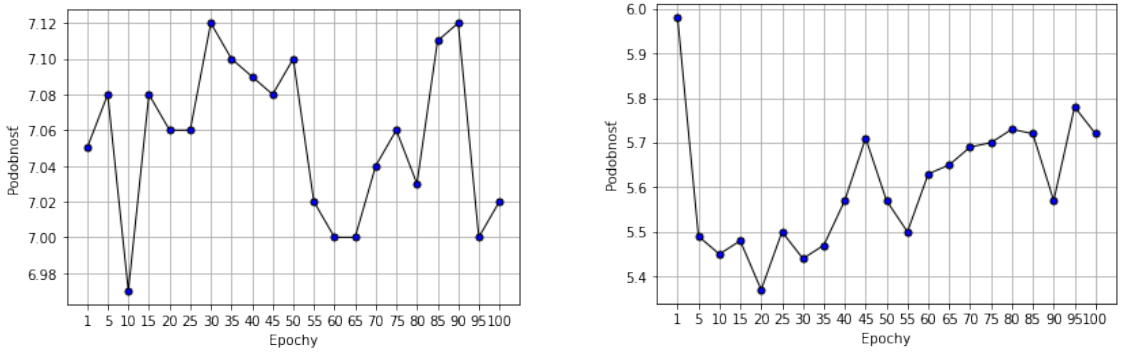


Obr. 4.11: Uber den Wolken, porovnanie prahov

4.2 Prvotný prístup na zúženom datasete

Ďalší potenciálny spôsob akým zvýšiť podobnosť generovanej stopy je zúženie tréningových datasetov len na najkvalitnejšie vzorky. Otázne však je čo sa dá považovať za najkvalitnejšie vzorky. Originálny dataset obsahoval všetky midi súbory, ktoré obsahujú aspoň jednu gitaru, basu, saxofón, je v nich menej ako 5 iných nástrojov a zastúpenie saxofónu je vyššie ako 20%. Datasety môžu obsahovať niekoľko negatívnych faktorov, ktorých eliminácia zvýši podobnosť s originálom.

Prvým z nich je odstránenie piesní, ktoré obsahujú ostatné, nepotrebné nástroje. Pomocou už spomínaného súboru so štatistikami som vybral všetky, ktoré obsahujú aspoň jednu gitarovú, saxofónovú, basovú a maximálne jednu klavírnu stopu okrem nich neobsahujú žiadne iné nástroje. Jednu klavírnu stopu som bol nútený ponechať, pretože bez nej by bol počet súborov na učenie a trénovanie príliš malý. Po filtrovaní mi ostalo 460 midi súborov. Aby sme dokázali zistiť aspoň trocha validnú informáciu o zlepšení alebo zhoršení podobnosti modelu v porovnaní s predchádzajúcim, rozhodol som sa ho otestovať na 400 testovacích súboroch z predchádzajúcej podkapitoly. Zo 460 vyfiltrovaných súborov som odstránil všetky ktoré sa nachádzajú v spomínaných 400 testovacích súboroch, z čoho mi ostalo 421 adekvátnych piesní na učenie, tie som ďalej konvertoval do 23 datasetov pomocou metódy spomínanej pri začiatku tejto kapitoly. Model som nechal iterovať 100 epoch a po pár hodinách som mal výsledky.



(a) Výsledok Keras funkcie na prvotnom modele

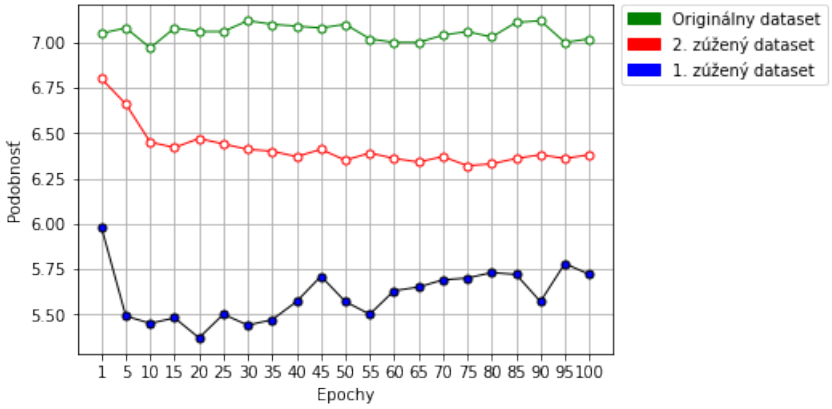
(b) Výsledok Keras funkcie na zúženom modele bez ostatných nástrojov

Obr. 4.12: Porovnanie podobností výstupov Keras funkcie

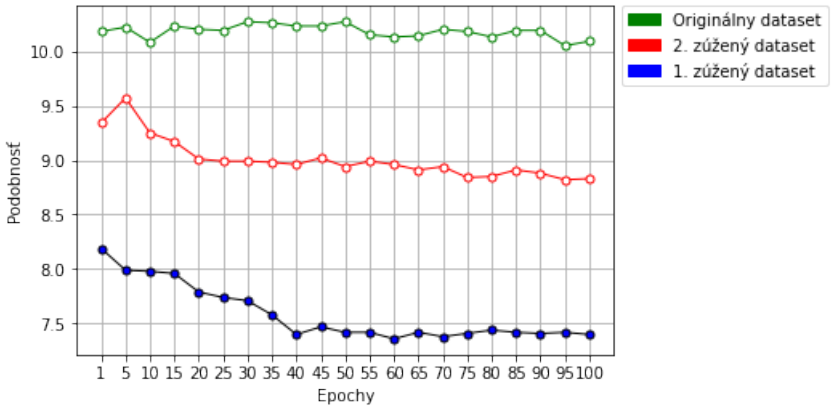
Už pri porovnaní podobnosti Keras funkcií na obrázku 4.12 môžme vidieť, že sa nejedná o zlepšenie. Ľavý graf popisuje podobnosť modelu trénovanom na originálnych 4000 piesňach, graf vpravo vyjadruje podobnosť modelu trénovanom na zúženom datasete. Najvyššiu podobnosť dosiahol model hneď pri prvej epoche. Rovnako negatívne vyšli aj výsledky pri mojej funkcii, z tohto dôvodu som sa sem rozhodol zvyšné grafy ani neuviesť. Aritmetický priemer hodnôt podobností pri predchádzajúcom modeli s prahom o hodnote 0,25 je 10,18%, jedná sa o najvyššiu dosiahnutú podobnosť. Model so zúženým datasetom dosiahol pri prahu 0,25 podobnosť o hodnote iba 7,59%. S istotou môžme skonštatovať, že počet nástrojov nie je jeden z hlavných rozhodovacích faktorov.

Druhým prístupom je sprísnenie percentuálneho zastúpenia saxofónu v piesni. Zo súborov vyberieme všetky, ktoré obsahujú pôvodne požiadavky ako pri prvom modeli, t.j. aspoň jednu gitaru, basu, saxofón a menej ako 5 iných nástrojov. Rozdiel nastáva pri saxofónovej

stope, ktorá musí znieť aspoň 70% piesne čo je nárast o 50% v porovnaní s 20% v prvotnom prístupe. Po filtrovaní som získal 823 piesní, z ktorých som 50 odstránil, pretože sa nachádzajú v testovacom balíčku. Z výsledných 773 súborov som vytvoril 39 datasetov a započal 100 epoch učenia.



Obr. 4.13: Porovnanie výsledkov Keras funkcie



Obr. 4.14: Porovnanie výsledkov Mojej funkcie

Na obrázku 4.13 môžme vidieť porovnanie výsledkov podobnosti Keras funkcie pre originálny aj zúžený dataset. Výsledky sú pozitívnejšie ako pri prvom zúženom datasete avšak aj v tomto prípade je podobnosť modelu nižšia ako pri originále. Do grafu 4.14 som spracoval najvyššie dosiahnuté podobnosti všetkých modelov vypočítanej mojou funkciou a to pri prahu o hodnote 0,25. Môžme vidieť, že pri originálnom datasete sú výsledky medzi epochami relatívne stabilné, zatiaľ čo pri zúžených datasetoch tieto výsledky naprieč epochami klesajú.

Z výsledkov môžme usúdiť, že percentuálne zastúpenie saxofónu je podstatná hodnota, avšak počet tréningových piesní je stále najdôležitejšia premenná pri trénovaní modelu.

4.3 Druhotný prístup

Druhý spôsob, ktorý som sa rozhodol použiť by sa dal považovať za prechod medzi doprednými a rekurentnými sieťami. Prvotný model generoval trojsekundovú stopu bez vedomia

toho čo vygeneroval v predchádzajúcom kroku. Druhotný model má na vstupe aj 3 vygenerované sekundy z posledného kroku. V prípade prvých troch sekúnd, ktorým žiadny krok nepredchádzal je na vstup privedená matica plná núl. Model som znázornil, pomocou funkcie model.summary() z knižnice Keras na obrázku 4.15. Premenná *prev_sax_input* reprezentuje práve tieto predchádzajúce 3 sekundy.

Layer (type)	Output Shape	Param #	Connected to
prev_sax_input (InputLayer)	[(None, 288, 64)]	0	
guitar_input (InputLayer)	[(None, 288, 64)]	0	
bass_input (InputLayer)	[(None, 288, 64)]	0	
concatenate_2 (Concatenate)	(None, 288, 192)	0	prev_sax_input[0][0] guitar_input[0][0] bass_input[0][0]
dense_4 (Dense)	(None, 288, 64)	12352	concatenate_2[0][0]
dense_5 (Dense)	(None, 288, 64)	4160	dense_4[0][0]

Obr. 4.15: Druhotný model

Pri tomto prístupe som sa rozhodol opäť pristupovať dvoma spôsobmi. Buď dostane model na vstupe tri sekundy saxofónovej stopy, ktorú sám vygeneroval v predchádzajúcom kroku alebo dostane tri sekundy z originálnej saxofónovej stopy. Prvý spôsob je intuitívnejší avšak učenie by bolo veľmi zdĺhavé, keďže hodnota batch size by musela byť 1 a musel by som manuálne iterovať dataset. Po nemalej chvíle snahy som však tento spôsob zavrhol s dojmom, že nemôže fungovať a sústredil sa na druhý spôsob.

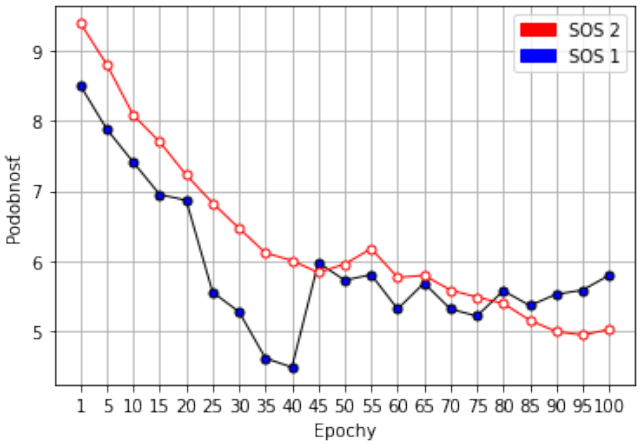
Pri druhom spôsobe kde model dostáva na vstup tri sekundy originálnej predchádzajúcej saxofónovej stopy je učenie modelu rýchlejšie a poskytuje možnosť pripraviť kvalitnejšie datasety, pre jednoduchšie rozlíšenie nadobudol názov a to spôsob s originálnou stopou (ďalej len spôsob SOS). Bohužiaľ pri príprave datasetov som jemne zlyhal ale rozhodol som sa to využiť vo svoj prospech. Pre vysvetlenie zlyhania musím najskôr priblížiť samotný dataset, ktorý je veľmi podobný predchodcovi z prvotného prístupu.

4.3.1 Spôsob SOS

Stopy jednotlivých nástrojov sú opäť rozsekané do trojsekundových úsekov, ktoré sú následne pridávané do troch matíc o veľkosti 1024x288x64 reprezentujúcich jednotlivé nástroje. K týmto trom maticiam som pridal ešte štvrtú, ktorá obsahuje tri sekundy predchádzajúcej saxofónovej stopy. Tu nastalo moje zlyhanie, piesne z ktorých pochádzajú úseky v datasete sa nestriedajú ale naopak ide jedna celá pieseň za druhou. Preto som sa rozhodol využiť tento omyl ako príležitosť zistenia vplyvu striedania piesní v datasete, na výsledok modelu.

Oba modely som nechal učiť 100 epoch. Model, ktorý bol trénovaný na datasete kde sa piesne striedali dosiahol priemernú podobnosť 6,32% zatiaľ čo druhý model dosiahol priemernú podobnosť 5,93%. Maximum bolo dosiahnuté tiež striedajúcim modelom a to o hodnote 9,39 % bohužiaľ hneď na prvej epoche, čo nenapovedá moc kvalitnému modelu. Modely boli opäť testované na spomínanom testovacom balíčku o veľkosti 400 súborov s prahom o hodnote 0,25. Výsledky som spracoval aj do grafu, ktorý môžme vidieť na ob-

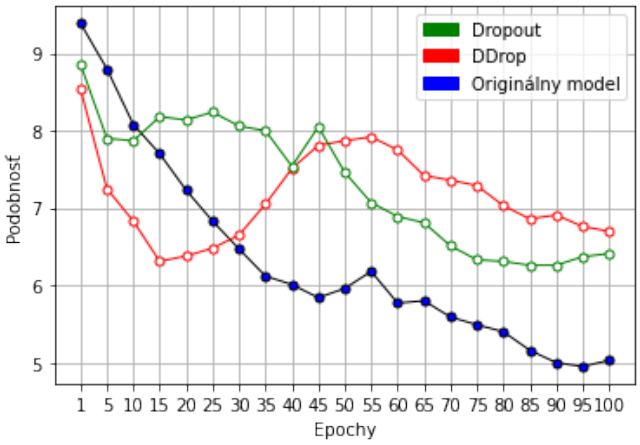
rázku 4.16. Má striedanie piesní na model pozitívny vplyv ? V prvých epochách určite avšak postupom trénovania táto výhoda klesá.



Obr. 4.16: Porovnanie podobností modelu s datasetom kde sa piesne striedajú (SOS 2) a naopak kde nie (SOS 1)

Nadobudnutie maxima modelu hneď na začiatku a nasledujúce klesanie je pravdepodobne prejavom pretrénovania siete. Pre overenie tejto teórie som sa rozhodol natrénovať ďalšie dva modely so striedajúcimi piesňami s dodatočnými **Dropout** vrstvami. Dropout [5] je technika, ktorá adresuje problém pretrénovania siete. Rieši ho odpojením náhodných neurónov spolu s jeho väzbami počas trénovania, čím zabráni ich nadmernej spolupráci.

Prvý model obsahuje Dropout vrstvu o hodnote 20% nachádzajúcu sa pred výstupnou vrstvou. Druhý model obsahuje dve Dropout vrstvy jednu o hodnote 20 % nachádzajúcou sa hneď za vstupom a druhú o hodnote 30 % nachádzajúcu sa pred výstupnou vrstvou. Na obrázku 4.17 môžme vidieť porovnanie modelu s Dropout vrstvami a bez. Síce sa hodnota počiatočného maxima znížila, prudkosť klesania bola výrazne potlačená. Model s dvoma Dropout vrstvami sa mi javí ako najlepšia možnosť, keďže dosahuje najstabilnejšie výsledky.



Obr. 4.17: Porovnanie podobností modelu s jednou Dropout vrstvou o hodnote 20% (Dropout), dvoma vrstvami 20 a 30%(DDrop) a originálnym modelom

Po vypočutí niekoľkých výtvorov druhotného prístupu môžem jednoznačne usúdiť, že jeho výsledky sú hudobne nekvalitnejšie v porovnaní s prístupom prvotným. Generuje podstatne menej tónov a pri vyšších prahoch skĺzne do opakovania jedného tónu stále za sebou v rytmickom súlade. Druhotný prístup pokladám za neúspech.

Kapitola 5

Generovanie saxofónu rekurentnou neurónovou sieťou

Ďalší spôsob trénovania, ktorému som chcel dať v mojej práci šancu je založený na využití rekurentnej neurónovej siete. V teórii by mala byť na tvorenie hudby adekvátnejším kandidátom ako dopredná neurónová sieť. Model si do istej miery dokáže zapamätať čo sa už v piesni odohralo a podľa toho ovplyvniť svoj výstup.

5.1 Tvorba modelu

Hlavným problémom pri tvorbe modelu rekurentnej neurónovej siete je tréningová doba, ktorá je pomerne vysoká. Z tohto dôvodu som bol nepriamo nútený vytvoriť čo najjednoduchší model. Vstup tohto modelu bude rovnaký ako u doprednej neurónovej siete a to dve matice o veľkosti 288x64 reprezentujúce gitaru a basu. Tie sú spojené do jednej matice a posúvané ďalej ako vstup pre LSTM vrstvu. Za ňou nasleduje Dense vrstva a finálna výstupná vrstva so sigmoid aktivačnou funkciou. Nad jednotlivými vrstvami využívam aj techniku Dropout o hodnote 0,3 pre zamedzenie pretrénovania modelu. Pre lepšiu predstavu som model vypísal pomocou Keras summary() funkcie na obrázku 5.1. Model rovnako ako aj predchádzajúce modely používa stratovú funkciu *categorical_crossentropy* a optimalizátor *Adam*.

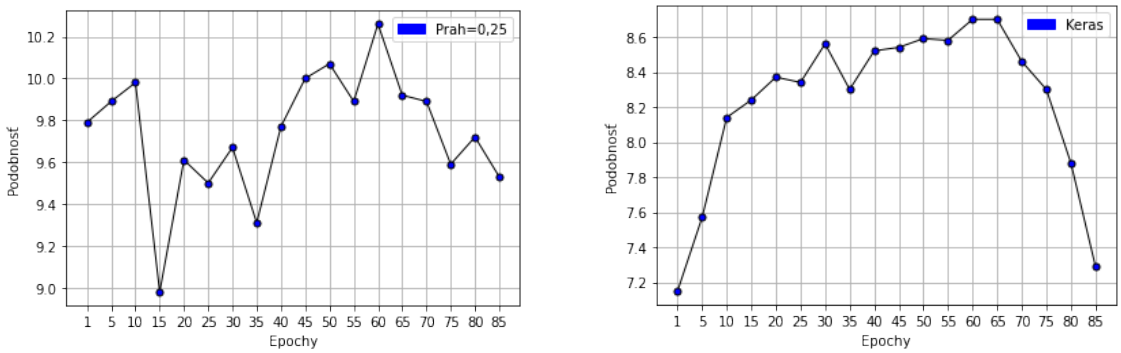
Model používa rovnaký dataset ako dopredná neurónová sieť z prvého prístupu. Aj keď je model blízko najjednoduchšiemu možnému prípadu, trvá trénovanie jednej epochy na google colabe 90 minút. Pre zrovnanie, epocha u prvotného prístupu dopredného modelu trvala 16 minút. Čo je celkom zásadný problém a budúcej modifikácii modelu nenapomáha.

5.2 Trénovanie a vyhodnotenie modelu

Model som mal v pláne trénovať opäť 100 epoch avšak po 87. mi začal vracať model samé nuly a bol som nútený jeho trénovanie predčasne ukončiť. Keďže prvotný model dosahoval najvyššiu podobnosť pri prahu o hodnote 0,25, použil som rovnaké nastavenie pri vyhodnotení aj tu. Výsledok som spracoval do grafu na obrázku 5.2, ktorý obsahuje aj podobnosť Keras funkcie. Prvotný model dosiahol maximálnej podobnosti o hodnote 10,28 %, rekurentnej neurónovej sieti sa podarilo maximum o hodnote 10,26 % a to po 60. epoche. Výsledok Keras funkcie je priemerne o 1 % vyšší ako u prvotného modelu.

Layer (type)	Output Shape	Param #	Connected to
guitar_input (InputLayer)	[(None, 288, 64)]	0	
bass_input (InputLayer)	[(None, 288, 64)]	0	
concatenate_1 (Concatenate)	(None, 288, 128)	0	guitar_input[0][0] bass_input[0][0]
dropout_3 (Dropout)	(None, 288, 128)	0	concatenate_1[0][0]
LSTM1 (LSTM)	(None, 288, 128)	131584	dropout_3[0][0]
dropout_4 (Dropout)	(None, 288, 128)	0	LSTM1[0][0]
DENSE1 (Dense)	(None, 288, 96)	12384	dropout_4[0][0]
dropout_5 (Dropout)	(None, 288, 96)	0	DENSE1[0][0]
output (Dense)	(None, 288, 64)	6208	dropout_5[0][0]

Obr. 5.1: Model rekurentnej neurónovej siete



(a) Výsledok mojej funkcie pri prahu o hodnote 0,25

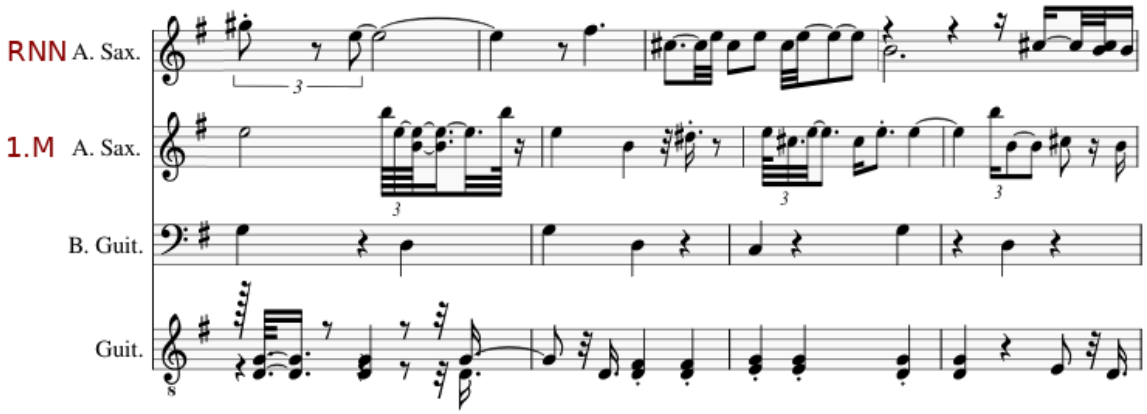
(b) Výsledok Keras funkcie

Obr. 5.2: Porovnanie podobností výstupov Keras a mojej funkcie

Keďže podobnosť s originálom vyšla percentuálne blízko prvotnému modelu s doprednou neurónovou sieťou, čakal som približne rovnako kvalitné výsledky. Pre ich subjektívne porovnanie som si vygeneroval rovnaké piesne ako u prvotného modelu s prahom o hodnote 0,7 a započal moje hodnotenie. Niektoré z piesní neobsahovali žiadne saxofónové tóny. Sieť si nebola ani z ďaleka taká istá ako u prvotného modelu. Piesne vygenerované s hodnotou prahu vyššou ako 0,3 obsahujú z podstatnej väčšiny len jeden opakujúci sa tón a u prahov vyšších ako 0,5 nie sú vygenerované tóny skoro žiadne. Prahová hranica, ktorá oddeľovala v prvotnom modeli piesne vygenerované sólovým prístupom s ladiacimi tónmi (prípady, kedy bol prah v rozmedzí 0,6 až 0,7) od piesní, ktoré sú už presýtené často aj falošnými tónmi (prah o hodnote menej ako 0,5) je v tomto modeli veľmi úzka na to, aby bol nájditeľný kompromis pre všetky piesne. Práve z tohto dôvodu má väčšina vygenerovaných piesní buď primálo tónov na to, aby bol doprovod hudobne zaujímavý alebo ich je tam príliš veľa a znie to ako Free jazz (bez urážky fanúšikov tohto žánru). Výsledok tohto modelu utvrdil aj fakt, že hodnotiť model generujúci hudbu objektívne je veľmi zložité a podobnosť s originálom nie je dôkazom kvalitných vygenerovaných piesní.

Pre kompletnosť som sa rozhodol opäť pridať porovnanie notového zápisu. Na obrázku

5.3 môžme vidieť dve saxofónové stopy vygenerované do piesne Bad Moon Rising. Vrchná z nich (**0,25T_BadMoonRising.mid**) je produktom tohto modelu a stopa pod ním (**0,25T_30.epoch_BadMoonRising.mid**) je vygenerovaná pomocou prvotného modelu, obe pri prahu o hodnote 0,25. Noty tohto modelu znejú useknuté a kratšie ako by mali, čo sa však javí ako celkový problém modelu s LSTM vrstvou.



Obr. 5.3: Porovnanie saxofónovej stopy RNN modelu a prvotného modelu oba s prahom o hodnote 0,25 na piseni Bad Moon Rising

Model s LSTM vrstvou má však aj pozitívne stránky, jeho výsledné noty sú príjemne odlišné od prvotného modelu a našiel som vo vygenerovaných skladbách aj zaujímavé úseky, ktoré si nemyslím, že by boli v prvotnom modeli reálne. Bohužiaľ časová náročnosť trénovania modelu mi nedovoľuje sa mu ďalej venovať, avšak stále si myslím, že má na generovanie doprovodu veľký potenciál, ktorý nie som schopný plne využiť.

Kapitola 6

Model generujúci klavírny doprovod

Tento model bol vytváraný z dôvodu uvedomenia si pár nedostatkov v mojej práci, ich následného odstránenia a v snahe využiť moje novonadobudnuté skúsenosti o algoritmickom komponovaní hudby. Primárny problém predchádzajúcich kapitol je zakorenený v nedostatočnej schopnosti saxofónu vystupovať ako doprovodný nástroj. Riešením tohto problému bolo generovanie viacero saxofónových tónov v jednom okamihu čo pôsobilo orchestrálnym dojmom a bohatším zvukom. Avšak tento spôsob nebol objektívne ohodnotiteľný, keďže podstatná väčšina tréningových piesní neobsahovala viac saxofónov ale naopak používala saxofón ako zdroj melodickej linky alebo sólového úseku. Zo subjektívneho hľadiska u mňa tento spôsob jednoznačne vyhrával a preto som sa rozhodol naňho nadviazať v podobe modelu, ktorý bude generovať klavírny doprovod. Klavír sa na túto úlohu hodí podstatnejšie viac z dôvodu možnosti hrania akordov čo je práve v mojom ponímaní hlavné čaro doprovodu. Taktiež tréningový dataset je obšírnejší z dôvodu frekventovanejšieho výskytu klavíru v porovnaní so saxofónom.

6.1 Tvorba datasetu

Pri tvorbe datasetu som sa rozhodol zamerať na najfrekventovanejšie používané nástroje a to vyhráva práve zloženie v podobe gitara, basa a klavír. Využil som svoj skript na filtrovanie MIDI súborov, opísaný v tretej kapitole tejto práce, na vygenerovanie nových štatistík. Tieto štatistiky namiesto saxofónového zastúpenia obsahujú percentuálne znenie piana, basy a gitary, t.j. aké percento piesne znie daný nástroj. Počet piesní ktoré obsahujú gitaru, basu, piano a menej ako 4 iné nástroje vyšiel na 21172 kusov. Z toho som sa rozhodol vybrať všetky, v ktorých znejú požadované nástroje aspoň 70% času a ostalo mi 6719 dúfam, že kvalitných piesní na trénovanie. Z nich som odobral náhodných 600 skladieb na testovanie a zo zvyšku vytvoril datasety.

Keďže prvotný prístup dosahoval najvyššiu podobnosť s originálom a jeho výsledky sa mi aj subjektívne najviac páčili, rozhodol som sa ho využiť aj v tomto prípade a práve z toho dôvodu sú mu datasety veľmi podobné. Namiesto saxofónovej stopy obsahujú stopu klavírnu, počet striedajúcich piesní som zvýšil na hodnotu 16, keďže striedanie sa osvedčilo, že ma na výsledný model pozitívny vplyv a počet trojsekundových stôp v datasete som zdvojnásobil z hodnoty 1024 na 2048. Výsledkom je 196 datasetov. Model som doplnil aj o dve Dropout vrstvy a jeho finálna podoba je vidieť na obrázku 6.1, na ktorom môžme

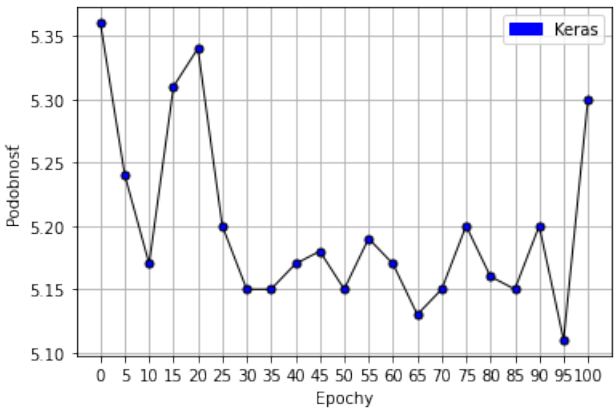
pozorovať prvú Dropout vrstvu o hodnote 20 % hneď za vstupnou vrstvou a druhú Dropout vrstvu o hodnote 30 % nachádzajúcou sa pred výstupnou vrstvou obsahujúcou aktivačnú sigmoid funkciu.

Layer (type)	Output Shape	Param #	Connected to
guitar_input (InputLayer)	[(None, 288, 64)]	0	
bass_input (InputLayer)	[(None, 288, 64)]	0	
concatenate_7 (Concatenate)	(None, 288, 128)	0	guitar_input[0][0] bass_input[0][0]
Dropout_20_percent (Dropout)	(None, 288, 128)	0	concatenate_7[0][0]
dense_9 (Dense)	(None, 288, 64)	8256	Dropout_20_percent[0][0]
Dropout_30_percent (Dropout)	(None, 288, 64)	0	dense_9[0][0]
Output_Sigmoid_Layer (Dense)	(None, 288, 64)	4160	Dropout_30_percent[0][0]

Obr. 6.1: Model obsahujúci dve Dropout vrstvy

6.2 Trénovanie a vyhodnotenie modelu

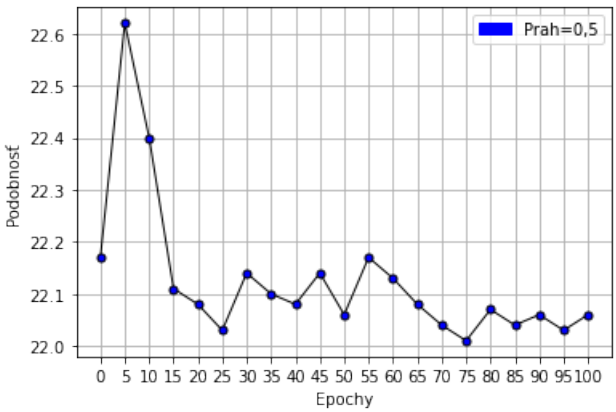
Model opäť prekonal sto iterácií v podobe sto epoch, trénovanie jednej trvalo cca 19 minút, čo je v porovnaní s prvotným modelom obsahujúcim saxofón navýšenie len o 3 minuty. Avšak datasety obsahujú o 50% piesní viac, z čoho sa dá usúdiť, že zvýšenie počtu vzoriek v jednom datasete pozitívne ovplyvnilo rýchlosť učenia. Opäť som vypočítal podobnosť s originálom pomocou Keras funkcie, ktorú môžme vidieť na obrázku 6.2. Výsledky sú celkom stabilné počas všetkých epoch, pohybujúce sa v rozmedzí 5,1 až 5,35 %, sú však nižšie ako u saxofónového modelu. Medzi 95. a 100. epochou nastalo zaujímavé stúpnutie, ktoré ma donútilo natrénovať model ešte ďalších desať epoch, avšak v nasledujúcich epochách som sa stretol len s miernym klesaním.



Obr. 6.2: Podobnosť s originálom Keras funkcie

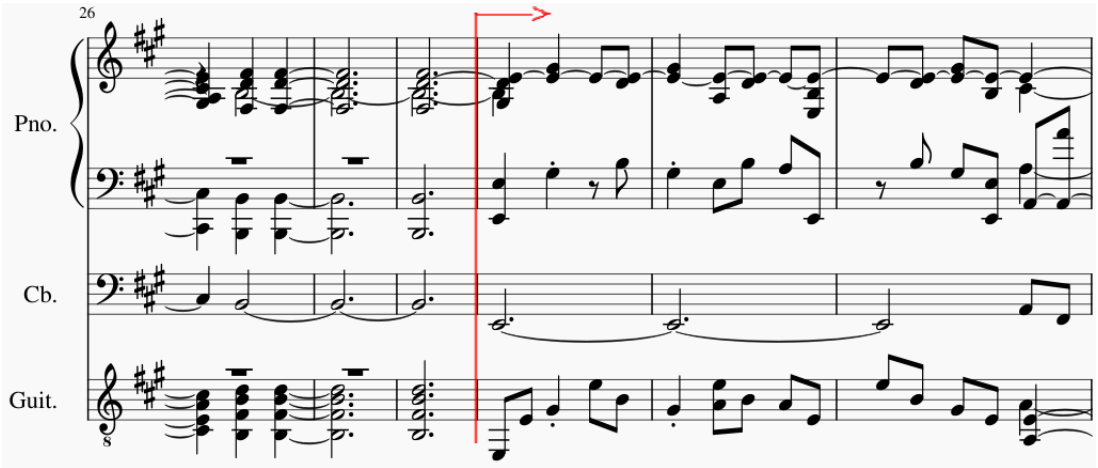
Ďalším krokom je vypočítanie podobnosti mojou funkciou, po prevedení pár zmien som funkciu upravil, aby fungovala aj na piano model. Za zahrané sú požadované všetky tóny, ktorých hodnota je vyššia ako 0,5 a taktiež možnosť sólového nástroju som nastavil na

False, čiže sa neberie len tón s najvyššou hodnotou u každej vzorky ale všetky tóny, ktorých hodnota je vyššia ako spomínaný prah, rovnako ako pri prahovom prístupe. Nasledovalo jedno z mojich šťastnejších období vo fáze vypracovávania tejto práce. Výsledky dosahovali podobnosť vyššiu ako 22% ! Opäť som ich spracoval do grafu na obrázku 6.3. Najprv som sa snažil nájsť chybu v mojej funkcii ale po chvíli som to vzdal a vypočul si vygenerované piesne, z ktorých som nadšený. Približne každá piata sa dá považovať za kvalitný doprovod. Noty sú v podstatnej väčšine dĺžkovo správne, neznejú sekavo a klavír je schopný prispieť do piesne aj po rytmickej stránke, hraním akordov na ľavej ruke, práve túto možnosť saxofón postrádal.



Obr. 6.3: Podobnosť mojej funkcie pri prahu o hodnote 0,5

Jednu z mojich obľúbených vygenerovaných stôp môžme vidieť na obrázku 6.4, úsek bol vygenerovaný do piesne This Time od Bryana Adamsa (**0,5T_ADAMS.mid**). Časť, ktorá ma zaujala sa nachádza za nenápadne dokreslenou červenou čiarou so šípkou a reprezentuje v mojom subjektívnom ponímaní ideálny vstup pre model. Jednoduchá basová linka doplnená o rýchlu melódiu gitarového pôvodu. Práve v tomto prostredí vyznie rytmická časť ľavej ruky a melódia je harmonicky doplnená tónmi ruky pravej.



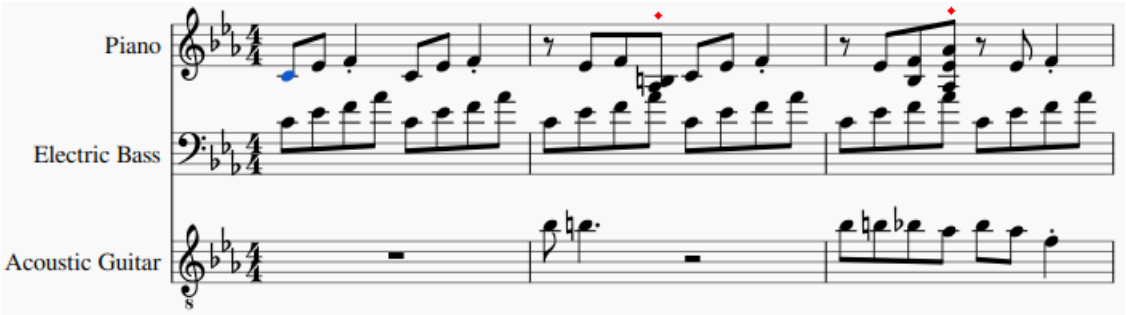
Obr. 6.4: Úsek z vygenerovanej stopy do piesne This Time od Bryana Adamsa

6.2.1 Vyhodnotenie modelu na vlastnej skladbe

Ako posledný test klavírneho modelu som sa rozhodol vytvoriť jednoduchú krátku skladbu, ktorá bude zameraná na silnejšie a slabšie stránky modelu, ktoré som si všimol pri počúvaní vygenerovaných stôp. Skladba pozostáva z troch hlavných častí:

- Prvá časť - je zložená z opakujúcej sa basovej linky a jednoduchej gitarovej melódie. Mala by predstavovať ideálny vstup pre model.
- Druhá časť - je zložená z basovej melódie, ktorá je doplnená o jeden gitarový akord. Slúži na otestovanie schopnosti modelu generovať klavír podľa basy.
- Tretia časť - je zložená z gitarových akordov a jednoduchej basovej linky. Mala by priblížiť jednu z najslabších stránok modelu - presýtenosť tónov pri snahe kopírovať gitarové akordy.

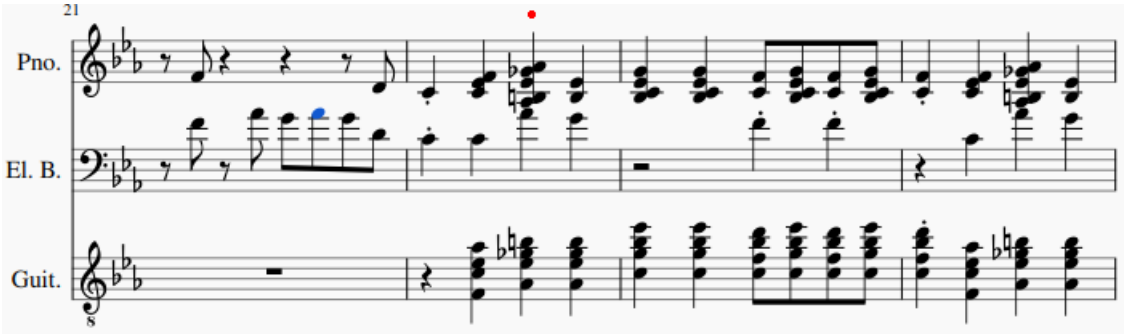
Skladba bola vytvorená v programe REAPER [1], ktorý umožnuje exportovať MIDI formát.



Obr. 6.5: Úsek z prvej časti

Na obrázku 6.5 môžme vidieť úsek z prvej časti skladby, do ktorej som vkladal najväčšiu nádej. Vygenerovaná stopa opakuje do istej miery basovú linku avšak nastali dve hudobne zaujímavé časti, ktoré som vyznačil červenými bodkami. Prvá bodka označuje dvojtónový akord, ktorý neopakuje basovú linku ale naopak robí to čo by mal – dotvára zaujímavú zmenu v melódii. Druhá bodka označuje príjemný akord, ktorý rozširuje basovú linku.

Doprovod vygenerovaný v druhej časti piesne mi neprišiel zaujímavý, pozostával len z pár pridaných tónov. Preto som sa sem obrázok z tejto časti rozhodol neuviesť.



Obr. 6.6: Úsek z tretej časti

[1] https://www.reaper.fm/

Do tretej časti som vkladal najmenšiu nádej ale bol som naopak príjemne prekvapený. Po prvom počutí pôsobil vygenerovaný doprovod opäť presýtený, avšak po viacerých vypočutiach som tento názor začal strácať. Presýtenosť, ktorú som vnímal aj z viacerých predchádzajúcich piesní bola pravdepodobne spôsobená nedostatočným vypočutím daných piesní. Na obrázku 6.6 môžme vidieť úsek z tejto časti.

Klavírny model považujem za úspešný a viem si predstaviť jeho využitie v mojej budúcej, hudobnej tvorbe. Do piesne som vygeneroval stopu aj pomocou iných modelov a nachádzajú sa v prílohovej časti tejto práce: **CD/Songs/Vlastná pieseň**.

Kapitola 7

Záver

Počas tvorenia tejto práce som bol vystavený nespočetnému množstvu zlých, primárne saxofónových ale aj klavírnych kreácií mojich modelov. Na druhú stránku sa medzi nimi nachádzalo aj nemalé množstvo kvalitných hudobných doprovodov, z ktorých som bol viac než príjemne prekvapený.

Problém pri generovaní hudby je nedostatočná schopnosť objektívneho hodnotenia vygenerovaných skladieb. Dosiahnutie vysokej podobnosti vygenerovanej skladby s originálom nie je záruka kvalitného doprovodu. Práve z tohto dôvodu som chvíľami spoliehal na svoje subjektívne hodnotenie viac ako bolo vhodné.

Podarilo sa mi úspešne skonvertovať MIDI nahrávky do matíc potrebných pre trénovanie neurónovej siete a z týchto matíc opäť do MIDI formátu. Vytvoril som systém, pomocou ktorého môžem jednoducho natrénovať model aj pre iné frekventovane používané nástroje. V práci som sa zameral primárne na saxofón, ktorý som generoval troma hlavnými spôsobmi. Prvotný prístup, predstavujúci klasickú doprednú neurónovú sieť, neobsahujúci žiadnu spätnú väzbu. Druhý prístup, kedy sieť dostávala na vstup svoje tri predchádzajúce vygenerované sekundy a tretí prístup využívajúci LSTM vrstvu. Prvý prístup dosahoval najlepšie výsledky ako po subjektívnej tak aj objektívnej stránke. Ako najväčší úspech tejto práce by som určite považoval klavírny model, ktorý som otestoval aj na vlastnej skladbe a ktorého výsledky prekonali moje očakávania.

7.1 Potenciálne budúce rozšírenia

Časť práce, ktorú by som vybavil rozšírením je generovanie doprovodu pomocou rekurentnej neurónovej siete. Jej model nevyšiel podľa mojich predstáv a začal po 87. epoche generovať len nuly. Tento problém by mohol byť riešený úpravou optimalizátoru, či už znížením jeho hodnoty learning rate, alebo vybratím úplne iného.

Ďalším vylepšením by mohlo byť napísanie funkcie, ktorá by automaticky upravovala vygenerovanú stopu. Častým problémom boli noty znejúce sekavo, kratšie ako by mali. Funkcia by ich automaticky zarovnala na adekvátnu dĺžku na základe tempa.

Zaujímavým experimentom by bolo natrénovanie modelu pre generovanie basy len na základe gitary. Na výstup takéhoto modelu by sa dalo nadviazať už existujúcim generovaním saxofónu alebo klavíru, t.j. vygenerovanie stopy na základe vygenerovanej stopy. Rozšírenie databázy MIDI súborov by malo určite tiež pozitívny vplyv na výsledok.

Literatúra

[1] Eck, D. a Lapamle, J. *Learning Musical Structure Directly from Sequences of Music* [online]. University of Montreal, december 2010 [cit. 2020-26-02]. Dostupné z: http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/lstm_music.pdf.

[2] Hermann Hild Johannes, F. a Menzel, W. HARMONET: A Neural Net for Harmonizing Chorales in the Style of l.S.Bach. [online]. 1992, [cit. 2020-16-05]. Dostupné z: http://papers.nips.cc/paper/576-harmonet-a-neural-net-for-harmonizing-chorales-in-the-style-of-j-s-bach.pdf.

[3] Järveläinen, H. *Algorithmic Musical Composition* [online]. Helsinki University of Technology, apríl 2000 [cit. 2020-22-01]. Dostupné z: https://engineering.purdue.edu/ece477/Archive/2006/Fall/F06-Grp01/files/alco.pdf.

[4] midiman. *The Largest MIDI Collection on the Internet* [online]. Reddit, jún 2015 [cit. 2020-04-03]. Dostupné z: https://www.reddit.com/r/WeAreTheMusicMakers/comments/3ajwe4/the_largest_midi_collection_on_the_internet.

[5] Nitish Srivastava, A. K. I. S. a Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. [online]. 2014, [cit. 2020-24-04]. Dostupné z: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf.

[6] Raffe, C. a Ellis, D. P. W. *INTUITIVE ANALYSIS, CREATION AND MANIPULATION OF MIDIDATA WITH prettyMidi* [online]. 2014 [cit. 2020-11-03]. Dostupné z: https://colinraffel.com/publications/ismir2014intuitive.pdf.

[7] Roberto De Prisco, A. T. a Zaccagnino, R. A Neural Network for Bass FunctionalHarmonization. [online]. 2010, [cit. 2020-16-05]. Dostupné z: https://link.springer.com/chapter/10.1007/978-3-642-12242-2_36.

[8] Sepp Hochreiter, J. S. LONG SHORT-TERM MEMORY. [online]. 1997, [cit. 2020-09-05]. Dostupné z: https://www.bioinf.jku.at/publications/older/2604.pdf.

[9] Tanner, G. *Introduction to Deep learning with Keras*. Dostupné z: https://towardsdatascience.com/introduction-to-deep-learning-with-keras-17c09e4f0eb2.

[10] Wikipedia. *Algorithmic composition* [online]. Wikipedia, december 2019 [cit. 2020-22-01]. Dostupné z: https://en.wikipedia.org/wiki/Algorithmic_composition.

[11] Wikipedia. *Evolutionary music* [online]. Wikipedia, december 2019 [cit. 2020-22-01]. Dostupné z: https://en.wikipedia.org/wiki/Evolutionary_music.

Príloha A

Obsah CD

CD obsahuje 3 hlavné súbory:

A.1 Program

Obsahujúci zdrojové kódy:

- **generator.py** - Primárny skript na spustenie, ďalej popísaný v nasledujúcej prílohe.
- **datasets.py** - Skript slúžiaci na generovanie datasetov.
- **instrument_recognizer.py** - Skript slúžiaci na generovanie štatistík o midi súboroch.
- **neural_networks.py** - Súbor obsahujúci funkcie pre vytvorenie a trénovanie použitých modelov neurónových sietí.
- **tensor.py** - Skript obsahujúci funkcie na vygenerovanie stopy, vyhodnotenie podobnosti a automatickú prácu s viacerými modelmi.

Súbor taktiež obsahuje podsúbor **models**, v ktorom sa nachádzajú vybraté epochy modelov, pomocou ktorých je generovaná stopa do piesne. V súbore sa tiež nachádza **requirements.txt** slúžiaci na nainštalovanie knižníc pomocou návodu v ďalšej prílohe.

A.2 Songs

Obsahujúci ukážky vygenerovaných piesní roztriedených do súborov podľa použitého modelu. Piesne sú pomenované vo formáte **0,5T_názov.mid** kde 0,5 označuje hodnotu prahu, pri ktorom bola pieseň generovaná.

A.3 Datasets

Obsahujúci datasety použité na učenie a testovanie.

A.4 Source

Obsahujúci zdrojové kódy technickej správy.

A.5 Ostatné

Na CD sa nachádza aj technická správa vo formáte pdf a výstupy skriptu instrument_recognizer: *pianoStats.txt a saxStats.txt* obsahujúce informácie o MIDI skladbách.

Príloha B

Návod na použitie

Na spustenie je potrebný interpret jazyka Python, pri písaní bola použitá verzia 3.6.9. Ďalej je potrebné nainštalovať potrebné knižnice príkazom:

```
pip3 install -r requirements.txt
```

Následne môžme spustiť program týmto spôsobom:

```
sudo python3 generator.py [-h] -m MIDI [-n NETWORK] [-t TRESHOLD] [-s SOLO] [-b BPM]
```

Povinné argumenty:

- **-m** Slúži na zadanie cesty ku súboru formátu MIDI, v ktorom sa nachádza gitara a basa.

Nepovinné argumenty slúžiace na:

- **-h** Vypísanie nápovedy.
- **-n** Výber modelu siete. Prednastavená je možnosť **sax** využívajúca model z prvotného prístupu. Ďalšie možnosti sú **fb** - druhotný prístup, **rnn** - model RNN, **piano** - model doprednej neurónovej siete generujúci piano.
- **-t** Nastavenie prahu, od ktorého sú noty považované za zahrané. Prednastavená hodnota je **0,5**.
- **-s** Nastavenie sólového prístupu. Prednastavená hodnota je **yes**, stopa obsahujúca takmer len jednu notu každú vzorku. Pri možnosti **no** bude stopa obsahovať akordy a rozmanitejší doprovod.
- **-b** Nastavenie tempa piesne v BPM. Pokiaľ nie je zadaná žiadna hodnota, program sa pokúsi odhadnúť tempo.