VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

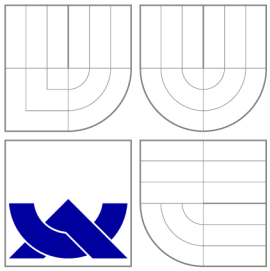
HARMONIZACE MELODIE S MELODICKÝM BASEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

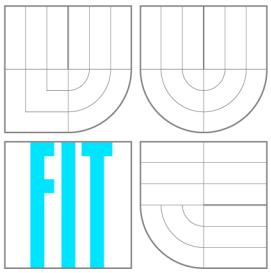
AUTOR PRÁCE
AUTHOR

EMIL KONDELA

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HARMONIZACE MELODIE S MELODICKÝM BASEM

MELODY HARMONIZATION WITH A MELODIC BASS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — EMIL KONDELA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL FAPŠO
SUPERVISOR

BRNO 2010

Abstrakt

Tato bakalářská práce se zabývá harmonizací melodie. Hlavním cílem je vyprodukovat harmonii pro libovolnou melodii v sopránu. Na harmonizaci je využito neuronové sítě. Jako vstup je použita melodie ve formátu midi a na výstupu je harmonie ve formátu MIDI, ABC a ps.

Abstract

This bachelor's thesis deals with melody harmonization. The aim is to produce harmony for previously unseen melody in soprano. Harmonization uses neural networks. For an input a melody in MIDI format is used and for an output a harmony in MIDI, ABC and ps formats is used.

Klíčová slova

Harmonizace melodie, melodický bas, neuronová síť, MIDI, ABC notace

Keywords

Melody harmonization, melodic bass, neural network, MIDI, ABC notation

Citace

Emil Kondela: Harmonizace melodie s melodickým basem, bakalářská práce, Brno, FIT VUT v Brně, 2010

Harmonizace melodie s melodickým basem

Prohlášení

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Fapša.

.....
Emil Kondela
19. května 2010

Poděkování

Rád by som týmto poďakoval vedúcemu mojej bakalárskej práce Ing. Michalovi Fapšovi, za pomoc pri práci a za svoj čas, ktorý mi ochotne venoval pri konzultáciach.

Obsah

Kapitola 1

Úvod

Na otázku, čo je hudba neexistuje len jedna konkrétna odpoveď. Názory na jej zodpovedanie sa výrazne odlišujú. Rovnako ako ju nevieme presne definovať, tak nepoznáme ani ako a kde vznikla. Jedno, čo vieme určite, že je už od dávnych čias bola zaraďovaná k podstate ľudského života. Môže sa nazvať aj najstarším a zároveň najuniverzálnejším jazykom ľudstva.

Na hudbu sa dá pozerať aj z fyzikálneho pohľadu. Tento pohľad definuje hudbu ako organizované kmitanie šíriace sa vzduchom, ktoré spĺňa určité fyzikálne pravidlá. Hudba je práve to miesto, kde sa veda s citom stretávajú a snažia sa vytvoriť niečo spoločné.

Cieľom tejto bakalárskej práce bolo vytvoriť nástroj, ktorého cieľom je zautomatizovať jedno z odvetví hudby a to hudobnú harmonizáciu. Snaží sa doplniť harmóniu k zadanej postupnosti nôt (melódií).

Vytvorený nástroj by mohol byt užitočnou pomôckou pre začínajúcich a aj pokročilých hudobných skladateľov a mohol by im ušetriť veľa času. Tvorba hudby počítačom je veľakrát odmietaná v kruhoch profesionálnych hudobných skladateľov a hudobníkov. Nakoľko považujú hudbu za niečo, čo vychádza iba z človeka, a zautomatizovať ju nie je možné. Tí pokrokovejší túto možnosť až tak nezavrhujú a používajú takéto nástroje na inšpiráciu pri svojej hudobnej tvorbe.

V hudbe síce platia hudobné pravidlá, ale mnohokrát sa stane, že porušenie niektorých pravidiel má za následok pozitívny vplyv na jej celkový výsledok. To je aj dôvod, prečo systémom, ako je tento, nikdy úplné nenahradíme hudobných skladateľov.

Vstupom do tohto nástroju je melódia uložená vo formáte MIDI. Tento formát je priemerne rozšírený a umožňuje jednoduchšie strojové spracovanie. Výstupom programu je zharmonizovaná melódia (k melódii je pridaná harmónia). Výstup je prevedený do ABC notácie, ktorá umožňuje jej ďalšie spracovanie či už do počúvateľného MIDI súboru alebo do notovej reprezentácie.

Kapitola 1 je venovaná základom hudobnej harmonizácie z hľadiska hudobnej teórie. Je tu pojem hudobná harmónia širšie vysvetlený. Popísaná je časť z histórie harmonizácie a jej základné rozdelenie. Bude tu pokiaľ možno jednoducho vysvetlený aj princíp hudobnej harmonizácie.

Kapitola 2 hovorí o strojovom učení. V nej je popísaný jeho základný princíp a jeho delenie. Táto kapitola je zameraná predovšetkým na jeden z typov strojového učenia, ktorý vo svojom programe využívam. Tým typom sú neurónové siete. V tejto kapitole je popísaný ich vývoj a princíp.

Kapitola 3 sa zaoberá predchádzajúcimi prácami, ktoré zoznamujú s problematikou harmonizácie pomocou počítača. Práce slúžia zároveň ako inšpirácia pre tvorbu tohto systému.

Kapitola 4 sa venuje nástrojom, ktoré boli v tejto práci použité. V krátkosti je opísané ich praktické použitie. Sú tu spomenuté výhody a nevýhody jednotlivých nástrojov.

Kapitola 5 sa zaoberá dátami a ich reprezentáciou. v krátkosti Sú tu popísané trénovacie dáta. Taktiež sa popisujú formáty MIDI a ABC, v ktorých sú dáta reprezentované.

V Kapitole 6 je popísaný návrh systému. Je tu vysvetlený presný postup tvorby tohto systému. Postup je rozdelený na 4 časti. Tiež sú tu spomenuté niektoré vylepšenia, ktoré boli v tomto systéme realizované.

Kapitola 7 sa zaoberá samotnou implementáciou systému. Je tu jednotlivo popísaný každý skript a vysvetlené niektoré jeho konštrukcie. Okrem toho je tu popísaný aj použitý programovací jazyk a vývojové prostredie.

V Kapitole 8 sa s vytvoreným systémom experimentuje.

V prílohách je vytvorený návod na použitie tohto systému a notové výstupy zharmonizovaných skladieb.

Kapitola 2

Harmónia

2.1 Hudobná teória

Slovo harmónia pochádza z Grécka a znamená hodiť sa k sebe, spojovať. Gréci slovo harmónia v minulosti používali na označenie pre hudbu. Vo všeobecnosti však harmónia znamená: súzvuk, súznenie viacerých tónov, ktoré sa ku sebe hodia. Harmonizácia sa snaží k jednej hudobnej línii (najčastejšie soprán) pridať tóny z ostatných hlasov. Snaží sa teda k horizontálnej zložke hudby (melódia) pridať vertikálnu zložku (akordy).

Z historického hľadiska môžeme rozlíšiť dve základne časti v hudobnej harmónii. Klasická harmónia, podľa ktorej sa harmonizovalo v období klasicizmu, baroka a romantizmu(J.S. Bach, A. Vivaldi, W.A. Mozart). Podľa modernej harmónie sa začalo harmonizovať od roku 1900 (S. Rachmaninov, A. Novák). [7]

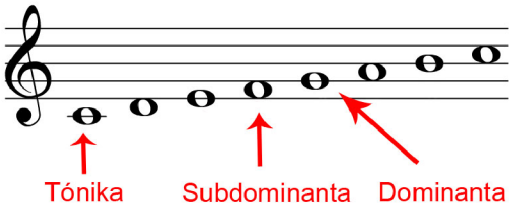
Tým ako tvoriť harmóniu (akordy) a spájať ich sa zaoberá samostatný obor: náuka o hudobnej harmónii a ako predmet sa študuje na hudobných školách. Táto náuka určuje pravidlá, ako spájať tóny, aby vznikli akordy. Harmónia sa inak nazýva tiež homofónia. Opakom homofónie je polyfónia. Polyfónia je súzvuk viacerých hlasov, ktoré na sebe rytmicky nezávisia, ale riadia sa určitými pravidlami, ktorými sa zaoberá kontrapunkt.

Štúdium harmónie je veľmi komplikované. Pravidlá sú síce presne dané, ale veľakrát sa stane, že niektoré pravidlá sa nemusia úplne striktne dodržiavať. Preto sa pri štúdiu harmónie odporúča zapojiť intuíciu a spoľahnúť sa na svoj hudobný sluch. To bol aj dôvod prečo sú niektorí hudobníci voči tvorbe hudby počítačom dosť skeptickí.

Na to, aby sme mohli harmonizovať nejakú melódiu, potrebujeme vedieť v akej tónine je daná skladba napísaná. Ďalšiu podstatnú vec, čo potrebujeme vedieť, je údaj o tom, či je skladba molová alebo durová. Pre durovú stupnicu je charakteristická veľká tercia – znamená to, že medzi prvým a tretím tónom durovej stupnice sa nachádzajú tri tóny (pre stupnicu C-dur je veľká tercia C-E). Obdobne pre molovú stupnicu je charakteristická malá tercia. Znamená to, že medzi prvým a tretím tónom molovej stupnice sa nachádzajú taktiež tri tóny (pre stupnicu A-mol je malá tercia A-C).

Na harmonizovanie používame tri základné harmonické funkcie. Sú nimi Tonika, Dominanta, Subdominanta. Tonika je durový kvint-akord – súzvuk troch tónov pozostávajúci z prvého tónu stupnice, veľkej tercie a kvinty (ide o 1., 3., 5. tón stupnice). Tonika je funkciou harmonického pokoja. Dominanta je kvint-akord v danej stupnici vybudovaný od piatého tónu stupnice. Subdominanta je vybudovaná od štvrtého tónu stupnice a spoločne s dominantou sú funkciou harmonického napätia. Pre stupnicu C-dur je tonika zložená z tónov C E G, dominanta je zložená z tónov G H D a subdominanta z tónov F A C obr.2.1. Obdobne je to pre molovú stupnicu. Na ich označenie sa používajú najčastejšie písmená T

D S.[21]



Obrázek 2.1: Začiatočné noty harmonických funkcií T D S v C-dur

2.2 Harmonizácia s melodickým basom

Harmonizovanie s melodickým basom je súčasťou barokovej harmonizácie. Najčastejšie sa tento prístup využíval pre nástroje ako kontrabas, violončelo alebo fagot. Pri harmonizácii s melodickým basom sa okrem základných prístupov obzvlášť dbá na stavbu basovej melodickej línie.[7]

Kapitola 3

Strojové učenie

3.1 Úvod do strojového učenia

Strojové učenie je rovnako ako aj veľa iných vedných disciplín inšpirované človekom. Ľudia získavajú o veci predstavu tým, že sa učia pojmy a hľadajú medzi nimi súvislosti.

Pre príklad uvediem, ako sa malé dieťa učí poznávať určíté veci. Chceme dieťaťu vysvetliť napríklad, čo je to pomaranč. Najprv mu povieme, že je to guľaté. Dieťa vezme zelené jablko a určí to ako pomaranč. Dieťa musíme opraviť a vysvetliť, že pomaranč má oranžovú farbu. Ďalej dieťa vezme oranžovú loptičku a označí, že je to pomaranč. Zjavne naša definícia nebola dostatočná, preto dieťaťu povieme, že pokiaľ po rozkrojení je vnútro oranžové, dá sa jesť a príjemne vonia, tak by si dieťa malo vytvoriť už predstavu o tom, čo je pomaranč. Je to niečo guľaté, oranžové a dá sa jesť. Toto sú informácie, ktoré dieťa získalo a pomocou nich je schopné si zadefinovať nový pojem pomaranč. Stroje podobne ako ľudia sú schopné sa učiť, ale potrebujú mať zadefinované vlastnosti a súvislosti medzi nimi.

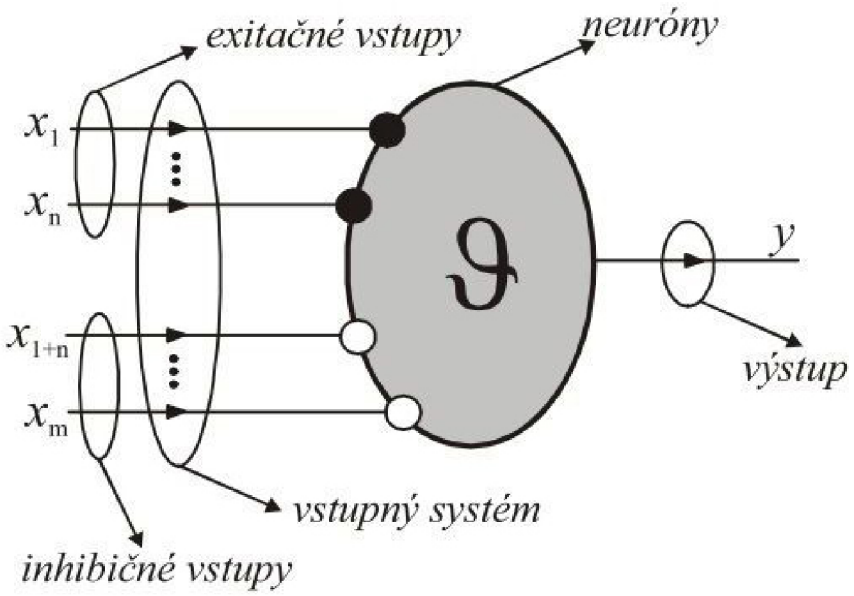
V strojovom učení vzniklo zopár odlišných smerov, v ktorých sa pozerajú na učenie rôzne. Jedna skupina sa zaoberala modelovaním mechanizmov, ktoré prebiehajú v ľudskom tele. Ďalšia skupina zvolila empirický prístup, kde sa nesnažila nájsť všeobecný princíp, na základe pokusov, ich výsledkov a manipulovaním s nimi vytvorili postupy na učenie. Iná skupina sa snažila na problém pozerať matematicky a dokazovaním teorém a vymyslieť nejaké všeobecné princípy.

Jeden z ďalších aspektov, ktoré ovplyvňujú učenie, je stupeň kontroly. Strojové učenie podľa toho delíme na kontrolované alebo nekontrolované učenie. Pri kontrolovanom (*supervised*) učení potrebujeme učiacemu algoritmu predložiť nejaké konkrétne trénovacie príklady, ktoré obsahujú vstupné a výstupné informácie. V prípade klasifikačnej úlohy chceme na základe vstupných dát čo najpresnejšie určiť, do ktorej triedy patria. Znamená to, že výstupná informácia obsahuje už určenú konkrétnu triedu. Ďalším cieľom je naučiť sa správne určiť triedu na základe vstupov, ale zároveň znalosti generalizovať. Znamená to, že systém bude určovať správne výsledné triedy aj pre doteraz nevidené vstupné informácie. Jedna z metód sú nižšie spomenuté neurónové siete .

Pri nekontrolovanom (*unsupervised*) učení nemáme sprístupnené v trénovacích dátach označenie do tried. Ale snažíme sa nájsť nejaké spoločné znaky v dátach, ktoré by mohli reprezentovať rozličné triedy. Jedna z metód nekontrolovaného učenia je *clustering*, kde sa hľadajú spoločné znaky a vytvárajú sa zhluky dát. [5]

3.2 Neurónové siete

Umelé Neurónové siete (*ANN – artificial neural networks*) sú jednou z foriem strojového učenia, ktorá je inšpirovaná fungovaním ľudskej mysle pomocou neurónov. Neuróny v mozgu sú navzájom poprepájané synapsiami, ktoré medzi sebou komunikujú a spracovávajú informácie od okolia. Významnou vlastnosťou neurónových sietí je dobrá aproximácia funkcie, ktorá systém popisuje. Začiatkom päťdesiatych rokov vytvorili McCulloch a Puitts prvý model neurónovej siete reprezentovaný pomocou logického neurónu, ktorý mohol mať len hodnoty 0 alebo 1 obr.3.1. Tento neurón obsahuje excitačné vstupy, ktoré odozvu zosilňujú a inhibičné vstupy, ktoré odozvu zoslabujú.



Obrázek 3.1: Logický neurón McCullocha a Pittsa [22]

Vnútorný potenciál neurónu (ξ) je definovaný ako rozdiel inhibičných a excitačných vstupov. Podľa toho, či je výsledok väčší ako určitý prah (ϑ), sa určuje výstup (y) ako 0 alebo 1. [22]

$$y = \begin{cases} 1 & (\xi = x_1 + \cdots + x_n - x_{1+n} - \cdots - x_m \geq \vartheta) \\ 0 & (\xi < \vartheta) \end{cases}$$

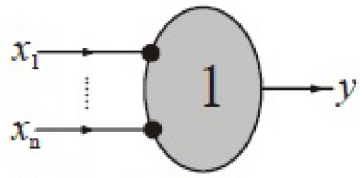
Uvádzam príklad neurónu, ktorý realizuje funkciu OR. obr.3.2

Nevýhodou neurónových sietí s logickými neurónmi je ich neschopnosť sa učiť. Rozloženie a parametre siete sú pevne stanovené a nemenné.

Prevrat v neurónových sieťach spôsobil Frank Rosemblatt. V jeho modeli neurónových sietí použil váhové koeficienty a prahy ako premenné. Tieto parametre sa nastavujú a upravujú pri trénovaní s cieľom, aby sieť čo najlepšie popísala zadaný problém. Nové parametre

$$y_{OR}(x_1, x_2) = s(x_1 + x_2 - 1)$$

#	x_1	x_2	$y_{OR}(x_1,x_2)$	$x_1 \vee x_2$
1	0	0	$s(-1)$	0
2	0	1	$s(0)$	1
3	1	0	$s(0)$	1
4	1	1	$s(1)$	1



$$y = x_1 \vee \ldots \vee x_n$$

Obrázek 3.2: Príklad neurónu Boolovej binárnej funkcie OR [22]

siete sa prepočítavajú pomocou zadanej funkcie, v ktorej vystupuje aj predchádzajúca hodnota.

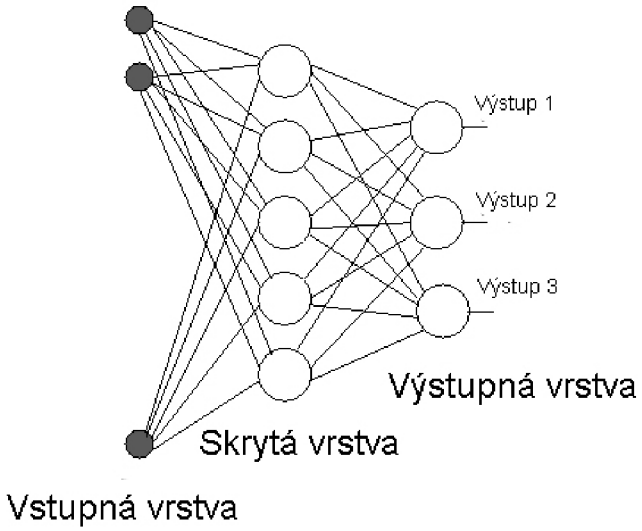
Dopredné neurónové siete (*multilayer neural networks*), ktoré boli vytvorené Davidom Rumelhartom spôsobili druhú revolúciu v neurónových sieťach. Na rozdiel od Rosemblattovych sietí dokáže byť použitá aj na nelineárne problémy. Riešenie je v pridaní skrytej vrstvy (*hidden layer*), čím sa zároveň zvýši kapacita siete. obr. 3.3

Neurónová sieť sa rozdelila na 3 rôzne vrstvy:

1. Vstupná vrstva – obsahuje vstupy siete
2. Skrytá vrstva – prepojenie medzi vstupnou a výstupnou vrstvou
3. Výstupná vrstva – obsahuje výstupy siete

Pridaním skrytej vrstvy sa objavuje nový problém, ako trénovať jednotky v skrytej vrstve, pokiaľ nepoznáme ich vstupy ani výstupy. Riešením je algoritmus Backpropagation. Cieľom tohto algoritmu je minimalizovať celkovú chybu pri spätnom priechode sieťou pre trénovacie dáta.

Pri neurónových sieťach môžeme rozlišovať 2 fázy:



Obrázek 3.3: Dvoj-vrstvá neurónová sieť [13]

1. Fáza trénovania – sieť sa trénuje, upravujú sa jednotlivé váhy. Trénovanie prebieha ako kontrolované učenie. Cyklicky prepočítavame váhy pre každú trénovaciu vzorku. Jeden cyklus cez všetky trénovacie dáta sa nazýva epocha.

2. Fáza vyhodnocovania – na už natrénovanú sieť sa aplikujú vstupné dáta, ktoré chceme vyhodnotiť a vo výstupe získame informácie, ktoré udávajú pravdepodobnosti jednotlivých priradených tried.

Kapitola 4

Predchádzajúce práce

Pred tým, ako som začal pracovať na praktickej časti bakalárskej práce bolo potrebné zoznámiť sa s touto tématikou. Okrem základov hudobnej teórie, ktorú nájdete výborne spracovanú v knihe [21], bolo potrebné zoznámiť sa s tým, ako sa k harmonizácii pristupuje z pohľadu informatiky. Veľké šťastie je, že táto téma je celkom populárna a nie je problém nájsť potrebnú literatúru. K veľkej väčšine minulých prác som sa dostal cez odporučenia od vedúceho práce alebo ako bibliografické odkazy použité v minuloročných bakalárskych prácach. Z veľkého množstva som sa pokúsil vybrať tie práce, ktoré ma zaujali a mohli byť prínosom pre moju prácu.

4.1 Chorálová harmonizácia s použitím prístupu strojového učenia

Strojové učenie sa používa na veľa známych úloh ako sú napríklad rozpoznávanie reči, počítačové videnie alebo ovládanie robota. Napríklad v tejto práci bolo použité strojové učenie pomocou stromových metód. Samotné učenie bolo urobené rovnako ako v iných prácach ako *blackbox* čiže, ako nástroj, ktorý sa priamo neimplementuje, ale využíva sa už niektorý vytvorený. Ako nástroj na učenie bol použitý TiMBL (Tilburg Memory-based Learner).

V tejto práci [2] boli dáta reprezentované pomocou **kern formátu. Medzi lokálne aspekty, ktoré ovplyvňujú harmóniu melódie použil Alex Chilvers momentálnu výšku tónu a výšku predchádzajúceho a nasledujúceho tónu, rovnako tak aj dĺžku tónov a vzdialenosť od taktovej čiary. Medzi globálne aspekty bola použitá tónina a metrum.

4.2 Harmonizácia s použitím Markových modelov

Kaan Biyikoglu's [1] použil na harmonizáciu melódie Markove modely. Použitie Markových modelov vychádza z predpokladu, že pravdepodobnosť udalosti závisí na konečnom počte nadradených udalostí.

Na reprezentáciu tónu je použitých 12 stavov (rovnako ako poltónov v jednej oktáve). Všetky pesničky sú transponované do jednej spoločnej oktávy na trénovanie. Pravdepodobnosti sú nakoniec vyhodnocované použitím algoritmu MLE (The Maximim Likelihood Estimate). Konečný výstupný akord je vyhodnotený Vitrebiho algoritmom. [20]

V ďalšej práci od Moray Allana a Christophera Williamsa [6] boli taktiež Markové modely použité na vytvorenie harmónie. Vychádza z predpokladu, že závisia nielen na pozoro-

vanom (observed) stave ale aj na skrytých (hidden) stavoch. Pozorované stavy sú melódia a skryté stavy sú akordy. Na základe toho sa snaží obmedziť počet možných akordov, ktoré môžu vo výsledku nastať. Pri harmonizácii si skladbu rozdelil na dve časti, podľa toho či skladba bola durová alebo mólová. Najpravdepodobnejšie stavy sú znova vybraté Vitrebiho algoritmom.

Okrem vytvárania akordov bolo v tejto práci vytvorené aj zdobenie harmónie. Pridaním nejakých ďalších tónov alebo zmenou dĺžky tónu. Vo svojej práci som sa rozhodol tento prístup nepoužiť, nakoľko sa mi javil ako veľmi náročný a pre jeho správne fungovanie je potrebné mať veľa dát na trénovanie.

4.3 Harmonizácia s využitím neurónových sietí

Pri štúdiu problematiky som sa taktiež stretol s použitím neurónových sietí na harmonizáciu. Tento prístup sa mi zdal veľmi zaujímavý a tak som si ho aj zvolil vo svojej práci. Preto, bude bližšie popísaný v nasledujúcich kapitolách. Zistil som, že s týmto prístupom sa už pracovalo napríklad pri vytvorení nástroju Harmonet [4]. Ide o nástroj vytvorený Hermann Hildom, Johanesom Feulnerpom a Wolframom Menzelom. Na trénovanie využíva už známe chorály J.S Bacha. Namiesto jednoduchého kódovania každej noty v akorde sa na reprezentáciu používa harmonická funkcia, čo spôsobí, že nikdy nebudú znieť v akorde tóny, ktoré ku sebe nepatria. Funguje tak, že najprv sa vyhodnotí zo sopránovej línie basová línia. Potom sa pomocou pravidiel dorobia ostatné hlasy. Ďalším vylepšením je špeciálna neurónová sieť slúžiaca priamo na ornamentáciu.

Kapitola 5

Nástroje použité v systéme

Táto časť popisuje nástroje, ktoré som použil vo svojom programe na predspracovanie vstupných dát, trénovanie siete a spracovanie výstupov. V krátkosti sa pokúsim vysvetliť základnú funkčnosť a použitie týchto nástrojov.

5.1 Predspracovanie vstupných dát

5.1.1 Nástroj mftext

Hlavné využitie programu mftext je na prevod skladby do textovej reprezentácie, ktorá je pre nás lepšie čitateľná a pochopiteľnejšia. Program na vstupe očakáva súbor vo formáte MIDI a spustí sa nasledujúcim príkazom.

```
mftext [input_file]
```

Na výstupe vytvorí zoznam nôt nazvaný *Note list*. Každá nota na výstupe sa popisuje troma číslami. Prvé číslo *ontime* určuje v akom čase má daná nota začať znieť. Číslo *offtime* určuje kedy má nota prestať znieť. Posledné číslo *pitch* určuje údaj o výške tónu vo forme MIDI čísla (napr. note C1 je vo formáte MIDI čísla pridelené čísla 60. Prehľadná tabuľka spolu s frekvenciou je uvedená v odkaze [16])

```
Note [ontime] [offtime] [pitch]
```

Program mftext dokáže okrem zistenia informácií o note zistiť aj hlavičku MIDI súboru. V hlavičke sa vyskytujú napríklad: názov skladby, informácia či je skladba molová alebo durová, v akej je tónine, informácia o takte a iné. Nakoľko nie všetky MIDI súbory majú poriadne vyplnenú hlavičku, nemôžeme sa na ňu spoliehať a je potrebné tieto informácie získať inak.

Tu je výstup programu mftext pre prvý verš slovenskej ľudovej piesne Kohútik jarabý (5 dvojtónových akordov):

```
Header format=0 ntrks=1 division=192
Track start
```

```
Time=0  Sysex, leng=6
Time=0  Tempo, microseconds-per-MIDI-quarter-note=273972
Time=0  Program, chan=1 program=29
Time=0  Parameter, chan=1 c1=7 c2=127
Time=6496  Meta event, end of track
Track end
Note    273    502 79
Note    273    502 67
Note    547    776 81
Note    547    776 69
Note    821   1301 83
Note    821   1301 71
Note   1369   1849 84
Note   1369   1849 72
Note   1917   2146 84
Note   1917   2146 72
```

5.1.2 Melisma music analyzer

Melisma music analyzer[12] v skutočnosti nie je jeden nástroj ale súbor samostatných nástrojov, ktoré na seba nadväzujú a slúžia hlavne na spracovanie hudobných súborov. Melisma obsahuje nástroje ako sú: grouper, streamer, key, meter, harmony. Nástroje, ktoré som z tejto sady využíval sú tu popísané.

Nástroj meter

Program meter berie ako vstup *Note list* ktorý vyprodukoval mftext a snaží sa skladbu rozdeliť z hladiska taktovania na rovnaké malé časti *Beats-doby*. Môžeme si to napríklad predstaviť, ako keby sme chceli všetky noty v skladbe rozdeliť na šestnástinové doby. Potom sa snaží usporiadať tieto malé časti pravidelnej mriežky dôb, kde sa pravidelne opakujú rôzne úrovne *Level of beats*. Týchto úrovní môže byť 5 (0,1,2,3,4). Každá vyššia úroveň obsahuje aj nižšie úrovne. Takže každá doba bude mať úroveň 0, každá druhá ma úroveň 1 (čiže má aj úroveň 0), každá štvrtá má úroveň 2 (teda aj 0 a 1) ... atď. Toto priradenie je veľmi dobre vidieť, ak si necháme zobraziť aj pseudografický výstup.

Výstup z programu meter je určený rovnako ako pri mftext číslami *ontime* a *offtime* a už spomínaným parametrom *level of beats*

```
Beat [ontime] [offtime] [level_of_beats]
```

Výstup pre skladbu Kohútik jarabý:

```
Beat      0  2
Beat    140  0
Beat    280  1
Beat    420  0
Beat    560  4
```

```
Beat     665  0
Beat     805  1
Beat     945  0
Beat    1085  2
Beat    1225  0
```

Nástroj harmony

Hlavným cieľom nástroju harmony je spraviť harmonickú analýzu zadanej piesne. Ako vstup programu zadávame *Note list* a *Beat list* vygenerovaný programom meter. Pre každú malú časť tzv. Beat sa snaží nájsť jej harmóniu v podobe akordu, ktorý ju bude reprezentovať. Nakoľko Beaty sú často rovnaké, stáva sa, že pre viac Beatov je pridelený jeden akord. Program ma veľmi zrozumiteľný grafický výstup, z ktorého už môžeme získať lepšiu predstavu o skladbe, ktorú chceme harmonizovať.

Výstup programu je okrem grafického aj textový, ktorý je lepšie čitateľný. Okrem časových údajov o trvaní noty udáva aj akord v TPC notácii [19].

```
Chord [ontime] [offtime] [TPC-chord]
```

Výstup pre skladbu Kohútik jarabý:

```
    Chord      0    140  5
    Chord    140    280  5
    Chord    280    420  5
    Chord    420    560  5
    Chord    560    945  6
    TPCNote      0    245  79   3
    TPCNote      0    245  67   3
    TPCNote    280    490  81   5
    TPCNote    280    490  69   5
    TPCNote    560   1015  83   7
Analysis:         Harmonic Rep           TPC Rep
     0  x x x           A          < G   G >       +
   140  x               A          <       >
   280  x x             A            < A   A >             +
   420  x               A            <       >
   560  x x x x x          E             < B   B >                   +
   665  x                  E             <       >                   2
   805  x x                E             <       >                   2
   945  x                  E             <       >
```

Nástroj key

Tento nástroj slúži na zistenie a analýzu tóniny v akej je skladba napísaná. Existuje veľké množstvo metód na hľadanie tóniny. Program key má implementované 4 rôzne metódy. Pre svoju prácu som si vybral analýzu pomocou Krumhansl-Schmuckler algoritmu, nakoľko táto

metóda je jednoduchá a dáva veľmi dobré výsledky. Ďalšie metódy delia skladbu na rôzne *key segmenty* a ich spracovanie by bolo príliš komplikované.

5.2 Trénovanie siete

5.2.1 The NICO toolkit

Tento nástroj vyvinutý švédskym ústavom spracovania reči a hudby na Štokholmskej univerzite slúži na vytváranie umelých neurónových sietí. Hlavným cieľom tohto projektu bolo vyvinúť nástroj na riešenie konkrétnych problémov v spracovaní reči. Jeden z hlavných dôvodov prečo som začal používať tento program bolo, že dokáže veľmi jednoducho vytvoriť rekurentné a time-delay neurónové siete. Druhou výhodou je jeho zverejnenie pod BSD licenciou, čiže je voľne šíriteľný. Síce nemá grafické užívateľské prostredie, ale určite nemá problém poradiť si s veľkým počtom dát pri veľkom počte prepojení.

Chceme vytvoriť jednoduchú neurónovú sieť, ktorá bude mať vstupnú, výstupnú a skrytú vrstvu. Na to potrebujeme pomocou príkazov `AddGroup` a `AddUnit` jednotlivé časti neurónovej siete, v ktorých určíme počet jednotiek v jednotlivých vrstvách. Na pospájanie použijeme príkaz `Connect`. Na trénovanie siete slúži príkaz `BackProp`. Pri trénovaní, ktoré zaberie nemálo času si môžeme nechať vypísať ako sa nám zmenšujú jednotlivé chyby s postupujúcimi iteráciami.

NICO je určite jednoduchý nástroj na prácu neurónovými sieťami, ale vďaka tomu, že nemá dostupnú vhodnú dokumentáciu a nakoľko som nedokázal získať niektoré potrebné informácie, rozhodol som sa tento nástroj nepoužiť.

5.2.2 Neural Network Toolbox

Neural Network Toolbox [14] je rozšírenie softwaru Matlab. Umožňuje navrhovanie, implementovanie, vizualizáciu a simulovanie umelých neurónových sietí. Tento program sa použije tam, kde by formálna analýza bola príliš zložitá alebo aj nemožná. Veľkou výhodou je textové aj grafické užívateľské rozhranie. Grafické rozhranie povoľuje upravovať a vytvárať neurónové siete.

Veľkou výhodou je výborne napísaná dokumentácia [15], kde okrem podrobného zoznamu použitých funkcií, ktorých nie je málo, sú vytvorené aj intuitívne tutoriály na riešenie konkrétnych problémov aj pre začiatočníkov. Tento toolbox ponúka aj výbornú možnosť testovania neurónovej siete, nakoľko si sám rozdelí dáta a pomocou nich sa snaží sieť validovať. Malou nevýhodou je, že tento softvér nie je poskytovaný zadarmo a jeho zadováženie stojí nemalú sumu. Vo väčšine škôl je táto licencia zakúpená.

Návod na vytvorenie jednoduchej siete pomocou textového užívateľského rozhrania [18]:

1. Po spustení programu matlab zadáme vstupné a výstupné vektory

```
inputs = [0 1 0 1; 0 0 1 1];
  targets = [0 1 0 1];
```

2. Vytvoríme neurónovú sieť, ktorá bude mať 20 neurónov v skrytej vrstve.

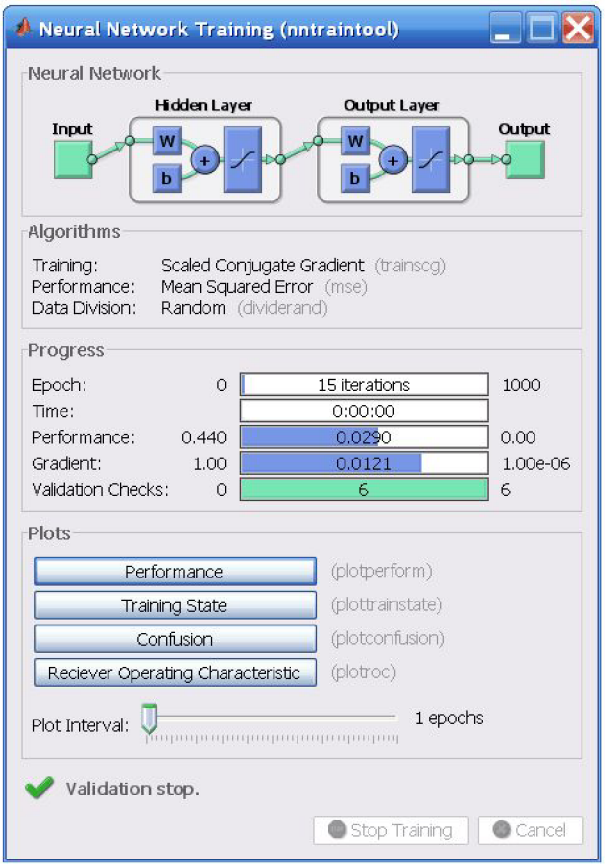
```
net = newpr(inputs,targets,20);
```

3. Spustíme trénovanie siete a zobrazí sa nám grafické rozhranie obr. 5.1, kde môžeme vidieť priebeh trénovania a pri stlačení tlačidla Performance vidíme priebeh zmenšovania chyby siete.

```
net=train(net,inputs,targets);
```

4. Nakoniec chceme použiť natrénovanú sieť na nové dáta a získať potrebný výstup.

```
eval_targets = sim(net,eval_inputs)
```



Obrázek 5.1: Grafické prostredie Matlabu slúžiace na trénovanie siete [18]

5.3 Spracovanie výstupov

5.3.1 abc2midi

Jeden z programov z balíčka abcMIDI [8] vytvorený Jamesom Allwrightom. Program slúži na spracovanie súborov v ABC notácii. Konkrétne tento program prevádza skladbu zo súboru v abc notácii na skladbu do MIDI formátu, ktorý sa už dá prehrať obyčajným prehrávačom.

5.3.2 abc2ps

Je program, ktorý prevádza skladbu v ABC notácii do formátu PS, ktorý je jednoduchšie a intuitívnejšie zobrazovaný. Prípadným prevedením pomocou programu ps2pdf dosiahneme prevod na klasickú notáciu vo formáte PDF, podľa ktorého si skladbu môžeme zaspievať alebo zahrať na nejakom hudobnom nástroji.

Kapitola 6

Dáta a ich reprezentácia

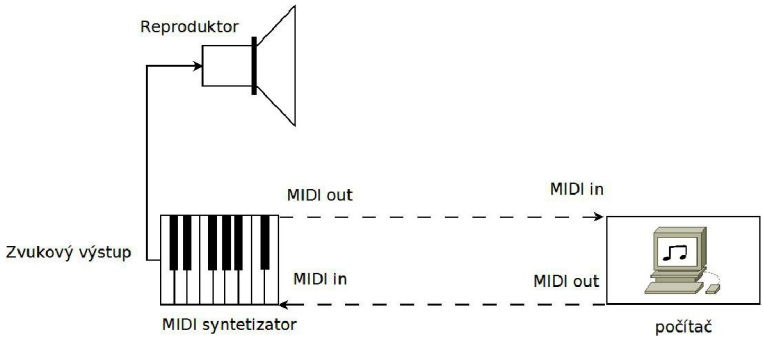
V tejto časti sú v krátkosti opísané dáta, ktoré boli použité v tejto práci. Popísaný je aj základ informacií o formátoch, v ktorých sú tieto dáta uložené.

6.1 Formát MIDI

MIDI (Musical Instrument Digital Interface) je protokol alebo aj druh počítačového jazyka, ktorý slúži na komunikáciu medzi počítačom a elektronickým hudobným nástrojom ako je napríklad keyboard alebo syntetizátor. Hlavný rozdiel od digitálneho zvuku je, že sa neprenáša žiaden audio signál, ale iba správa o vzniknutej udalosti. Správy, pomocou ktorých medzi sebou rozhrania komunikujú, udávajú: výšku tónu, trvanie tónu, silu tónu, vibrato a mnoho iných. Výhodou, ktorú využívam vo svojej práci je, že na modulovanie skladby stačí jednoducho pripočítať k tónu určitú hodnotu. Táto vlastnosť by sa pri digitálnom zvuku použiť nedala.

Ako to funguje:

Predstavme si, že ako nástroj je použitý klasický klávesový syntetizátor s MIDI rozhraním. Pri stlačení klávesy sa zopne elektrický spínač na klávese. Ten odošle správu na čip nástroja. Čip potom pošle signál do MIDI rozhrania, ktoré signál preformátuje a pošle ho do počítača. obr. 6.1 Komunikácia funguje podobne aj opačným smerom.[3]



Obrázek 6.1: Systém nahrávania a prehrávania pomocou MIDI správ [3]

6.2 ABC notácia

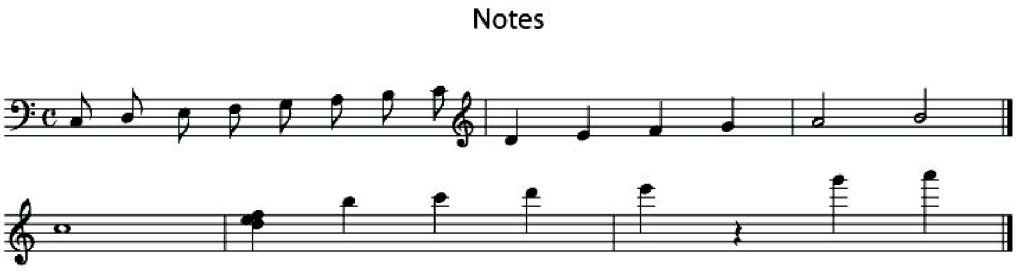
ABC notácia je jazyk, ktorý bol vyvinutý Chrisom Walshawom. Jeho hlavnou úlohou je prepísať skladbu zapísanú v notách na skladbu zapísanú v znakoch do formátu ASCII. Veľkou výhodou tejto notácie je, že je ľahko spracovateľná strojom, ale zároveň neprestáva byť čitateľná pre človeka. Na túto notáciu existuje veľké množstvo softvéru pre rôzne operačné systémy ako napríklad: abc2midi, abc2mtex, abc2ps ...

Pochopenie základov [10] vôbec nie je ťažké, nakoľko je veľmi intuitívne:

1. Na začiatku skladby potrebujem vyplniť základné údaje o tónine(K:), o takte(M:) o dĺžke jednej noty(L:), prípadne o názve pesničky(T:).
2. Noty sú zobrazené písmenom a majú vždy rovnaký názov ako v hudobnej teórii. Oktáva sa udáva veľkosťou písmena. Malé písmená sú vyššie, veľké písmená nižšie. Ďalšie úpravy spravíme pridaním apostrofu (zvýšenie) alebo čiarky (zníženie). Pridaním ďalších môžeme oktávu ešte znížiť alebo zvýšiť. Noty, ktoré sú zahrané súčasne, obalíme do hranatých zátvoriek.
3. Takty sa oddeľujú znakom | ,riadky |] a na konci skladby sa dáva znak ‖
4. Dĺžka každej noty sa pridáva hneď za notu a určuje sa ako násobok základnej noty, ktorá bola definovaná v hlavičke.
5. Pomlčka sa zobrazuje písmenom z a pre jej dĺžku platí to isté ako pre noty
6. Krížiky sa reprezentujú znakom horná strieška a béčka sú reprezentované znakom podtržítko, ktorý sa umiestni pred notu.

Skladba na obr. 6.2 zapísaná v abc notácii:

```
X:1
T:Notes
M:C
L:1/4
K:C
C,/2 D,/2 E,/2 F,/2 G,/2 A,/2 B,/2 C/2 | D E F G | A2 B2 | ]
c4 |[dfe]b c' d' | e' z~g' a' | ]
```



Obrázek 6.2: Notový výstup pre skladbu napísanú v abc notacii

6.3 Trénovacie dáta

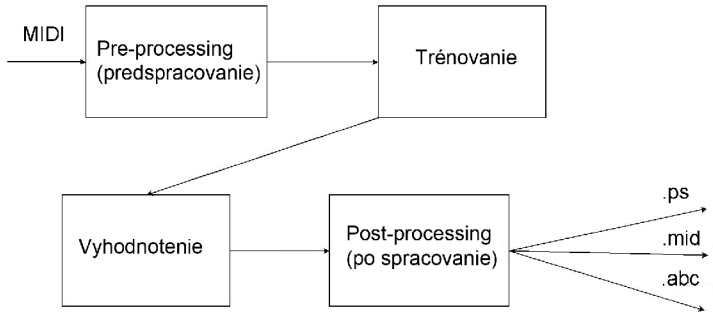
Podľa viacerých článkov je pre harmonizovanie vhodné použiť Bachové chorály [6]. Pôvodne išlo o jednohlasné skladby luteránskej cirkvi naspievané celou kongregáciou. Boli napísané neznámymi autormi. J. S. Bach do týchto monofonických skladieb pridal harmóniu a tým získali jeho meno. Dôvod, prečo sa používajú pravé tieto skladby je, že sú vo veľkej väčšine podobné a spĺňujú pravidlá klasickej harmonizácie. Všetky sú napísané v barokovom štýle a väčšina má malú veľkosť, čo je taktiež výhodné. [2]

Kompletnú databázu chorálov v rôznych formátoch nájdete na internete.[11]

Kapitola 7

Návrh systému

Svoj systém som sa rozhodol podľa vzoru klasifikátora rozdeliť na štyri základné fázy: Pre-processing (Predspracovanie), Trénovanie, Vyhodnotenie a Post-Processing (Pospracovanie) obr. 7.1. Časti na seba nadväzujú a výstup z jednej fázy slúži ako vstup do druhej. Každá z týchto častí je jednotlivo popísaná. V každej časti okrem jej popisu sú uvedené aj vylepšenia, ktoré pomohli dosiahnúť lepší výsledok.



Obrázek 7.1: Návrh rozdelenia systému

7.1 Fáza pre-processingu (predspracovania)

Neurónovej sieti nemôžeme dať na vstup skladbu v MIDI formáte. Je potrebné tento vstup upraviť. V tejto fáze sa snažíme vstupné dáta pripraviť na vstup do neurónovej siete do formátu, ktorý bude pre sieť vhodnejší.

7.1.1 Spracovanie sadou nástrojov Melisma

Skladba vo formáte MIDI nie je dobre čitateľná pre človeka. V MIDI (binárnom) formáte je vhodná na spracovanie počítačom. Na extrahovanie dôležitých informácií slúži sada nástrojov Melisma, ktoré sú popísané v kapitole o použitých nástrojoch. Melisma zistí potrebné informácie o jednotlivých tónoch a ich trvaniach. Tiež vykonáva analýzu tóniny, v akej je skladba napísaná, metra, akordov, ...

7.1.2 Reprezentácia tónov

Podľa zadania máme harmonizovať melódiu s použitím melodického basu. Reprezentovať tón bitovým vektorom s dvanástimi bitmi [1] nie je správnym riešením. Pri prechode medzi oktávami by dochádzalo k melodickým skokom. Prvým riešením, ktoré bolo použité, bolo zväčšiť rozsah bitového vektora na trojnásobok za predpokladu, že dokážeme nájsť optimálny rozsah troch tónin pre každý z jednotlivých hlasov. Tým pádom vznikli 36-bitové vektory. Napríklad komorné A v sopráne by bolo zobrazené nasledovne:
`[000000000100000000000000000000000000]`

Ďalším riešením, ktoré bolo neskôr aj realizované, bolo pridanie dvoch bitov. Prvých 12 bitov popisuje tón v rámci chromatickej stupnice. Ďalšie dva bity samostatne popisujú oktávu. Tak isto aj pre neurónovú sieť je to vylepšenie, nakoľko sa zmenší dimenzionalita a docielime tým, že nám bude stačiť menej dát pri rovnakom počte neurónov. Komorné A by bolo zapísané: `[00000000100010]`

Jednotlivé hlasy boli rozdelené na 3 časti:

1. Soprán
2. Bas
3. Alt a tenor

Soprán a bas obsahujú každý len jeden tón. Alt a tenor môže obsahovať žiaden alebo aj viacej tónov. Soprán je určený ako najvyšší tón zo všetkých, ktoré aktuálne znejú a naopak bas ako najnižší. Alt a tenor sú tóny, ktoré sa vyskytnú medzi basom a sopránom.

7.1.3 Rozdelenie na Dur alebo Mol

Na harmóniu ma veľký vplyv aj to, či je skladba v tónine durovej alebo molovej. Stačilo pridať pred vektor v sopráne jeden bit, ktorý určuje či je skladba dur alebo mol.

7.1.4 Modulácia do tóniny C-dur/A-mol

Ide o jedno z vylepšení, kde sa všetky trénovacie skladby prevedú do spoločnej tóniny C-dur/A-mol. Vychádza sa z predpokladu, že skladba je napísaná v jednej tónine a nie je problém pričítaním určitého koeficientu a tým skladbu posúvať medzi tóninami. Touto normalizáciou podstatne vylepšíme celkový systém a výsledkom je, že nie je potreba mať až taký veľký počet trénovacích dát.

7.1.5 Vylepšenie 1: Odstránenie nesprávne určených tónin

Určovanie tónin je veľmi komplikovaná záležitosť. Ani nástroj Melisma nie je v tomto smere neomylný. Teoreticky sa dalo tóninu určiť z informácií v MIDI hlavičke. Tá však nie je vždy dobre a dostatočne vyplnená. Tým pádom by sa nedalo harmonizovať melódiu, ktorá nemá v hlavičke zadefinovanú tóninu.

Za predpokladu, že trénovacie dáta majú v hlavičke informáciu o tónine (ako napríklad dáta J.S Bacha), dalo sa odstrániť dáta, v ktorých sa Melisma pomýlila alebo počas skladby nastala modulácia do inej tóniny.

[1] V jednej oktáve je 12 poltónov a každý bit reprezentuje jeden poltón. Bitový vektor pre tón D by vyzeral napríklad takto: 001000000000

7.1.6 Vylepšenie 2: Vytvorenie histórie

V tomto vylepšení som sa inšpiroval prácami [2]. Pre lepšie výsledky neurónovej siete je dobre vedieť aké tóny boli zahrané pred aktuálnym tónom.

Jedno z riešení bolo, pripojenie za vektor dva predchádzajúce tóny. Nevýhodu tohto prístupu je, že sa zvýši veľkosť vektorov. Na zníženie veľkosti vektoru sa pridáva na koniec len číslo, o koľko je predchádzajúci tón od aktuálneho posunutý.

7.2 Fáza trénovania

V tejto fáze sa vytvára neurónová sieť a začína sa trénovať. Trénuje sa pomocou dát (vektorov) vytvorených v predchádzajúcej fáze. Vytvorenie, ako aj používanie neurónovej siete pre nás funguje ako čierna skrinka *blackbox*, nakoľko nič neimplementujeme, iba využívame už použité nástroje alebo príkazy. Maximálne vytvárame skript a používame preddefinované príkazy.

V tejto fáze som sa pokúšal použiť dva rôzne nástroje na spracovanie neurónových sietí. Jedným z nich bol NICO a druhým Neural Network Toolbox z Matlabu. Dôvody prečo je lepšie uprednostniť Matlab boli spomenuté už v kapitole 5 o použitých nástrojoch.

7.2.1 Rozdelenie na 2 siete

Podľa zadania bolo potrebné harmonizovať sieť pomocou melodického basu. Z toho vyplýva, že najlepšie bude, ak sa systém rozdelí na dve oddelené siete. Jedna bude z hlasu v sopráne vytvárať hlas v base. Ďalšia so spoločného vektoru sopránu a basu vytvorí ostatné hlasy. Pre jednoduchosť si prvú sieť pomenujme Basová a druhú sieť Alto-tenorová. Sieť Alto-tenorová ma na vstupe spoločný basový aj sopránový vektor. Na výstupe alto-tenorový vektor.

7.2.2 Rozdelenie dát pri trénovaní

Pri trénovaní Matlab rozdelí trénovacie dáta na 3 časti:

1. Trénovacie dáta – tieto dáta sú systému prístupné počas trénovania a podľa nich sa sieť trénuje, čiže sa nastavujú jednotlivé váhy v sieti.
2. Validačné dáta – sú použité na testovanie generalizácie. V prípade, že sa systém prestane zlepšovať, trénovanie sa zastaví.
3. Testovacie dáta – počas trénovania sietí nie sú k dispozícii. Dostaneme ich až po skončení a slúžia na vyhodnotenie siete. Určujú nám, ako dobre je sieť natrénovaná.

7.3 Fáza vyhodnotenia

Rovnako ako v predchádzajúcej fáze aj tu bol použitý Matlab. Vo vyhodnotení sa už netrénovanej sieti predložia dáta, pre ktoré chceme dosiahnuť nejaký výsledok. Výstupné dáta sú ohodnotené určitým číslom. Toto číslo po prevedení na pravdepodobnosti určuje, aká je pravdepodobnosť danej triedy.

7.4 Fáza post-processingu (po spracovania)

V tejto fáze sa z výstupov predchádzajúcej neurónovej siete po prevedení do formátu pravdepodobnosti vytvoria vektory, ktoré sú pre nás lepšie spracovateľné. Vytvoria sa požadované zharmonizované súbory skladieb.

7.4.1 Zaokrúhľovanie vektorov

Z vektorov, ktoré sú prevedené do formátu pravdepodobnosti potrebujeme vybrať dôležité informácie. Pre každý výstup siete osobitne.

Pre basovú sieť je nutné vybrať aspoň jeden tón a určíme ho ako triedu, ktorá má najväčšiu pravdepodobnosť. Oktávu určíme podľa toho či presahuje určitú zadanú hranicu.

Pre altovo-tenorovú sieť je to komplikovanejšie. Nakoľko môže obsahovať aj viacej ako jeden tón, je potrebné zvoliť si nejakú hranicu pravdepodobnosti. Tá určuje, či je tón dostatočne pravdepodobný, aby v danom mieste zaznel.

7.4.2 Odkódovanie vektorov

Samotný vektor je potrebné odkódovať na jednu hodnotu vo forme výšky tónu. To sa deje presne opačným spôsobom ako bol vektor kódovaný.

7.4.3 Dĺžka tónu

Okrem zachovania tónu sa v prvej fáze uložila zo sopránovej melódie aj informácia o dĺžke jednotlivých tónov. Pre samotnú harmóniu nie je až tak veľmi dôležitá, ale pre lepší hudobný zážitok je určite veľmi podstatná. Kompromis bol urobený v tom, že sa určila rovnaká dĺžka tónu pre všetky tóny v harmonických hlasoch pod daným sopránom.

7.4.4 Vytvorenie abc notácie

Na vytvorenie abc notácie tento skript používa informácie z pôvodného súboru sopránu, novo vytvorených súborov altu a tenoru a údajov o dĺžke tónu. Všetky tieto informácie spojí a prekóduje na abc notáciu. Pokiaľ vo vstupnej melódii bola nejaká medzera medzi tónmi, tak toto miesto nahradí pomlčkou.

Kapitola 8

Implementácia

V tejto časti sú popísané všeobecné informácie o implementácii a taktiež vysvetlený význam jednotlivých skriptov a ich samotná realizácia.

8.1 Programovací jazyk a vývojové prostredie

Na začiatku som chcel na programovanie použiť už pre mňa známy programovací jazyk C++. Na základe odporučení od vedúceho som tento svoj úmysel nakoniec rozhodol zmeniť a prešiel na pre mňa úplné nový jazyk Perl. Toto bola pre mňa zároveň aj možnosť rozšíriť si svoj obzor a naučiť sa nový skriptovací jazyk.

Perl je interpretačný jazyk, ktorý bol vytvorený za cieľom spracovania súborov. Autorom bol Larry Wall, ktorý si chcel spraviť jednoduchšiu alternatívu k jazyku C. Pôvodne bol jazyk navrhnutý pre UNIX ale v dnešnej dobe sa rozšíril aj na ostatné platformy. Perl ma výhodu, že je objektovo orientovaný. Hlavnou myšlienkou, s ktorou bol jazyk vytvorený je, že existuje veľké množstvo riešení na jednu vec. Perl dáva programátorovi väčšiu voľnosť napríklad tým, že nie je potrebné definovať pre premennú typ .

Nevýhodou tohto jazyka je, že programátor môže veľmi často napísať neprehľadný program, ktorý je ťažko čitateľný. Pokiaľ sa ale budú dodržiavať pravidlá programovania a poctivo komentovať zdrojový kód, nebude program menej prehľadný ako v iných jazykoch. Určite nie je vhodný pre začínajúcich programátorov. [17] Na niektoré jednoduchšie časti bol použitý jazyk Bash.

Svoj program som vyvíjal pod operačným systémom Linux s distribúciou Ubuntu 10.04. Na písanie som použil jednoduchý textový editor Kate, ktorý ma výhodu v zobrazovaní syntaxe.

8.2 Jednotlivé skripty

8.2.1 skript.sh

Skript slúži na celkové spustenie systému, kde stačí na vstupe zadať trénovacie dáta do zložky *midi* a dáta, ktoré chceme vyhodnotiť do zložky *midi_test*. Na výstupe v zložke *midi_out* budú uložené výstupné zharmonizované nahrávky. Jednotlivé skripty sú vytvorené vždy na spracovanie jedného súboru. `skript.sh` spúšťa skripty vždy pre všetky súbory v danej zložke.

8.2.2 parser.pl

Tento skript slúži na predspracovanie vstupných súborov, aby boli vhodné na spracovanie neurónovou sieťou. Vo svojom tele spúšťa už popísané nástroje Melisma. S využitím regulárnych výrazov sa vytiahnú potrebné informácie o dĺžke trvania, výške tónu. Systém sa snaží každéj dobe(*beatu*) priradiť tón, ktorý v danom momente znie. Často sa stane, že naraz znie aj viacej tónov. Pokiaľ znie jeden tón, určí sa buď ako soprán alebo ako bas. Záleží na jeho výške. V prípade dvoch tónov sa vyšší z nich určí ako soprán a nižší ako bas. Ostatné hlasy sú nulové. Pri viacerých tónoch obdobne sa určuje bas a soprán a ostatné tóny sú uložené do alto-tenorového vektoru. Každý z troch vektorov sa ukladá do osobitného súboru v osobitnej zložke. Tón sa normalizuje, aby sa zmestil do rozsahu troch oktáv, ktorý je pre každý hlas osobitne určený. Do špeciálneho súboru *others* sa ukladá informácia o akorde a o tónine, ktoré sú určené na budúce časové spracovanie. Na trénovanie druhej siete je potrebné, aby na vstupe bol bas a soprán spoločne. Preto sa aj pre tento vektor vytvorí nový súbor. Informácia o tónine zistená z nástroju Key sa porovnáva s tóninou získanou z hlavičky MIDI súboru. Toto vylepšenie predpokladá , že MIDI hlavička je poriadne vyplnená. Naše trénovacie dáta ju vyplnenú poriadne majú. Hlavným cieľom tohto vylepšenia je odstrániť dáta, ktoré by mohli veľmi negatívne ovplyvniť trénovanie.

8.2.3 filter_data.sh

Skript slúžiaci na odstránenie vektorov, ktoré sa opakovali a boli rovnaké pre všetky štyri skupiny: bas, soprán, bas-soprán a alto-tenor. Dáta sa taktiež rozdelia na dve skupiny podľa toho, na trénovanie ktorej neurónovej siete sa budú používať.

8.2.4 parse_Sopr.pl

Skript takmer totožný so skriptom `parse.pl`. Rozdiel je len v tom, že tento skript sa používa na vyhodnocovacie dáta. Pri tom nám stačí spracovať iba jeden hlas, nakoľko predpokladáme, že v zdrojovom súbore sa nachádza melódia v sopráne. Výstup bude použitý na natrénovanú neurónovú sieť.

8.2.5 copysb.pl, copysopr.pl

Tieto skripty sa používajú na vytvorenie histórie pre vektory, ktoré slúžia ako vstup do jednotlivých sietí. Vychádza sa z predpokladu, že na lepšie natrénovanie siete budú mať vplyv aj predchádzajúce noty. Namiesto toho, aby sa k vektoru momentálnej noty pridali ďalšie vektory, pridávajú sa iba vzdialenosti od predchádzajúcich nôt. Rapídne sa tým zníži ich dimenzionalita. Docielime tým, že nám bude stačiť menej dát pri rovnakom počte neurónov. Ako optimálny počet predchádzajúcich nôt boli určené 2 noty.

8.2.6 zaokruhli_bass.pl

Skript slúžiaci na spracovanie výstupu z basovej neurónovej siete. Z prvých dvanástich bitov určí najväčší a ten sa vyberie ako najpravdepodobnejší tón. Oktáva sa určí z posledných dvoch bitov, pokiaľ je pravdepodobnosť bitu väčšia ako 50% .

8.2.7 zaokruhli_at.pl

Skript fungujúci podobne ako zaokruhli_bass.pl. Rozdiel je v určovaní tónov. To či bude daný tón znieť určuje, či je pravdepodobnosť daného bitu väčšia ako 50%. Oktáva sa určuje úplne rovnako ako v predchádzajúcom skripte.

8.2.8 vectors2abc.pl

Táto časť slúži na vytvorenie, dekódovanie vektoru a vytvorenie súboru v abc notácii. Na vytvorenie hlavičky abc notácie slúži funkcia `createHead`. Tá určuje tóninu, názov skladby a metrum. V tomto skripte sa spájajú 4 súbory. Súbor, ktorý určuje dĺžky tónov, obsahuje vždy začiatočný čas a koncový čas. Nakoľko sa tu osobitne nezachováva informácia o pomlčkách, je potrebné si ju určiť. Pokiaľ je začiatočný čas jednej noty rozdielny ako konečný čas predchádzajúceho, vložíme do výsledného vektoru pomlčku.

Pri spájaní súborov sa vychádza z predpokladu, že súbory majú rovnaký počet vektorov. Na základe toho môžeme naraz prechádzať všetky jednotlivé súbory. Každý tón znejúci v rovnakom čase sa dekóduje pomocou funkcie `toABC` na konkrétnu výšku. Súbor je v abc notácií pripravený na spracovanie ostatnými nástrojmi.

8.2.9 neural1.m, neural2.m

Ide o skripty pre nástroj Matlab, ktoré definujú trénovanie neurónových sietí. Trénovacie vstupné vektory sa načítavajú z adresárov *data_1* a *data_2*. Testovacie vektory sa načítavajú zo zložky *data_3*.

Vytvárame *feed-forward* siete pomocou príkazu `newff`, kde nastavujeme: vstupné dáta, výstupné dáta, počet neurónov v skrytej vrstve, kriteriálnu a aktivačnú funkciu.

V prvom skripte sa sieť trénuje pomocou kriteriálnej funkcie *crossentropy*, ktorá sa využíva spoločne s aktivačnou funkciou *softmax*. Výhodou tohto prístupu je, že výsledný vektor bude vo formáte pravdepodobností pre jednotlivé triedy a celkový čas trénovania sa výrazne zníži. Nevýhodou je, že na vstupe a výstupe je potrebné mať vždy iba jednu triedu.

Vyhodnocujeme pomocou príkazu `sim`, ktorý do natrénovanej siete pošle vektory, pre ktoré chceme vedieť výsledok.

Výsledné vektory sú uložené do zložiek *eval_data_1* a *eval_data_2*.

Kapitola 9

Experimenty

Na zistenie toho, ako dobre tento systém harmonizuje, je potrebné si systém nejako otestovať a robiť na ňom rôzne experimenty.

9.1 Experimentovanie s výsledkom

9.1.1 Empirický test

V tomto teste sa nepoužívajú žiadne metriky. Ide čisto o vypočutie zharmonizovanej skladby vo formáte MIDI a na základe empirických skúseností s hudbou sa snažíme určiť, či je skladba zharmonizovaná správne. Pri tomto experimente sa predpokladá aspoň nejaký hudobný sluch. Je potrebné vedieť rozlíšiť úplnú disharmóniu od harmónie.

9.1.2 Test podľa výstupných nôt

Tento experiment vychádza z výstupnej zharmonizovanej skladby. Preštudovaním tohto výstupu sa zisťuje, či vo výsledku nie je príliš veľa tónov, ktoré do danej tóniny nepatria. Predpokladá sa, že výstupná skladba bude v jednej tónine. Napríklad pre C-dur môžeme ako kritérium určiť, či sa vo výslednej ABC notácii nachádzajú tóny bez krížikov a béčiek.

Toto kritérium bohužiaľ nie je až tak spoľahlivé, nakoľko už v úvode som spomínal, že nedodržanie určitých pravidiel nemusí vždy znamenať zlý výsledok, a naopak môže viesť k jeho zlepšeniu.

9.2 Experimentovanie so sieťou

Veľký vplyv na výsledok systému má kritérium, ako dobre je sieť natrénovaná. Toto testovanie prebieha v skriptoch pre Neural Network toolbox. Na experimentovaní so sieťou som pre jednoduchosť pracoval už iba s neurónovou sieťou, ktorá vytvárala k sopránu bas.

Pri trénovaní siete bola v prvých fázach tvorby systému zvolená kriteriálna funkcia *mse* (mean-squared error). Kriteriálna funkcia určuje, nakoľko sieť správne určila výstupný vektor. Pri trénovaní sa trénovací algoritmus snaží túto chybu minimalizovať po každej epoche.

Je potrebné tu spomenúť, že *mse* nie je najvhodnejšiou kriteriálnou funkciou pre klasifikačné úlohy.

Uvediem príklad: Máme sieť a na výstupe iba tri neuróny. Cieľom je dostať vektor `[1 , 0 , 0]`. Sieť vygeneruje na výstupe vektor s hodnotami `[0.6 , 0.5 , 0.5]`. Zna-

mená to, že pokiaľ berieme vo výstupe maximum, tak nám sieť výsledok určí správne. *Mse* berie rozdiel medzi vstupným a výstupným vektorom `[0.4, -0.5 , -0.5]`. Po umocnení na druhú `[0.16 , 0.25 , 0.25]`. Celková priemerná chyba z toho bude 0.22, čo je dosť veľká chyba na to, že sieť určila výsledok správne.

V sieti na vytvorenie basu k sopránu bola ako kriteriálna funkcia zvolená *crossentropy* [9], ktorá nie je v Neural Network toolboxe implementovaná, ale podarilo sa mi nájsť k nej zdrojové súbory. Výhodou je jej spojenie s metódou *softmax*, kde sa ich spoločná derivácia vypočíta ako jednoduchý rozdiel, čo nám zásadne ovplyvní rýchlosť trénovania. Nevýhodou je, že na výstupe musí byť len vektor, ktorý kóduje len jednu triedu z N.

9.2.1 Test určenia správne klasifikovaných nôt

V tomto teste sa počíta pomer správne klasifikovaných vektorov k celkovému počtu vektoru. Výsledná chyba v mojích experimentoch vychádza okolo 10%, ale toto číslo nemusí znamenať nesprávne zharmonizovanú melódiu. Určenie tónu o oktávu vyššie neznamená zásadnú chybu harmonizácie. Tiež je dôležité si uvedomiť, že z hľadiska interpretácie chyby natrénovanej siete, môže k jednej melódii existovať viacero správnych harmonizácií. To, že testovanie určí výstup siete ako nesprávny, neznamená, že je nesprávna harmónia, ale iba to, že sieť neurčila presne takú harmóniu, aká bola v testovacích dátach.

9.2.2 Experiment s chybovou maticou

Pomocou programu Matlab si necháme vypísať chybovú maticu, v ktorej bude zobrazené koľkokrát bol určitý tón v skutočnosti na výstupe z trénovacích dát vzhľadom na tón, ktorý určila sieť.

Parametre siete: počet neurónov v skrytej vrstve: 20, počet iterácií: 25, kriteriálna funkcia: crossentropy, aktivačná funkcia: softmax.

```
note confusion matrix:
      c   c#    d   d#    e    f   f#    g   g#    a   a#    h
c  8634  324 3945 1040 3720 3369  557 4630  873 4088  738 2200
c#    0    0    0    0    0    0    0    0    0    0    0    0
d   144   13   83   68   90  278    6   28   58    9   68   48
d#    0    0    0    0    0    0    0    0    0    0    0    0
e   155   42  180  134   95  110    3  289   33   32  118    0
f     0    0    0    0    0    0    0    0    0    0    0    0
f#    0    0    0    0    0    0    0    0    0    0    0    0
g  1594   18 1412  341  852 1484  369 4641  395  635  408 1196
g#    0    0    0    0    0    0    0    0    0    0    0    0
a     0    0    0    0    0    0    0    0    0    0    0    0
a#    0    0    0    0    0    0    0    0    0    0    0    0
h     0    0    0    0    0    0    0    0    0    0    0    0
```

V riadkoch je výstupná nota zo siete(bas) a v stĺpcoch správna basová nota definovaná v trénovacích dátach.

Príklad: Keď sieť určila, že výstupný tón má byť G, tak v 1594 prípadoch bola na výstupe nota C.

9.2.3 Experiment s maticou mapujúcou vstupy siete na výstupy

Matica získaná z Matlabu hovorí o tom, že ak máme zadanú notu v sopráne, tak v koľkých prípadoch bude k nej určená daná nota v base.

output translation matrix:

	c	c#	d	d#	e	f	f#	g	g#	a	a#	h
c	8168	0	0	0	0	0	0	0	0	0	0	0
c#	0	0	0	0	0	0	0	327	0	0	0	0
d	0	0	0	0	0	0	0	7729	0	0	0	0
d#	0	0	0	0	0	0	0	1869	0	0	0	0
e	6312	0	0	0	0	0	0	0	0	0	0	0
f	6104	0	0	0	0	0	0	0	0	0	0	0
f#	652	0	0	0	0	0	0	0	0	0	0	0
g	8489	0	0	0	0	0	0	0	0	0	0	0
g#	0	0	913	0	0	0	0	0	0	0	0	0
a	4719	0	0	0	0	0	0	0	0	0	0	0
a#	0	0	0	0	1201	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	3690	0	0	0	0

V riadkoch sú vstupné vektory z trénovacích dát (soprán) a v stĺpcoch výstupné vektory zo siete(bas).

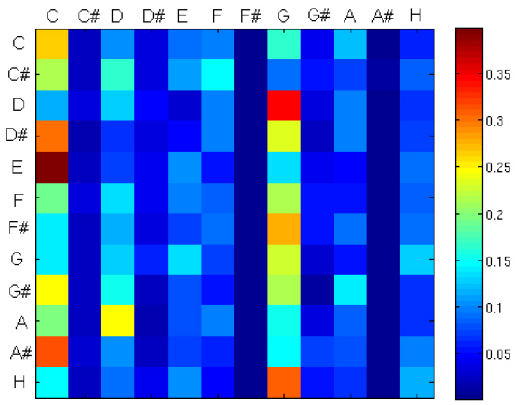
Príklad: Ak bola zadaná nota E v sopráne, tak jej bola pridelená v base nota C v 6312 prípadoch.

Maticu output translation matrix môžeme previezť na pseudo-pravdepodobnosti:

output probability translation matrix:

	c	c#	d	d#	e	f	f#	g	g#	a	a#	h
c	1912.7	103.0	857.7	177.1	1228.0	803.0	28.2	1141.3	0.4	714.6	343.1	858.8
c#	39.1	5.5	50.1	5.6	24.0	40.7	1.0	101.6	0.0	29.5	9.0	20.8
d	824.2	135.2	837.6	157.0	271.5	1132.6	16.1	3026.2	0.3	766.1	217.3	345.2
d#	411.7	20.1	135.9	40.0	117.3	314.4	3.7	427.1	0.0	185.6	66.6	146.5
e	2035.4	102.8	505.4	140.7	588.8	710.6	35.1	825.0	0.5	670.9	289.2	407.6
f	1273.0	130.1	799.9	137.5	795.3	559.3	53.9	991.0	0.5	810.5	191.9	361.0
f#	147.5	10.5	57.4	15.2	41.8	142.0	1.0	94.3	0.0	55.8	36.6	49.9
g	1806.8	105.0	892.7	129.6	982.2	1172.5	26.6	1370.7	0.1	921.4	299.9	781.4
g#	133.6	15.1	151.7	21.8	143.1	57.1	5.7	132.9	0.2	109.7	30.3	111.8
a	994.1	87.3	546.8	102.0	470.4	536.5	17.7	705.0	0.3	661.2	195.8	401.9
a#	177.5	20.3	180.2	30.3	201.8	88.5	7.6	197.7	0.1	129.3	39.0	128.6
h	624.5	67.8	406.0	83.1	477.2	414.7	11.7	911.4	0.1	291.5	133.7	268.4

Každý riadok po prevedení na pravdepodobnosti [1] je rozloženie pravdepodobnosti pre danú notu v sopráne.



Obrázek 9.1: Grafické znázornenie pravdepodobnostnej translačnej matice

Tieto pravdepodobnosti sa dajú pekne graficky zobraziť na obr. 9.1.

Príklad: Ak je na vstupe systému nota D, tak na výstupe v base jej bude najpravdepodobnejšie priradená nota G. Čím má nota viac červenej zložky, tým je pravdepodobnejšie, že bude aj vo výsledku.

9.2.4 Experiment s tvrdým rozhodovaním

Tento experiment určuje počet nôt, ktoré by boli určené na výstupe, ak by sa vyberala vždy najpravdepodobnejšia trieda. Z tohto môžeme určiť, ktoré noty búdu najčastejšie vybrané aj v mojom systéme.

9.2.5 Experiment s počtom neurónov v skrytej vrstve

Pri tomto experimente meníme počet neurónov a sledujeme ako sa nám mení kriteriálna funkcia *crossentropy*. Pri tomto experimente na obr. 9.2 sa snažíme nájsť optimálny počet neurónov v skrytej vrstve.

Parametre siete pri testovaní: počet iterácií: 25, kriteriálna funkcia: crossentropy, aktivačná funkcia: softmax.

Ako najvýhodnejšie som sa rozhodol zvoliť počet neurónov v skrytej vrstve na 20. V tom sa potvrdila aj moja hypotéza, že ide o veľmi jednoduchú sieť a na jej natrénovanie stačí aj málo neurónov v skrytej vrstve.

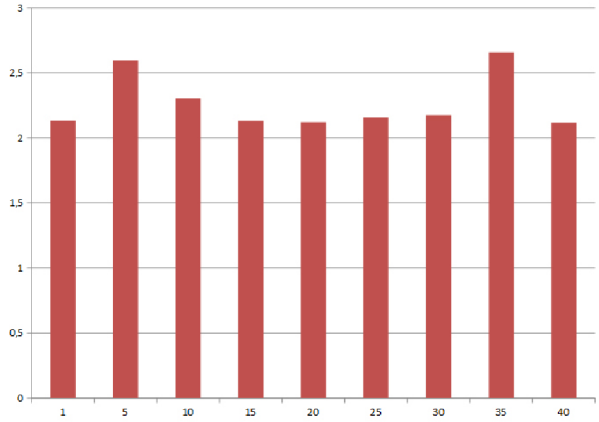
9.2.6 Experiment s počtom iterácií

V tomto experimente na obr. 9.3 som sa snažil nájsť optimálny počet iterácií pre 20 neurónov v skrytej vrstve.

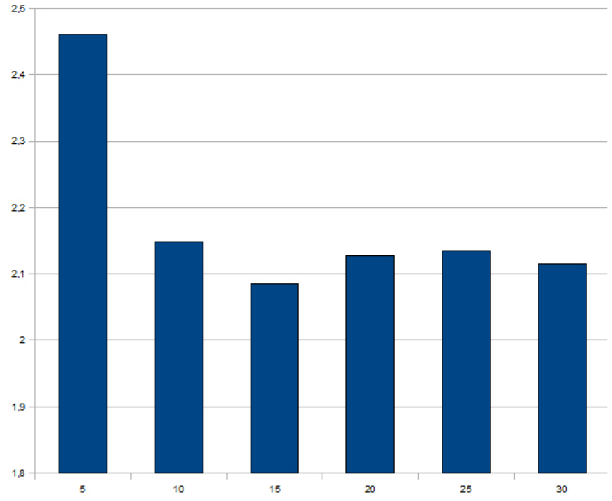
Parametre siete pri testovaní: počet neurónov v skrytej vrstve: 20, kriteriálna funkcia: crossentropy, aktivačná funkcia: softmax.

Z grafu sme zistili, že najlepšie bude použiť počet iterácií v rozmedzí 15–20. Bol to kompromis medzi časom potrebným na natrénovanie a chybou siete.

[1] každé číslo v riadku podelíme sumou celého riadku



Obrázek 9.2: Graf výsledkov experimentov s meniacim sa počtom neurónov v skrytej vrstve. Os x určuje počet neurónov v skrytej vrstve a os y je crossentropy.



Obrázek 9.3: Graf výsledkov experimentov s meniacim sa počtom iterácií. Os x určuje počet neurónov iteracií a os y je crossentropy.

Kapitola 10

Záver

Vo svojej práci som vytvoril systém, ktorý na základe melódie dokáže vytvoriť harmóniu. Uvedomujem si, že výsledky tohto systému nikdy nebudú absolútne dokonalé. Miestami môžu vzniknúť situácie, pri ktorých nastane disharmónia, ale vo väčšine tónov je harmónia vytvorená správne a výsledok je poslucháčom dobre počúvateľný.

Veľký dôraz som kládol na jednoduchosť systému, aby ho mohol používať aj bežný užívateľ. Stačí zadať jeden príkaz a systém vygeneruje melódiu s požadovanou harmóniou.

10.1 Budúci vývoj práce

Počas svojej práce som si uvedomil, že tvorba hudby počítačom je veľmi rozsiahla problematika. Existuje na jej riešenie veľa rôznych prístupov. Preto som si už počas vývoja ukladal dáta, ktoré by v budúcnosti mohli byť použité na zlepšenie harmonizácie.

V budúcnosti by sa dalo napríklad podľa vzoru Alexa Chilversa [2] zväčšiť počet parametrov, od ktorých závisí tvorba harmónie.

Ďalším zdokonalením by bolo tónom v sopráne nepriradzovať len rovnaké dĺžky ostatných hlasov, ale ich dĺžku nejakým spôsobom meniť.

Tejto téme by som sa určite rád venoval aj vo svojej diplomovej práci. Je množstvo nápadov a vylepšení, ktoré by sa dali do programu implementovať. Potrebujem však ešte získať a spracovať ďalšie informácie.

10.2 Celkový prínos

Prínosom mojej práce je vytvorenie systému, ktorý dokáže jednoducho vytvoriť harmóniu ku zadanej melódii aj pre ľudí, ktorý si harmóniu nevedia vytvoriť sami. Nástroj bude rovnako nápomocný aj pre mňa na tvorbu basovej melódie pri hre na basgitaru.

Výrazne som sa zlepšil v používaní skriptovacích jazykov ako sú Perl alebo Bash.

Tiež to je pre mňa užitočná skúsenosť do budúcnosti, napríklad pri písaní diplomovej práce.

Literatura

[1] Biyikoglu, K. M.: A Markov model for chorale harmonization. [online], [cit. 2010-05-14].
URL http://www.epos.uni-osnabrueck.de/music/books/k/klww003/pdfs/216_Biyikoglu_Proc.pdf

[2] Chilvers, A.: Chorale Harmonisation in the Style of J.S. Bach - A Machine Learning Approach. [online], [cit. 2010-05-14].
URL http://web.science.mq.edu.au/~achilver/project/THESIS.pdf

[3] Garrigus, S. R.: *Cakewalk Sonar*. Brno, Computer Press, 2007, ISBN 978-80-251-1491-9.

[4] H. Hild, W. M., J. Feulner: HARMONET: A Neural Net for Harmonizing Chorales in the Style of J. S. Bach. [online], [cit. 2010-05-14].
URL http://books.nips.cc/papers/txt/nips04/0267.txt

[5] Machová, K.: Strojové učenie Princípy a algoritmy. [online], [cit. 2010-05-14].
URL ftp://math.chtf.stuba.sk/pub/vlado/Book_Machova_SU/Machova_SU.pdf

[6] Moray, A.: Harmonising Chorales in the Style of Johann Sebastian Bach. [online], [cit. 2010-05-14].
URL http://www.tardis.ed.ac.uk/~moray/papers/allan2002.pdf

[7] Tichý, V.: *Harmonicky myslet a slyšet*. Praha, HAMU, 1996, ISBN 80-85883-10-4.

[8] WWW stránky: abcMIDI. [online], [cit. 2010-05-14].
URL http://abc.sourceforge.net/abcMIDI/

[9] WWW stránky: Cross-entropy method. [online], [cit. 2010-05-14].
URL http://en.wikipedia.org/wiki/Cross-entropy_method

[10] WWW stránky: How to understand abc (the basics). [online], [cit. 2010-05-14].
URL http://abcnotation.com/blog/2010/01/31/how-to-understand-abc-the-basics/

[11] WWW stránky: J.S. Bach midi set. [online], [cit. 2010-05-14].
URL http://www.jsbchorales.net/down/sets/jsb403.zip

[12] WWW stránky: The Melisma Music Analyzer. [online], [cit. 2010-05-14].
URL http://www.link.cs.cmu.edu/music-analysis/

[13] WWW stránky: Multilayer Neural Network. [online], [cit. 2010-05-14].
URL http://www.teco.edu/~albrecht/neuro/html/node18.html

[14] WWW stránky: Neural Network Toolbox 6.0.4. [online], [cit. 2010-05-14].
URL http://www.mathworks.com/products/neuralnet/

[15] WWW stránky: Neural Network Toolbox, Documentation. [online], [cit. 2010-05-14].
URL http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/nnet_product_page.html

[16] WWW stránky: Note names, MIDI numbers and frequencies. [online], [cit. 2010-05-14].
URL http://www.phys.unsw.edu.au/jw/notes.html

[17] WWW stránky: Perl. [online], [cit. 2010-05-14].
URL http://www.linuxsoft.cz/article.php?id_article=675

[18] WWW stránky: Recognizing Patterns. [online], [cit. 2010-05-14].
URL http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/getting6.html#26592

[19] WWW stránky: TPCNote. [online], [cit. 2010-05-14].
URL http://www.link.cs.cmu.edu/music-analysis/key.html

[20] WWW stránky: Viterbi Algorithm. [online], [cit. 2010-05-14].
URL http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/viterbi_algorithm/s1_pg1.html

[21] Zenkl, L.: *ABC hudební nauky*. Praha, Editio Bärenreiter, 2003, ISBN 80-86385-21-3.

[22] Čerňanský, M.: Logické neuróny a neurónové siete (podľa McCullocha a Pittsa). [online], [cit. 2010-05-14].
URL http://www2.fiit.stuba.sk/~cernans/nn/nn_texts/neuronove_siete_priesvitky_01.pdf

Dodatek A

Obsah CD

1. **database** – obsahuje trénovacie a harmonizované dáta
2. **tools** – obsahuje nástroje potrebné na správny chod systému
3. **scripts** – obsahuje jednotlivé skripty potrebné na harmonizáciu
4. **textBac** – obsahuje technickú správu vo formátoch .tex a .pdf

Dodatek B

Manuál

Skripty spolu s trénovacími dátami (v formáte MIDI) sú uložené na priloženom CD nosiči. Po ich rozbalení dostneme nasledujúcu štruktúru adresárov:

```
harmonizer
       |
       +-- database
       |
       +-- tools
       |
       +-- scripts
       |
       +-- textBac
```

Priložené sú aj voľne dostupné nástroje (uložené v zložke *tools*) používané skriptami okrem programu Matlab, ktorý nie je voľne šíriteľný. Pre správne fungovanie systému je potrebné ho mať nainštalovaný.

B.1 Kompilácia

```
./compile.sh
```

Priložené nástroje je potrebné pred ich použitím skompilovať. Na ich kompiláciu bol vytvorený skript `compile.sh`, ktorý uživateľ spustí z adresára *scripts*.

B.2 Celkové spustenie

```
./skript.sh
```

Pre celkové spustenie v druhom kroku je možné použiť už predpripravený skript `skript.sh`, ktorý spúšťa postupne všetky vytvorené skripty. Uložený je v adresári *scripts*. Spúšťanie týmto spôsobom je skutočne odporúčané, nakoľko ide o jednoduchý spôsob. Keď uživateľ chce aby neprebehol celý proces harmonizácie ale len niektorá časť, nevyužité skripty označí ako komentár.

Využíva už vytvorenú zložku *database*, v ktorej budú uložené trénovacie dáta (v zložke *midi*) a dáta na vyhodnotenie (v zložke *midi_test*). Tu by som chcel upozorniť, že vstupné

dáta vo formáte MIDI na harmonizáciu musia byť uložené len v jednom hlase. Zharmonizované dáta vo formáte MIDI sú k dispozícií vo vytvorenej zložke *midi_out*, ktorá bude umiestnená v adresári *database*.

B.3 Postupné spustenie

Jednotlivé skripty je potrebné spúšťať pre každý vstupný súbor samostatne. Tu bude ukázaný spôsob vždy pre jeden súbor.

B.3.1 Predspracovanie

Spracovanie testovacích MIDI nahráviek skriptom:

```
./parser ../database/midi/song.mid
```

výsledok bude uložený do zložky *data_0*, kde bude uložený a rozdelený na jednotlivé hlasy.

Odstránenie opakujúcich sa vektorov a rozdelenie siete:

```
./filter_data.sh
```

rozdelí jednotlivé trénovacie dáta do adresárov *data_1* pre trénovanie prvej siete a *data_2* pre trénovanie druhej siete.

Vytvorenie zoznamu dát na trénovanie:

```
ls ../database/data_0/sopr/ | sed 's/\..*//' > ../database/train.list
```

Spracovanie harmonizovaných MIDI nahráviek:

```
./parse_Sopr.pl ../database/midi_test/song02.mid
```

výsledok bude uložený v zložke *data_3*.

Vytvorenie zoznamu harmonizovaných dát:

```
ls ../database/midi_test/ | sed 's/\..*//' > ../database/test.list
```

Vytvorenie histórie:

Užívateľ si vytvorí následujúce adresáre:

```
mkdir ../database/data_1/sopr_hist
mkdir ../database/data_2/sb_hist
mkdir ../database/data_3/sopr_hist
```

Skript vytvorí históriu sopránu:

```
./copysopr.pl ../database/data_1/sopr/song01.sopr
```

Skript vytvorí históriu pre soprán a bas:

```
./copysb.pl ../database/data_2/sb/song01.sb
```

Skript vytvorí históriu pre výsledné dáta:

```
./copysopr.pl ../database/data_3/sopr/song02.sopr
```

B.3.2 Trénovanie siete

Skripty z Matlabu na trénovanie sietí sú uložené v zložke database.

```
cd ../database
```

Uživateľ vytvorí adresáre na výstupné vektory zo siete:

```
mkdir eval_data_1
mkdir parsed_bass
```

Spustíme Matlabom skript na trénovanie prvej siete:

```
matlab -nodesktop < neural1.m
```

Zo získaných vektorov skript určí výslednú notu:

```
cd ../scripts
./zaokruhli_bass.pl ../database/eval_data_1/song02.outb
```

Obdobným spôsobom postupujeme aj pre druhú sieť.

```
cd ../database
mkdir eval_data_2
mkdir data_3/sb
```

Program paste vytvorí vstupné vektory pre druhú sieť (z výstupu prvej siete).

```
paste -d' ' ../database/data_3/sopr/song02.sopr
../database/parsed_bass/song02.bassNew
| cut -f2-29 -d' ' > ../database/data_3/sb/song02.sb

mkdir data_3/sb_hist
 cd ../scripts
./copysb.pl ../database/data_3/sb/song02.sb
```

Spustíme Matlabom skript pre trénovanie druhej siete.

```
cd ../database
matlab -nodesktop < neural2.m
```

Uživateľ vytvorí výstupné adresáre:

```
 mkdir parsed_at
 mkdir abc_notation
 mkdir midi_out
```

Skript spracuje výstup z druhej siete:

```
cd ../scripts
./zaokruhli_at.pl ../database/eval_data_2/song02.outat
```

B.3.3 Po-spracovanie

Skript vytvorí výstupné súbory v abc notácii.

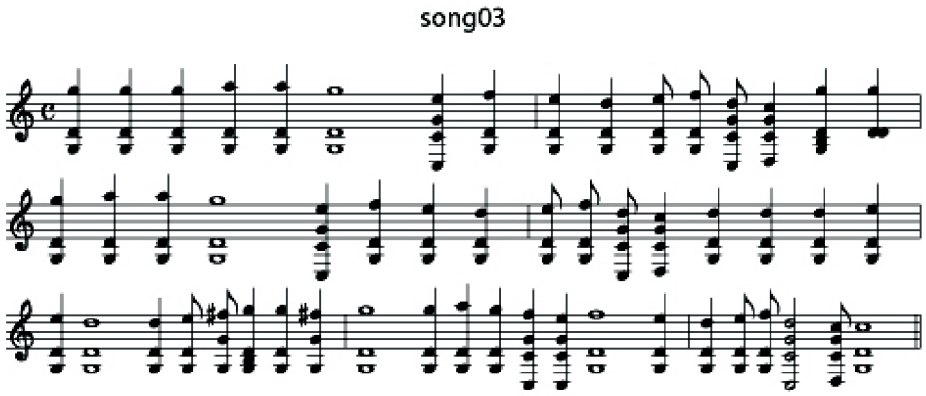
```
./vectors2abc.pl ../database/data_3/times/song02.times
```

Prevod do formátov midi a ps:

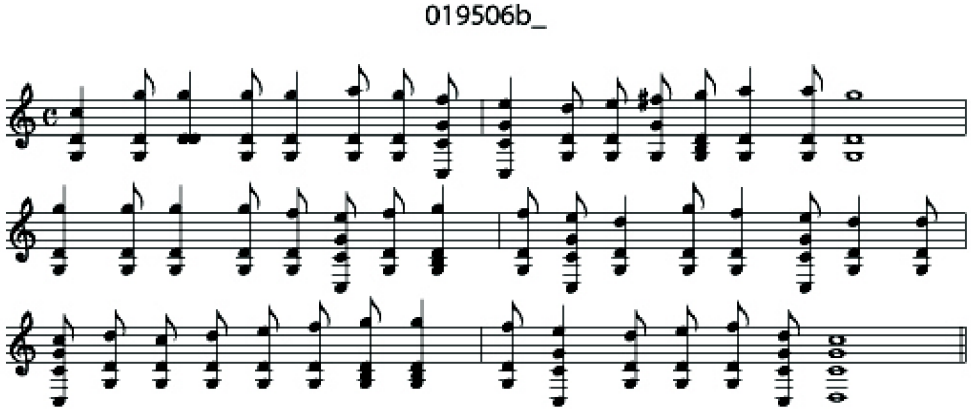
```
cd ../tools/abcmidi/
./abc2midi.exe ../../database/abc_notation/song02.abc
-o ../../database/midi_out/song02.mid

./abcm2ps ../../database/abc_notation/song02.abc -O ../../database/ps_out/=
```

Dodatek C

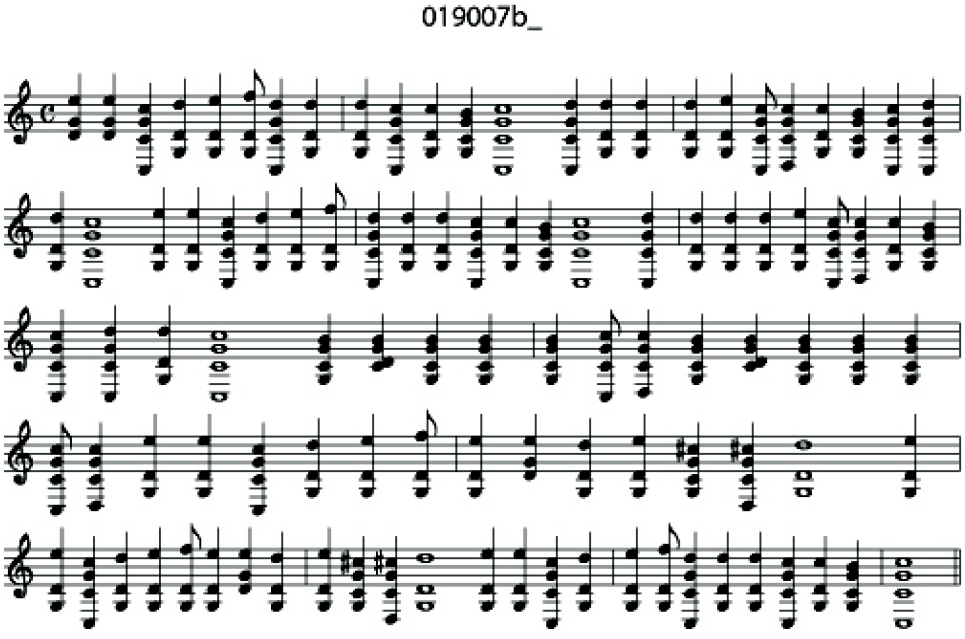
Ukážky harmonizovaných skladieb v notovom formáte



Obrázek C.1: Harmonizovaná melódia song03.mid uložená v adresári *database/midi_test*.



Obrázek C.2: Harmonizovaná melódia 019506b.jpg uložená v adresári *database/midi_test*.



Obrázek C.3: Harmonizovaná melódia 019007b.jpg uložená v adresári *database/midi_test*.